2017

Enhancing Rating-Prediction by Incorporating User Concerns Discovered from User Reviews

Ligaj Pradhan
University of Alabama at Birmingham

ENHANCING COLLABORATIVE FILTERING-BASED RATING-PREDICTION BY DISCOVERING AND INCORPORATING USER CONCERNS FROM USER REVIEWS

by

LIGAJ PRADHAN

CHENGCUI ZHANG, COMMITTEE CHAIR
STEVEN BETHARD
ALAN SPRAGUE
XUAN HUANG
WEI-BANG CHEN

A DISSERTATION

Submitted to the graduate faculty of The University of Alabama at Birmingham,
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy

BIRMINGHAM, ALABAMA

2017

ENHANCING RATING-PREDICTION BY INCORPORATING USER CONCERNS DISCOVERED FROM USER REVIEWS

LIGAJ PRADHAN

DOCTOR OF PHILOSOPHY IN COMPUTER SCIENCE

ABSTRACT

Collaborative filtering (CF) based rating prediction discover similar users and items to predict unknown ratings based on how similar users have rated similar items. Discovering and infusing additional knowledge to supplement the discovery of similar users can potentially improve the accuracy of CF-based rating prediction. Neighborhood discovery and non-negative matrix factorization (MF) are very popular techniques for collaborative filtering. Hence, we try to improve these two techniques to increase rating prediction accuracy by using multi-view clustering to better discover neighborhoods and infusing additional user behavior knowledge, respectively. Additional user behavior knowledge is extracted by mining User-Concern vectors (UC-vectors) which represent hierarchical relationships between hidden user concerns in user reviews. To further improve the rating prediction accuracy, we incorporate the extracted UC- vectors into Deep Neural Network (DNN) based architectures. We propose and experiment with a DNN architecture consisting of two parallel branches to learn user and item latent vectors before aggregating them to make the final rating predictions. The initial input to our DNN are the user and item rating behavior vectors constructed by collecting all the ratings given or received by the users or the items, respectively. We then regulate the learning of the latent user vectors using UC-vectors. In addition to using only rating behavior vectors as the initial input to our DNN model, we also explored using Term Frequency – Inverse Document Frequency (TF-IDF) vectors as the input to DNN. However, TF-IDF is based

on statistical learning alone and does not directly capture the conceptual contents of the text or the behavioral aspects of the writer. Hence, we also propose a novel weighing scheme to extract relatively low dimensional user behavioral vectors similar to our previously extracted UC-vectors and append them to the corresponding user or item TF-IDF vectors. Such additional user behavioral knowledge will allow TF-IDF to better capture user similarity and to further improve rating predictions. Our experiments and results on standard neighborhood-based, NMF-based and DNN-based architectures clearly showed that infusing additional user behavioral aspect can significantly improve the rating prediction accuracy.

Keywords: multi-view clustering, rating prediction, matrix factorization, User-Concerns, deep neural networks, TF-IDF

DEDICATION

To my loving wife Vijaya Bijukchhe and parents Mr. Thamesh Kumar Pradhan and Mrs. Kalyani Shrestha Pradhan

TABLE OF CONTENTS

Page

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATOINS

CB	content-based
CBOW	continuous bag-of-words
CF	Collaborative filtering
CoNMF	Co-Regularized Non-Negative Matrix Factorization
CTR	click through rate
DAG	directed acyclic graph
DL	deep learning
DNN	Deep Neural Network
DSPRP	Deep Semantic Projection based Rating Predictor
DSSM	Deep Structured Semantic Models
eNMF	extended Non-Negative Matrix Factorization
ESA	Explicit Semantic Analysis
HFT	Hidden Factors as Topics
hLDA	Hierarchical Latent Dirichlet Allocation
HSO	Hirst St-Onge
IDF	Inverse Document Frequency
IEF	Inverse Identity Frequency
IL	Incremental Learning
JCN	Jiang-Conrath
k-NN	k-Nearest Neighbors
LCH	Leacock-Chodorow

LCS	least common subsume
LDA	Latent Dirichlet Allocation
LOD	Linked Open Data
LSA	Latent Semantic Analysis
MAE	mean absolute error
MF	matrix factorization
MKL	Multiple Kernel Learning
nBOW	normalized bag-of-words
NCRP	nested Chinese restaurant process
NMF	non-negative matrix factorization
NMPI	normalized point-wise mutual information
ODP	Open Dictionary Project
PAM	Pachinko Allocation Model
PMF	Probabilistic Matrix Factorization
PMI	point-wise mutual information
RES	Resnik information content
RMSE	Root Mean Squared Error
SGD	stochastic gradient descent
SPrank	semantic path-based ranking
SR	Selective Regularization
SVD	Singular Value Decomposition
TF	Term Frequency
TF-IDF	Term Frequency - Inverse Document Frequency
TP	Topic Proportion
TP-IEF	Topic Proportion - Inverse Entity Frequency

UC	User-Concern
WMD	Word Mover's Distance
WUP	Wu-Palme

CHAPTER 1

INTRODUCTION

With tremendous growth of e-commerce and information over the Internet, recommendation and rating prediction systems have become an integral part of almost every online system. CF is a type of system which exploits past rating behaviors of users to recommend items liked by similar users. CF is increasingly popular in recommendation and rating prediction systems because it is better suited to make personalized recommendations than CB systems. CF explicitly focuses on what a user likes or dislikes by identifying similar users and what the similar users like. CF based rating prediction systems also consider how similar users rate similar items to make rating predictions.

The second chapter of this dissertation thesis presents some background on recommendation and rating prediction systems. As this dissertation work is mainly focused on CF-based recommendation and rating prediction models, this section particularly elaborates the description of CF based rating prediction models. Clustering based neighborhood discovery and matrix factorization (MF) based rating prediction are further explained as these techniques are very popular for CF-based models [1, 2]. As this dissertation will be exploring regularization based on user behavior, a background of regularization is also presented in this chapter.

The first effort to improve CF-based rating prediction in this dissertation is made by improving the neighborhood discovery. This work relies on multi-view clustering to

discover deeper and more intricate relationship between users and enhance neighborhood discovery. Hence in the third chapter, this dissertation work explores multi-view clustering to discover better user and item clusters. Multi-view clustering refers to the clustering process where multiple modalities will be used collectively during the clustering phase. Multi-view clustering uses information from various sources and makes the neighborhood search more effective and accurate. It is expected to improve the accuracy of cluster-based CF by improving the user and item clustering.

This dissertation work also tries to improve CF-based rating prediction by improving MF based techniques. All the known ratings can be represented in a ratings matrix *R,* by arranging users in the rows and items in the columns. The rows would then represent user rating vector and the columns would represent item rating vector. After factorization, sub matrices are obtained and used to regenerate the original matrix back again, along with the predictions for the previously unknown rating values. The dissertation work initially focuses on NMF and to incorporate user behavioral knowledge along with traditional NMF. Hence, the fourth chapter of this dissertation work presents how user concerns and interests can be extracted from their user reviews using topic modeling techniques and arranged in a hierarchical tree based on their inter-relationships. Chapter five then focuses on extracting UC-vectors from the hierarchical tree of user concerns and incorporating them with NMF based rating prediction model.

Different users care about different aspects such as cost, proximity, location, time and cleanliness in the items they buy/visit. They express their concerns regarding such aspects while writing reviews for these items. In the rest of the dissertation work, such concerns expressed in the user reviews are referred to as User-Concerns. It is presumed

that similar users have similar User-Concerns. Therefore, the first step is to extract various User-Concerns hidden in user reviews. Subsequently, the dissertation explores if the accuracy of rating prediction systems can be improved by relating users using the discovered User-Concerns.

To discover User-Concerns, this dissertation starts by extracting topics from user reviews by using state-of-the-art topic modeling techniques. In many cases, the extracted topics are a mixture of different concepts and hence likely to be unclear and incomprehensible [3]. Hence, the dissertation work proposes a method to extract a large number of topics and then rely on hierarchical clustering that exploits semantic similarities between these topics to generate a relatively small number of more coherent topic clusters. Furthermore, central words are computed for such clusters to generate User-Concerns. A hierarchical tree (UC-Tree) is also created for such User-Concerns, which represents their inter-relationships. Depending upon the User-Concerns discovered in the user reviews and their corresponding edge distances computed using the UC-Tree, this dissertation explores to discover deeper relationships among users. It is expected that deeper and more intricate relationships among users or items can be represented by relating them in the realm of User-Concerns. NMF is frequently used to factorize ratings matrix (whose elements are the ratings given by the user in the corresponding row to the item in the corresponding column) and to predict unknown ratings using the factorized matrices [4]. In this dissertation work, conventional NMF approach will also be extended with the knowledge discovered from the User-Concern hierarchy to improve the rating prediction.

DNNs have achieved significant success in predicting user sentiments and ratings over various range of items. Several neural models project the initial user and item vec-

tors into semantic spaces before predicting how much the user likes the items [5, 6]. To further improve the rating prediction accuracy, the extracted UC- vectors are incorporated into Deep Neural Network (DNN) based architectures. In the sixth chapter of this dissertation a DL techniques called Autoencoder is explored to project the input vectors to some latent space and perform ratings prediction. Shallow to Deep networks can be trained to produce the input itself at the output with Autoencoders and in this process, some intermediate layer learns a low dimensional representation of the input. The approach taken in this dissertation work can be described with a two-step process. First, an unsupervised training is performed to train user-based autoencoders and item-based autoencoders to generate latent representations of the user and item rating vectors. Then as a second step a supervised learning is performed to train a DNN to predict rating values for the latent user and item vectors.

Furthermore, a deep architecture which is referred to as Deep Semantic Projection based Rating Predictor (DSPRP), consisting of two parallel branches to learn user and item latent vectors is also experimented with in this dissertation. Finally, the dot product of the two latent representations are used to predict the corresponding rating values. Initially such DNN was trained using user and item rating vectors obtained from the rating matrix R. UC-vectors were then incorporated to regulate the learning of latent user vectors and improve the overall prediction accuracy. Chapter seven presents the DSPRP model and various experiments conducted for such deep rating predictions.

In addition to using only rating behavior vectors as the initial input to our DNN model, we also explored using TF-IDF vectors as the input to DNN. TF-IDF computes weight for each word in a document which increases proportionally to the number of

times the word appears in a specific document but is counterbalanced by the number of times it occurs in the collection of documents. TF-IDF is the state-of-the-art for computing relevancy scores between documents. TF-IDF is also used for making rating predictions [6-9]. However, it is based on statistical learning alone and doesn't directly capture the conceptual contents of the text or the behavioral aspects of the writer. Hence, in chapter eight of this dissertation work relatively low dimensional user behavioral vectors are extracted from the same text, from which TF-IDF vectors are extracted, and used to enrich the performance of TF-IDF. User-Concerns embedded in user reviews are extracted and appended to TF-IDF vectors to train a deep rating prediction model. Experiments show that adding such conceptual knowledge to TF-IDF vectors can significantly enhance the performance of TF-IDF vectors by only adding very little complexity.

The work presented in this dissertation are evaluated using user-restaurant review dataset from the Yelp challenge dataset [37] and user-hotel review dataset crawled from TripAdvisor [52]. The proposed approaches for making rating predictions are compared and evaluated with rating predictions systems made by using several state-of-the-art techniques on neighborhood discovery, NMF and DNNs. The proposed approaches and techniques are shown to be able to discover deeper and more intricate relationships among users and enhance the accuracy of rating prediction.

CHAPTER 2

INTRODUCTION TO RECOMMENDATION AND RATING PREDICTION SYSTEMS

Recommendation systems basically work by predicting item ratings for users and suggesting items with high predicted ratings to the users. Today, people rely on such automatic rating prediction systems in numerous daily activities. Many different kinds of rating prediction systems have evolved which will be briefly discussed in this section. Model based techniques such as Cluster Model builds clusters of similar users and items to make the prediction process faster and more effective. Cluster Models and an additional popular rating prediction technique called MF are introduced in this chapter. MF has become very popular in building rating prediction systems because of its high accuracy and scalability.

Recommendation and Rating Prediction Systems

With an explosive growth of information, e-commerce and the Internet, recommendation and rating prediction systems have been a part of almost every online activity. People reply on some form of recommendation or rating prediction system to get personalized recommendations in numerous online activities such as shopping, finding research papers, reading news, watching movies, exploring vacation destinations, listening to the music, choosing restaurants, and searching web pages. Due to its growing practical applications, there has been tremendous interest in rating prediction systems and various ap-

proaches have been developed for making such rating predictions. However, despite these advances, such systems still require improvements and thus are still a hot research area [22]. Recommendation systems generally work by predicting user ratings for various items and then by suggesting items with the highest ratings. Recommendation and rating prediction systems can broadly be classified into three types:

a) Content-based System

b) Collaborative Filtering System

c) Hybrid System

Content-based Systems (CB)

In the CB approach, items similar to the ones previously liked or rated higher by the user are rated higher and finally recommended. Basically, for each item an item-profile is created which consists of various feature-value pairs. Such features are important characteristics of the items. For example, a movie can be described by its title, language, actors, genre, length, date, etc. Such item features can broadly be classified as categorical and continuous types. Categorical features are of Boolean types. For example, if a feature for movies is language, then there is a component for each language. The value of a component is '1' if the movie is in that language, otherwise '0'. Continuous features are numerical values of items that cannot be represented by Boolean values, for example, the length of a movie. Additionally, for each user a user-profile is maintained which consists of user preferences and various characteristics of the user. For example, in the MovieLens dataset a user is described by age, gender, occupation and zip code [23]. Hence most CB systems will contain a database of item-profiles, user-profiles and some

technique to compare what items may be of interest to a particular user-profile. User profiles can be collected either explicitly or implicitly. In explicit user information collection, users are asked to provide their preferences and information. However, this approach will burden the user to enter his details. In the implicit user information collection method, the system automatically collects and updates user information as the user uses the system. CB's greatest advantage is that it can make recommendations or predict ratings as soon as user profiles and item profiles are known, even when the user has not given a single rating. However, user behaviors and experience are never considered for recommendation purposes, which is a major disadvantage of CB. Hence, it is not well suited for making personalized recommendations. Figure 1 shows a generic structure of CB where a reader reads a news and then is recommended other news which are similar to the one he read. In other words, those news that have higher similarity to the news read by the users will be rated very high to make recommendations.

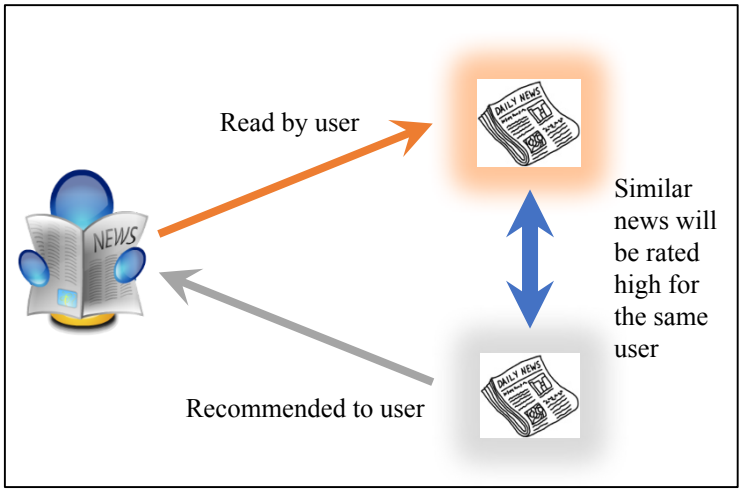

Figure 1. The general structure of a CB recommendation system.

Collaborative Filtering Systems (CF)

CF systems exploit the rating behavior of users to discover similar users and recommend items liked by similar users. Hence the focus is to discover other like-minded users by analyzing their rating behaviors. CF systems represent the entire user-item space in a rating matrix R, where entry R_{ij} represents the rating or a preference score in some scale given by i^{th} user to j^{th} item. All the entries that represent items not yet rated by corresponding users are generally replaced with zeros for computational purposes [22]. Table 1 shows an example of a ratings matrix for movies. The items that are not yet rated are denoted as '?'.

Table 1. Sample ratings matrix for movies

	The Avengers	The Dark Knight	The Notebook	Roman Holiday	A Walk to Remember
Alice	5	5	?	1	1
Bob	?	5	1	1	1
Charlie	2	1	?	5	5
David	1	2	5	?	5

CF approach exploits such rating matrices to solve two related problems: finding similar users and finding similar items. The output of such systems can be either a prediction of a numerical value (such as rating prediction) or a list of top N recommended items. CF approaches can broadly be divided into Memory-based and Model-based approaches.

Memory-based approaches use the entire or a sample of user-item ratings to discover users with or items of similar interest and predict an unknown rating or generate a top N recommendation list. Memory based approaches can be broadly divided into two types:

a) User-based

b) Item-based

In the user-based approach, the task is to find similar users by comparing the ratings given by them to the same items. The prediction for a previously unrated item is made by taking some weighted average of ratings given by similar users who have already rated that item. Similarly, in item-based approach, the task is to find similar items by comparing ratings given by the same user to different items. The prediction can thus be made by taking some weighted average of the most similar items rated by the same person.

Model-based approaches build some statistical or probabilistic models beforehand and do not have to use the entire ratings dataset at the time of prediction. Model-based approaches are usually fast, accurate and less sensitive to sparsity. However, a lot of time is spent in learning the model which makes online learning inefficient. Clustering models, decision trees, aspect models, latent factor models, dimension-reduction models and Bayesian Network models are examples of such model-based approaches [4].

Clustering Models

In Clustering Models, like-minded users are clustered into one group, and the unknown ratings for individuals that belong to a particular cluster can be predicted using known ratings in that cluster. Getting some weighted average of the ratings from the nearest neighbors within the same cluster can be one way of making predictions for an unrated item [24-26]. The underlying expectation from such cluster models is to improve the quality of CF and increase its scalability [24]. The clustering process partitions a

large-dimensionality space into smaller groups with lesser users, items and ratings. Traditional CF algorithms can thus be applied to these smaller groups to obtain greater scalability and quality for CF.

Matrix Factorization (*MF*)

Latent Factor Model and MF based techniques have become popular in CF because of their high accuracy and scalability [4]. Such approaches assume that a small set of latent features can describe the user-item rating behavior. Thus, a low-rank MF with rank-k (referring to the number of latent features of users and items) is performed on the user-item rating matrix $R_{n\times m}$, where n is the number of users and m is the number of items. MF and Singular Value Decomposition (SVD) are highly associated [1]. SVD basically decomposes a matrix P of size $n\times m$ into a product of three matrices as $P = A\Sigma BT$, where A is of size $n\times n$, Σ is of size $n\times m$ and finally B is of size $m\times m$. A and B are orthogonal matrices and Σ is a diagonal matrix. The diagonal of Σ is comprised of K non-zero elements. Hence, effectively these three matrices have dimensions of $n\times K$, $K\times K$ and $K\times m$, respectively. The values of the k diagonal elements of Σ are in decreasing order. The product of these three matrices obtained by choosing only the $r<<K$ first singular elements gives $\hat{R}$, which is the closest rank-r approximation of R in terms of Frobenius Norm [1] as expressed in equation 1.

$$\left|\left|\hat{R} - R\right|\right|_F^2 = \sum_{i=1}^{n}\sum_{j=1}^{m}\left|\hat{p}_{ij-}p_{ij}\right|^2 \qquad (1)$$

The ratings matrix R is generally a very sparse matrix with many unknown entries. Hence the above SVD approach cannot be applied directly to factorize such sparse

matrices. It needs additional processing such as replacing the unknown entries with zeros or several other methods similar to the ones addressed in [27-29]. A very different technique called Incremental Learning (IL) is generally used to avoid this problem of unknown entries. The decomposed matrices are learned by iterating over the available ratings data one by one. The 'Base Algorithm' presented by Patrick Ott decomposes matrix R into only two feature-matrices unlike three matrices in SVD [1]. The factorization produces two smaller matrices $U_{n\times K}$ and $V_{K\times m}$. The n rows of $U_{n\times K}$ matrix represent the n user vectors with K latent features each and the m columns of $V_{K\times m}$ matrix represent the m item vectors with K latent features. The exact meaning of these latent features is difficult to interpret; however, they ideally capture inherent features of users and items [30]. The matrices V and U are learnt so that their product is an approximation of R. Basically, the user and item vectors are learnt iteratively so that their inner product is approximate to the known rating (the corresponding entry in matrix R). Only the known ratings are used to learn these user and item vectors. This is also equivalent to filling the unknown entries (as represented by '?' in Table 1) in the sparse rating matrix R with zeros. The predicted rating by user u_i to item v_j is obtained by the dot product of the i^{th} row vector of matrix U and the j^{th} column vector of matrix V. Hence, provided that the real rating r_{ij} of user u_i to item v_j is known, the quadratic error can be computed as the following equation 2.

$$\varepsilon_{ij}^2 = (r_{ij} - \sum_{k=1}^{K} u_{ik} v_{kj})^2 \qquad (2)$$

MF is carried out by minimizing this loss function. In order to find the update values for each element of u and v vectors, partial derivatives of the above error function are taken in terms of u and v as shown in equations 3 and 4.

$$u'_{ik} = u_{ik} + \alpha \frac{\partial \varepsilon_{ij}^2}{\partial u_{ik}} = u_{ik} + 2\alpha \varepsilon_{ij} v_{kj} \quad (3)$$

$$v'_{kj} = v_{kj} + \alpha \frac{\partial \varepsilon_{ij}^2}{\partial v_{kj}} = v_{kj} + 2\alpha \varepsilon_{ij} u_{ik} \quad (4)$$

where α is called the learning rate, and it controls the rate of approaching the minima. Typically, the value of α is very small. For example, Ott uses 0.001 for α [1]. u'_{ik} and v'_{kj} denote the updated u_{ik} and v_{kj} entries, respectively.

Regularization

The above is a very basic algorithm for MF. There are plenty of variants of this basic algorithm. One such variant is to introduce a regularization term by introducing another new parameter β to control the magnitudes of the user and item feature vectors. β is generally set around 0.02 and this extension to the basic algorithm helps to avoid overfitting [1, 31]. This would suppress overtraining and improve the prediction accuracy for unseen data. The squared error function with the regularization parameter is expressed in equation 5 as follows:

$$\varepsilon_{ij}^2 = \left(r_{ij} - \sum_{k=1}^{K} u_{ik} v_{kj} \right)^2 + \frac{\beta}{2} \sum_{k=1}^{K} (||u||^2 + ||v||^2) \quad (5)$$

The new update rules for this squared error function are also obtained similarly to the basic loss function by taking the partial derivatives with respect to *u* and *v* alternately, they are expressed with equations 6 and 7 as follows:

$$u'_{ik} = u_{ik} + \alpha \left(\frac{\partial \varepsilon_{ij}^2}{\partial u_{ik}} - \beta u_{ik} \right) = u_{ik} + \alpha \left(2\varepsilon_{ij} v_{kj} - \beta u_{ik} \right) \quad (6)$$

$$v'_{kj} = v_{kj} + \alpha\left(\frac{\partial \varepsilon_{ij}^2}{\partial v_{kj}} - \beta v_{kj}\right) = v_{kj} + \alpha\left(2\varepsilon_{ij}u_{ik} - \beta u_{v_{kj}}\right) \quad (7)$$

The algorithm for MF using the above procedure is presented below:

Algorithm 1 Basic Algorithm for MF using IL

1: Initialize matrices U and V for the first time using fixed procedure or randomly
2: for k =1 to K, do
  repeat
    for $\forall (u_i, v_j, r_{ij}) \in T$, do
      Compute $\boldsymbol{\varepsilon_{ij}}$.
      Update u_{ik}, v_{kj}.
    end for
    Re-compute $\sum_{(\boldsymbol{u_i}, \boldsymbol{v_j}, \boldsymbol{r_{ij}}) \in \boldsymbol{T}} \boldsymbol{\varepsilon_{ij}^2}$
  until some terminal condition
 end for

Note: K denotes the total number of latent features considered and T denotes the training dataset.

Matrix $\hat{R}$ matrix as shown in Table 2 is obtained by the product of the U and V matrices and has values very close to the already existing ratings. In addition, it also contains predicted values for the previously unknown ratings. It can be observed that Alice and Bob are similar in their preference to movies, as both of them prefer action hero based movies and have low preference for romantic movies. Similarly, Charlie and David have similar preferences exactly opposite to Alice and Bob. The CF based on the Basic Algorithm for MF using IL seems to be able to capture user similarity and predict movie-ratings similar to how other similar users have rated the movies. CF approach has significant advantages as it can make recommendations without knowing the properties of the items by utilizing the rating behavior of users. In a way, it makes recommendations based on people's experience. CF is specially adapted to make personalized recommendations as it is trained on the basis of user behaviors and experience. However, it also suffers from some problems such as sparsity, cold start, and scalability. Sparsity refers to a very

sparse user-item ratings matrix as most users rate only a very small portion of all the items. Similarly, cold start refers to the situation when the user does not have previous ratings and hence CF will not be able to find any similar users. Scalability issues arise when the number of users and items grow enormously resulting in an exponential growth in computations [22]. Hence, CF and CB are often combined as hybrid systems to over-come these limitations.

Applying the above Basic Algorithm for MF using IL with $\alpha = 0.002$, $\beta = 0.02$ and K=2 produced the following predictions for the missing ratings after 5000 iterations:

Table 2. Approximated $\hat{R}$ matrix for the ratings matrix R presented in Table 1

	The Avengers	The Dark Knight	The Note-book	Roman Holiday	A Walk to Remember
Alice	4.94	5.03	1.01	1.00	1.00
Bob	4.87	4.94	1.00	1.00	1.00
Charlie	1.50	1.48	5.10	4.95	5.02
David	1.52	1.50	5.00	4.87	4.94

Probabilistic Matrix Factorization

Probabilistic Matrix Factorization (PMF) is a probabilistic approach to CF. PMF is shown to scale linearly with the number of observations and perform very well on large, sparse and imbalanced datasets [29]. It also generalizes well for users with very few ratings. Suppose there exists a rating matrix $R_{n\times m}$ and user and item matrices $U_{n\times K}$ and $V_{K\times m}$ as discussed while explaining MF. In PMF, the ratings matrix R is modeled as draws from a Gaussian distribution. The conditional distribution over the observed ratings are defined as presented in equation 8.

$$p(R|U,V,\sigma^2) = \prod_{i=1}^{n}\prod_{j=1}^{m}\left[N(R_{ij}|U_iV_j,\sigma^2)\right]^{I_{ij}} \qquad (8)$$

where mean for R_{ij} is given by $U_i V_j$and σ^2 is the noise variance. $N(x|\mu,\sigma^2)$ denotes the probability density function of the Gaussian distribution with mean μ and variance σ^2. I_{ij} is 1 if the i^{th} user have rated the j^{th} item, otherwise it is 0. The user and movie feature vectors are given zero-mean spherical Gaussian priors as presented in equation 9.

$$p(U|\sigma_U^2) = \prod_{i=1}^{n} N(U_i|0,\sigma_U^2 I), p(V|\sigma_V^2) = \prod_{j=1}^{m} N\left(V_j\middle|0,\sigma_V^2 I\right) \qquad (9)$$

This can also be interpreted as each row of U and each column of V are drawn from a multivariate Gaussian with mean μ=0. The variance for both cases are some multiple of identity matrix I. The log of the posterior distribution over the user and item features can also be expressed as equation 10.

$$\ln p(U,V|R,\sigma^2,\sigma_V^2,\sigma_U^2)$$

$$= -\frac{1}{2\sigma^2}\sum_{i=1}^{n}\sum_{j=1}^{m} I_{ij}\left(R_{ij} - U_i V_j\right)^2 - \frac{1}{2\sigma_U^2}\sum_{i=1}^{n} U_i^T U_i$$

$$-\frac{1}{2\sigma_V^2}\sum_{i=1}^{m} V_j^T V_j - \frac{1}{2}\left(\left(\sum_{i=1}^{n}\sum_{j=1}^{m} I_{ij}\right) ln\sigma^2 + nKln\sigma_U^2 + mKln\sigma_V^2\right)$$

$$+ C \qquad (10)$$

where C is a constant independent of the parameters. Maximizing the log-posterior probability over the user and item features with fixed σ^2, σ_U^2 and σ_V^2 is equivalent to minimizing the following objective function as presented in equation 11.

$$E = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{m} I_{ij}(R_{ij} - U_i V_j)^2 + \frac{\lambda_U}{2}\sum_{i=1}^{n} ||U_i||_{Fro}^2 + \frac{\lambda_V}{2}\sum_{j=1}^{m} ||V_j||_{Fro}^2 \qquad (11)$$

where $\lambda_U = \sigma^2/\sigma_U^2$, $\lambda_V = \sigma^2/\sigma_V^2$, and $||.||_{Fro}^2$ denotes Frobenius norm. The dot product of U_i and V_j is passed through a logistic function which bounds the output in the range

[0,1]. This is further fed to the following function as presented in equation 12 to map the actual rating values.

$$t(x) = \frac{x - 1}{K - 1} \qquad (12)$$

where x is the output from the logistic function and the actual ratings are, whole numbers ranging from 1 to *K*. This objective function can be minimized using steepest descent which is linear to the number of observations.

Hybrid Systems

Hybrid Systems combine various rating prediction techniques and focus on achieving higher accuracy and performance. There has been extensive research in combining CF and CB to overcome the disadvantages of one approach by the advantages of the other. Feature augmentation, weighted, mixed, switching and cascading are some of the many techniques to combine CF and CB into Hybrid Systems [22]. Meenakshi et al. summarize different ways of hybridization as follows [22]:

a) Implementing CF and CB separately and combining their predictions.
b) Incorporating some CB characteristics into a collaborative approach.
c) Incorporating some collaborative characteristics into CB approach.
d) Constructing a general unifying model that incorporates both CB and collaborative characteristics.

CHAPTER 3

ENHANCE CF-BASED RATING PREDICTION SYSTEMS WITH MULTI-VIEW CLUSTERING

Background

Multiple views represent different sources or the natures of data regarding the items of interest. Multiple views can be collectively used to learn models by considering the diverse information offered by such different views. On the basis of when the information from different views are fused together, multi-view learning approaches can be of early integration, intermediate integration or late integration types. Features are combined together at the feature level to create a unified feature set in early integration. Results from models created using individual views are combined to give a common consensus in the late integration approach. The information from multiple views is used during the model learning phase itself in intermediate integration. Xu et al. classify various multi-view learning techniques into three types [32] - Co-training, Multiple Kernel Learning (MKL), and subspace learning. Co-training is similar to the intermediate integration where various views are used to train alternately to maximize mutual agreement. In MKL, kernels corresponding to different views are linearly or non-linearly combined to improve learning performance. Subspace learning tries to obtain latent subspace shared by multiple views.

Literature Review: Clustering Models and Multi-View Learning

Connor et al. proposed to use clustering techniques to partition the item-space so that a large-dimensional space is reduced into a smaller space with fewer items, ratings and users, to reduce the computation for prediction [24]. They use existing data partitioning and clustering algorithms to partition sets of items in the basis of user ratings. Finally, the predictions are made separately for each partition. They have shown that clustering improves scalability of CF. In this dissertation, the neighborhood search space is reduced by clustering similar items and users together, by leveraging information from multiple views to improve the clustering quality and produce better rating prediction for cluster-based CF.

CF system exploits the rating behavior of users to discover similar users and recommends items liked by similar users. Typically, users rate only a very small number of items and because of that very less information is available to cluster similar users by computing their similarities. The similarity between users is drawn from only a small number of overlapping ratings. This problem is also called 'data sparsity problem' in CF-based models and results in imprecise and inaccurate predictions. Hence, Pham et al. have used information from social networks to cluster users into sub clusters for obtaining the neighborhood of the active users [25]. They aim to exploit social relationships to cluster users and use these clusters to make recommendations. Trust network like in Epinion where users state how much they trust each other and scientific collaboration between scholars and citations between publications to recommend research papers are some examples of social relationships discussed in their work. In a practical scenario, a large number of users and items may be introduced continuously.

Shenoy et al. proposed clustering users and items using additional features from the user reviews that represent user preferences [21]. They have shown that ratings predicted using the average of similar users and similar businesses from such cluster models produced more accurate predictions than clustering performed without considering such additional features that incorporate user preferences.

George et al. proposed a novel CF approach based on weighted Bregman co-clustering algorithm [33]. The idea is to simultaneously obtain both user and item neighborhood via co-clustering and use the average ratings of the co-clusters to predict ratings. They have designed incremental and parallel versions of co-clustering and used them to build an efficient real-time CF framework. The recommendation system is thus efficient and dynamic which can adapt to changes based on new users and items. Our goal is also to use improved clustering approach to form better sub clusters of users and items to use them for recommendations. However, this dissertation focuses more in gathering information from different sources to group similar user and items into groups. The main goal of this work is to show how effective information fusion from multiple sources can improve accuracy of cluster-based CF rather than focusing on improving efficiency or dynamic adaptation to new users and items.

He et al. combined multiple views and proposed a framework called Co-Regularized Non-Negative Matrix Factorization (CoNMF) which is an extension over NMF [10]. CoNMF perform multi-view clustering by jointly factorizing the multiple matrixes through co-regularization. They measured clustering accuracy using extrinsic criteria such as evaluating against human labeled ground truth and show the effectiveness of this clustering method. This dissertation work also employs their clustering method to

boost the clustering quality of users and items and show that effectively discovering and utilizing information from multiple sources to construct user and item clusters can boost the accuracy of cluster based CF and thus increase the accuracy of the rating prediction systems.

The Proposed Approach

CF based rating prediction systems predict unknown ratings of a user for an item, based on the analysis made on how similar users rate similar items. Model-based approaches such as cluster models provide efficient means to find out similar users or similar items as they inherently reduce the search space by previously grouping similar users or items into clusters. The final rating prediction is estimated using these clusters instead of the entire user-rating space. Hence, the accuracy of such cluster-based approaches can be improved by improving the clustering process itself. This dissertation work presents how multi-view clustering can be used to cluster users or items by leveraging information from multiple modalities to improve the accuracy of CF-based rating prediction systems. Yelp business rating dataset is used to test our approach on user-restaurant rating prediction. This chapter begins by identifying multiple views for both users and restaurants. Each view represents a distinct source of information regarding the user or the restaurant. These views are then used to perform multi-view clustering for both users and restaurants, respectively. To predict the unknown rating of a user for a restaurant, the averages of k-Nearest Neighbors (k-NN) are computed from the respective user-cluster and the restaurant-cluster.

In multi-view clustering, multiple views provide different sources of information regarding the items to be clustered. These multiple views can be used together during the clustering process, which could be more accurate than clustering performed using individual views or simply by combining these features at the feature level [22, 34].

In this dissertation work, User-Restaurant ratings from Yelp dataset are used to demonstrate the effectiveness of our approach. User-Restaurant rating matrix, in which each element is the ratings given by the user in the corresponding row to the restaurant in the corresponding column are vital to discover similar users in many CF-based recommendation systems. However, this matrix is very large and sparse in reality [22]. Hence, categories are used to reduce the dimension of this matrix. In particular, restaurants (14,531 in total in our dataset) can be grouped into way fewer categories (240 in total in our dataset). Therefore, User-Restaurant matrix is replaced by User-Category matrix, whose elements are the average ratings given to restaurants in the category in the corresponding column, by the user in the corresponding row. This greatly reduces the dimensionality and the sparsity of this matrix. This is similar to reducing the item-space by partitioning items into various clusters as in [24]. Similarly, for the item-based approach Restaurant-Category matrix is used to discover similar restaurants.

Multiple Views

Views refer to separate sources of information (feature sets) that can be used for clustering. Two views UC and UMS are generated for users and two views RC and RMS for restaurants. UC $\in R_{m\times c}$, where m is the number of unique users and c is the number of restaurant categories. The corresponding value in matrix UC_{ij} is '1' if the user U_i has

rated a restaurant in the category C_j, otherwise it is '0'. UMS $\in R_{m\times 2}$, and contains the mean and the standard deviation of the ratings given by each unique user. Similarly, RC $\in R_{n\times c}$, where n is the total number of unique restaurants and c is the number of restaurant categories. The corresponding value in matrix RC_{ij} is '1' if the restaurant R_i belongs to category C_j, otherwise it is '0'. One restaurant can belong to multiple categories. RMS$\in R_{n\times 2}$, and contains the mean and the standard deviation of the ratings received by each unique restaurant.

Multi-View Clustering

Multiple modalities or views may be collectively used to cluster items. Each such view may present different nature of information regarding the items. There has been significant amount of work with multi-view learning and multi-view clustering [10,34-35]. Depending upon when the information from various views is fused together, multi-view clustering can have Early Integration, Intermediate Integration or Late Integration. Features are fused together at the feature level to create a unified view in Early Integration. Results from individual view's clustering are fused for common consensus in Late Integration. Closeness in multiple views is considered during the clustering process itself in Intermediate Integration. CoNMF as proposed by He et al. is of Intermediate Integration type [10]. It is an extension over NMF. NMF factorizes a data matrix into two non-negative matrices [2, 10]. If $V \in R_{m\times n}$ is a data matrix with m items and n attributes, NMF factorizes V into W and H sub matrices such that $V \approx WH$, where W and H are of $m\times\gamma$ and $\gamma\times n$ dimensions, respectively. γ is a predefined desired number of clusters. The columns of W represent γ clusters and its elements give the membership to the γ clusters. There-

fore, an item is associated with the cluster given by the column number for which it has the highest value in *W*. CoNMF performs joint factorization for multiple data matrices from multiple views through co-regularization [10]. Pair-wise CoNMF is used, where, coefficient matrices learned using two views could complement each other during factorization. This approach is shown to be tolerant to the varying feature dimensions in different views and avoid degrading the clustering quality when views with higher clustering utility are used together with some other view with lower clustering utility [10].

Cluster-Based Prediction

Multi-view clustering lies at the heart of the proposed approach. The goal is to cluster both users and items into groups based upon their similarity. Various views for both users and items, which represent various sources of information regarding users and items are identified first. Two views UC and UMS are used for clustering users and two views RC and RMS ae used for clustering restaurants. Once various views are identified, feature sets are extracted for each view and perform clustering by using pair-wise CoNMF presented in [10]. CoNMF is the extension over NMF for multi-view clustering. One of the targets of this dissertation work is to explore if multi-view clustering can be used to improve cluster-based CF. Two rating prediction models are built which use user clusters and restaurant clusters to predict user-based and item-based rating predictions, respectively. In particular, the cluster containing the user/restaurant (for which the rating is being predicted) are discovered and the k-NNs from this cluster are computed. Euclidean distance is used to compute the distance between users (restaurants) within the user (restaurant) clusters, to compute the k-NNs of that user/restaurant. The means of the average ratings given (by users) and received (for restaurants) by these top k-NNs are used

as the predictions from the user-based and the item-based models, respectively. These predictions can be used separately or combined to make the final user-restaurant prediction by using a fusion technique as in [36].

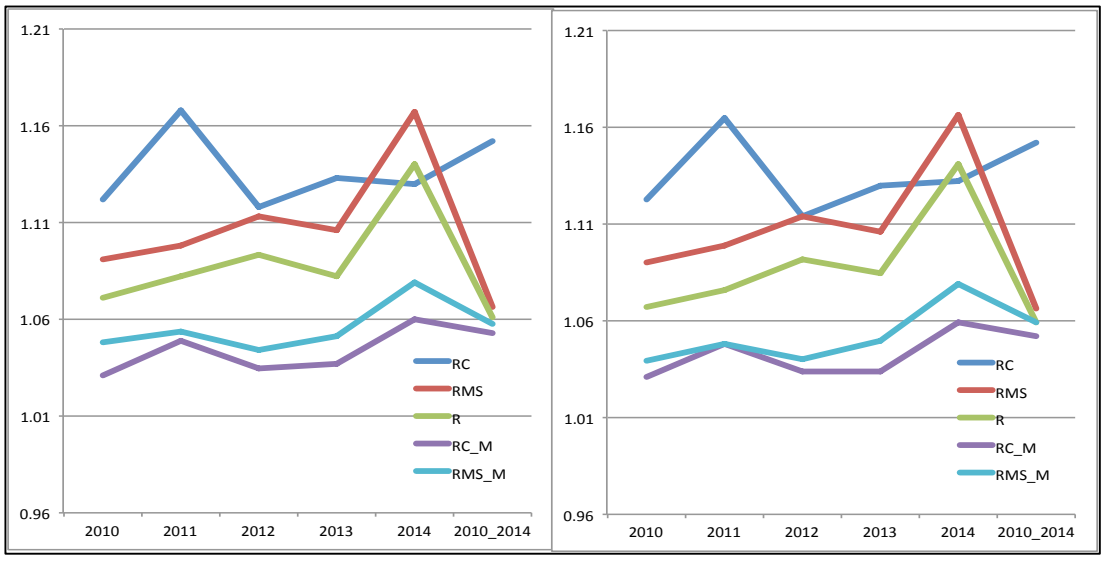

Figure 2. RMSE plot for item-based CF at cluster size 5 and 10.

Figure 3. RMSE plot for user-based CF at cluster size 5 and 10.

Experiments and Results

The proposed approach to predict user-restaurant ratings was tested using the Yelp challenge dataset [37]. The original dataset was filtered and only the restaurants, which received user reviews between 2010 and 2014 were included. Finally, only those reviews, which were from users who have at least ten ratings in total and at least one in each year from 2010 to 2014 were considered in this study. The final dataset consisted of 1.5K unique users, 14.5K unique restaurants and 86K reviews. Experiments were conducted on datasets of two different time intervals. First, the whole five-year dataset (referred to as '2010_2014') was used to train and test our approach. Five subsets; each containing reviews from one single year between 2010 and 2014 for training and testing, respectively were next considered for the experiments. For each review in the datasets, the user, restaurant, categories the restaurant is associated with, average user rating, standard deviation in user rating, average restaurant rating, standard deviation in restaurant rating, year and month of the review and the star rating in the review were extracted. For each dataset, 10% of the total dataset were randomly chosen as test set. However, it was also made sure that every unique user and unique item (restaurant) were present in the training set at least once, by randomly picking at least one instance of each user and restaurant to add to the training data. This process was repeated for each yearly dataset and the '2010_2014' dataset.

For the '2010_2014' and the yearly datasets, three different experiment settings including clustering the users and the restaurants using individual views, clustering by combining feature sets of multiple views, and multi-view clustering using pair-wise CoNMF were tried. Under the second setting, feature sets of UC and UMS were combined to cluster users as 'U', and RC and RMS to cluster restaurants as 'R'. Multi-view

clusters were represented as UC_M, UMS_M, RC_M and RMS_M; indicating respective views whose coefficient matrix is used for clustering after performing pairwise CoNMF [10]. Clustering using individual views (UC, UMS, RC and RMS), i.e., the first setting, and the combination of features from multiple views (U and R: the second setting) were performed using k-means clustering. Different numbers of clusters ranging from 2 to 10 were tried for all three settings. To predict the ratings for each user-restaurant pair in the test dataset, two different predictions were made - one using only user-based clusters and another using only item-based clusters (restaurant clusters). Hierarchical clustering performed with individual views suggested that the number of clusters ranges mostly from 3 to 10. As the number of clusters is needed as a priori for pair-wise CoNMF clustering, different numbers of clusters ranging from 2 to 10 were tested in our experiments. Also for fair comparison regarding the number of clusters, k-means clustering with *k* ranging from 2 to 10 were used while clustering using the individual views UC, UMS, RC, RMS, and the feature combined views U and R. The error of the predictions was measured using Root Mean Squared Error (RMSE). Figure 2 and 3 show the various RMSE errors plotted for various clustering techniques and datasets when the numbers of clusters were 5 (left subfigure) and 10 (right subfigure), respectively.

Discussion

Figures 2 and 3 show that multi-view clustering produces lower RMSE in almost every case. RC_M consistently produces better results than RMS_M in item-based approach. Similarly, UC_M produces lower RMSE than UMS_M in user-based approach. Another interesting observation is the superior performance of multi-view clustering compared to the combined view clustering (R and U). Figure 2 shows that R generates

RMSE lower than both RC and RMS, which indicated that combining features from these two views can improve the performance. Interestingly, both multi-view clustering (RC_M and RMS_M) produce significantly better results than R. Figure 3 shows that U generates RMSE in between UC and UMS, indicating that UMS is significantly better when used alone. In this case simply combining UMS with UC at feature level decreases the accuracy due to added noise. However, by using multi-view clustering the results are either comparable to UMS or better. This demonstrates the advantage of using multi-view clustering in finding the nearest neighbors over directly combining features from different views at the feature level for clustering. Also, when yearly dataset was used, multi-view clustering almost always produced better results than other methods used in our experiments. However, when the whole dataset (2010 to 2014) was used, RMS in item-based and UMS in user-based were better or comparable to multi-view clustering. This may suggest that multi-view is especially advantageous when the dataset is limited, or when there is no conclusive evidence which view would perform the best.

The results and observations suggest that multi-view clustering could be used to produce better clusters of users and items by leveraging the clustering abilities of multiple views. This in turn, is demonstrated to improve the results of CF-based recommendation systems that use cluster models. As similar users and items are placed in the same group, the average of the top k-NN seems to produce better prediction of a user's rating for an item (restaurant).

CHAPTER 4

EXTRACT COHERENT AND INTERPRETABLE USER-CONCERNS FROM USER REVIEWS AND CONSTRUCT HIERARCHICAL TREE FOR THE EXTRACTED USER-CONCERNS

Users write reviews about items regarding various aspects. These aspects may differ from user to user and depend upon what the particular user cares about in an item (restaurant in this chapter). Each user may be thinking about a certain aspect, which is important to him while writing the review. Mining user reviews to discover what the user does or does not care is vital to understanding user behaviors. Topic modeling techniques have been extensively used to discover meaningful topics from user reviews and to interpret user behaviors. Although the discovered topic word distribution may be intuitively meaningful, it may be challenging to accurately interpret the meaning of each topic [3]. Moreover, when the number of topics to be generated is small, Probabilistic Topic Modeling (PTM) techniques are forced to fit all the concepts into a small set of topics, which makes the topics too general [38]. This is also likely to make the generated topics rather noisy and less coherent because of mixed knowledge, thus not suitable for applications that intend to uncover a small number of clear user concerns from short documents such as user reviews. In order to capture clearer topics, PTMs require a larger number of topics, which will be able to capture finer grained and more focused concepts present in the user reviews. However, a larger number of topics may create greater inconvenience to a human trying to understand a concise list of generic user concerns from user reviews.

Literature Review: Auto Labelling of Topics and Generating Hierarchies From Topic Models

Mei et al. have pointed out why automatic and accurate labeling of topics from multinomial topic modeling is necessary. They proposed an automatic process to generate labels for traditional topic word distributions from topic modelling and hence make them more interpretable. Their approach is to extract a set of candidate labels from a reference collection which is related to the current domain, find a good relevance scoring mechanism to find the semantic relevance between the generated topics and the candidate labels, and finally select the top ranked labels for each topic as the corresponding labels. The labels are also auto-generated in this dissertation work for each topic generated by LDA. However, the method proposed in this chapter varies from their approach as we compute the centroid for each topic and represent the topic with the top 10 terms closest to the topic centroid. Word2vec is used for vector representation of each word and cosine similarity to compute the distances. The centrality of each topic is emphasized and labels are effectively created such that they are able to express the essential meaning of each topic.

Zavitsanos et al. used LDA to discover topics at different levels, i.e., with different number of topics (K), and computed conditional independence between topics at adjacent levels to determine if two topics at lower level merge into a single topic at the next higher level in the hierarchy [38]. Using this approach, they organized the discovered topics hierarchically. This dissertation work intends to discover User-Concerns in user reviews and organize them into a hierarchical tree structure. However, in contrast to Zavitsanos et al. LDA is used only once in this dissertation work to generate a large number of topics (e.g. 200 topics). Word2vec and word-movers distance are then used to

compute the distances between topics (group of words representing User-Concerns). These distances are further exploited to perform an agglomerative hierarchical clustering. Different thresholds are used to cut the tree to obtain different numbers of clusters. Inherently information about what clusters merge at each level is also obtained. As such, following the agglomerative tree the hierarchical tree for user concerns are discovered in a bottom to top fashion. The user concerns at the lower level are finer grained and the ones at the upper levels are more generic. The proposed approach directly considers semantic closeness between User-Concerns (topics) and hence it is expected to generate more coherent set of terms describing User-Concerns hidden in the user reviews and their hierarchical organization.

Several variants of hierarchical topic modeling have also emerged following the popularity of LDA. PAM generates a nested hierarchy of topics and captures correlations between topics using a directed acyclic graph (DAG) [39]. The leaf nodes are words, but the intermediate nodes represent correlation between its children, which can be words or topics. Hence PAM is able to represent groups of correlated topics in a hierarchy. hLDA represents the distribution of topics in documents by arranging the topics in a tree. A document is modelled as a mixture of topics in a single path in the tree [40]. Only the plain LDA which is much faster to learn is intended to be used in this dissertation work and hierarchical clustering is to be performed with appropriate semantic distance measures to construct the hierarchy of User-Concerns. The topic coherence for the topics computed using different topic modeling are also compared to our approach in this dissertation work. By using semantic measures to construct the hierarchy it is expected that the topics would be more human interpretable and coherent.

Background

Topic Modeling

Topic Modeling has grown into a very popular technique to extract hidden topics from a collection of documents. The uncovered topics can then be used for various purposes such as document navigation, document classification, trend analysis, and user behavior analysis. The assumption behind topic modeling is that a document consists of multiple topics, and each topic appears in different proportions in each document. Likewise, different words have different probabilities to appear in each topic. Hence, each topic can be represented as a probability distribution of words and each document as a probability distribution of topics. Screening only the words with high probabilities in topics and topics with high probabilities in documents, such topic modeling techniques can discover hidden themes in each document. Latent Dirichlet Allocation (LDA) is a widely-used technique for topic modeling to generate word distribution for topics and topic distribution for documents [41]. Several hierarchical variants such as Pachinko Allocation Model (PAM) and Hierarchical Latent Dirichlet Allocation (hLDA) have also been proposed which generate a nested hierarchy of such topics.

Latent Dirichlet Allocation (LDA)

LDA is an unsupervised and generative probabilistic topic modeling technique based on the assumption that the documents are a mixture of topics (T) and each topic is a distribution over words in a vocabulary (V). LDA uses the bag-of-words concept as each document is represented as a vector of word counts in V. Considering a predefined number of topics K, the generative process for LDA defined in [41] is presented here:

1. A distribution over topics is randomly chosen.
2. For each word in each document
   a. Choose a topic z_n from the distribution over topics
   b. Choose a word w_n from the distribution over words associated with the chosen topic

Documents are assumed to be generated according to this generative process. The latent variables in LDA are the combination of topics, topic distribution per document and the word distribution per topic. The observed variable is the word distribution per document. Now, the goal is to learn the best set of latent variables that can explain the observed word distribution per document. This learning process gives us the word distribution for topics and topic distribution for documents as outputs. Figure 4(a) shows the general structure of LDA.

Pachinko Allocation Model (PAM)

PAM generates a nested hierarchy of topics and captures correlations between topics using a directed acyclic graph (DAG) [39-40]. The concept of topic is not only limited as distribution over words like in LDA, but is also extended to distribution over other topics. The leaf nodes in PAM are words in the vocabulary similar to LDA. The intermediate nodes represent the correlation between its children, which can be either words or topics. Figure 4(b) shows a general structure for a four-level PAM. The immediate parents of the leaf nodes are sub-topics (*s*) which capture the correlations between words. The immediate parents of sub-topics are called super-topics (*S*), and they capture the correlation between topics.

Hierarchical Latent Dirichlet Allocation (hLDA)

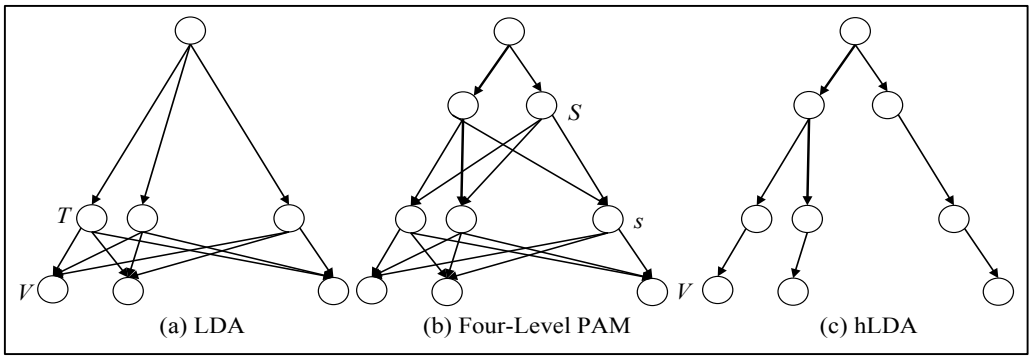

Figure 4. Model structures for LDA, Four-Level PAM and hLDA.

hLDA represents the distribution of topics in documents by arranging the topics in a tree. Documents are modeled as a mixture of topics along a single path in the tree. Hence, each document is generated by topics along a single path in the tree [42]. When learning the model from data, the sampler alternates over an existing or a new path through the tree. While alternating over the paths, hLDA assigns each word in each document to a topic along the chosen path. The structure of the tree is learned along with the topics using a nested Chinese restaurant process (NCRP). This algorithm needs to know the depth of the hierarchy a priori. Figure 4(c) shows the general structure of hLDA.

Word2vec-Mapping Words into Vectors

Word2vec computes vectors for words using techniques such as continuous bag-of-words (CBOW) and an architecture called a Skipgram. The CBOW model is a bag-of-words model where the order of the words does not matter, and the goal is to use surrounding words to predict the word in the middle. The Skipgram model learns the word vectors by trying to optimize the classification of words based on another word in the

same sentence. Mikolov et al. show that these techniques can be used to learn high-quality word vectors from much larger datasets in significantly less time [43].

Word Mover's Distance

Word Mover's Distance (WMD) is a novel distance function based on the word embedding learned with word2vec models from local occurrences in sentences. WMD measures the dissimilarity between two text documents as the minimum amount of distance that the embedded words from one document have to travel to be similar to the word embedded in the other document [44]. WMD is based upon the concept similar to Earth Mover's Distance. As vectors represent words, it is possible to compute the similarity between each pair of words and hence incorporate similarity between individual word pairs into document distance metric. If *d* and *d'* are considered as normalized bag-of-words (nBOW) representation of two documents in (*n*-1)-simplex and $c(i, j)$ as the distance between words *i* and *j*, a matrix $T \in R^{n*n}$ representing the flow of word contents from *d* to *d'* can be built, such that *d* becomes identical to *d'*. Word content from word *i* in document *d* can flow to multiple words in document *d'*. $T_{ij} \geq 0$ denotes how much of word *i* in *d* travels to word *j* in *d'*. Finally, the distance between the two documents *d* and *d'* can be defined as the minimum cumulated cost to move all words from *d* to *d'* as represented by the following expression 13.

$$\min_{T \geq 0} \sum_{i,j=1}^{n} T_{ij} c(i,j) \qquad (13)$$

$$subject\ to: \sum_{j=1}^{n} T_{ij} = d_i \quad \forall i \in \{1, \ldots., n\}$$

$$\sum_{i=1}^{n} T_{ij} = d'_j \quad \forall j \in \{1, \ldots, n\}$$

where d_i and d'_j represent the i^{th} and the j^{th} words in d and d', respectively.

As suggested in [44], several cheap lower bounds of WMD allow speeding up the computation. A lower bound of WMD termed as Relaxed word moving distance (Relaxed WMD) is also implemented where the flow of word content from word i in document d is allowed to flow to only one word in document d', which will be its nearest neighbor (among all the words in d'). This gives a tight bound to the original WMD with lower computational complexity. In this dissertation chapter, WMD is used to compute the distance between two topics representing user concerns.

Finding semantic relatedness between words

Different scoring methods to measure semantic relatedness between words have emerged based on structures of products such as WordNet and Wikipedia. Some of the popular techniques that use WordNet and Wikipedia to score semantic relatedness between words are presented below:

WordNet Similarity

WordNet is a lexical ontology that represents words via 'synsets' [45]. These synsets are structured in a hypernym/hyponym hierarchy (nouns) or hypernym/troponym hierarchy (verbs). With the development of WordNet, a number of methods to compute the semantic relatedness between synset pairs (i.e., sense specific word pairs) have been developed.

a) Vector. In Vector measure, the surrounding words in a piece of text are used to form a context vector [46]. This context vector describes the context in which the word sense appears. Cosine distance between two word senses is computed to find their relatedness.

b) Hirst St-Onge (HSO). HSO measures the semantic relatedness based on the length and tortuosity of the path between nodes [47]. The relatedness score is calculated by a weighted sum of path length between the two words under comparison and the number of turns the path makes.

$$rel_{hso}(w_1, w_2) = C - len(w_1, w2) - k*turns(w_1, w2) \qquad (14)$$

where C and k are constants.

c) Resnik information content (RES). Resnik presented a method RES for weighting the edges between nodes in WordNet by their frequency of use in the text [48]. He compared two concepts by measuring the Information Content of the subsumer with the greatest Information Content from the set of all concepts that subsumed them.

$$sim_{res}(w_1, w_2) = \max_{w \in S(w_1, w_2)} \left(-\log p(c)\right) \qquad (15)$$

d) Lin (LIN). Lin expanded on RES by scaling the Information Content of each node by the information content of their least common subsumer (LCS) as LIN measure.

$$sim_{lin}(w_1, w_2) = \frac{2 * \log p(lcs_{w_1, w_2})}{\log p(w_1) + \log p(w_2)} \qquad (16)$$

e) Jiang-Conrath (JCN). Unlike LIN, JCN measures the relatedness using specificity of the two nodes compared with their LCS.

$$sim_{jcn}(w_1, w_2) = \frac{1}{IC(w_1) + IC(w_2) - 2 * IC\left(lcs_{w_1,w_2}\right)} \quad (17)$$

f) Lesk (LESK). LESK utilizes lexical overlap in dictionary definitions to disambiguate word sense. The sense definitions that have the most words in common are most likely to be similar. Word sense glosses are used rather than dictionary definitions for this purpose, which also extends the dictionary definitions via WordNet ontological links.

g) Leacock-Chodorow (LCH). LCH measures the semantic relatedness between two WordNet synsets by finding the shortest path using hypernym and synonym relationships. This path is further scaled by maximum depth of WordNet (D), and log likelihood is taken to compute the final similarity measure.

$$sim_{lch}(w_1, w_2) = \frac{sp(w_1, w_2)}{2 * D} \quad (18)$$

h) Wu-Palmer (WUP). WUP measures the semantic relatedness between two synset nodes by scaling their depth with the depth of their LCS. This measurement emphasizes that two nodes closer to each other deeper in the hierarchy (specific terms) are semantically more related than those closer at lower depths (general terms).

$$sim_{wup}(w_1, w_2) = \frac{2 * depth\left(lcs_{w_1,w_2}\right)}{depth_{w_1} + depth_{w_2} + 2 * depth\left(lcs_{w_1,w_2}\right)} \quad (19)$$

Wikipedia Similarity

Recently, there has been a huge surge in exploiting Wikipedia article content, in-article links and document categories to compute semantic relatedness between words. There are a number of Wikipedia-based scoring methods and several implementations available. Palmetto [49] and Wikipedia Miner [50] are used for this dissertation study. Scoring methods based on Wikipedia used in our study are described briefly as follows:

a) ***Wikipedia miner toolkit.*** Milne and Witten utilize the count of links pointing to an article. If the out links from two Wiki articles are compared, some overlaps can be observed. These overlaps indicate the relatedness of the two concepts, and some out links that are unique to each concept which indicate relatedness. The basic approach is to use such sets of common and distinct links to generate features and predict the relatedness between the two concepts.

b) ***Coherence measures using palmetto library.*** The Palmetto library [49] offers a client for the REST interface of their web service, which exposes six interesting methods to calculate the coherences among a set of words: C_A, C_V, C_P, C_UCI, C_NPMI and C_UMass.

C_A retrieves the co-occurrence counts for the words in the word set using a context window of size five. These counts are then used to compute the normalized point-wise mutual information (NMPI) of every word to every other word resulting in vectors for each word. The cosine similarity of all word pairs gives the measure for coherence.

C_V retrieves the co-occurrence counts for the words in the word set using a context window of size 110. These counts are then used to compute the normalized point-wise mutual information (NMPI) of every word to every other word resulting in vectors for each word. The average of the cosine similarity between the vector of each word and the sum of all word vectors give the measure for coherence.

C_P retrieves the co-occurrence counts for the words in the word set using a context window of size 70. For each word, the confirmation to its preceding word is computed using the confirmation measure of Fitelson's coherence. The arithmetic mean of the confirmation measure gives the measure for coherence.

In the C_UCI coherence measure, word co-occurrence counts are derived using a sliding window of size 10. Point-wise mutual information (PMI) is calculated for each pair, and the arithmetic mean of all these PMI values gives the coherence measure. C_NMPI is an enhanced version of C_UCI as it uses NPMI instead of PMI.

The main idea of the C_UMass measure is that the co-occurrence of a topic word would be supported by every preceding word that has a higher order ranking in the topic. Therefore, for each word, the logarithm of its conditional probability using every other word preceding it is computed, and the arithmetic mean of this sum gives the measure of coherence.

UMBC Semantic Similarity Service

UMBC provides a hybrid approach to compute semantic similarity between short noun or verb phrases combining a thesaurus (e.g. WordNet) and statistics from large corpus [51]. Their statistical method is based upon distributional similarity and Latent Se-

mantic Analysis (LSA). They also expose a Web API to query the UMBC Similarity service. Given two words or phrases, it returns a number between 0.0 and 1 representing semantic similarity.

The Proposed Approach

Hence, in this section aims to discover a relatively small number of user concerns that are generic and at the same time clear and coherent compared to that produced by conventional PTMs. This section starts by exploiting LDA to generate a relatively large number of topics, e.g., 200. Then the word mover's distance [44] is used to compute semantic distances between the generated topics and perform an agglomerative clustering to come up with a smaller number of clusters, e.g., 15. In the final step each such cluster is represented with a few central words, i.e., words that are nearest to the cluster centroid. Each such group of words represents a user concern, similar to a topic generated by PTMs. A word2vec model is trained using all the reviews text available in Yelp Challenge Dataset [37] to generate vectors for each term. Each term is represented with a vector of size 200. Trained word2vec model allows us to find the vector similarity between different words.

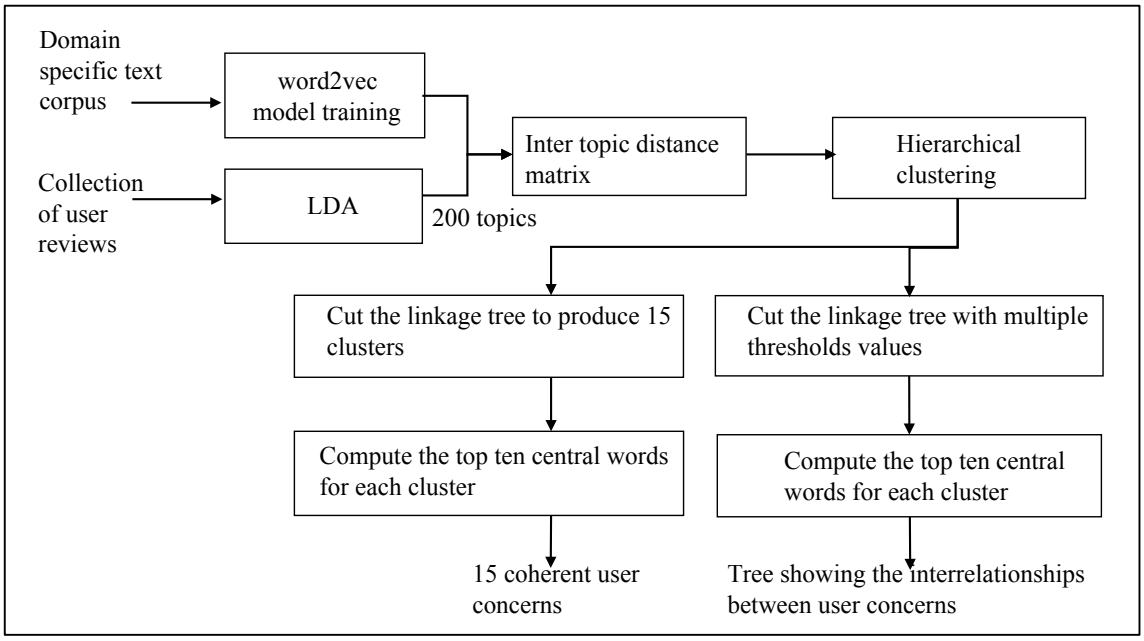

Figure 5. Proposed method to extract coherent user concerns and inter-relationships between User-Concerns.

Furthermore, how a hierarchy of user concerns can be automatically generated by exploiting this approach is also demonstrated in this section. Different sets of topics and topic clusters are generated while performing the agglomerative clustering of the initial 200 topics generated using the standard LDA. Various numbers of topic clusters can be extracted by setting different threshold levels as cutoff values to generate clusters from the linkage tree constructed during agglomerative clustering process. A hierarchy of topics is generated by tracking what clusters merge at these levels. Such a hierarchy is then shown to be able to capture the interrelationship between various user concerns in review texts and facilitate computing semantic distances between the discovered user concerns.

Steps Involved in Extracting Hierarchy of User-Concerns

The proposed method first extracts a relatively large number of topics using a topic modeling technique and then relies on hierarchical clustering to exploit semantic distances between topics, to generate a smaller number of highly coherent and semantically clear topics is proposed in this section. It is expected that such a coherent set of topics represent user concerns hidden in the user reviews with much clarity and high interpretability. The steps involved in our approach are presented in Figure 5 and described briefly as follows.

a) Perform LDA on a collection of user reviews to extract a relatively large number of topics (e.g., 200 topics in this study).

b) Train a word2vec model with a huge text corpus. In this study, the total user reviews available was used, i.e., 1.5 million user reviews containing more than 200 million words.

c) Compute a distance matrix ($D \in R^{200*200}$) representing pairwise distance between topics discovered by LDA, using WMD and the trained word2vec model.

d) Perform agglomerative hierarchical clustering using the computed distance matrix D.

e) Cut the linkage tree formed during agglomerative clustering as shown in Figure 6, at a level where all the 200 topics merge to give exactly 15 clusters. This gives us a representation of a relatively small number of potentially coherent and highly interpretable user concerns.

f) Cut the linkage tree at different threshold (θ) levels to extract different numbers of clusters by choosing values between 0 to 1 for a in the following equation 20. The same linkage tree will give the additional information regarding which clusters merge

when parsed bottom up in the tree. This will give a hierarchy of user concerns, which can capture their interrelationships.

$$\theta = a * max \begin{pmatrix} linkage\ distances\ between \\ merging\ clusters\ pairs \end{pmatrix} \qquad (20)$$

Figure 7 illustrates a linkage tree formed during agglomerative clustering and shows the linkage distances for cluster pairs to merge.

g) The centroid of each non-leaf cluster (cluster of words formed by the merging of two or more clusters) is also computed using their word2vec vectors. The ten words closest to the centroid (representing the most central words) are suggested as the label for each such non-leaf cluster. The closeness to the centroid is computed using cosine similarity between vectors. Hence the user concerns represented by this approach naturally tend to be coherent and closely related.

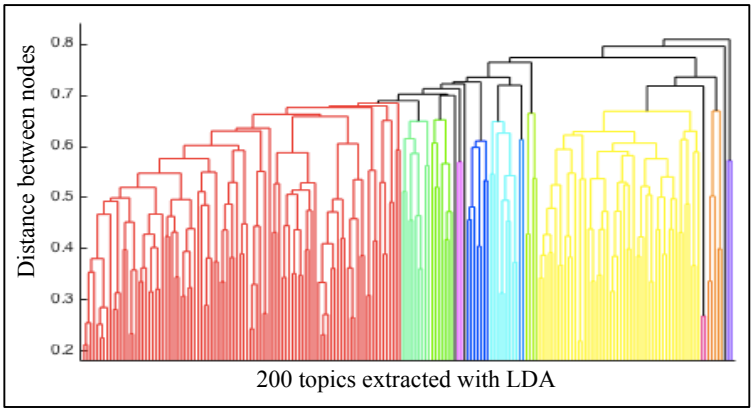

Figure 6. Hierarchical clustering of topics discovered using standard LDA.

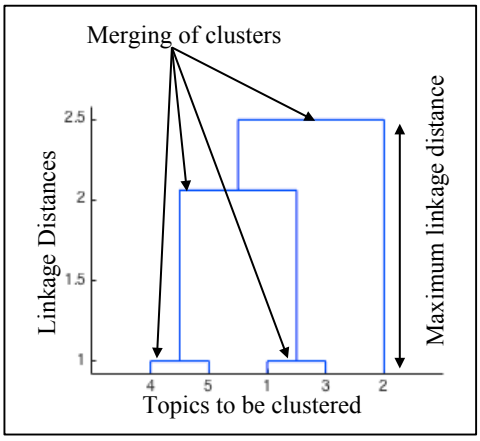

Figure 7. Linkage tree formed during agglomerative clustering.

Experiments and Results

Our approach to extract user concerns is tested with 2014 Yelp challenge dataset. The original dataset is filtered to include reviews for only those restaurants that received at least ten ratings in total and at least one in each year from 2010 to 2014. As such, the final dataset consists of 86K reviews from 1.5K unique users for 14.5K unique restaurants. 200 topics are learned by using standard LDA and followed the steps described in the proposed approach section to extract 15 sets of words to represent user concerns. Standard LDA was also used to produce 15 topics and a four level PAM to generate a total of 15 super topics and 200 sub topics. The 15 super topics and the 200 sub topics from PAM resembled the setting used in our approach that produced 15 user concerns (topics) from 200 LDA topics. Also for PAM, the probability distribution of words across the top 10 sub-topics are added and sorted in decreasing order to obtain the top words for each super-topic. The interest in this section is to observe how the proposed approach performs when applications demand a relatively small number of coherent and human interpretable user concerns from user reviews in comparison to some existing well-

known topic modeling approaches. Stemming is performed to replace any topic word with its root form by using NLTK Lemmatization functionality, which is based on WordNet's built-in morphology. The top 10 words for each topic discovered by LDA, each super topic discovered by PAM and each user concern extracted by our approach, are compared in terms of semantic relatedness or coherence based on metrics described in the background section. These measures are shown to be highly correlated with human judgment for coherence and semantic relatedness. The results for coherence and semantic relatedness using various metrics used in our study are summarized in Tables 3 and 4.

Table 3. Mean relatedness measure for various user concern extraction methods using various similarity measures

Resource	Method	Our Approach	LDA	PAM
WordNet	Vector	0.109	0.097	0.078
	HSO	0.563	0.397	0.306
	RES	1.484	1.188	0.768
	LIN	0.123	0.108	0.074
	JCN	0.063	0.061	0.058
	LESK	53.433	39.114	32.526
	LCH	1.308	1.260	1.302
	WUP	0.406	0.370	0.362
Wikipedia	Wikipedia Miner toolkit	0.577	0.542	0.5263
	C_A	0.218	0.214	0.156
	C_V	0.470	0.417	0.353
	C_P	0.417	0.367	0.141
	C_UCI	0.717	0.830	-0.111
	C_NPMI	0.096	0.081	0.011
	C_UMass	-2.916	-2.314	-2.218
UMBC Semantic Similarity	WordNet +LSA	0.281	0.160	0.098

Table 4. Median relatedness measure for various user concern extraction methods using various similarity measures

Resource	Method	Our Approach	LDA	PAM
WordNet	Vector	0.107	0.084	0.076
	HSO	0.216	0.194	0.25
	RES	1.201	1.024	0.619
	LIN	0.081	0.100	0.063
	JCN	0.060	0.068	0.059
	LESK	39.382	37.036	32.786
	LCH	1.254	1.241	1.345
	WUP	0.381	0.364	0.358
Wikipedia	Wikipedia Miner toolkit	0.579	0.543	0.507
	C_A	0.187	0.176	0.139
	C_V	0.440	0.422	0.338
	C_P	0.415	0.444	0.091
	C_UCI	0.872	0.832	-0.134
	C_NPMI	0.097	0.087	0.005
	C_UMass	-2.589	-2.418	-2.298
UMBC Semantic Similarity	WordNet +LSA	0.257	0.132	0.085

The green highlighted boxes show the highest semantic relatedness values among the three methods compared in our study. In Figure 8, a portion of the hierarchy derived from the agglomerative tree generated during the hierarchical clustering process is presented. As described in the sixth step of our approach, the linkage tree presented in Figure 6 is cut at different threshold levels in order to produce different numbers of clusters. For demonstration purposes, 5 threshold levels are chosen which is computed using equation 20 presented in the sixth step of proposed approach, i.e., *a* varied from 0.6, 0.65, 0.7, 0.75 to 0.8. These thresholds gave 99, 83, 59, 42 and 24 clusters (depicted in red color in Figure 8), respectively, for a total of 200 topics. A trace of which nodes/clusters merge and at what level is computed and the hierarchy is drawn as demonstrated in Figure 8.

Each node represents a cluster of topics, which is formed by merging several topics or topic clusters, and they represent user concerns hidden in the user reviews. The labels for each node are also automatically generated using the approach described in the proposed approach section. Labels generated for each user concern are shown inside each box (see Figure 8), which represent individual user concerns at various levels of granularity. The boxes with bold borders are leaf nodes, which represent fine grained or specific user concerns. They remain unchanged and also persist at lower levels, but are shown only at the highest levels in Figure 8 for clarity purposes. These user concerns merge into more generic user concerns as the hierarchy is traversed bottom-up.

Discussion

In this work, our main goal is to extract a small but coherent set of user concerns, which is clear and highly human interpretable. PTMs are widely used for extracting such hidden information from text, but they are likely to be forced to fit multiple concepts into a small set of topics. Hence, if a very concise set of topics is needed, PTMs are likely to yield noisy and mixed topics. This effect on PTMs can be observed from the few examples provided in Table 5. Coherence measures from Tables 3 and 4 also support our observations. In the first topic from LDA in Table 5, it can be noticed that the user concern seems to be a mixture of concepts such as drinking beer and place/location. With PAM, the topics generated look even less clear as PAM also tries to group sub-topics under each super-topic. For example, in the first topic from PAM, it can be noticed that the user concerns here could be a mixture of several concepts such as food, location, service and friendliness. User concerns extracted using our approach tend to be focused around a

more specific concept. For example, in the first topic extracted by the proposed approach it can easily be noticed that the user concern here is strictly opening days and hours. Similarly, the second user concern is clearly about ice cream or cakes (desserts). One reason the proposed approach is able to generate such a coherent set of concepts to describe the user concern could be because semantic relatedness is also considered in the process of clustering the fine-grained topics. PTMs on the other hand are based on only co-occurrences of words in the given text corpus.

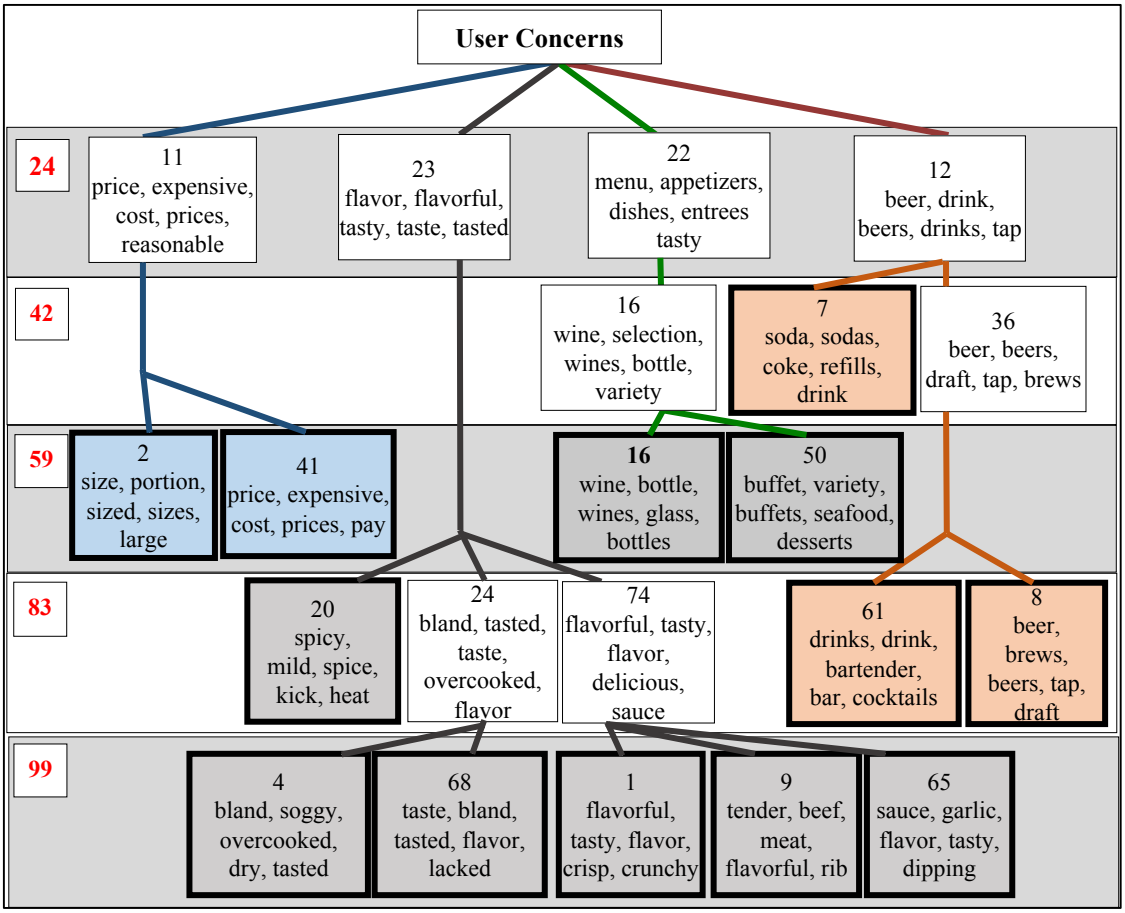

Figure 8. Portion of the hierarchical tree showing the interrelationships between various user concerns discovered by exploiting the linkage tree formed during the agglomerative clustering process.

Tables 3 and 4 clearly demonstrate that most of the relatedness and coherence measures used in our study show that our approach is able to generate more coherent and clear human readable user concerns. Another advantage our approach has is that it can exploit an external knowledge base to compute the relatedness of words and topics. In this study, the external knowledge base is simulated by using all the Yelp reviews available at to train a word2vec model. The trained word2vec was used to compute the distance between words in the clustering process. Any other related text corpus can be used for this purpose. Word2vec models can be trained from a very large dataset in significantly less time. It is difficult to use such external knowledge base with PTMs such as standard LDA or PAM after the models have been computed using only the word co-occurrences in the available documents. Looking at the hierarchical tree in Figure 8, it can easily be noticed how specific user concerns merge into more generic concepts as this tree is traversed bottom-up. For example, in the rightmost user concern 12 at level 24 (beer, drink, beers, drinks and tap), it can be noticed that this user concern depicts a general concept of drinks. Concerns involving sodas quickly branch away at level 42. User-Concerns that are closer to each other such as User-Concerns 61 (drinks, bar, bartender, cocktails) and 8 (beer, brews, tap, draft) only branch out much later in the hierarchy. Similar patterns can be observed with other branches in the hierarchy as well. As such, this hierarchical model can be used to understand what user concerns are closer to each other and what user concerns are more distant from each other. Based on this kind of insight similar people might be discovered, assuming people with similar concerns expressed in their reviews are similar. Similarly, it may also be assumed that restaurants with user reviews that show concerns about similar aspects are similar. Hence, such a hierarchy showing the interrela-

tionships between user concerns could be useful in building various user and item centric models such as CF-based recommendation systems.

Table 5. Examples of the user concerns extracted by Our Approach, Standard LDA and PAM while extracting a relatively small number of user concerns (topics)

Our Approach	*late sunday pm saturday friday monday dinner lunch night afternoon* *chocolate vanilla banana cinnamon whip caramel cream syrup butter maple* *paper plastic container bag dirty togo bathroom box bathroom table* *size portion generous large small portion small huge price* *girl ladies people customer work guys waitress girl owner staff*
Standard LDA	*bar happy night hour beer drink place patio fun* *table order back server minute order time ask wait want* *food good chip tacos salsa mexican bean chicken taco place* *breakfast coffee cream egg menu ice bacon day french chocolate* *burger fry sandwich cheese lunch hot good order sandwich*
PAM	*food place salad location great good bar service wait friendly* *good food time star place night back pretty friend service* *good location food night back place pretty time late* *good great place food pretty nice service back sauce time* *place food location beer great tacos service good friendly taco*

CHAPTER 5

EXTEND THE CONVENTIONAL NMF APPROACH WITH ADDITIONAL KNOWLEDGE FROM USER REVIEWS TO IMPROVE THE RATING PREDICTION

This chapter focuses on extracting UC-vectors from the UC-Tree generated in chapter 4 and incorporate this additional knowledge to extend conventional NMF. User behavioral knowledge based regularization is proposed to extend conventional NMF and several ways of extracting UC-vectors from the UC-Tree is also discussed in this chapter. Finally, results from the experiments that show the effectiveness of the proposed approach are presented.

Literature Review

The literature review for this chapter is presented in two sections. The first section discusses various works that have experimented with using user review text for rating prediction and recommendation tasks. The second section describes works that incorporated some form of hierarchical knowledge into rating prediction or recommendation models.

Rating Prediction and Recommendation Using Review Text

In the quest of discovering user rating behavior and recommending items more precisely, many people have considered incorporating user reviews into the recommendation and rating prediction process [16-20].

Trioshi et al. proposed to improve rating prediction using graph based features such as degree of centrality, page rank, clustering coefficient, and node redundancy [20]. They represented standard user-item ratings matrix and some domain metadata into graphs and used graph-based features to predict business ratings. They extracted several features such as number of reviews by users, average ratings by users, number of reviews for businesses and average ratings of businesses, to construct the graph. Fan et al. proposed to predict star ratings for the business from their online reviews in Yelp using only the review text [36]. Their approach is to create a bag of words from the most frequent words in all text reviews or most frequent words/adjectives from results of Parts-of-Speech analysis, and train machine learning models to predict star ratings.

McAuley et al. proposed to combine latent rating dimensions obtained from latent factor models with latent review topics obtained from topic models such as LDA [17]. They also aim to improve upon the state of the art models based on MF and LDA by optimizing for rating error during MF and topic modeling together. Their model, called Hidden Factors as Topics (HFT), discovers topics that are correlated with the hidden factors of users and items. They defined an objective function whose first part represents the rating error (similar to objective functions in MF/latent-factor recommendation systems) and there is a second component which represents the topic modeling component. Hence, the second component which they referred to as ‘corpus likelihood’ acts as regularizer for

the ratings prediction model. The proposed approach to extend NMF is also similar in a sense that it will also have a similar objective function and an additional regularizer component to regularize the learned ratings prediction models. However, unlike 'corpus likelihood', the approach is to regularize with user and item vectors discovered from the hierarchical tree of User-Concerns. While learning the user and item factors, the additional component in the objective function also checks that similar users/item vectors from User-Concern hierarchy should also have similar user/item factors. Unlike their work, the proposed approach relies on topic modeling methods such as LDA together with hierarchical clustering and semantic distance measures between User-Concerns in our UC-Tree to discover user and item latent factors.

Bauman et al. proposed to use a word-based and an LDA-based methods to extract contextual information from specific reviews which are pre-classified as context-rich reviews to complement traditional recommendation systems with context relevant information [16]. They proposed to classify reviews into 'context-rich' (i.e., specific) and 'context poor' (i.e., generic) reviews. An LDA model is built from only context-rich reviews which are then used to discover topics in all user-generated reviews. Topics are scored in terms ratios between their specific and generic frequencies and sorted in descending order. The topics at the top of the sorted list are expected to be rich in contextual information. The proposed work also uses LDA-based methods to discover latent User-Concerns in user reviews. However, it also employs hierarchical clustering and several semantic distance measures to construct a hierarchy of discovered User-Concerns, which represents the inter-relationships between User-Concerns. The final goal is to exploit these inter-relationships in order to boost the accuracy of rating prediction systems.

Huang et al. also used LDA to discover latent subtopics from Yelp restaurant reviews to provide meaningful insights about what customers care about in order to increase Yelp ratings [18]. They show how the overall rating of a restaurant may be average but the restaurant may have very bad rating in some subtopics and improving in that particular area may increase the overall restaurant rating. While this chapter also investigate, what customers care about, the main goal is to discover similar users based on what the users care about the most.

Shenoy et al. also incorporated reviews data into ratings prediction using LDA [21]. They have created a 50-dimensional feature vector (representing 50 topics) for both user and business using topic modeling. They use reviews for each user/business and perform LDA over it and sum up the probability distribution over all topics. Finally, they normalize these sums and create user/business vectors before incorporating the user-business vector product into their existing recommendation model. The proposed work differs from this work as it aims to incorporate interrelationships between the discovered User-Concerns into existing rating prediction models to improve their accuracy.

Incorporating Hierarchies Into Recommendation and Rating Prediction Models

Several works that incorporate hierarchical information into recommendation models have emerged. Hierarchical structures add additional information to the recommendation process and help improve the accuracy of recommendation predictions. They are especially useful to mitigate problems such as cold start and sparsity. Cold start is a problem faced by CF-based systems when the user is new, and has no previous rating data available to compare the user's ratings behavior with the other users. Sparsity prevails

when there are very few items rated by each user, which makes the process of computing similar users and items ineffective based on the rating behaviors of the users.

Cheekula et al. have integrated Linked Open Data, i.e., incorporating hierarchical knowledge into traditional content based recommendation model to infuse background knowledge [11]. They intend to use hierarchical knowledge derived from crowd sourced knowledge bases. They use the DBpedia category structure extracted from crowd sourced Wikipedia to infuse knowledge in a recommendation system built for Movielens dataset. The mapping between Movielens entities and DBpedia categories is borrowed from another paper [11]. They utilize an adaptation of spreading activation algorithm to assign activation values to categories in the taxonomy/hierarchy from DBpedia. Movies are at the bottom of the hierarchical tree. And for a user with the movies he has rated the activation function spreads the activation value to all the categories (upwards). Then with 'reverse spreading' the activation values are spread from the categories to the unrated movies at the leave nodes. Hence, they get the values to rank the unrated movies. The proposed approach does not use an existing hierarchy from an external knowledge base but rather derive from the same data for which the rating prediction model is trained. The proposed approach also intends to discover the relationships between User-Concerns of users and discover similar users, rather than categorical information of items and discover similar items. Unlike computing activation values for categories, the proposed approach intends to compute semantic distances between categories to measure distances between adjacent categories.

Ostuni et al. presented SPrank (Semantic path-based ranking) which is a hybrid recommendation algorithm that considers implicit feedback effectively incorporating on-

tological knowledge from Linked Open Data (LOD) to compute top-N item recommendations [14]. They extract the ontological knowledge from DBpedia in terms of path-based features and combine this semantic description of items with implicit feedback from users to make top-N recommendations using learning to rank algorithm. Hence, they propose a unified graphical representation of the hybrid model by unifying a bipartite graph representing user-item relationships with the extracted hierarchical tree holding the semantic relations between items. Basically, they want to explore paths in this proposed semantic graph to find other items the user may be interested in, based on his available implicit feedback data. Unlike finding relationships between items using a semantic graph from DBpedia, the proposed approach aim to explore user behaviors and concerns expressed in user reviews to discover user similarities. The proposed approach aims to infuse such semantic knowledge into CF process by extending MF techniques. The UC-Tree proposed in chapter 4 is also constructed specifically using the review texts from the same dataset from which the rating prediction system is constructed. Hence, it is not needed to establish relationships or mappings between the knowledge base in the LOD and the items for which the actual rating prediction model is being built. Similarly, such external knowledge may add extra information to items but user behavioral data would be rather difficult to find from in LOD.

Kanagal et al. proposed a taxonomy-aware latent factor model by combining taxonomies with latent factor model to learn user purchase behavior [12]. As with latent factor models they learn factors for users and items. However, they introduce factors for every interior node in the taxonomy and enforce the taxonomy prior over the learned item factors. The latent factor for each item is dependent upon all of its ancestors up to the

root. The sum of all the factors for the ancestral nodes in the hierarchy gives the latent factor for each item. This addition to the latent factor model incorporates taxonomical knowledge in computing user and item factors. The proposed approach intends to discover and represent user behavioral information with hierarchies and use a similar approach to incorporate such hierarchal knowledge into latent factorization models. This dissertation work also explores further into different ways of incorporating such hierarchical information into rating prediction systems including various vector representations for users using the hierarchical tree.

Menon et al. proposed a response prediction approach using MF techniques by seamlessly combining side-information and hierarchies for pages and ads into the objective function for the MF process [13]. The side-information is hierarchical relationships, e.g., the ads belong to various campaigns and campaigns can be run by various advertisers. Hence, such hierarchy can encode correlations such as two ads shown as part of the same campaign may have similar click through rates (CTRs). Their proposed model is a hybrid model which combines several ideas such as hierarchical regularization, agglomerate fitting, and residual fitting, to incorporate hierarchies into the recommendation model. The proposed work also intends to incorporate similar techniques into the objective function for MF. However, some vector representations for users are to be precomputed using the User-Concern hierarchy and used to add an additional constraint to regularize the objective function. The idea behind this approach is that if the users have similar User-Concerns then their vector representation computed using the User-Concern hierarchy will be similar. This in turn will force the MF of the ratings matrix to construct similar latent vectors for such similar users. Hence, the intention is to combine similarity

from the User-Concern's hierarchy tree with similarity in terms of available ratings data while computing the latent vectors for users and items.

Zhang et al. proposed to automatically discover the taxonomies from online shopping data and jointly learn a taxonomy-based recommendation system [15]. They have shown that their model called 'HF model' generates high-quality and human readable taxonomies and even outperforms latent factor models based on human induced taxonomy. HF Model automatically generates a hierarchical categorization for all items based on their descriptions and their purchase data using nested Chinese Restaurant Process (nCRP). The item descriptions are generated via a language model and the user purchase data is generated by a latent factor model. They propose an inference algorithm to jointly optimize the tree structure and latent factors. They alternate until convergence using two steps: i) sampling a path for each item over the tree and ii) optimizing the latent factors of items and categories while keeping the tree fixed. They also show that the accuracy further increases while incorporating available human induced taxonomy in addition to automatically generated hierarchy. By tuning the hyper parameters, they can control the influence of the human induced taxonomy in the global objective function. Unlike this work, the proposed work intends to focus more on user's behavioral similarities rather than categorizing items into hierarchies. The final goal would be to construct hierarchies of User-Concerns and then discover similar users according to their concerns.

The Proposed Approach

Once the UC-Tree similar to the one presented in Figure 8 is generated, conventional NMF is extended with the additional knowledge that can be extracted from the hi-

erarchy of User-Concerns. This extension is expected to increase the accuracy of rating prediction systems that use MF techniques by incorporating UC-Tree in MF.

The initial goal is to represent all users with some vector representation by using the UC-Tree. Then distance measures such as cosine distance are to be used to compute the vector distances to express the level of similarity between any pair of users. The cosine distance between the user vectors will be smaller if the uses are similar and larger if the users are dissimilar, as cosine distance measures the dissimilarity between two vectors. Our goal is to add this additional information into the MF process. The proposed work aims to infuse this information by adding an additional parameter to the cost function of the IL algorithm for MF discussed in chapter two. This additional parameter will minimize the difference between the latent factors of users (computed in the MF process) if their user vectors (computed from UC-Tree) are similar. Hence, after adding this additional constraint into the MF process, MF is expected to produce more similar latent factors for similar users. The rating predictions are computed by the dot product of user latent factors and item latent factors as presented in chapter two. Thus, two users with similar latent factors will produce very similar ratings for the same item. Thus, it is desirable to have very similar latent factors for users if the users are very similar. This new component which is incorporated into the MF process is shown in the following equation 21. This equation is an extension of the equation 5 presented in chapter 2.

$$\varepsilon_{ij}^2 = \left(r_{ij} - \sum_{k=1}^{K} u_{ik} v_{kj} \right)^2 + \frac{\beta}{2} \sum_{k=1}^{K} (||u_k||^2 + ||v_k||^2) + \lambda 1 \sum_{l=1}^{n} UC(u_i).UC(u_l).||u_i - u_l||^2 \qquad (21)$$

where $\lambda 1$ is a parameter which controls the influence of this additional component in the MF process. $UC(u_i)$ and $UC(u_l)$ are the vectors derived for users u_i and u_l from UC-Tree. n is the total number of users. The extended version of MF using the proposed approach is referred as 'eNMF' in this dissertation.

The new updates for the user and the item latent factors which are computed by taking the partial derivatives of the above equation in terms of u and v, respectively, are:

$$u'_{ik} = u_{ik} + \alpha \frac{\partial \varepsilon_{ij}^2}{\partial u_{ik}} = u_{ik} + 2\left(\alpha \varepsilon_{ij} v_{kj} - \beta u_{ik}\right) + 2.\lambda 1.\sum_{l=1}^{m} (UC(u_i).UC)(u_l)(u_i - u_l) \quad (22)$$

$$v'_{kj} = v_{kj} + \alpha \frac{\partial \varepsilon_{ij}^2}{\partial v_{kj}} = v_{kj} + 2(\alpha \varepsilon_{i\square} u_{ik} - \beta u v_{kj}) \quad (23)$$

Explore Different Ways of Representing Similarities Between Users Using the UC-Tree

A hypothetical UC-Tree is presented in Figure 9. Nodes 1, 2, 3 and 4 represents four user concerns and node 0 is the root node. User-Concerns represented by nodes 3 and 4 merge into User-Concern represented by node 2. Similarly, User-Concerns represented by nodes 1 and 2 merge into User-Concern represented by node 0. The edge distances are also labelled with the distance between the respective nodes. These distances can be computed using Word-Mover's distance between the labels for respective User-Concern pairs. Users that have particular User-Concerns are also presented alongside respective nodes. For example, users U_1 and U_3 have user concerns represented by node 3, and users U_2 and U_3 have user concerns represented by node 4. As node 2 is formed by merging User-Concerns represented by nodes 3 and 4, all users U_1, U_2 and U_3 are shown

to have User-Concern represented by node 2. Now, some approaches to extract user vectors from the UC-Tree are presented as follows:

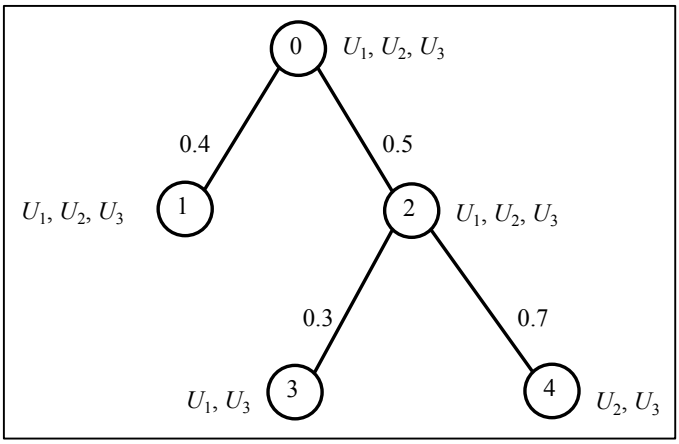

Figure 9. A hypothetical UC-Tree.

Considering Only the Presence of the User in Each Node

Our first representation of user vectors from the UC-Tree consists of a number of vector elements equal to the number of nodes in the UC-Tree. Each vector element represents a separate individual node. The values of the vector element will be '0' if the user does not belong to the particular node; otherwise the value is '1'. For example, Table 6 denotes the presence of each user in each node of the UC-Tree presented in Figure 9.

Table 6. UC-Tree to user vectors considering only the presence of users in each node

	Node 1	Node 2	Node 3	Node 4
U_1	1	1	1	0
U_2	1	1	0	1
U_3	1	1	1	1

Note: Node 0 is not included in the vector representation as it will be common to all the users. The user vectors will thus be represented as:

$UC(U_1) = \{1,1,1,0\}$, $UC(U_2) = \{1,1,0,1\}$ and $UC(U_3) = \{1,1,1,1\}$

Considering Only the Immediate Parent

The second method proposes to represent the user vectors from the UC-Tree considers the distance from the immediate parent, in addition to the first approach. If a user is present in a node, then the value of the user's vector element representing this node will be the edge distance from its immediate parent; otherwise it will be '0'. For example, Table 7 denotes the presence of each user in each node by the distance from the node and its immediate parent.

Table I. UC-Tree to user vectors considering only the immediate parent

	Node 1	Node 2	Node 3	Node 4
U_1	0.4	0.5	0.3	0
U_2	0.4	0.5	0	0.7
U_3	0.4	0.5	0.3	0.7

Note: The user vectors for each user will thus be represented as:

$UC(U_1) = \{0.4, 0.5, 0.3, 0\}$, $UC(U_2) = \{0.4, 0.5, 0, 0.7\}$ and $UC(U_3) = \{0.4, 0.5, 0.3, 0.7\}$

Considering the Entire Path From the Root

Our third approach to represent the user vectors from the UC-Tree considers the entire path from the root to each node. If a user is present in a node, then the value of the vector element representing this node will be the sum of all edges from the root to this node; otherwise it will be '0'. For example, Table 8 denotes the presence of each user in each node by the total edge distance from the root to the respective nodes.

Table 8. UC-Tree to user vectors considering the whole path from the root

	Node 1	Node 2	Node 3	Node 4
U_1	0.4	0.5	0.8	0
U_2	0.4	0.5	0	1.2
U_3	0.4	0.5	0.8	1.2

Note: The user vectors for each user will thus be represented as: $UC(U_1) = \{0.4, 0.5, 0.8, 0\}$, $UC(U_2) = \{0.4, 0.5, 0, 1.2\}$ and $UC(U_3) = \{0.4, 0.5, 0.8, 1.2\}$

The rationale behind these three levels of vector representation is to use the hierarchy of User-Concerns at different levels. The first approach uses the hierarchy at the minimum level. It only considers the presence of users at each node in the hierarchy. Users with only common generic User-Concerns (common nodes nearer to the root node) should be less similar to the users with common fine-grained User-Concerns (common nodes nearer to the leaf nodes). Our first approach cannot distinguish if the closeness is at a finer-grained level or at a more generic level. For example, U_3 has similar User-Concerns with both U_2 and U_1 as depicted by nodes 3 and 4, respectively. It will be tricky to determine who is more similar to U_3 by using only the first approach. This can be observed when the cosine distance between the user vectors pairs ($UC(U_3)$, $UC(U_1)$) *and* ($UC(U_3)$, $UC(U_2)$) are computed by using the vector representations from the first approach. The cosine distance between user U_3 and U_1 will be same as the distance between the users U_3 and U_2. Therefore, both U_1 and U_2 would be equally similar to U_3. Node 3 and node 4 have a common parent (node 2) but node 4 is at higher distance from its parent than node 3. This means that node 4 is more dissimilar to its parent than node 3. As we are heading from generic to more specific User-Concerns when we parse the UC-Tree top to bottom, we infer that U_3 sharing node 4 with U_2 depicts that it shares more specific User-Concern with U_2, than U_1. Our second approach can look up to the immediate parent, and will be able to discover this kind of knowledge. This can be observed when the cosine distance between the user vectors pairs ($UC(U_3)$, $UC(U_1)$) *and* ($UC(U_3)$, $UC(U_2)$)

is taken by using the vector representations from the second approach. Cosine distance between U_3 and U_1 is 0.29 and the cosine distance between U_3 and U_2 is 0.05. Hence the second approach can discover that U_3 is more similar to U_2 which is similar to our inference.

Similarly, the second approach is expanded by considering the whole path distance from the root to the node in our third approach. The rationale behind this is again to incorporate more information from the hierarchy into vector representation than the second approach. When the cosine distance between the user vectors pairs ($UC(U_3)$, $UC(U_1)$) *and* ($UC(U_3)$, $UC(U_2)$) representations are computed from the third approach, we get 0.35 and 0.14, respectively. Hence this approach also discovers that U_3 is more similar to U_2 which is also similar to our inference.

Comparing Different Ways of Representing Similarities Between Users Using the UC-Tree

Table 9. Comparison different techniques of using the UC-Tree for Yelp Dataset

Techniques	RMSE
Considering only the presence of users in each node	1.00431
Considering only the immediate parent	1.00218
Considering the entire path from the root	1.00200

Experiments and Results

Experiments show that considering the entire path from the root is probably the best approach among the three different approaches discussed in previous section. Hence, the entire path from the root is used to represent UC-vectors in all the experiments in rest of the chapter.

Datasets

Our approach is tested with two datasets - one from Yelp Challenge Dataset 2014 and the other one is crawled review dataset from TripAdvisor [37, 52]. The original Yelp dataset is filtered to include reviews for only those users who reviewed and rated at least ten ratings in total. As such, the final Yelp dataset consists of 98,251 reviews and ratings from 1,525 users for 14,531 restaurants. The crawled TripAdvisor dataset consisted of 22,920 reviews and ratings from 6,095 users for 1,761 hotels. Each user has rated at least three hotels. The average ratings in the Yelp and TripAdvisor datasets are 3.65 and 3.99, respectively. User ratings vary from 1 to 5 for each item and there is a review text for each corresponding user rating, in each dataset.

Experiments Setup

Test sets consisting of random 10 percent from the total available reviews for both the Yelp and TripAdvisor datasets are created for testing. The remaining 90 percent of the reviews were used for training purposes. It is also made sure that at least one review from each user always appeared in each training dataset. For both datasets, 200 topics are learned by using the standard LDA from the training sets as it was observed that 200 topics were able to capture topics at a fairly fine- grained level from our previous study [53-54]. Then by following the steps described in chapter four UC-Trees with four levels containing 15, 30, 45 and 60 nodes (User-Concerns) at each level, respectively, was extracted. UC-vectors with 150 elements were generated and finally rating predictions were made using the proposed ‘eNMF’ approach for the user-item pairs in the test datasets [55]. In order to show the effectiveness of the proposed method, the results obtained by

using the proposed 'eNMF' is compared with that of MeanUser, MeanItem, NMF and PMF. MeanUser and MeanItem use the average ratings given by each user and the average rating received by each item to predict the rating. The base Iterative-Learning algorithm presented by Ott with added non-negative constraints for the factorized matrix elements was used as the baseline NMF [1]. PMF as proposed by Salakhutdinov et al. which uses a probabilistic approach using Gaussian assumptions on the known ratings data and factor matrices was also compared with our approach [29]. α and β for both NMF and eNMF were set to 0.002 and 0.02, respectively. $\lambda 1$ for eNMF was set to 0.002 for all the experiments as experiments conducted with a range of values for $\lambda 1$ showed that 0.002 would perform the best.

Table 10. RMSE obtained for various values of $\lambda 1$ with eNMF

$\lambda 1$ for eNMF	RMSE for TripAdvisor Dataset	RMSE for Yelp Dataset
0.2	0.9709	1.0400
0.02	0.9723	1.0296
0.002	0.9671	1.0020
0.0002	1.0469	1.0048

Results

Rating predictions from our proposed eNMF approach were obtained for different *K* latent features during the MF process. For the experiments, *K* was varied from 10, 15, 20, 25 to 30. Rating predictions from the baseline approaches were also obtained with different *K* latent features and compared with eNMF. Figure 10 shows plots of mean absolute error (MAE) and root mean squared error (RMSE) for both TripAdvisor and Yelp datasets with varying *K*. Figure 10 shows that our proposed method almost always performed the best for both datasets.

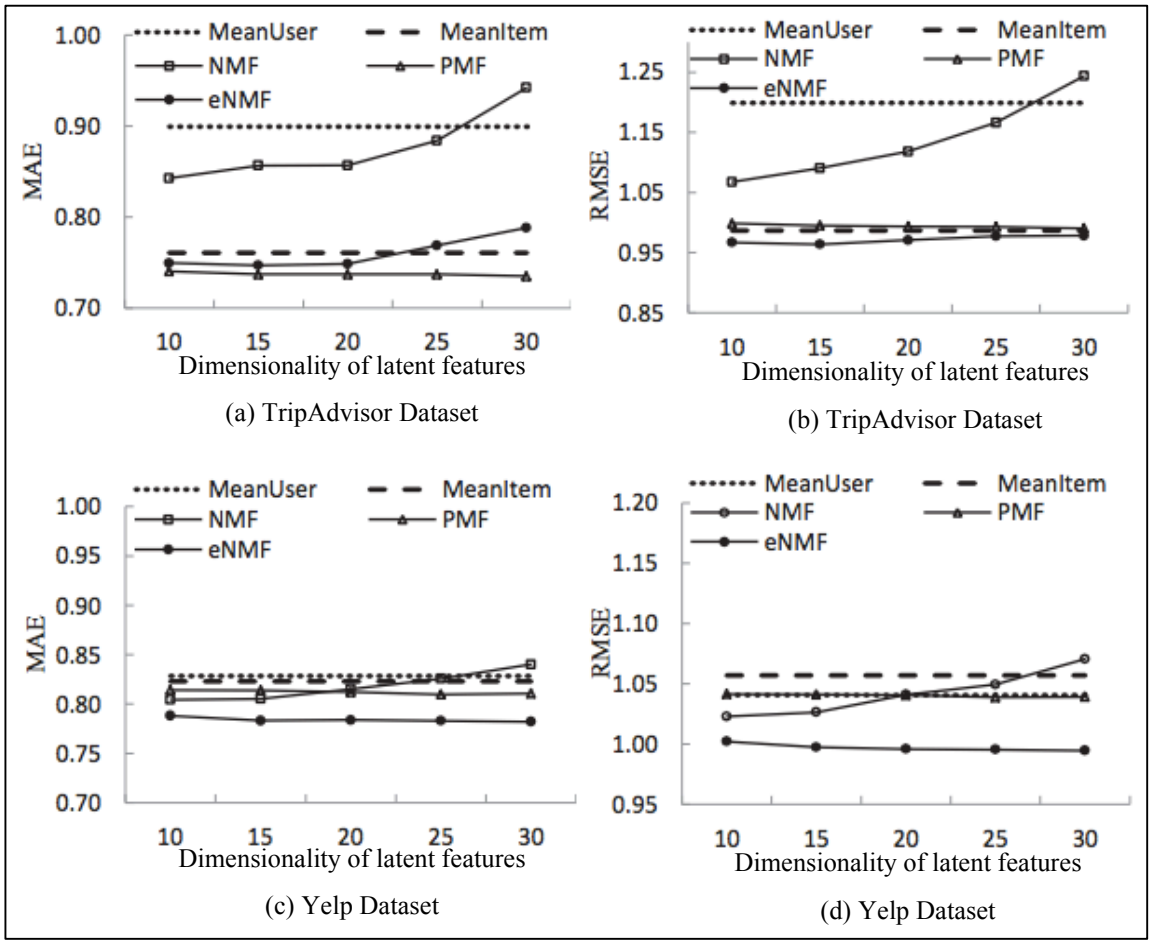

Figure 10. MAE ad RMSE plots for the baseline and eNMF approaches.

The best performance among all the tested *K* values for all the methods are compared in Table 11. For RMSE metric in the TripAdvisor dataset, eNMF performs the best with RMSE 0.96. MeanItem with RMSE 0.98 showed far better performance in comparison to MeanUser with RMSE 1.19. For MAE metric in the TripAdvisor dataset, PMF performed the best with MAE 0.73, however eNMF's performance is also almost comparable with MAE 0.74. MeanItem with MAE 0.76 again showed significantly better performance than MeanUser with MAE 0.89. PMF and NMF showed better performance than MeanUser and MeanItem for both MAE and RMSE metrics in the Yelp dataset, and

eNMF outperformed all the methods with MAE 0.78 and RMSE 0.99.

Table 11. Performance comparison (best performance of all the models)

Dataset	TripAdvisor		Yelp	
Metric	MAE	RMSE	MAE	RMSE
UserMean	0.8993	1.1990	0.8285	1.0405
ItemMean	0.7609	0.9865	0.8230	1.0570
NMF	0.8426	1.0675	0.8044	1.0227
PMF	0.7351	0.9906	0.8098	1.0387
eNMF	0.7470	0.9637	0.7820	0.9945

Discussions

The proposed work shows how user behaviors regarding their concerns can be extracted from their user reviews. This chapter also presents how such User-Concerns can be organized into a hierarchical tree (UC-Tree) based on their semantic closeness and interrelationships. Figure 8 shows a portion of the UC-Tree derived for the Yelp dataset. It can easily be observed that specific User-Concerns merge into more generic concerns as we traverse bottom-up. For example, when the right child from the root, i.e., 'beer, drink and tap' is considered, it can be noticed that this User-Concern represents a general concept of drinks. Concerns involving soda quickly branch away at the next level. User-Concerns that are closer to each other such as the User-Concerns 'drink, bartender, bar, cocktails' and 'beer, brews, tap and draft' only branch out much later in the hierarchy. Similar patterns can be observed with other branches in the tree as well. Hence, this kind of hierarchical model can be used to understand what User-Concerns are closer to or more distant from each other. By conducting experiments with two datasets, one from Yelp Challenge Dataset 2014 and the other crawled review dataset from TripAdvisor [37, 52], it can be seen that the discovery of similar users can be enriched by using rating behaviors along with the user similarity knowledge extracted from such UC-Tree. This

chapter demonstrates the infusion of such User-Concern knowledge representation into conventional NMF based CF techniques by adding an extra component in the loss function of the Iterative Learning based MF as shown in Equation 21. This additional parameter will reduce the difference between the latent factors of users (computed in the MF process) if their UC-vectors (computed from UC-Tree) are similar. The proposed method is compared with several states of the art techniques for CF including MeanUser, MeanItem, NMF and PMF. MAE and RMSE plots from Figure 10 clearly show that our technique performed better than all the other techniques, which validates the effectiveness of infusing such extra User-Concern knowledge into recommendation and rating prediction systems. The addition of the second regularization term into the loss function as shown in Equation 21 will introduce additional computation. In future work, instead of comparing each user's UC-vector to every other user's UC-vector, the plan is to use some heuristics to select only a small portion of users to compare with. Similarly, automatic calibration and estimation of the regularization parameter $\lambda 1$ that determines the relative weight of the additional User-Concern based similarity term on MF could also be implemented.

CHAPTER 6

AUTOENCODER BASED DEEP RATING PREDICTORS

In the recent years, there has been a growing craze over DNNs in the machine learning (ML) community. The success of DNN in winning numerous contests in pattern recognition and ML over other state-of-the-art traditional ML algorithms has boosted its growth and popularity. Deep learning (DL) architectures such as DNN, deep belief networks, and recurrent neural networks have been increasingly applied in various fields including computer vision, natural language processing, speech recognition, recommendation and rating prediction systems, and bioinformatics. Incorporation of DL by huge tech companies like Google and Facebook to effectively leverage their massive amounts of data has also contributed in generating immense interest towards DNN.

DL not only learns task-specific algorithms like classification and clustering but it can also learn data representations that are most effective for learning task-specific problems. It is known to discover and learn features that best represent the problem. For example, in DL based image recognition tasks the inputs are raw pixels and the DNN learns various features and patterns in the images to correctly identity or classify images. There are other kinds of DL approaches such as Autoencoders that can learn latent representations of the input features which can then be used to train the final task-specific prediction models [56-57].

As User-Concerns vectors and ratings behavior vectors are already extracted in the previous chapters, it would be interesting to observe how DL can learn features from our vector sets and apply such learned representations in recommendation and ratings prediction tasks. Hence, DL technique called Autoencoder is explored in this chapter to project the input vectors to some latent space and perform ratings prediction in this chapter.

Autoencoders

It was observed that random initialization of weights in DNN was not a good idea [57]. Hence, autoencoders became popular to pre-train the layers of a DNN and the research in this area got a lot of attention from 2006 to 2011. Shallow to Deep networks can be trained to produce the input itself at the output with Autoencoders. For example, a simple two-layer network shown in Figure 11 can be trained to generate the input vector at the output.

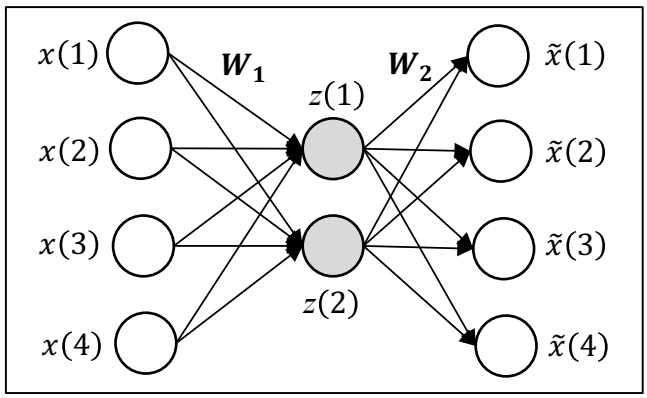

Figure 11. Two layered autoencoder.

If the input $x(i)$ is a vector of length four and if it is used to train the network to generate itself $\tilde{x}(i)$, an objective function can be set up using the sum of squared differences like expressed by equation 24.

$$f(W_1, b_1, W_2, b_2) = \sum_{i=1}^{m} \left(x(i) - \tilde{x}(i)\right)^2$$

$$= \sum_{i=1}^{m} (W_2 z(i) + b_2 - \tilde{x}(i))^2$$

$$= \sum_{i=1}^{m} (W_2(W_1 x(i) + b_1) + b_2 - \tilde{x}(i))^2 \qquad (24)$$

where W_1 and W_2 are weights for the first and the second layer, b_1 and b_2 are biases.

This particular architecture can be called a linear autoencoder if it uses linear activation functions. If a nonlinear activation is desired, the activations can be replaced by a nonlinear function. If the data is highly nonlinear, several hidden layers can also be used and such architecture will be known as deep autoencoder. Pre-training of a DNN using shallow autoencoders can take place by learning initial weights for each layer at a time using unsupervised data. The final layer is trained using the supervised data and the whole network is then fine-tuned using back propagation.

For example, let's suppose there exists a DNN of two hidden layers as demonstrated by Figure 12. W_1 and W_2 are the set of weights to be initialized. Such network can be trained by using two autoencoders A_1 and A_2. First autoencoder A_1 has parameters W_1 and W_1'. After training A_1 to produce the input at its output, the weights W_1 will be used in the network shown in Figure 12 and W_1' will be neglected. The first half of the autoencoder that encodes the input to the latent representation is also called the 'Encoder'. Similarly, the second half of the autoencoder that decodes the latent representation to the out-

put (which is same as input) is also called 'Decoder'. Then the outputs produced at hidden layer of the trained autoencoder A1 (by the encoder) will be the input for the next autoencoder A_2 with layers W_2 and W_2'. It will also be trained to produce the fed input at its output. W_2 will be used as initial weights for the second layer in the given network shown in Figure 12 and W_2' will be neglected. Hence the initial weights to be used in the network shown in Figure 12 are obtained using autoencoders A_1 and A_2.

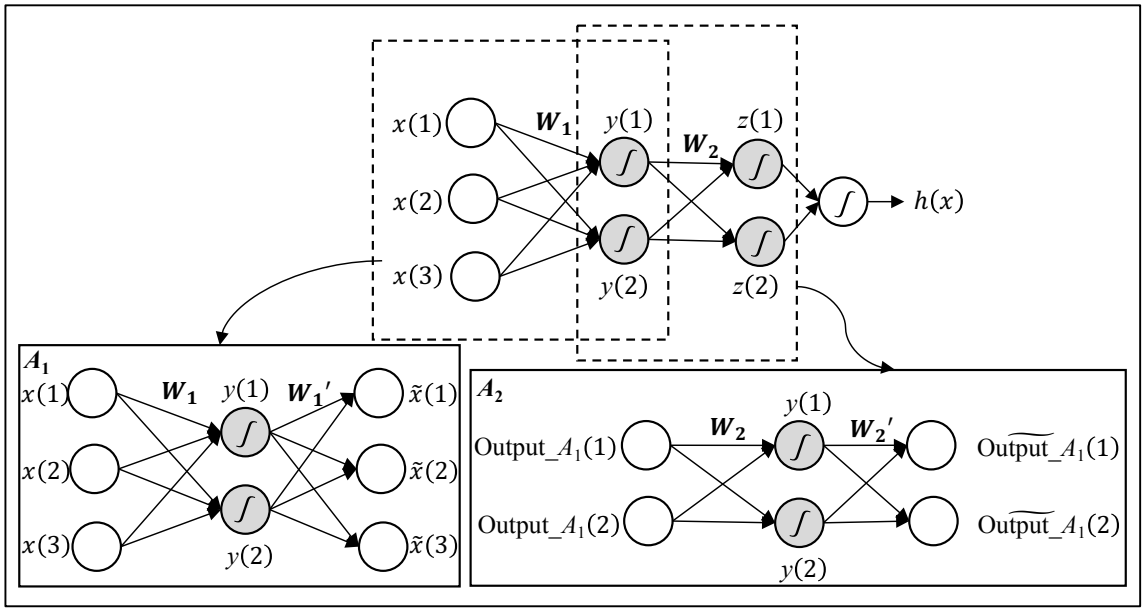

Figure 12. Weights initialization for DNN with two hidden layers.

Autoencoders are not only used for pre-training, but they can also be used for data compression [57]. The outputs produced at the hidden layers in the autoencoder shown in Figure 10 can be viewed as the latent representation of the actual input. As it can be noticed that the input is a four-element vector but the latent representation is only two-element vector.

AutoRec

An example of using autoencoders into CF was presented by Sedhain et al. [58]. They proposed AutoRec, an autoencoder based framework for CF to predict known ratings for items from individual users. The rating matrix $R_{n\times m}$ is a partially observed user-item rating matrix where each user u could be represented by a partially observed row vector $r^{(u)} = (R_{u1}, \ldots R_{un})$ and each item i can be represented by a partially observed column vector $r^{(i)} = (R_{1i}, \ldots R_{mi})$. Now the goal of their work is to train an item-based (or a user-based) autoencoder which can take a partially observed item vector (or user vector), project it into a low-dimensional latent space and then again reconstruct the original item vector $r^{(i)}$ (or user vector $r^{(u)}$).

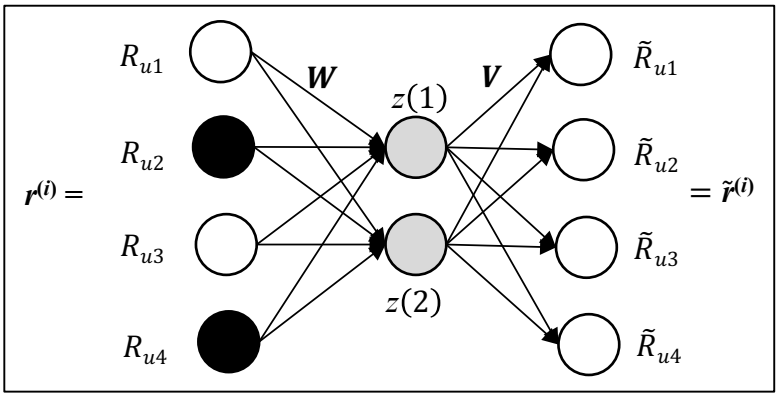

Figure 13. Item-based AutoRec model.

The autoencoder had a single k-dimensional hidden layer and the transformations W and V and the biases μ and b were to be learned. The reconstruction of the input item vector r was represented as

$$h(r;\theta) = f(W.g(Vr + \mu) + b)$$

where $\theta = \{W, V, \mu, b\}$ and $f(.)$ and $g(.)$ are activation functions. The known ratings as depicted by black nodes at the input in Figure 13 and only these nodes are used for

learning the autoencoder. As the reconstructed item vector contains values for all the unobserved ratings, these values can be used to make predictions of unknown ratings from user for this item. Hence, only items vectors are used in item-based autoencoders and only user vectors are used in user-based autoencoders to make rating predictions. The authors have also claimed that AutoRec is compact and efficient for training. Additionally, AutoRec is also shown to outperform state-of-the-art CF techniques like biased MF, RBM-CF and LLORMA on Movielens and Netflix datasets.

Autoencoder Based Rating Predictor

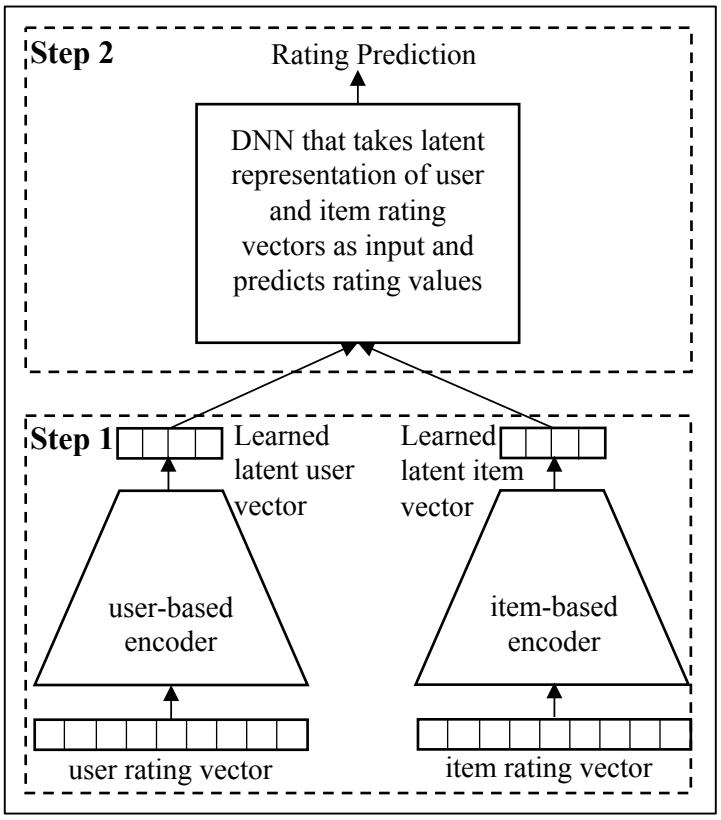

Figure 14. Autoencoder based rating predictor.

Encouraged by success stories similar to that presented by Sedhain et al. [58], it would be interesting to try to learn features from the rating behavior vector for items and

users (from the rating matrix R) using Autoencoders. The autoencoder-based model proposed in this chapter can be described with a two-step process as shown in Figure 14. First, an unsupervised training is performed to train user-based autoencoders and item-based autoencoders to generate latent representations of the user and item rating vectors. Then as a second step a supervised learning is performed to train a DNN to predict rating values for the latent user and item vectors.

Experiments

In this chapter experiments are performed with the same TripAdvisor dataset that was used in the previous chapters and autoencoders are trained for the users and items separately. There are 6095 users and 1761 items in the TripAdvisor dataset. The initial user and item vectors are the rows and columns of the partially observed ratings matrix $R_{n\times m}$, where all the unobserved ratings were set to zeros. For the experiments with autoencoders the 'UNSUP' package for unsupervised learning in Torch is used [59]. A simple linear autoencoder with the encoder layer consisting of a fully connected linear transformation module is trained with input size equal to the length of user or item vector and output size of 40, a 'Tanh' transfer module and a 'Diag' module. The decoder layer simply consists of a linear transformation module with input size of 20 and output size of the original vector size of user or item vectors. 5K iterations are executed for the user vectors and 50K iterations are executed for the item vectors. User-based autoencoders is iterated less number of times as it consumed more time per iteration due to larger dimensionality than the item-based autoencoder. MAE for the training data with the user-based and item-based auto encoders are presented in the following Figure 15.

Then for Step 2, the latent vectors are used for the users and items produced by the trained autoencoders to a DNN for predict their corresponding rating values. The DNN trained has two spatial convolution layers each followed by a RELU transfer function and a spatial max pooling layer, and three fully connected linear transformation layers as shown in Figure 16. SpatialConvolution (*inputPlane* => *outputPlane*, *kW*, *kH*) defines a spatial convolution layer that takes *inputPlane* input layer and produces *outputPlane* output layers and has kernel of width *kW* and height *kH* [64]. Similarly, SpatialMaxPooling (*kW*×*kH*, *dW*, *dH*) defines a MaxPooling layer that operated in *kW*×*kH* region with step size *dW*×*dH* [64].

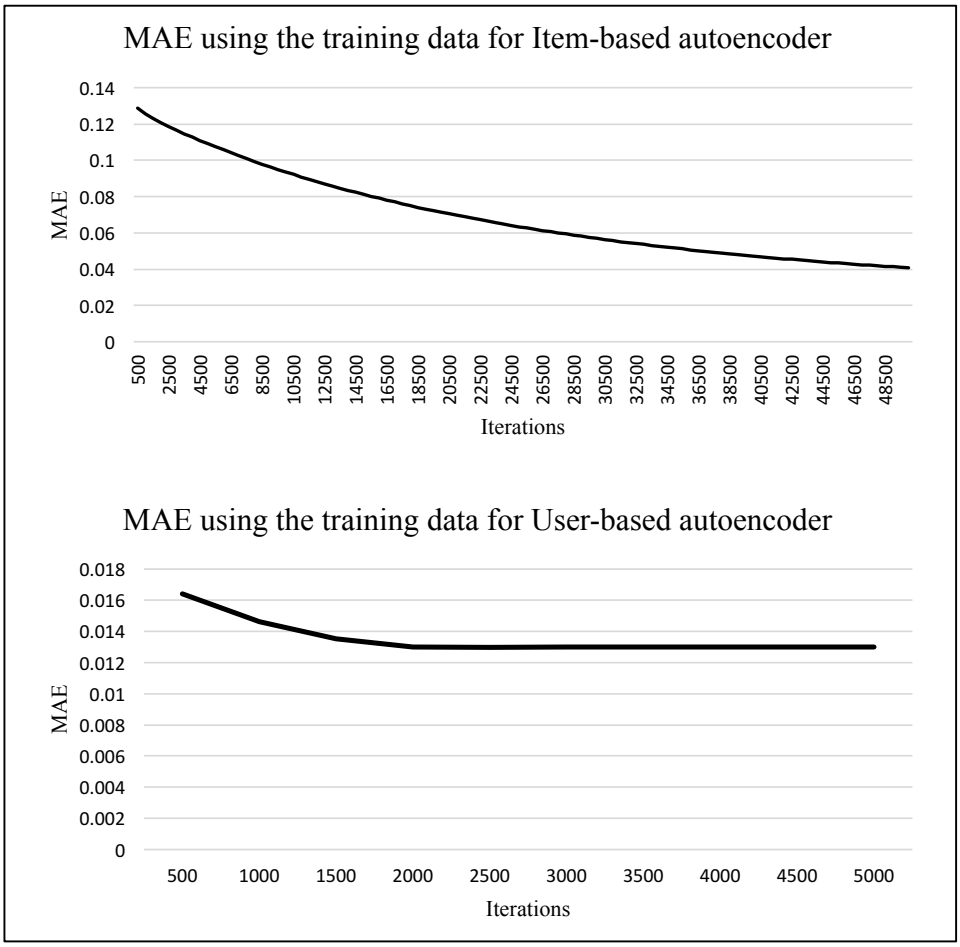

Figure 15. MAE for training of user-based and item-based autoencoder.

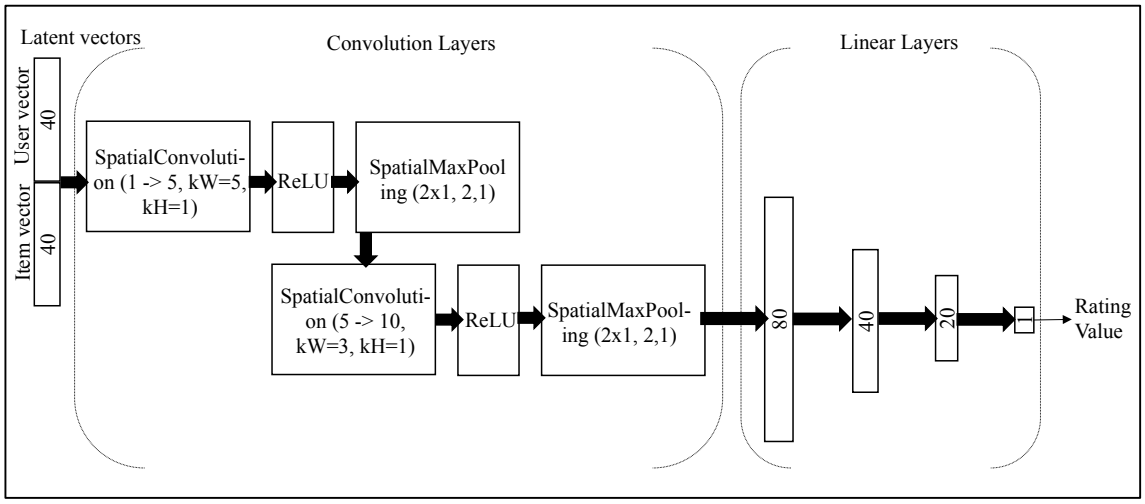

Figure 16. DNN used to train rating prediction by using latent vectors for user and items produced by user-based and item-based autoencoders.

Observations

Our observations as presented in Figure 17 show the prediction error both in terms of MAE and RMSE. The error values do not seem to decrease and were also significantly larger than that achieved by our eNMF approach.

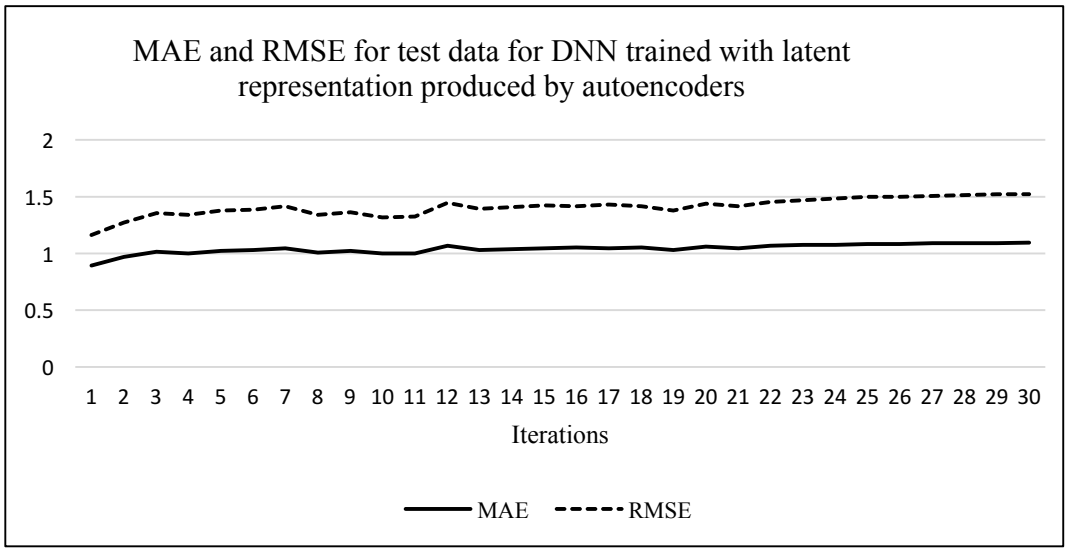

Figure 17. MAE and RMSE for rating predictions made using the DNN trained with latent user and item vectors from autoencoders.

Discussion

Some of the probable reasons why the approach proposed in this chapter has high prediction errors are discussed in this section. The proposed approach does not explicitly avoid using the unobserved ratings to train the network as in AutoRec [58]. The unobserved rating values are assigned as zeroes and hence the autoencoders also predict these zeroes while replicating the input in its output. Another reason might be the depth of the autoencoders. A simple two layered autoencoder is used in the proposed approach. As suggested by Sedhain et al., autoencoders with deeper networks might replicate inputs at the output with higher accuracy [58]. Salakhutdinov and Hinton also used deep autoencoders in their work to extend semantic modeling [60]. Additionally, autoencoders adopts an unsupervised approach to learn the model parameters for reconstruction of the input at the output, rather than optimizing over the actual task like rating prediction in our case. The two steps presented in Figure 14 viz. unsupervised latent representation learning using autoencoders (step 1) and rating predictor using the latent representations (step 2) are trained separately. It might be necessary to train both the autoencoding step and the rating prediction step jointly to further improve the accuracy of the prediction. However, that would be similar to pre-training the network where the network weights are initialized using autoencoders before the final training of the whole network. Hence, the next chapters proceed to train DNN where the latent representation is learned jointly with rating prediction.

CHAPTER 7

DEEP SEMANTIC PROJECTION BASED RATING PREDICTOR (DSPRP)

DNNs have achieved significant success in predicting user sentiments and ratings over various range of items (e.g., restaurants and hotels). Several neural models that project the initial user and item vectors into semantic spaces before predicting how much the user likes the item have recently become very popular [6]. Some additional side information (e.g., similarity in user preferences) regarding the user or item can be very valuable if they can be effectively incorporated into the neural models along with the initial user and item feature vectors. Such features can then jointly train a DNN to generate effective latent representation for the users and items and predict unobserved ratings. Proper incorporation of such user-specific or item-specific side information into a CF based rating prediction model could be very helpful in further boosting up the learning of such networks.

Several approaches such as directly appending the side information to the initial input feature set or treating the side information as a separate modality are popular ways to incorporate them into a DL framework. There might be scenarios where some user-specific side information like personal preferences, interests and concerns can be collected from several user reviews written for different items. Such additional side information may not be directly related to overall item rating values but rather more related to individual user traits and thus treating them as an extra modality or an additional user view to

train rating prediction models might not be very suitable. Hence, this dissertation chapter proposes a novel framework that uses such additional side information about the users to perform regularization while learning the projections of the initial user vectors into semantic spaces. The additional side information targeted in this chapter is user concerns (e.g., roominess and staff friendliness of the restaurant) mined from the user reviews. It is hypothesized that injecting such user-specific side information to regularize the learning of the network layer weights that are directly linked to generating latent user vector representations can effectively improve the accuracy of the model learned. This additional regularizer will force the network to learn similar latent user vectors not only if their initial input vectors are similar, but their side information should also suggest that they are similar. Our framework will then be compared with some existing methods that also try to add additional information about users and items into the DL framework.

Literature Review

Using learned user and item vector projections to predict rating or user sentiment is very popular in DL based approaches [6, 60-63]. Salakhutdinov and Hinton approached semantic modeling with deep autoencoders to handle document search problem [60]. They successfully demonstrated that the semantic structure embedded in query and the document can be captured by using autoencoders. They learned to reconstruct the document and query vectors using unsupervised learning and used the intermediate low dimensional representation learned by the autoencoders as semantic representations. They even showed that such approach can perform better than conventional LSA. However, such DL approach optimized for the reconstruction of the documents rather than the final

task of identifying relevant documents for queries. Hence, their learning can be limited to generating reconstructions at the output and not optimized to detect relevant documents for the search queries.

Huang et al. presents Deep Structured Semantic Models (DSSM) which maps query and document into a common semantic space. The relevance of each document to a given query is then computed as the cosine similarity between the latent vectors in that semantic space. Then using a clickthrough data consisting of list of queries and their clicked documents, a DNN model is trained such that the conditional likelihood of the clicked document is maximized given a query [6]. A typical architecture as proposed by Huang et al. to map raw text features to latent semantic space is shown in Figure 18. The input to this DSSM model is high dimensional term vector and the output from such DSSM is a low-dimensional latent semantic vector. Cosine similarity of such latent vectors would then generate a relevance score at the final node of such neural model. The authors have compared their approach to state-of-the-art latent semantic techniques like LSA, PLSA and DAE that are learned in an unsupervised way, and BLTM-PR and DPM that are learned in a supervised way. Their experiment show that DSSM based architecture can beat these state-of-the-art techniques with a significant margin [6].

This model is further extended by Elkahky et al. to incorporate multiple views for cross-domain user modeling in recommendation systems where common users express their preferences for items from multiple domains like news, apps and movies [61]. Users and items are then mapped to a latent space by maximizing their similarity. The model jointly learns a single latent user representation for user features using item features from different domains. This is shown to improve recommendation across all domains. This

approach is also claimed to generate more compact and semantically richer user latent vector representation. By leveraging more user preference data across multiple domains this approach can also tackle data sparsity problem. A typical architecture used by Elkahky et al. is shown in Figure 19.

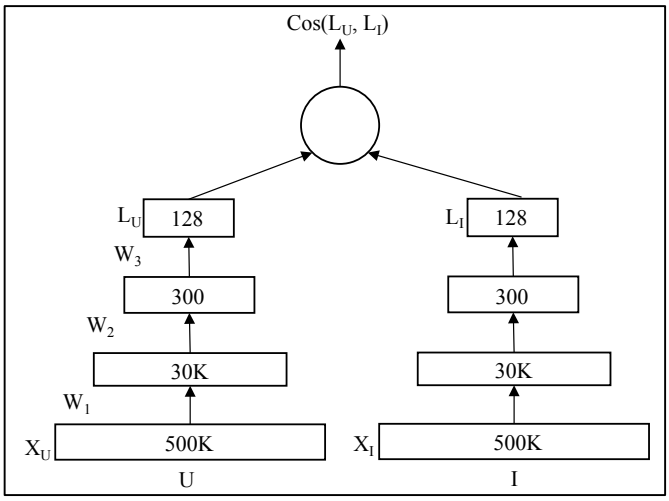

Figure 18. Illustration of DSSM. [6]

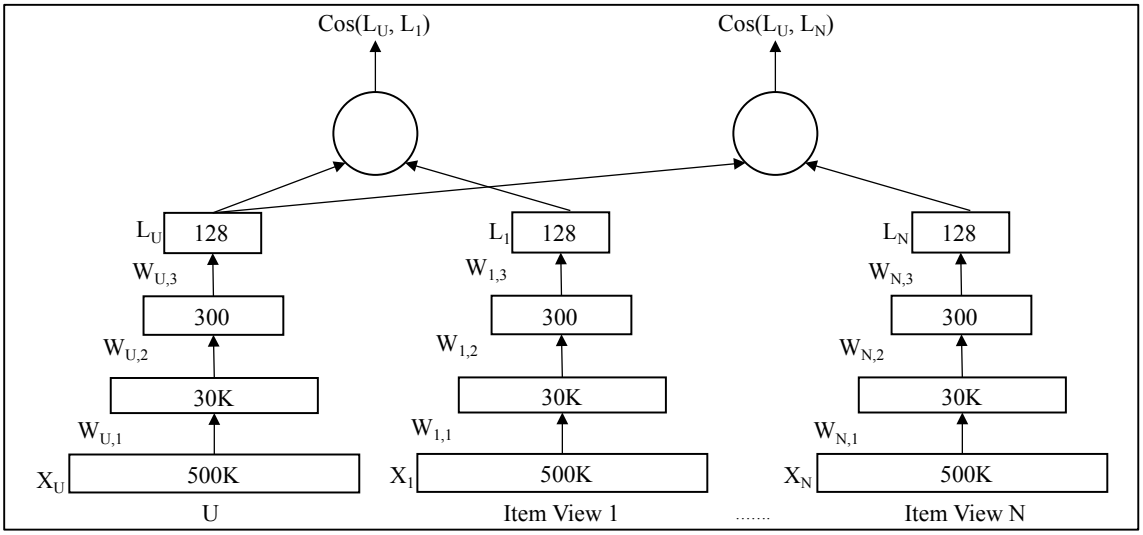

Figure 19. Illustration of Multi-View DSSM. [61]

Tang et al. introduces a user-word composition vector model that incorporates user information [62], in which they represent each word in user reviews as a word vector

and maintain a matrix for each user that can transform the word into a user specific word vector. These representations are further fed into a feed-forward neural network to learn the review rating prediction.

The Proposed Approach

The proposed work will initially use an approach similar to the DSSM model presented by Huang et al. The proposed work will also have two parallel networks to learn the latent user and item representations which will take initial user and item feature vectors as inputs. The user branch will generate the low-dimensional latent user vector representation and the item branch will generate the lo-dimensional latent item vector representations. Finally, a dot product of the two latent representations is taken to predict rating values as shown in Figure 20. The network shall be learned using known rating values made by different users for different items. This approach is referred to as 'DSPRP_1' in this dissertation and used as the first baseline.

One of the approaches we propose to improve on the 'DSPRP_1' architecture is by introducing a regularizer parameter along with the prediction error as represented by equation 25.

$$Error = (RealRating - Prediction) + \delta \sum_{l=0}^{q} (UC_i).(UC_l)|u_i - u_l| \quad (25)$$

This model is referred to as 'DSPRP_2' in this dissertation work and is shown in Figure 21. The additional error value introduced by the added regularization would be propagated back to the network with backpropagation and would only affect the learning of weights that are linked to computing the latent user representations, i.e., weights inside the M hidden layers bounded by a thick solid rectangle on the left in Figure 21. The regu-

larizer would influence the model to learn similar latent user vector representations if the additional side information tells us that they are similar in some nature. UC_i and UC_l are vectors representing such side information, which in our experiments will be UC-vectors. The extraction of such UC-vectors from user reviews have already been explored and successfully incorporated into matrix completion techniques for more accurate rating predictions in chapter four and five [53-55]. Examples of user concerns could include things that users generally want in items such as roominess and cleanness of restaurants/hotels, price, cuisine style, parking, location and so on. U_i and u_l are latent vector representations for the i^{th} and the l^{th} users learned by the trained neural network. δ is the weight associated to control the influence of the additional regularization term.

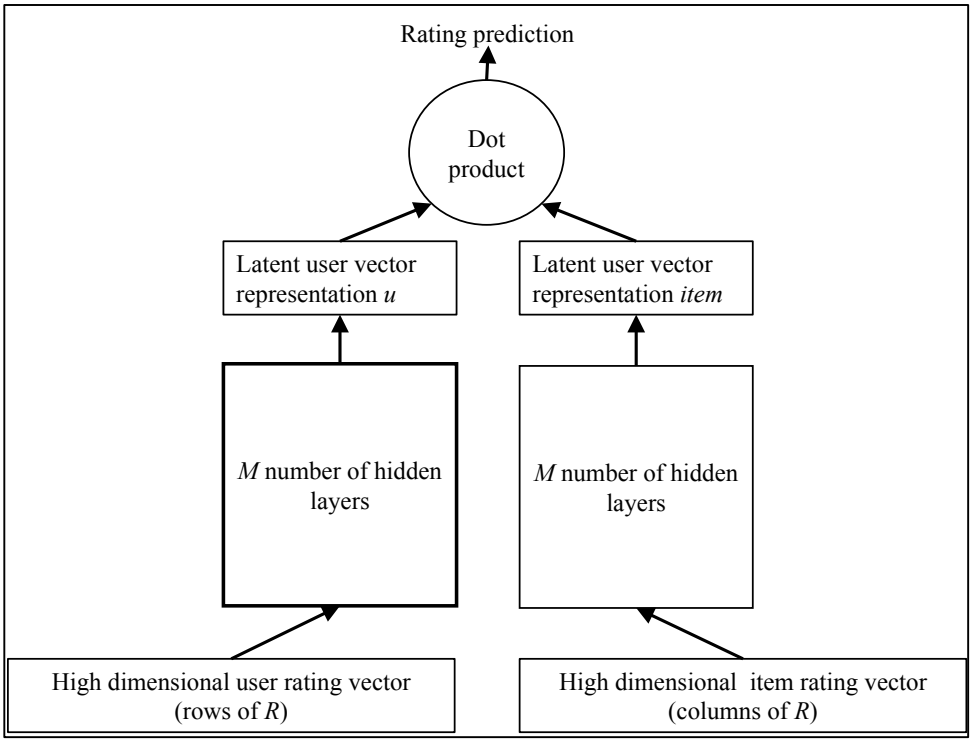

Figure 20. DSPRP_1 approach to predict rating using architecture similar to DSSM model.

The proposed DNN is trained using minibatches of q training instances, i.e., q user and item pairs with known rating values. Each user in each minibatch is compared with D other users in the same minibatch (where $D<<q$) to compute the regularization component. Computing the regularization component by comparing each user to all other users included in the minibatch would be computationally very expensive.

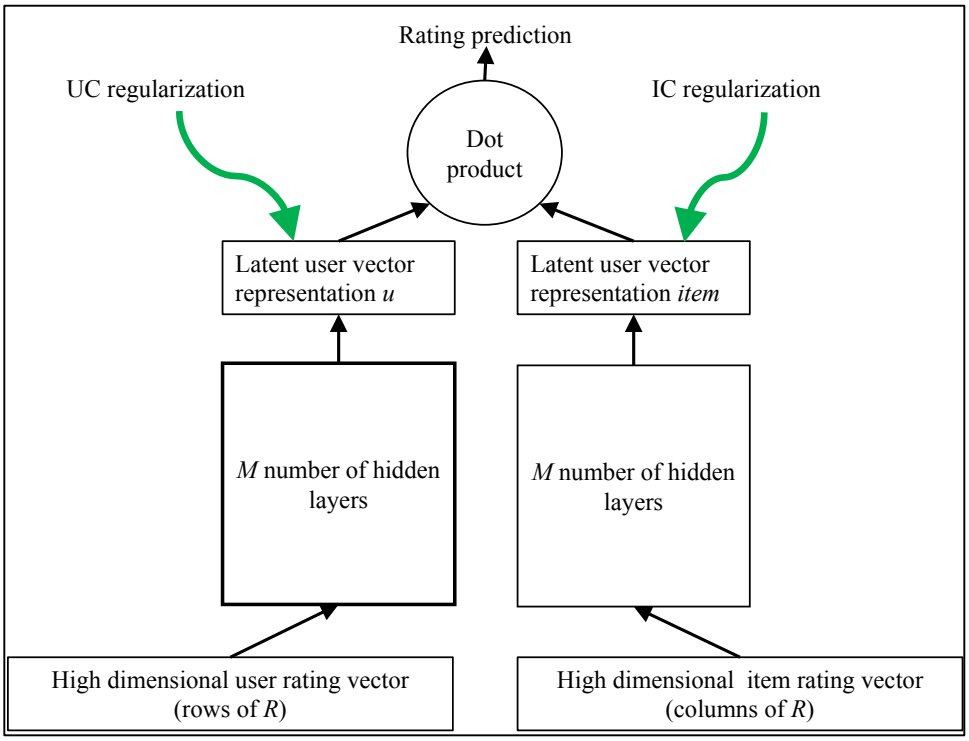

Figure 21. Proposed DSPRP_2 model as an improvement to the baseline approach by introducing regularization using the UC and IC vectors.

Our initial training vectors for user and item pairs will be derived from the rating matrix 'R' that has users in the rows, items in the columns. The known ratings given by users to items are in the corresponding cells. Unknown ratings will be treated as zeros. Hence the user vectors extracted from this rating matrix will be the corresponding row vectors containing all the ratings given by this user to all the items. Similarly, column

vectors will give us the initial item vectors. Additional side information would be User-Concern vectors that will be mined from each user's review texts.

Effectively the network would learn similar vectors for u_i and u_l not only because they have similar rating behavior (denoted by initial high dimensional input user vectors), but also if the dot product of UC_i and UC_l is higher. Hence, in this way additional user specific side information can be incorporated into the DL framework by explicitly specifying that the side information is user specific and not directly mixing it with how the latent item vectors are generated.

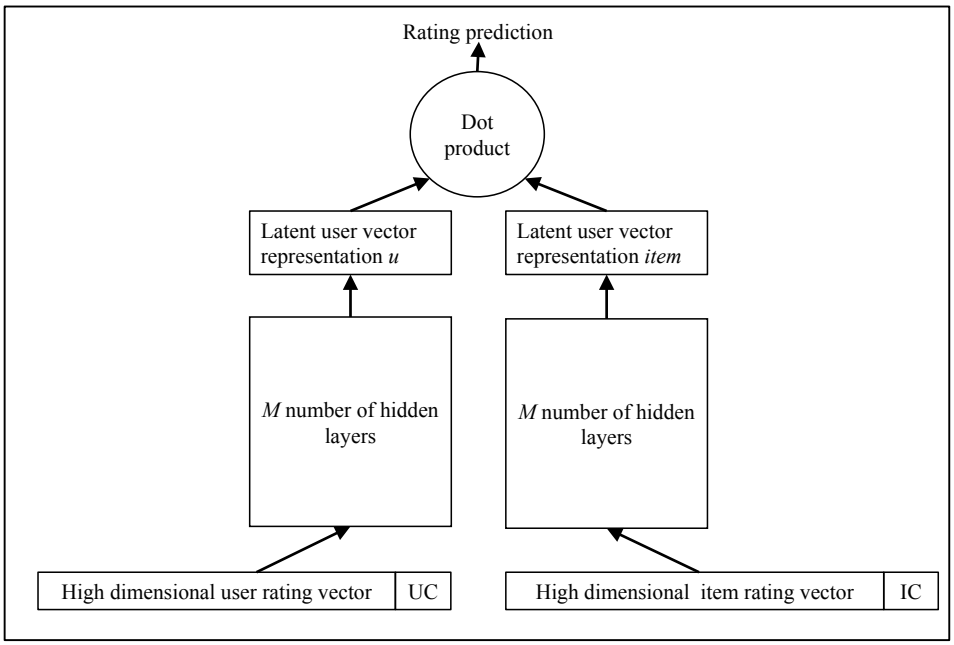

Figure 22. DSPRP_3 approach where the model is learned using the appended rating behavior and User-Concern vectors.

Another approach as shown in Figure 22, where we feed DSPRP_1 with user-ratings vectors appended with UC-vectors as input user vectors and item-ratings vectors appended with IC-vectors as input item vectors is also introduced in this chapter. This is similar to early fusion technique, where features from different modalities are merged

before training a model. This approach will be referred to as ‘DSPRP_3’ in this dissertation work.

Training DSPRP Models

In this section, the framework to train the DNN models is described along with hidden units used in our DNN architecture, the choice of minibatch, gradient computation and stochastic gradient descent (SDG) used for training.

Torch

Torch is used for building, training and testing our DNN models [64]. It is a scientific computing framework which supports large number of machine learning algorithms while making the use of GPUs easy and efficient. The scripting language used for the implementations is LuaJIT with underlying C/CUDA implementation. Torch comes with a large ecosystem of community-driven packages for wide variety of areas such as ML, computer vison, signal processing parallel processing and networking. Most importantly, Torch consist of popular neural network and optimization libraries with very high flexibility to implement complex neural network topologies. Its graph package also supports arbitrary graphs of neural networks with streamlined support to parallelize over CPUs and GPUs. It is also continuously growing and has already been used within several big companies and research labs like Facebook, Google, Twitter and IDAP [64].

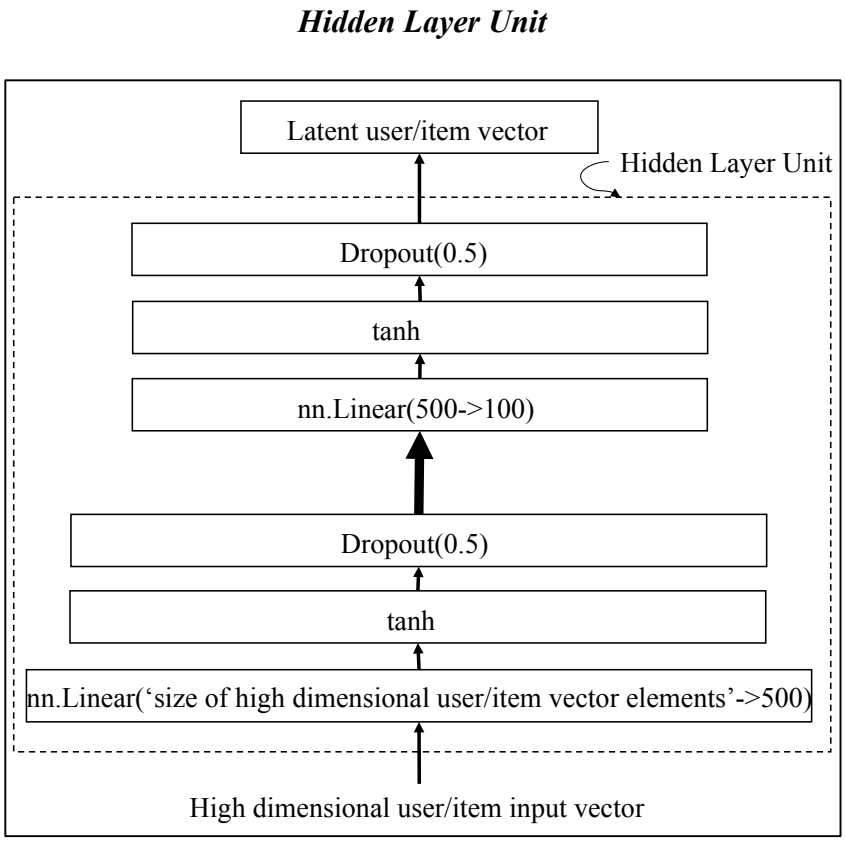

Figure 23. Hidden layer unit used in our experiments for DSPRP_1, DSPRP_2 and DSPRP_3.

A similar structure for the '*M* Hidden Layer Unit' as shown in Figure 23 was used in both user and item branches for all DSPRP models in our experiments. The high-dimensional input vectors for users and items are fed into the hidden layer unit. The hidden layer unit consists of two sets of modules, each comprising of a linear transformation module followed by a 'Tanh' transfer function layer and a 'Dropout(*p*)' layer. The linear transformation module in the first block reduced the vector length to 500 nodes and the linear transformation module in the second block reduced the 500 nodes to 100 nodes. The transfer function layer introduces non-linearity and the 'Dropout(*p*)' layer masks part of the inputs with a probability '*p*' to introduce regularization and prevent co-adaptation

of neurons [65]. The output of the hidden layer unit is the low-dimensional latent vector representation for users and items.

SGD and Minibatches

In addition to SGD, minibatches are used to train our DSPRP models. In general batch gradient descent, the gradient is computed over the entire dataset and the weight parameters are updated. In pure SGD, the weight parameters are updated with single randomly chosen instance of the dataset. Since it is based on one random data point it is very noisy and may go off in a direction far from the batch gradient. Although this randomness is very effective to avoid local minima with SGD, this process is extremely inefficient as it needs to iterate over the entire dataset numerous times to get a good solution. Hence, minibatch approach is a tradeoff between batch gradient descent and SGD. Instead of using one single random data point with SGD, a set of random data points is used and the gradient for each of them is computed. The weight parameters are then updated with their average. This approach injects enough noise to each gradient update so that it is still enough to avoid local minima but at the same time allows lesser number of updates, allowing higher efficiency and relatively faster convergence.

Gradient Computation

Gradient is computed by taking partial derivatives of the error function at the output and propagated to the network weights with backpropagation. The output of the DSPRP models are the q-dimensional rating predictions for a minibatch of size q. The error at the output is computed using MSECriterion in Torch which measures the mean

squared error between the q elements in the model's output (predicted) and the known ratings (ground truth) for the user and item pairs included in the minibatch. The error function is expressed by the following equation 26.

$$error(x, y) = \frac{\sum_{i=1}^{m} |x_i - y_i|^2}{q} \qquad (26)$$

For DSPRP_2 model Selective Regularization (SR) is introduced with an additional regularization term as shown by equation 27. The additional term computes similarity of each user in the minibatch with at most D number of other unique users included in the minibatch of size q and regularizes the latent vector generation by making sure that the latent vectors should be similar not only because the initial input vectors of the users are similar but also because they have similar User-Concerns. Hence, partial derivatives of this additional component are computed and corresponding update values are also back propagated to update the weights of the nodes in the user branch along with the updates computed because of the original error function expressed by equation 26. However, this additional update values due to the regularization component only affects the weights that are used to compute the q latent user vectors chosen for SR.

$$Error = \frac{\sum_{i=1}^{m} |x_i - y_i|^2}{m} + \delta \sum_{l=0}^{q} (UC_i).(UC_l)|u_i - u_l| \qquad (27)$$

The above equation only shows the addition of the regularization term in the user branch. If wish to add regularization in the item branch, a similar expression can be added to the error function.

Experimental Setup

Experiments are run with the same TripAdvisor dataset from the previous chapters. It consists of 22,920 reviews and ratings from 6,095 users for 1,761 hotels. Each user has rated at least three hotels. The average rating is 3.99. User ratings vary from 1 to 5 for each item and there is a review text for each corresponding user rating, in each dataset. These reviews are used to extract UC and IC vectors for each user and item, respectively. Test sets consisting of random 10 percent are created from the total available. The remaining 90 percent of the reviews are used for training purposes. It is also made sure that at least one review from each user at random always appeared in the training dataset.

The learning rate for all DSPRP models are set to 0.001. Dropouts in the hidden layer units are also set to 0.5 for all the experiments. Each minibatch is selected to consist of 200 pairs of users and items from the training dataset. The dimension of the low-dimensional latent vector is set to be 20. For the DSPRP_2 model the size of D is set to be 70. δ which controlled the influence of SR is set after experimenting with values ranging between 0.01 and 0.0001 at intervals decreasing by a decimal place. Experiments are run for DSPRP_1, DSPRP_2 and DSPRP_3 and trained models are saved at intervals of 1000, for testing purposes.

Results and Observations

Experiments run with different values of δ ranging from 0.01 to 0.0001 at intervals of a decimal place suggest that 0.001 would be an appropriate value for δ as shown in Figure 24.

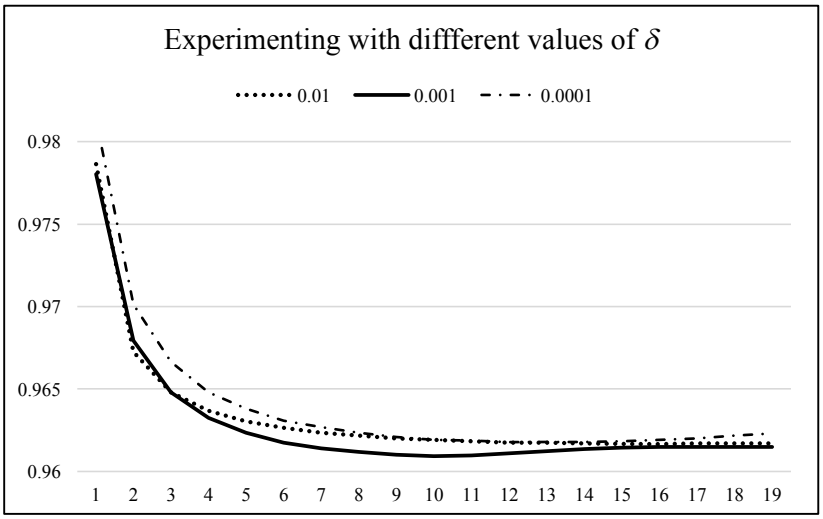

Figure 24. Effect of different values of δ ranging from 0.01 to 0.0001

Experiments are executed with DSPRP_3 by appending only UC vectors to the high-dimensional user input vectors (DSPRP_3_UC), by appending only IC vectors to the high-dimensional item input vectors (DSPRP_3_IC) and by appending both UC and IC vectors to the user and item input vectors, respectively (DSPRP_3_UC_IC). The performance of these models in terms of RMSE and MAE are presented in Figure 25. DSPRP_3_IC seems to perform slightly better than the other models.

Another experiment conducted is to compare the performance of all the approaches discussed in this chapter. DSPRP_1 only uses the high-dimensional input rating vectors for both users and items. SR is introduced with DSPRP_2. Adding regularization terms for both users and items are tested in this chapter. However, the DNN model does not converge properly while adding the regularization terms for items. Hence, SR is only used in the user branch and referred to as DSPRP_2_UC. DSPRP_3_IC is the best performer among different versions of DSPRP_3 from Figure 25, so DSPRP_3_IC is used for comparison. Using SR on the user branch and appending IC vectors to the item input

vectors is also tried as DSPRP_2_UC+DSPRP_3_IC because SR performed better when added in the user branch alone and DSPRP_3 performed better when appending IC vectors to the items vector alone. The expectation is that this would add extra side information for both user and item and would perform better.

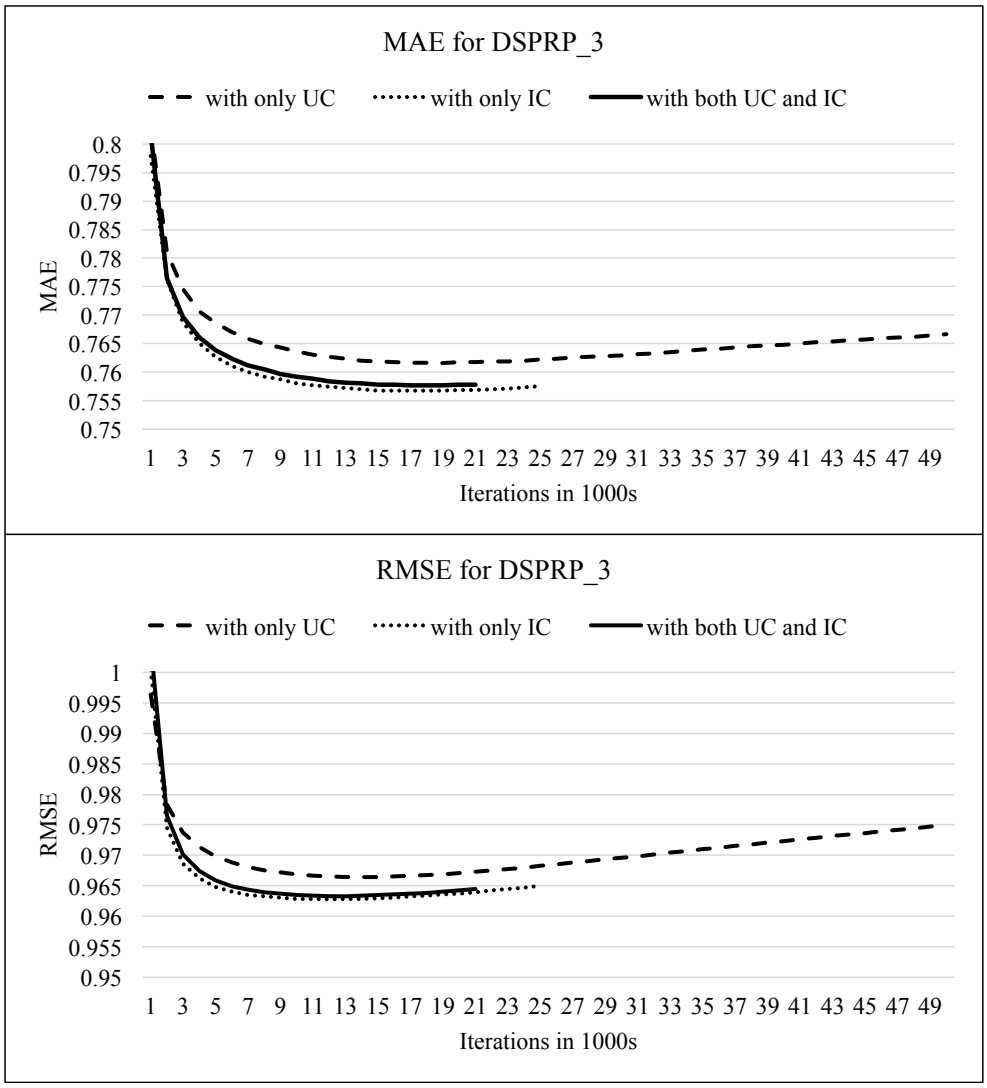

Figure 25. RMSE and MAE plots for DSPRP_3_UC, DSPRP_3_IC and DSPRP_3_UC_IC.

As shown by Figure 26, DSPRP_1 performs the worst as it doesn't use any UC or IC vector. DSPRP_3 performs slightly better than DSPRP_1 as a result of appending ad-

ditional side information. However, SR with DSPRP_2_UC seems to perform better than DSPRP_3_IC. This suggests that SR can perform better than just appending the side information with the high-dimensional input vectors. Finally, DSPRP_2_UC+DSPRP_3_IC performs the best among the four models as expected. The reason might be that this model could incorporate the additional UC and IC information the best.

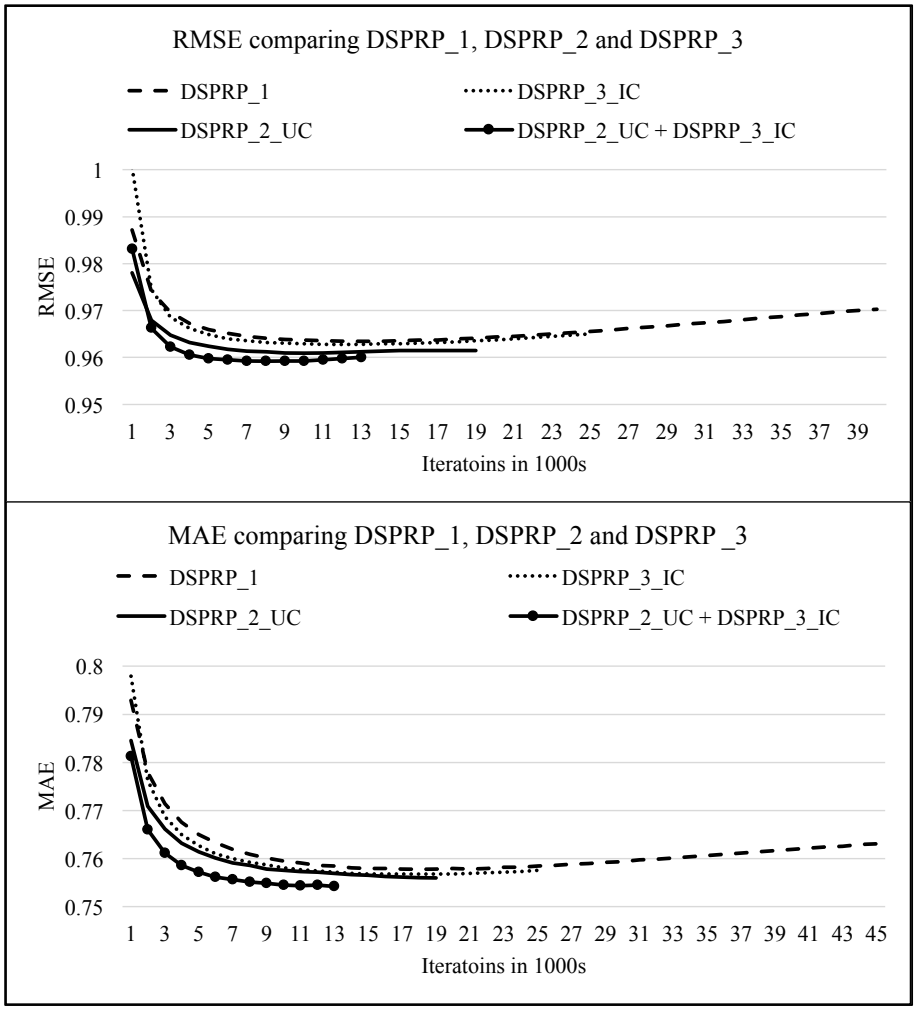

Figure 26. RMSE and MAE plots for DSPRP_1, DSPRP_2_UC, DSPRP_3_IC and DSPRP_2_UC+DSPRP_3_IC.

Discussion

RMSE achieved by DSPRP_1 on the same TripAdvisor dataset is slightly better than the matrix completion counterparts from chapter 5. NMF and eNMF achieved RMSE of 1.0675 and 0.9637, respectively, while DSPRP_1 achieved RMSE of 0.9634. DSPRP_1 doesn't even uses the User-Concerns vectors and performs better that the eNMF approach which enhances NMF approach by incorporating User-Concern vectors. This shows that DNN based models could achieve better accuracies when it comes to predicting user ratings based on their rating behavior alone compared to matrix completion counterparts. Hence, CF-based rating prediction and recommendation systems certainly looks very promising. After incorporating User-Concern information simply by appending it to the initial ratings behavior vectors, i.e., with DSPRP_3_IC approach a RMSE of 0.9627 is achieved. Further incorporating User-Concern vectors using SR as proposed in this section, a RMSE of 0.9609 is achieved. This demonstrates the effectiveness of DNN based approach to fuse multiple modalities and benefit from extra available user-based information.

As presented by Elkahky et al. multiple views can be incorporate for cross-domain user modeling in recommendation system [61]. But it will not be very easy to link each user from different domains. Jointly learning a user representation from multiple domains can be difficult when such linkage may not be easily established. Moreover, users should have explicitly indicated their preferences in each domain. There might be conditions where the additional available information may be more about describing the user rather than explicitly expressing the preference scores. It might be much practical and easy to extract user behavioral information from user reviews in the same domain

and incorporate such additional information into the existing rating prediction or recommendation model similar to our SR approach. It can also be interesting to collect user reviews from multiple domain in cases where same users from different domains cane be linked up successfully and their behavioral component can be extracted. Such approaches might be even more powerful to associate users according to their behavior because it might be able to discover more fundamental user interests and choices.

Incorporating UC-vectors and performing SR on the user branch improved the accuracy of our DNN-based rating prediction. However, performing similar regularization on the item branch using similar IC-vectors couldn't improve the accuracy. There can be multiple reasons behind this. Perhaps the behavioral vectors extracted from the user reviews explicitly represents user interests and concerns and might not work well with items. The reviews were user reviews and users might be more inclined to express their feelings, thoughts and emotions rather than item description. In this dissertation work, only UC-based SR in the user branch is explored.

While performing SR, each user in the minibatch of size q is compared with D other users in the same minibatch. Comparisons can be made with all other q users but it would add more computation and make SR infeasible. As the training data is continuously shuffled, it is very likely that completely different set of users are included in the minibatches in each iteration, each user is compared with only D other users. In the experiments, D was selected to be 70. Furthermore, users might also be pre-clustered based on their UC-vectors and only few other users can be chosen from their corresponding or other closest clusters and included in the same minibatch to reduce the number of computa-

tions. In this case, only very few similar users would be used to impose that the latent vectors should be similar as the users have similar user concerns.

Our results as presented in Figure 26 demonstrates that SR, i.e., DSPRP_2 approach, is more effective than simply appending the additional UC-vectors with the rating behavior vectors. One reason might be that the rating behavior vector is a very sparse but high dimensional vector and a dropout factor of 0.5 is used. Hence, the effectiveness of the dense but relatively low dimensional UC-vector might be minimized when it is appended at the end of the rating behavior vector. SR compares UC-vector of a user with UC-vectors of *D* other users using all the elements in the vector and influences the way latent user vectors are learned.

Conclusion

This work showed how DNN can be used to make unobserved ratings prediction from users to items by using their past rating behavior. Initially an architecture similar to that proposed by Huang et al. to project the initial high dimensional input vectors was used for both users and items into low dimensional latent representation [6] and aggregate over these latent vectors to make the final prediction. This dissertation chapter proposed an enhancement to this approach by using User-Concerns vectors generated in chapter 5 to regulate the latent user vector generation. This approach is referred to as SR because different type of user specific or item specific knowledge can be used for regularization at the user or item branch. This approach is compared with the initial base model and also to another model where the additional UC and IC vectors were appended with the initial input feature vectors. Our experiments suggest that SR can predict more

accurate ratings than the initial model and the feature appended model. Hence, it can be concluded that SR poses a promising direction for further investigation on properly integrating additional user behavioral knowledge into existing user centric DNN models. It can also be understood that infusing additional user behavioral knowledge together with appropriate ways to incorporate them into an existing model is vital to improve the accuracy of an existing rating prediction or recommendation and rating prediction models.

CHAPTER 8

TOPIC PROPORTION - INVERSE ENTITY FREQUENCY (TP-IEF)

Term Frequency - Inverse Document Frequency (TF-IDF) computes weight for each word in a document which increases proportionally to the number of times the word appears in a specific document but is counterbalanced by the number of times it occurs in the collection of documents. TF-IDF is the state-of-the-art for computing relevancy scores between documents. However, it is based on statistical learning alone and doesn't directly capture the conceptual contents of the text or the behavioral aspects of the writer. Hence, this dissertation work shows how relatively low dimensional user behavioral vectors extracted from the same text, from which TF-IDF vectors are extracted, can be used to enrich the performance of TF-IDF. This chapter proposes to extract User-Concerns embedded in user reviews and append them to TF-IDF vectors to train a deep rating prediction model. The experiments executed in this chapter show that adding such conceptual knowledge to TF-IDF vectors can significantly enhance the performance of TF-IDF vectors by only adding very little complexity.

Introduction

TF-IDF is the state-of-the-art when it comes to representing documents with vectors and computing relevancy scores between documents. Some variations of TF-IDF are often used by search engines to score and rank documents for user queries [66]. However,

as indicated by Sun et al., TF-IDF is only based on statistical learning, hence it doesn't directly capture conceptual contents of the texts [7]. Neither does it consider the behavioral aspects of the writer that can be derived from the text. In non-traditional documents, such as web pages of Amazon, Yelp, and TripAdvisor, the item listed is often followed by user reviews. A plethora of behavioral aspects embedded in user reviews cannot be adequately utilized by TF-IDF. Sun et al. also indicated that TF-IDF can suffer when the review length is very short as it would generate a very sparse vector. Therefore, the motivation for this dissertation chapter is to explore extracting user behavioral aspects from user reviews on items and inject such extracted knowledge into the TF-IDF features that are extracted from the same set of user reviews. A deep rating prediction model is trained to show how incorporating such behavioral aspects into TF-IDF can be very effective and improve rating prediction for new unrated items. Such additional behavioral vectors can be much denser and are expected to enrich TF-IDF with conceptual knowledge about various User-Concerns extracted from their reviews. In the previous chapters, topic modeling technique such as Latent Dirichlet Allocation (LDA) are successfully used together with hierarchical clustering to extract user interest and concern vectors from their user reviews [53-54]. It is also shown that such user behavioral vectors can be incorporated into traditional CF-based rating prediction models, particularly NMF [55].

This dissertation chapter proposes a novel approach which assigns weights to topics discovered by LDA from user and item reviews to represent concerns expressed by users. The collection of reviews from a user or for an item represents that user or item in topic modeling. The topics extracted from topic modeling are used as tokens instead of words (terms) and perform TF-IDF like weighting, which is referred to as Topic Propor-

tion - Inverse Entity Frequency (TP-IEF), to generate User-Concern vectors for each user and item. The weights for each topic in TP-IEF increases with such Topic Proportion (TP) for each user or item review, which is similar to the Term Frequency (TF) component of TF-IDF. Likewise, the weights are counterbalanced by the number of times this topic is repeated among users and items, i.e., Inverse Entity Frequency (IEF), which is similar to the Inverse Document Frequency (IDF) component. Subsequently, the vocabulary size and the dimension of User-Concern vector representation would be the number of topics discovered by topic modeling.

The effectiveness of such vectors is then analyzed by using it to learn a deep rating prediction model. This approach is compared with a similar deep rating prediction model learned using only TF-IDF vectors and some state-of-the-art CF-based rating prediction models trained using the rating values collected for the same set of user and item reviews. The effectiveness of TP-IEF is also validated by comparing its performance with that of TP vectors obtained only using LDA to learn similar deep rating prediction model. Our experiments demonstrate the validity of our TP-IEF approach by showing that such low dimensional User-Concern vectors can be as effective as high-dimensional TF-IDF vectors. When used together, it can significantly boost the performance of TF-IDF while adding very little overhead. Experiments are executed with a dataset collected from TripAdvisor which consists of ratings given to hotels by different users along with their corresponding user reviews to demonstrate the effectiveness of our proposed approach.

Literature Review

Plenty of literature have reported the use and effectiveness of TF-IDF for representing documents with vectors and computing relevancy scores between documents. Certain variants of TF-IDF are often used by search engines to score and rank documents for user queries [66].

Huang et al. use TF-IDF as one of their baseline models. They represent both document and queries with TF-IDF term weighting and use cosine similarity between the documents and queries to rank the documents for each query [6]. Qu et al. pointed out that unigram and *n*-gram representations of text, which is common in opinion mining, cannot capture opinions that are expressed with some typical phrases and thus fail to yield robust predictors when predicting a user's numeric rating for a product using text reviews [8]. They proposed to mine opinions within a review which consists of a root word, a set of modifier words in the same sentence, and one or more negation words. They assigned a numeric score to each opinion and finally aggregated such scores to predict the rating value. This dissertation work also tries to overcome similar shortcomings of the TF-IDF approach by injecting concepts like User-Concerns and preferences into learning rating prediction for each user. Our approach uses topic modeling to extract User-Concerns from review texts and proposes a weighting scheme to effectively capture the User-Concerns.

Sun et al. presented an interesting work for classifying short texts into a large taxonomy such as Open Dictionary Project (ODP) or Wikipedia category system [7]. They argued that statistical learning from TF-IDF weighting would suffer from feature sparseness when the lengths of text snippets are very short with only several words. Each cate-

gory having very few documents to train a classifier with TF-IDF vectors could also be a problem. They relied on Explicit Semantic Analysis (ESA) to map fragments of text to a set of Wikipedia concepts. A vector representing association of each document with Wikipedia concepts was generated. An ensemble of SVM classifier trained using both TF-IDF and ESA features was used to predict final categories for short texts. The approach proposed in this dissertation chapter is similar to this approach in that it also relates each user and item to topics derived by topic modeling. However, it mainly aims at associating behavioral information, in addition to the TF-IDF vector, with each user and item and train a DNN to predict the corresponding rating. It is also expected that the proposed method will be helpful when TF-IDF suffers from having very few and short reviews for a user or an item.

Alshari et al. explored rating prediction using user comments with the goal of improving it by leveraging sentiment analysis [9]. They assumed that rating prediction is a classification problem with five classes ranging from 1 to 5. TF-IDF vectors are generated for all the comments and aggregated over all the comments belonging to the same rating class. This would create a surrogate vector for each rating class. For a new comment, its TF-IDF vector was computed and the similarity with each surrogate rating vector is computed. It was then assigned the class of the most relevant surrogate rating vector. However, in the proposed work, users and items are represented with TF-IDF vectors extracted from their reviews and train a DNN to predict their corresponding rating values. Furthermore, this dissertation chapter tries to improve the prediction accuracy by injecting concept vectors extracted using topic modeling into TF-IDF.

Background

Term Frequency-Inverse Document Frequency (TF-IDF)

TF-IDF computes weights for each word in a collection of documents. It evaluates how important a word is to each document in the collection of documents. The weight increases proportionally to the number of times a word appears in the document but is counterbalanced by the number of times it occurs in the collection of documents [66]. It is typically computed for each word *t* in each document *d*, as shown in equation 28.

$$\text{TF} - \text{IDF}\,(t, d) = 1 + \log(tf)\ * log_{10}\left(\frac{N}{df_t}\right) \qquad (28)$$

where tf is the term frequency (number of times word *t* appears in document *d*), *N* is the total number of documents in the collection and df_t is the number of documents in the corpus with term *t* in it (document frequency). Many variations of the TF-IDF scheme are used for tasks like scoring and ranking documents according to the relevance to some search query or relevance matching between documents. Each document or query can be represented as a vector of the TF-IDF weights of different words in the corpus and vector similarity can be computed to find the similarity between the query and the documents.

The Proposed Approach

This dissertation chapter proposes a novel model which considers replacing the words in TF-IDF with topics discovered by topic modelling techniques like LDA as tokens instead of words as in TF-IDF. It evaluates how important a topic is to each user or item in the collection of reviews from various users to various items. The vocabulary size

is the number of topics discovered by topic modeling. TP-IEF weight vectors are then used to represent each review or the user who wrote the review.

Topic modelling is first performed over all the reviews in our dataset to generate *T* number of topics. Topic modelling will generate a topic distribution for each review written by each user. This topic distribution for each review contains *T* TPs which suggest the likelihood of each topic describing the review. Aggregation over all the reviews made by a single user is performed by selecting the maximum TP for each topic to generate a *T*-dimensional vector for the user. This vector is referred to as the TPs related to that user. The weight given to a topic increases proportionally to this TP, similar to the 'Term Frequency' component in TF-IDF. However, this topic might be a popular topic and be present in all the reviews. This would make such a topic less useful in representing the unique view of a user. In other words, the effect of such topic should be counterbalanced by the number of times it occurs among all the users. In order to count the number of times a particular topic occurs among each user, a threshold Ψ is set for the value of TP to determine if a topic belongs to a user. If the TP is larger than Ψ, it is assumed that the topic is indeed discussed in the review written by the user. This counterbalancing term is referred to as Inverse Identity Frequency (IEF), which is similar to 'Inverse Document Frequency' in TF-IDF. The proposed weighting scheme TP-IEF is represented by equation 29.

$$\text{TP} - \text{IEF} \ = 1 + (tp) * log_{10}\left(\frac{total\ users\ or\ items}{1 + users\ or\ items\ with\ this\ topic\ @\Psi}\right) \quad (29)$$

A *T*-dimensional TP-IEF vector is generated for each user. Similarly, by considering reviews written for each item, TP and IEF can be generated for items. Each item is represented by the proposed weighting technique with *T* TP-IEF vectors. Such vectors are

then used to represent users and items, and learn models for rating prediction and recommendation purposes.

Experiments and Results

Experiments were conducted with the same TripAdvisor dataset as in our previous experiments [52-55]. It consists of 22,920 reviews and ratings from 6,095 users for 1,761 hotels (items). The dataset is filtered to include only those users who have rated at least three hotels. The average rating is 3.99. User ratings vary from 1 to 5 for each item and there is a review text for each corresponding user rating. Test sets are created consisting of random 10 percent of the total available user to item ratings and reviews. The remaining 90 percent of the dataset is used for training purposes. It was also made sure that at least one randomly-picked review from each user is in the training dataset.

To compare the proposed approach with CF-based rating prediction models that are learned using the rating behavior of the users on items, rating vectors are extracted. A user-item rating matrix $R_{n\times m}$ is obtained, where n is the number of users and m is the number of items from the rating values for all corresponding reviews in our dataset. Entry R_{ij} represents the rating value between 1 to 5 given by i^{th} user to j^{th} item. The rows and the columns of this matrix represent user rating vectors and item rating vectors, respectively. Such rating vectors are used to learn a NMF and PMF based rating prediction models as in our previous work [55]. extended Non-Negative Matrix Factorization (eNMF) was also presented in the previous chapters as an improvement over NMF by incorporating User-Concerns vectors extracted from user and item reviews. Table 12 compares the best performances of the approach proposed in this dissertation chapter and

other CF-based approaches in terms of root mean squared error (RMSE) and mean absolute error (MAE). All the unknown ratings are represented with zeros.

A DSPRP model is also learned using the rating behavior vectors and indicated as DSPRP_rating in this dissertation work. The input at the user branch is a 1761-dimensional user rating vector and the input at the item branch is a 6095-dimensional item rating vector. *M* was arbitrarily chosen to be two and *p* was chosen to be 0.5 in all the experiments in this chapter.

A similar network is then used to learn rating predictions using only the TF-IDF vectors extracted from the user and item reviews and indicated as DSPRP_TF-IDF. The only modification made over DSPRP_rating is to accept TF-IDF input vectors of length 10K instead of the rating vectors. The top 10K unigrams in the corpus are considered while extracting TF-IDF vectors for each user and item.

Table 12 Comparison of TP-IEF with TF-IDF and some CF-based approaches

Metric	RMSE	MAE
NMF	1.0675	0.8426
PMF	0.9906	0.7351
eNMF	0.9637	0.7470
DSPRP_rating	0.9634	0.7577
DSPRP_TF-IDF	0.9544	0.7462
DSPRP_TP-IEF	0.9539	0.7445
DSPRP_TF-IDF+TP-IEF	0.9497	0.7421

The network was also modified to accept inputs of length 200 for both user and item vectors and learned a rating prediction model with only TP-IEF vectors extracted by setting *T* to 200. This model is indicated as DSPRP_TP-IEF in this dissertation work. In order to find the most suitable value for the threshold Ψ, experiments were conducted with values ranging from 0.02 to 0.06 at the interval of 0.01, and it was observed that

0.04 is the most suitable value. Figure 28 shows the performance of TP-IEF with varying Ψ.

Furthermore, TP-IEF vectors for users and items are appended to their TF-IDF vectors, and a similar DNN model is trained by modifying the DNN to accept 10200-dimensional input vectors in both user and item branches. This model is indicated as DSPRP_TF-IDF+TP-IEF. Performance of TP-IEF are compared with other models discussed above. The results are demonstrated in Figure 27. To validate the effectiveness of the proposed weighting scheme and verify that it is better than using the *T*-dimensional TP values directly obtained from LDA as user and item vectors, pure TPs vectors were also generated to train a similar DNN model. To obtain such vector representations, all the reviews made by a single user are aggregated by selecting the maximum TP for each topic to generate a *T*-dimensional vector for the user. Similarly, all the reviews written for an item are aggregated to generate a *T*-dimensional pure TP vector.

Discussion

This section discusses the results from various experiments performed in the experiments section and analyze the significance of TP-IEF vector representation.

The deep rating prediction model DSPRP_rating trained with rating behavior vectors achieved a minimum RMSE of 0.9634. This performance is better than NMF and other MF based methods in the experiment. However, when the same network is trained with the TF-IDF vectors as DSPRP_TF-IDF, the RMSE was further reduced to 0.9544. This shows that TF-IDF based vector representation can also be very effective in training rating prediction models. Though TF-IDF do not directly represent rating values like the

ratings vectors from ratings matrix *R*, it is able to train the DNN to predict more accurate rating values.

Our experiments also demonstrate that using TP-IEF alone produces very similar results with a much lower dimensionality than using TF-IDF. The RMSE achieved by DSPRP_TP-IEF is 0. 9539 as compared to 0.9544 achieved by DSPRP_TF-IDF. In comparison to the 10K dimensional TF-IDF vectors, TP-IEF used in our experiments only has 200 dimensional vectors. Such low dimensional representation also makes the training process much faster and efficient. TP-IEF is a much denser representation compared to such high-dimensional TF-IDF vectors. Hence our results intuitively suggest that representing User-Concerns and conceptual knowledge from the text with such compact vector representation might be more useful than simply relying on statistical learning like TF-IDF to learn sparse vector representations.

However, the greatest impact of TP-IEF is seen when used in conjunction with TF-IDF as DSPRP_TF-IDF+TP-IEF. When the same network is trained with both TF-IDF and TP-IEF vectors combined, a lowest RMSE of 0.9497 is achieved on the same test set, which is significantly lower than the RMSE achieved by TF-IDF or TP-IEF individually. Therefore, it can be inferred that TP-IEF can add some extra knowledge to TF-IDF when used together. As TF-IDF is based on plain statistical learning, it is not aware of the conceptual knowledge regarding user behavior expressed in the text. TP-IEF is designed to capture such user behavioral aspect by representing User-Concerns with compact vector representation. The vector representation is achieved by introducing a TF-IDF like weighting scheme for topics discovered by topic modeling from user and item reviews. The weighting scheme increases the weights proportionally to the TP and counter-

balances it with the number of times it occurs among all the users or items. Hence, TP-IEF intelligently addresses themes and concepts regarding user interests from their reviews.

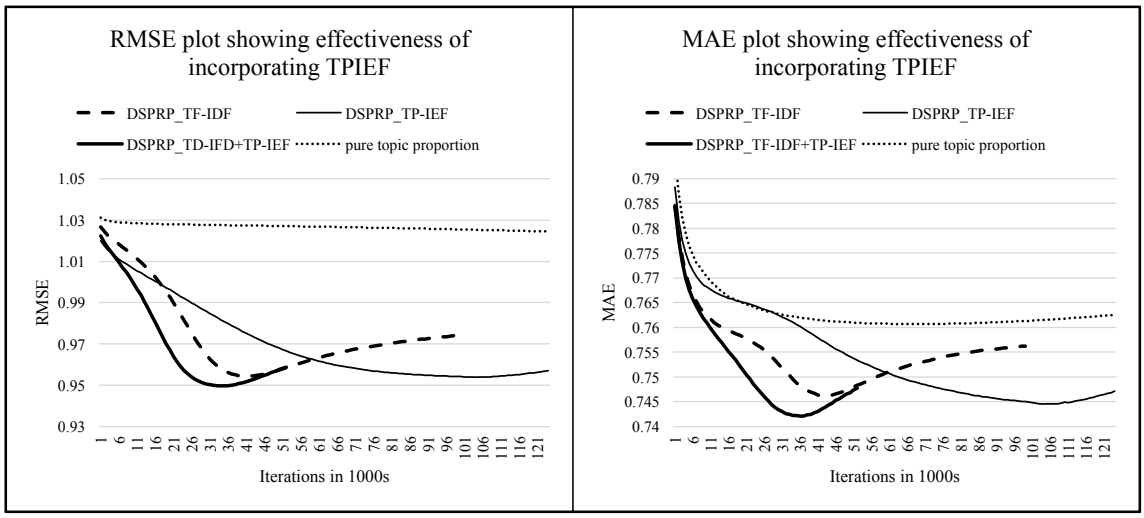

Figure 27. Performance comparison between TP-IEF, TF-IDF, TF-IDF + TP-IEF and TP alone.

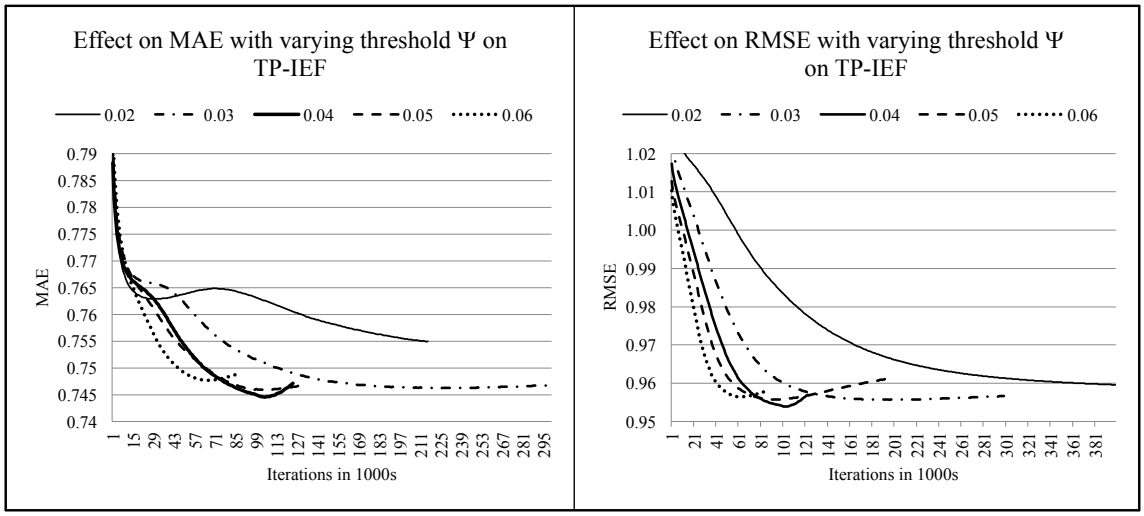

Figure 28. Performance of TP-IEF at various threshold values.

TP-IEF shows drastically improved performance compared to pure TPs vector obtained directly from LDA as shown in Figure 27. Pure TPs vector contains TPs for all topics for a particular set of user or item reviews. This also validates the effectiveness of the proposed weighting scheme used for TP-IEF and demonstrates that it has significant benefits.

Conclusion

In this dissertation chapter, a novel approach which considers topics discovered by topic modeling technique such as LDA from user and item reviews to represent concerns expressed by users is proposed. The main contribution in the chapter is a TF-IDF like weighting scheme called TP-IEF which generates User-Concern vectors from topics generated by LDA. It evaluates how important a topic is to each user or item in the collection of reviews from various users to various items. Topics are used as tokens instead of words (terms) to compute weights for each topic for each user and item reviews. The weight for each topic increases with the magnitude of TP for each user or item review but is counterbalanced by the number of times this topic is repeated among users and items. A deep rating perdition model is trained to predict user ratings using TP-IEF vector representation. Our experiments show that such vector representation can outperform CF-based rating predictions models like NMF, PMF and eNMF built using ratings behavior vectors when used to train models to predict rating values. It is also shown that such low-dimensional behavioral vector representation can perform slightly better than or as good as high-dimensional TF-IDF vectors when used for learning rating prediction models. Most importantly, they can add additional information to TF-IDF vectors when used to-

gether to learn such deep rating prediction models. The improvement introduced by TP-IEF in learning rating prediction by injecting concepts such as User-Concerns and preferences into base statistical learning from TF-IDF is clearly visible from our results.

FUTURE WORK

This dissertation has incorporated user associations and user behavioral knowledge into various techniques like neighborhood discovery, MF and DNN to enhance the accuracy of rating predictions done using such techniques. However, this study has shown multiple promising directions to pursue in order to improve the incorporation of user associations and behavioral knowledge into rating prediction systems. Handling dynamically changing User-Concerns and interests which is expressed in user reviews into UC-Tree can be a very interesting area. This will keep the UC-Tree up-to-date with the current user interests. Incorporating selective regularization using item concerns in the item branch of the DSPRP architecture can be another area to explore. Multi-view item and user concerns vectors can also be incorporated with DSPRP architecture to incorporate user and item specific behavioral knowledge from multiple areas. This dissertation has demonstrated that incorporating user behavioral into predictive systems is a very interesting area and opens up a lot of opportunities.

LIST OF REFERENCES

[1] P. Ott, "Incremental matrix factorization for collaborative filtering", *Science, Technology and Design*, Anhalt University of Applied Sciences, January, 2008.

[2] M. H. Aghdam, M. Analoui, and P. Kabiri, "A Novel Non-Negative Matrix Factorization Method for Recommender Systems," *Applied Mathematics & Information Sciences,* vol. 9, pp. 2721-2732, 2015.

[3] Q. Mei, X. Shen and C. Zhai, "Automatic labeling of multinomial topic models", *in Proc. of the 13th ACM SIGKDD International Conference on KDD*, pp. 490-499, San Jose, CA, USA, August 13-17, 2007.

[4] M. H. Aghdam, M. Analoui and P. Kabiri, "Application of nonnegative matrix-factorization in recommender systems", *in Proc. of the 16th International Symposium Telecommunications (IST)*, pp. 873-876, Tehran, Iran, November 6-8, 2012.

[5] Q. V. Le. (2015). A tutorial on deep learning part 2: Autoencoders, convolutional neural networks and recurrent neural networks. [Online]. Available: https://cs.stanford.edu/~quocle/tutorial2.pdf {Autoencoders, A tutorial on Deep Learning}

[6] P. S. Huang, X.D. He, J.F. Gao, A. Acero, and L.P. Heck, “Learning deep structured semantic models for web search using clickthrough data,” in *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management*. ACM, pp. 2333-2338, 2013. {DSSM}

[7] X. Sun, H. Wang, and Y. Yu, "Towards effective short text deep classification,” in *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*. ACM, pp. 1143-1144, July 2011.

[8] L. Qu, G. Ifrim, and G. Weikum, “The bag-of-opinions method for review rating prediction from sparse text patterns,” in *Proceedings of the 23rd International Conference on Computational Linguistics (COLING)*, pp. 913-921, 2010.

[9] E. M. Alshari, A. Azman, and N. Mustapha., "Prediction of rating from comments based on information retrieval and sentiment analysis,” *Information Retrieval and Knowledge Management (CAMP)*, pp. 32-36, 2016.

[10] X. He, M.-Y. Kan, P. Xie, and X. Chen, "Comment-based multi-view clustering of web 2.0 items," *in Proc. of the 23rd International Conference on World Wide Web*, pp. 771-782, Seoul, Republic of Korea, April 07-11, 2014.

[11] S.K. Cheekula, P. Kapanipathi, D. Doran, P. Jain, and A.P. Sheth, "Entity Recommendations Using Hierarchical Knowledge Bases", *in Proc. Workshop on Knowledge Discovery and Data Mining Meets LinkedOpen Data co-located with 12th Extended Semantic Web Conference (ESWC)*, vol. 1365, Portoroz, Slovenia, May 31, 2015.

[12] B. Kanagal, A. Ahmed, S. Pandey, V. Josifovski, J. Yuan and L. G. Pueyo, "Supercharging Recommender Systems using Taxonomies for Learning User Purchase Behavior", *in Proc. of the VLDB Endowment (PVLDB)*, vol. 5, no. 10, pp. 956-967, Istanbul, Turkey, August 27-31, 2012.

[13] A. K. Menon, K.-P. Chitrapura, S. Garg, D. Agarwal and N. Kota, "Response prediction using collaborative filtering with hierarchies and side-information", *in Proc. of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 141-149, San Diego, CA, USA, August 21-24, 2011.

[14] V. Ostuni, T. Di Noia, E. Di Sciascio, and R. Mirizzi, "Top-n recommendations from implicit feedback leveraging linked open data", *in Proc. of the 7th ACM Conference on Recommender Systems (RecSys)*, pp. 85-92, Hong Kong, China, October 12-16, 2013.

[15] Y. Zhang, A. Ahmed, V. Josifovski, and A. Smola, "Taxonomy discovery for personalized recommendation", *in Proc. of the 7th ACM International Conference on Web Search and Data Mining (WSDM)*, pp. 243-252, New York, USA, February 24-28, 2014.

[16] K. Bauman and A. Tuzhilin, "Discovering Contextual Information from User Reviews for Recommendation Purposes", *in Proc. Workshop on New Trends in Content-based Recommender Systems (CBRecSys'14)*, pp. 2–9, Sillicon Valley, CA, USA, October 6, 2014.

[17] M. Fan and M. Khademi, "Predicting a business star in Yelp from its reviews text alone," *arXiv preprint arXiv:1401.0864,* 2014.

[18] J. Huang, S. Rogers, and E. Joo, "Improving restaurants by extracting subtopics from yelp reviews", *iConference*, March 2014 (Social Media Expo), Berlin, Germany, March 4-7, 2014.

[19] J. McAuley and J. Leskovec, "Hidden factors and hidden topics: understanding rating dimensions with review text", *in the 7th ACM Conference on Recommender Systems (RecSys)*, pp. 165-172, Hong Kong, China, October 12-16, 2013.

[20] A. Tiroshi, S. Berkovsky, M. A. Kaafar, D. Vallet, T. Chen and T. Kuflik, "Improving business rating predictions using graph based features", *in Proc. of 19th International Conference Intelligent User Interfaces*, pp. 17-26, Haifa, Israel, February 24-27, 2014.

[21] S. Shenoy and A. Aras, "From Reviews to Ratings: A model to predict ratings based on review text", [Online]. Available: http://cseweb.ucsd.edu/~jmcauley/cse255/reports/wi15/Sanjeev_Shenoy_Anwaya_Aras.pdf.

[22] M. Sharma and S. Mann, "A survey of recommender systems: approaches and limitations," *International Journal of Innovations in Engineering and Technology,* vol. 2, no. 2, pp. 8-14, 2013.

[23] J. Riedl, and J. Konstan, "Movielens dataset", [Online] 2004. Available: http://www.cs.umn.edu/Research/GroupLens.

[24] M. O'Connor and J. Herlocker, "Clustering items for collaborative filtering", *in Proc. of the ACM SIGIR Workshop on Recommender Systems*, Berkeley, CA, vol. 128, August, 1999.

[25] M. C. Pham, Y. Cao, R. Klamma, and M. Jarke, "A Clustering Approach for Collaborative Filtering Recommendation Using Social Network Analysis", *Journal of Universal Computer Science,* vol. 17, pp. 583-604, 2011.

[26] L. Pradhan, C. Zhang, and P. Chitrakar, "Multi-View Clustering in Collaborative Filtering based Rating Prediction', *in 2016 IEEE Tenth International Conference on Semantic Computing (ICSC)*, pp. 250-253, Laguna Hills, California, USA, February 3-5, 2016.

[27] M. W. Berry, "Large-scale sparse singular value computations ", *The International Journal of Supercomputer Applications*, vol. 6, no. 1, pp. 13-49, 1992.

[28] M. Kurucz, A.A. Benczur and K. Csalogany, "Methods for Large Scale SVD with Missing Values", *in Proc. of the 13th ACM SIGKDD Conf. KDD Cup and Workshop,* pp. 31-38, San Jose, California, USA, August 12-15, 2007.

[29] R. Salakhutdinov and A. Mnih, "Probabilistic matrix factorization," *in NIPS '07: Advances in Neural Information Processing Systems*, vol. 20, pp. 1-8, Vancouver, Canada November 6, 2007.

[30] V. Sindhwani, S.S. Bucak, J. Hu, and A. Mojsilovic, "A Family of Nonnegative Matrix Factorizations for One-Class Collaborative Filtering Problems", *in RecSys '09: Recommender based Industrial Applications Workshop*, New York, USA, October 22-25, 2009.

[31] Recommendation Systems, [Online] 2016. Available: http://infolab.stanford.edu/~ullman/mmds/ch9.pdf.

[32] C. Xu, D. Tao and C. Xu, "A survey on multi-view learning", *in Neural Computing and Applications*, vol. 23, pp. 2031-2038, 2013.

[33] T. George and S. Merugu, "A scalable collaborative filtering framework based on co-clustering", *in Fifth IEEE International Conference on Data Mining (ICDM'05)*, pp. 625- 628, Houston, Texas, USA, November 27-30, 2005.

[34] S. Bickel and T. Scheffer, "Multi-view clustering", *in Proc. of Internatoinal Conference Data Mining*, pp. 19-26, Brighton, UK, November 1-4, 2004.

[35] K. Chaudhuri, S. M. Kakade, K. Livescu, and K. Sridharan, "Multi-view clustering via canonical correlation analysis," *in Proc. of the 26th Annual International Conference on Machine Learning (ICML)*, pp. 129-136, Montreal, Canada, June 14-18, 2009.

[36] J. Wang, A. P. De Vries, and M. J. T. Reinders, "Unifying user-based and item-based collaborative filtering approaches by similarity fusion," *in Proc. the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 501-508, Seattle, WA, USA, August 6-10,2006.

[37] "Yelp Dataset Challenge | Yelp", *Yelp*, [Online] 2016. Available: http://www.yelp.com/dataset_challenge/.

[38] E. Zavitsanos, G. Paliouras, G.A. Vouros and S. Petridis, "Discovering Subsumption Hierarchies of Ontology Concepts from Text Corpora", *in Proc. IEEE/WIC/ACM International Conference on Web Intelligence (WI'07)*, pp. 402-408, Sillicon Valley, CA, USA, November 2-5, 2007.

[39] W. Liu and A. McCallum, "Pachinko allocation: DAG-structured mixture models of topic correlations", *in Proc. of the 23rd Intl. Conf. on Machine learning*, pp. 577-584, Pittsburgh, Pennsylvania, USA, June 25-29, 2006.

[40] A. M. Wei Li and D. Blei, "Nonparametric bayes pachinko allocation", *in Proc. of Conference on Uncertainty in Artificial Intelligence (UAI)*, Vancouver, Canada, July 19-22, 2007.

[41] D. Blei, A. Ng, and M. Jordan, "Latent dirichlet allocation", *in Journal of Machine Learning Research*, vol. 3, pp. 993-1022, 2003.

[42] T. Griffiths, M. Jordan and J. Tenenbaum, "Hierarchical topic models and the nested Chinese restaurant process", *Advances in neural information processing systems,* vol. 16, pp. 106-114, 2004.

[43] T. Mikolov, K. Chen, G. Corrado, and J. Dean, “Efficient estimation of word representations in vector space”, *in Proc. Intl. Conf. on Learning Representations (ICLR'13)*, Scottsdale, AZ, USA, May 2-4, 2013.

[44] M. J. Kusner, Y. Sun, N. I. Kolkin, Nicholas, and K. Q. Weinberger, "From word embeddings to document distances", *in Proc. Intl. Conf. on Machine Learning (ICML)*, pp. 957-966, Lille, France, July 6-11, 2015.

[45] C. Fellbaum, "Bradford Books", *WordNet: An Electronic Lexical Database*, 1998.

[46] H. Schü,tze, "Automatic Word Sense Discrimination," *Computational Linguistics*, vol. 24, no. 1, pp. 97-124, 1998.

[47] G. Hirst and D. St-Onge, &ldquo,Lexical Chains as Representations of Context for the Detection and Correction of Malapropisms,&rdquo, *WordNet: An Electronic Lexical Database*, MIT Press, pp. 305-332, 1998.

[48] P. Resnik, "Using Information Content to Evaluate Semantic Similarity In A Taxonomy", *in Proc. International Joint Conference on Artificial Intelligence (IJCAI)*, Quebec, Canada, August 20-25, 1995.

[49] M. Röder, A. Both and A. Hinneburg, "Exploring the space of topic coherence measures", *in Proc. the Eighth ACM Intl. Conf. on Web search and Data Mining*, pp. 399-408, Shanghai, China, January 31-February 6 2015.

[50] D. Milne and I. H. Witten, "An open-source toolkit for mining wikipedia", *Artiftificial Intelligence*, vol. 194, pp. 222-239, January 2013.

[51] L. Han, A. Kashyap, T. Finin, J. Mayfield and J. Weese, "Umbc ebiquity-core: Semantic textual similarity systems", *in Proc. of the Second Joint Conference on Lexical and Computational Semantics*, vol. 1. pp. 44-52, Atlanta, Georgia, USA, June 13-14, 2013.

[52] H. Wang, Y. Lu, and C. Zhai, “Latent aspect rating analysis without aspect keyword supervision,” in *Proceedings of the 17th ACM SIGKDD international conference on Knowledge* discovery *and data mining*. ACM, 2011, pp. 618–626. {TripAdvisor Data}

[53] L. Pradhan, C. Zhang, and S. Bethard, “Towards extracting coherent user concerns and their hierarchical organization from user reviews,” in *Proceedings of the 17th Interna- tional Conference on Information Reuse and Integration (IRI)*. IEEE, 2016, pp. 582–590.

[54] L. Pradhan, C. Zhang, and S. Bethard, "Extracting Hierarchy of Coherent User-Concerns to Discover Intricate User Behavior from User Reviews,” *International Journal of Multimedia Data Engineering and Management (IJMDEM)*, vol. 7, no. 4, pp. 582–590, 2016.

[55] L. Pradhan, C. Zhang, and S. Bethard, “Infusing Latent User- Concerns from User Reviews into Collaborative Filtering,” in *Proceedings of the 18th International Conference on Information Reuse and Integration (IRI)*, to be published.

[56] J. Schmidhuber, "Deep learning in neural networks: An overview", *Neural Networks*, vol. 61, pp. 85-117, 2015.

[57] Q. V. Le, A tutorial on deep learning part 2: Autoencoders convolutional neural networks and recurrent neural networks, 2015, [online] Available: https://cs.stanford.edu/~quocle/tutorial2.pdf.

[58] S. Sedhain, A. K. Menon, S. Sanner, L. Xie, "Autorec: Autoencoders meet collaborative filtering", *Proceedings of the 24th International Conference on World Wide Web*, ACM, pp. 111-112, 2015.

[59] UNSUP, 2017. [Online]. Available: https://github.com/koraykv/unsup

[60] R. Salakhutdinov, G.E. Hinton, "Semantic Hashing", *Proc. Int'l ACM SIGIR Conf. Research and Development in Information Retrieval*, 2007.

[61] Elkahky, Ali Mamdouh, Yang Song, and Xiaodong He. "A multi-view deep learning approach for cross domain user modeling in recommendation systems." *Proceedings of the 24th International Conference on World Wide Web*. ACM, 2015.

[62] Tang, Duyu, et al. "User Modeling with Neural Network for Review Rating Prediction." *IJCAI*. 2015.

[63] Tang, Duyu, Bing Qin, and Ting Liu. "Learning Semantic Representations of Users and Products for Document Level Sentiment Classification." *ACL (1)*. 2015.

[64] Torch7, 2017 [Online]. Available: http://torch.ch.

[65] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, "Improving neural net-works by preventing co-adaptation of feature detectors," http://arxiv.org/abs/1207.0580, 2012. {Dropout}

[66] H. Wu, R. Luk, K. Wong, and K. Kwok, “Interpreting TF-IDF term weights as making relevance decisions,” in *ACM Transactions on Information Systems*, vol. 26, no. 3, pp. 13, 2008.