Representation learning in Heterogeneous Information Networks for User Modeling and Recommendations

by

Surya Kallumadi

M.S., Kansas State University, 2010

AN ABSTRACT OF A DISSERTATION

submitted in partial fulfillment of the
requirements for the degree

DOCTOR OF PHILOSOPHY

Department of Computer Science
College of Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

2018

Abstract

Current research in the field of recommender systems takes into consideration the interaction between users and items; we call this the homogeneous setting. In most real world systems, however these interactions are heterogeneous, i.e., apart from users and items there are other types of entities present within the system, and the interaction between the users and items occurs in multiple contexts and scenarios. The presence of multiple types of entities within a heterogeneous information network, opens up new interaction modalities for generating recommendations to the users. The key contribution of the proposed dissertation is representation learning in heterogeneous information networks for the recommendations task.

Query-based information retrieval is one of the primary ways in which meaningful nuggets of information is retrieved from large amounts of data. Here the query is represented as a user's information need. In a homogeneous setting, in the absence of type and contextual side information, the retrieval context for a user boils down to the user's preferences over observed items. In a heterogeneous setting, information regarding entity types and preference context is available. Thus query-based contextual recommendations are possible in a heterogeneous network. The contextual query could be *type-based* (e.g., directors, actors, movies, books etc.) or *value-based* (e.g., based on tag values, genre values such as "Comedy", "Romance") or a combination of Types and Values. Exemplar-based information retrieval is another technique for of filtering information, where the objective is to retrieve similar entities based on a set of examples. This dissertation proposes approaches for recommendation tasks in heterogeneous networks, based on these retrieval mechanisms present in traditional information retrieval domain.

Representation learning in Heterogeneous Information Networks for User Modeling and Recommendations

by

Surya Kallumadi

M.S., Kansas State University, 2010

A DISSERTATION

submitted in partial fulfillment of the
requirements for the degree

DOCTOR OF PHILOSOPHY

Department of Computer Science
College of Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

2018

Approved by:

Major Professor
Dr. William H. Hsu

Copyright

Abstract

Current research in the field of recommender systems takes into consideration the interaction between users and items; we call this the homogeneous setting. In most real world systems, however these interactions are heterogeneous, i.e., apart from users and items there are other types of entities present within the system, and the interaction between the users and items occurs in multiple contexts and scenarios. The presence of multiple types of entities within a heterogeneous information network, opens up new interaction modalities for generating recommendations to the users. The key contribution of the proposed dissertation is representation learning in heterogeneous information networks for the recommendations task.

Query-based information retrieval is one of the primary ways in which meaningful nuggets of information is retrieved from large amounts of data. Here the query is represented as a user's information need. In a homogeneous setting, in the absence of type and contextual side information, the retrieval context for a user boils down to the user's preferences over observed items. In a heterogeneous setting, information regarding entity types and preference context is available. Thus query-based contextual recommendations are possible in a heterogeneous network. The contextual query could be *type-based* (e.g., directors, actors, movies, books etc.) or *value-based* (e.g., based on tag values, genre values such as "Comedy", "Romance") or a combination of Types and Values. Exemplar-based information retrieval is another technique for of filtering information, where the objective is to retrieve similar entities based on a set of examples. This dissertation proposes approaches for recommendation tasks in heterogeneous networks, based on these retrieval mechanisms present in traditional information retrieval domain.

Table of Contents

List of Figures x

List of Tables xii

Acknowledgements xiii

1 Introduction 1

1.1 Introduction 1

1.2 Representation Learning in Heterogeneous Networks 2

1.3 Recommender Systems Challenges 3

1.3.1 Sparsity and cold-start recommendations: 3

1.3.2 Diversity: 3

1.3.3 Interactive recommender systems: 3

1.3.4 Explanations to recommendations: 4

1.4 Objectives 4

2 Recommender Systems 6

2.1 Introduction 6

2.2 Overview of Recommender Systems Approaches 9

2.2.1 Providing Recommendations to Users 9

2.2.2 Collaborative Filtering Approach 10

2.2.3 Content-based Recommender Systems 13

2.2.4 Hybrid Recommender Systems 13

2.2.5 Domain Adaptation and Cross-Domain Collaborative Filtering 14

2.2.6 Deep Learning Approaches for Cross-Domain Recommendations . . . 20
2.3 Recommender Systems Evaluation, Metrics, and Measures 21
2.4 Taxonomy of Recommender Systems Metrics 22
2.4.1 Error and Accuracy metrics 23
2.4.2 Beyond Accuracy Metrics 29
3 Representation Learning in Networks 31
3.1 Embeddings and Distributed Representations 31
3.2 Language Modelling 33
3.2.1 Neural Network-Based Language Models 33
3.2.2 Word2Vec 35
3.3 Representation Learning in Networks 39
3.3.1 DeepWalk 41
3.3.2 Large-scale Information Network Embedding (LINE) 42
3.3.3 Node2vec 42
3.3.4 Deep Learning Approaches for Representation Learning in Information Networks 43
3.3.5 Pre-trained Embeddings and their Applications in Downstream Discovery Tasks 45
4 Representation Learning and Recommendations in Heterogeneous Networks 49
4.1 Introduction 50
4.1.1 User-driven Interactive Recommendations 51
4.2 Problem Definition 53
4.3 Distributed Representations and HIN embeddings 58
4.3.1 Skipgram 59
4.3.2 Skipgram for Metapaths 59
4.4 Experiments and Results 61

4.4.1 Dataset . . . 61
4.4.2 Evaluation Metrics . . . 63
4.4.3 Results . . . 63
4.5 Interactive Recommendations . . . 66
4.5.1 Exemplar-based Recommendations . . . 66
4.5.2 *Less Like this and More Like this* . . . 68
5 Semantic Diversity in Top-N Recommender Systems . . . 70
5.1 Introduction . . . 70
5.2 Background . . . 72
5.2.1 Determinantal Point Processes (DPPs) . . . 72
5.2.2 k-DPPs . . . 73
5.3 HIN Embeddings over User Curated Lists . . . 75
5.4 Data Set, Experiments and Results . . . 76
5.4.1 Experiments and Results . . . 77
5.5 Diversity Metrics and their Utility . . . 78
6 Music Playlist Completion and Continuation . . . 79
6.1 Introduction . . . 80
6.2 The RecSys 2018 Challenge . . . 82
6.3 Our approach . . . 85
6.3.1 Candidate generation . . . 87
6.3.2 Learning to Rank . . . 90
6.4 Results . . . 92
6.5 Conclusion . . . 93
7 Conclusions and Future work . . . 95
7.1 Summary and Conclusions . . . 95
7.1.1 Representation Learning in HIN using Constrained Random Walks . 96

7.1.2 New Modalities for Interaction for Interactive Search and Recommendations . . . 96
7.1.3 Using Representation Learning to Improve Semantic Diversity of Ranked Lists . . . 97
7.1.4 Using Semantic Embeddings for Recommendation Tasks, with Application to Playlist Completion and Continuation . . . 98
7.2 Future work . . . 99
7.2.1 Deep Semantic Models that Utilize Semantic Embeddings for Information Retrieval and Discovery . . . 99
7.2.2 High Accuracy Recall to Improve Performance of Search Systems . . . 99
7.2.3 Cross-Domain Recommendations using Domain-Aware Metapaths . . . 101
7.2.4 Retraining and Incremental Training for New Data, Users, and Items 101
Bibliography . . . 103

List of Figures

2.1 Ratings matrix 7

2.2 *Netflix* movie ratings distribution with power-law characteristics 8

2.3 Rank 3 SVD of a Matrix 12

3.1 Neural Language Model, Bengio et al.[1] 34

3.2 Continuous Bag-Of-Words (CBOW) architecture 37

3.3 Skipgram architecture 38

3.4 A search engine architecture - iterative refinement of results[2]. An example cascade architecture. After an initial ranking function H_0, each stage consists of two sequential operations: J_t prunes the input ranked documents, then a local ranking function H_t refines the rank order of the retained documents. The new ranked list is passed to the next stage. The size of the shaded area denotes the size of the candidate documents. Subscripts for each ranked list denotes the sequence of actions applied. 45

3.5 A recommender system architecture 47

4.1 Movie domain interaction between entities 51

4.2 Movie - Knowledge Graph Schema 53

4.3 (a) heterogeneous information network, (b) traces of fixed-length random walks 54

4.4 Citation network Metapaths 55

4.5 Movie Network Metapaths 55

4.6 precision@5 and precision@10 for varying sparsity in IMDb domain 64

4.7 precision@5 and precision@10 for varying sparsity in citation network 65

6.1 Demographics of users who contributed to the MPD by Left: gender and Right: age. 83

7.1 DSSM style model for playlist continuation task 100

List of Tables

2.1 Popular Recommender Systems Evaluation Metrics. 23

4.1 Metapaths captured from IMDB schema 57

4.2 Metapaths captured from Citation Network schema 57

4.3 IMDb dataset overview 61

4.4 Citation network dataset overview 61

4.5 Results for Precision@K with varying sparsity 63

4.6 Results for Kendall's τ, for ranked-order comparison 63

4.7 Exemplar-based user list generation for "Bond Movies" (Left) and "Film Noir" (Right) Movie titles in bold constitute the query set 66

4.8 Exemplar-based user refinement for the intent "newer romantic comedies", Movie titles in bold constitute the query set 67

5.1 Average dissimilarity measure of diversity 78

6.1 The full list of features that our learning to rank model considers. The features are categorized based on whether they depend only on the input playlist or the candidate track, or both. 86

6.2 Offline evaluation results for individual candidate sources and the combined LTR model output. For the combined model, we only measured the metrics at rank 500. The combined model achieves the best performance while QE emerges as the best candidate source. Note that for the clicks metric a lower value indicates a better performance. 91

6.3 Recall advantage of embeddings over QE. Recall advantage indicates the additional percentage of tracks retrieved by Embedding approaches over QE . . 91

6.4 The final RecSys 2018 spotify challenge leaderboards. Our submissions are highlighted in bold. Only the top 10 teams from the leaderboards are shown. The total number of participating teams was 112 and 31 for the main and the creative tracks, respectively. For the clicks metric a lower value indicates a better performance. 94

Acknowledgments

Firstly, I thank my advisor, Dr. William Hsu for his guidance, patience, and kindness. Dr. Hsu has always been generous with his time and resources. I will forever be grateful for all the opportunities he has provided me during my studies here at K-State. I owe my deepest gratitude to him for helping me attend DSSI, Machine Learning Summer School, and the other numerous conferences in the past few years.

I also thank my dissertation committee, Dr. Doina Caragea, Dr. David Gustafson, Dr. Caterina Scoglio, and Dr. Chris Pinner, for their valuable feedback. Special thanks to CIS academic advisor Sheryl Cornell, for her invaluable assistance over the years.

Bhaskar Mitra, Tereza Iofciu, Gabriel Necoechea, Andrew Trotman, and Felix Gräßer have been wonderful research collaborators. I thank them for their time, and thoughtful discussions.

Chapter 1

Introduction

"You cannot answer a question you cannot ask and you cannot ask a question that you have no words for."

- Judea Pearl, *The Book of Why: The New Science of Cause and Effect*

1.1 Introduction

Recommender systems play a vital role in modern day online ecosystems by surfacing relevant item recommendations to users. These ecosystems can be as diverse as online marketplaces such as Amazon.com, ebay.com etc. to content delivery platforms such as Netflix, Spotify and Google. In the presence of a large amount of information, recommender systems act as an information filtering mechanism that can predict a user's affinity to an unseen item. Recommender systems recommend items to a user based on a user's explicit and implicit preferences, the behaviour of other similar users, and user and item attributes[3].

Prevalent research in the field of recommender systems takes into consideration the interaction between users and items; we call this the homogeneous setting. But, in most real world systems these interactions are heterogeneous, i.e. apart from users and items there are other types of entities present within the system, and the interaction between the users and items occurs in multiple contexts and scenarios. For example, in the movie domain, a user can rate an item, but also a user can curate an item in a list based on a user defined context. In addition, a user might show preferences towards items based on the attributes associated with the items. In the movie domain, a user can like items because of the *topic* of the movie, the *director, actor* etc. In an ecommerce scenario, one can imagine the user preferences driven by not just the item, but attributes associated with the items. We can model such interaction domains as *heterogeneous information networks.*

The presence of multiple types of entities within a heterogeneous information network, opens up new modalities for generating recommendations to the users. The objective of this dissertation is to develop recommendation approaches in heterogeneous information networks. Query-based information retrieval is one of the primary ways in which meaningful nuggets of information are retrieved from large amounts of data. Here the query is represented by a user's information need. In a homogeneous setting, in the absence of *type* and contextual side information, the retrieval context for a user boils down to the user's preferences over observed items. In a heterogeneous setting, information regarding entity types and preference context is available. Thus query-based contextual recommendations are possible in a heterogeneous network. The contextual query could be type-based (e.g. directors, actors, movies, books etc.) or value-based (e.g. based on tag values, genre values such as "Comedy", "Romance") or a combination of types and values. Exemplar-based information retrieval is another modality of filtering information, where the objective is to retrieve similar entities based on a set of examples.

1.2 Representation Learning in Heterogeneous Networks

Heterogeneous information networks (HINs) are networks with multi-typed nodes and edges, which are more common in real life than homogeneous networks. In a HIN, relationships between nodes are more complex than those in a homogeneous network, thus more difficult to be represented. Representation learning of networks involves projecting nodes in the network into a latent dimension such that some property/set of properties of the nodes within the network is preserved. Classical network embedding methods like DeepWalk[?] and node2vec[?]

leveraged random walks to explore the structural information of the network and utilized Skipgram[4] to project the node into a low-dimensional vector. LINE represented the first-order proximity and second-order proximity of the network so as to capture the local and neighborhood network structures. All these models only aim to learn the representation for homogeneous networks, and perform poorly when applied to a HIN. In this dissertation I propose *metawalk*-based representation learning in HIN. The advantage of this approach is that metawalks are domain aware i.e. we can exploit the domain specific information and explicitly specify domain knowledge to learn representations over heterogeneous networks.

1.3 Recommender Systems Challenges

Following are some of the open research problems in recommender systems[5]:

1.3.1 Sparsity and cold-start recommendations:

Traditional recommender systems approaches have trouble with performance when not enough preference data is available with respect to users or items. This is known as the cold-start setting. Hybrid recommender systems have demonstrated their effectiveness in cold-start[6].

1.3.2 Diversity:

Providing novel and diverse recommendations has become an important RecSys research area because of the potential of current approaches resulting in filter bubbles[7].

1.3.3 Interactive recommender systems:

Current interfaces in recommender systems are passive, i.e. the systems takes into consideration the user, user history and the user context to provide recommendations to the users. This leaves no scope for the user to express their needs. Interactive recommender systems deal with the design of systems that allow the user to express their information needs[8;9].

1.3.4 Explanations to recommendations:

There is a need for interpretable recommendations to improve user trust in the system by making recommenders transparent. One way of achieving this objective is by providing explanations to the recommendations provided to the user.

1.4 Objectives

In this dissertation, based on retrieval mechanisms present in traditional information retrieval domain, we explore the following recommendation tasks in heterogeneous networks:

- Perform representation learning in heterogeneous information networks for the recommender systems domain based on implicit and explicit ratings. We learn latent representation of nodes based on random walks to encode relationships between various types of nodes.

- Query-based ranked recommendation of items to users, when type information is present within the network. Explore generic and personalized list curation based on a search query. Evaluate the performance of the approaches over implicit and explicit feedback data sets over multiple domains.

- Exemplar-based ranked retrieval of items in a heterogeneous setting. This is similar to the list completion task in machine learning and information retrieval. Given a set of example items of the same type, the objective is to retrieve the top-K items that fit with the exemplars based on latent representation of nodes in the network.

- User modeling-based on item preferences for user profile creation in heterogeneous information networks.

- Explore approaches for integrating explicit and implicit feedback preference data in heterogeneous networks. We focus on lists, which are a special case of implicit feedback with user specified context.

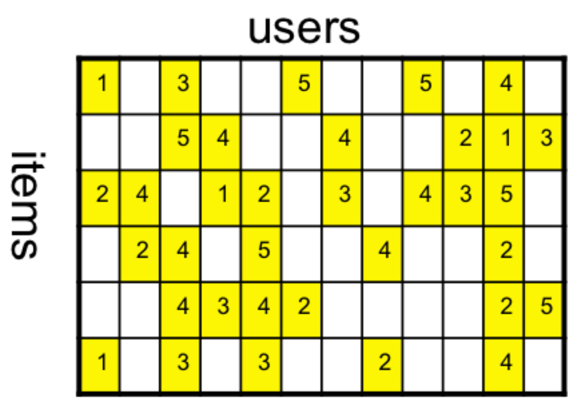
- New modalities for search and recommendation in heterogeneous information networks. Expose users to more intuitive ways of specifying user needs and context.
- Improve diversity of results of recommendation algorithms, by mapping entities in the domain to various domain driven representations.

Chapter 2

Recommender Systems

2.1 Introduction

Recommender systems play a vital role in modern day e-commerce ecosystems by providing relevant item recommendations to users. In the presence of large amounts of information, recommender systems act as an information filtering mechanism that can predict a user's affinity for an item. Recommender systems recommend items to a user based on a user's explicit and implicit preferences, the behaviour of other similar users, and user and item attributes[3]. Collaborative filtering data primarily consists of a set of users and their preferences over a set of domain items.[10] The user preferences over the items can be expressed in the form of either a rating range or a binary opinion (like/dislike) . Thus, each data point in this domain can be represented as a <**User, Item, Response**> *preference triple.* Items could be articles, web pages, advertisements, movies, books, etc. Items can be ephemeral or sticky depending on the application domain. For example, in a news recommender system, items are ephemeral and the item set is dynamically changing, whereas in a movie recommendation system, items are sticky and the item set is relatively stable. The preferences or ratings can be real valued or integer values in a predefined range. Preferences can also be implicit or explicit; in the case of explicit ratings, the users rate the items, whereas in the case of implicit ratings the rating score is determined based on past user behavior. For


users
items

1		3			5			5		4	
		5	4			4			2	1	3
2	4		1	2		3		4	3	5	
	2	4		5			4			2	
		4	3	4	2					2	5
1		3		3			2			4	

Figure 2.1: *Ratings matrix*

example, *Netflix* has a 1-5 explicit discrete Likert rating scale[11], whereas online e-commerce sites like eBay and Amazon have implicit ratings based on historical user behavior. This behavior could be user purchases, items the user has viewed previously on the site, number of page views, clicks etc.

The set of response triples form a sparse matrix (sparsity is domain-dependent) and in case of ratings, is called as the *ratings matrix*. Fig. 1 shows a sample ratings matrix. The ratings matrix cells are filled in with the user rating for the corresponding item that cell belongs to. The ratings matrix is usually sparse, as a user might not rate every item in the repository. Thus, the collaborative filtering task has two primary components: a) the prediction task b) the recommendation task. In the prediction task, the objective is to predict the rating preference for an unseen item given a user. In the recommendation task, the objective is to obtain a ranked list of most appropriate items that addresses a user need.

Cold-start and sparsity: A common bottle neck for recommender systems is the *cold-start*[12] problem, where recommendations are required for items that no user has rated yet or items need to be recommended to users who have not expressed any preference yet. Traditional CF approaches are ineffective in the presence of the cold-start setting due to the absence of item ratings and/or user preferences. For example, new users of the CF system must have expressed very few ratings or behavioral preferences over items, and items that

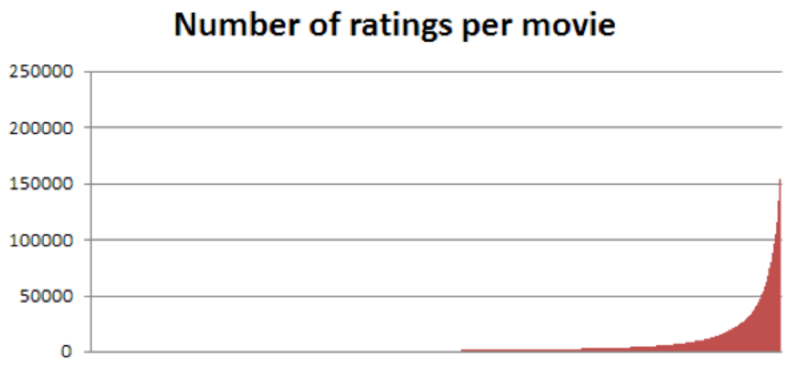


Figure 2.2: *Netflix movie ratings distribution with power-law characteristics*

are newly added to the system will have sparse ratings. Another problem of CF approaches is their sensitivity to sparsity[13]. Sparsity occurs when only a small subset of items are rated by most of the users, or when only a few users constitute the item ratings. Fig. 2.2 illustrates the sparsity problem within real-world data sets which exhibit power-law distribution characteristics. For example in *Netflix* data, a small set of movies have a lot of ratings, whereas a lot of movies have very few ratings.

One way to mitigate the cold-start problem is by using content-based and hybrid recommender approaches. In *content-based recommender systems*, we take the content and the meta-data associated with the items into account for the recommendations task. In *hybrid recommender systems*, the user-item references and the item content are both taken into account for the recommendations task. Another way of reducing the effects of sparsity in recommender systems is to perform cross-domain CF (CDCF). In CDCF we can have multiple source domains for each target domain, and the objective is to transfer knowledge in the form of user preferences from the source domains to the target domain. One central assumption to CDCF is that there should be an inherent relationship between the domains between which learning is performed. We can consider news, browsing activity, search and ads as closely related domains, where the type of news we consume, the online articles we browse, the queries we input for search and the ads we click on can be considered to be inherently related. Thus knowledge from one or more of these domains could be used as side information to drive recommendations within another domain.

2.2 Overview of Recommender Systems Approaches

Recommender systems can be viewed as a content optimization problem and the goal is to serve items to users. The most common approaches to recommending items to users can be broadly classified into *content-based filtering* (CBF) , collaborative filtering (CF) and hybrid recommender systems. Content-based filtering approaches use the items that a user prefers in order to recommend similar items to the user, based on item features that can be derived from aspects such as descriptions, social tags, price, *etc.* CBF methods create a profile for each user using the user's characteristics and behaviour, then use this profile to recommend more items to the user. Machine learning classification methods such as Naive Bayes classification and logistic regression have been used for CBF[14;15]. whereas collaborative filtering looks at the correlations between the ratings of items provided by the user[10]. CF is used when users associate quality and taste ratings to content, thus the context is more than just keywords or topics in the case of CF. Hybrid recommender systems are a combination of CBF and CF approaches[16].

2.2.1 Providing Recommendations to Users

There are two aspects to recommender systems: predicting a rating to a user-item pair, and providing item recommendations to a user in the target domain. The recommender systems field has moved away from the rating prediction task towards the item recommendation task. The evaluation metrics used for evaluating the effectiveness of recommender systems approaches reflects this shift. At any given time the user has limited bandwidth for consuming items recommended to them. Thus, what matters is not how accurately we can predict the ratings but how relevant the top-k items recommended to the user are. This shift is also largely dictated by the constraints imposed by the system user interface.

2.2.2 Collaborative Filtering Approach

The primary idea behind collaborative filtering is to suggest recommendations to a user by relying on the ratings and behavior of other similar users. The fundamental assumption about user behavior in collaborative filtering is that similar users will have similar interests and vice-versa. Therefore, if users agree in their opinion or ratings, then they are more likely to agree in their preferences about other items. Various CF approaches have been developed in previous work. *Memory-based* methods use nearest neighbor approaches to compute similarity between users or items[17,10]. *Model-based* methods learn the rating preference model for users based on training data[18]. Matrix factorization based methods try to find a low rank approximation of the ratings matrix[19,20].

Types of Collaborative Filtering

User-User collaborative filtering: The objective of User-User collaborative filtering[17] is to find other users with ratings similar to the current user. By identifying relevant users we can leverage their ratings on other items to model the current users' preferences. The item ratings for the current user are obtained by weighting the relevant user item ratings by the level of their agreement with the current user. Pearson correlation[21], Spearman correlation[21], cosine similarity are some of the similarity metrics used to measure agreement between users. As the number of users using the system grows, User-User collaborative filtering suffers from scalability issues. In practical applications, the number of users will be much greater than the number of items and the user set constantly changes whereas the item set is relatively stable.

Item-item Collaborative Filtering: Item-based collaborative filtering[22] tries to overcome the scalability drawback of user-user collaborative filtering. Item-item collaborative filtering uses similarities between the ratings of items. The rationale behind item-based filtering is that similar items tend to have similar ratings, then the users will have similar predilections towards similar items. In a domain where $|U| >> |I|$, item-based collaborative filtering has much better scalability than user-user collaborative filtering.

K-nearest neighbor approaches/Neighborhood-based Approaches are usually used in user-user CF and item-item CF, where all the pairwise correlations of all the pairs of users or items in the system are first calculated. These pairwise correlations provide a measure of agreement between two entities (user/item) . For any entity, the new item recommendations are calculated by averaging the ratings vectors for the K-closest entities. Pearson correlation is usually used as a distance metric for calculating the neighborhood of an entity (item or user) . Pearson correlation is defined as:

$$s_{ij} = \frac{\text{Cov}[r_i, r_j]}{\text{Std}[r_i]\text{Std}[r_j]} \tag{2.1}$$

where s_{ij} is the similarity of entities i and j, Cov is the co-variance of two entities and Std is the standard deviation of an entity. Similarity of two entities will be +1 if they are perfectly correlated and -1 if they are anti-correlated.

Konstan *et al.*[23][17] discuss the effectiveness of news recommendation in the Usenet domain using neighborhood recommendation approaches. These approaches are shown to be effective in the presence of sparse rating data, infrequent occurrence of news articles. Sarwar *et al.*[10] highlight the computational drawbacks of neighborhood exploration in user-based CF approaches and suggest item-based CF as a viable alternative that is both scalable and accurate for explicit data. Linden *et al.*[22] demonstrate the viability of using item-based CF approaches in the presence of implicit data (co-purchase information) .

Graph-based Approaches

Graph-based approaches have been proposed to exploit the user-item preference information as a bi-partite graph. The adsorption algorithm[24], proposed for a personalized video recommendation application in the *YouTube.com* domain, looks at the user-video content consumption graph. The random walk approach proposed in this work propagates preference information of users to their neighborhood (similar to label propagation[25]) , and is shown

$$
\begin{array}{|c|c|c|c|c|c|c|c|c|c|c|c|}
\hline
1 & & 3 & & & 5 & & & 5 & & 4 & \\ \hline
 & & 5 & 4 & & & 4 & & & 2 & 1 & 3 \\ \hline
2 & 4 & & 1 & 2 & & 3 & & 4 & 3 & 5 & \\ \hline
 & 2 & 4 & & 5 & & & 4 & & & 2 & \\ \hline
 & & 4 & 3 & 4 & 2 & & & & & 2 & 5 \\ \hline
1 & & 3 & & 3 & & & 2 & & & 4 & \\ \hline
\end{array}
\sim
\begin{array}{|c|c|c|}
\hline
.1 & -.4 & .2 \\ \hline
-.5 & .6 & .5 \\ \hline
-.2 & .3 & .5 \\ \hline
1.1 & 2.1 & .3 \\ \hline
-.7 & 2.1 & -2 \\ \hline
-1 & .7 & .3 \\ \hline
\end{array}
\begin{array}{|c|c|c|c|c|c|c|c|c|c|c|c|}
\hline
1.1 & -.2 & .3 & .5 & -2 & -.5 & .8 & -.4 & .3 & 1.4 & 2.4 & -.9 \\ \hline
-.8 & .7 & .5 & 1.4 & .3 & -1 & 1.4 & 2.9 & -.7 & 1.2 & -.1 & 1.3 \\ \hline
2.1 & -.4 & .6 & 1.7 & 2.4 & .9 & -.3 & .4 & .8 & .7 & -.6 & .1 \\ \hline
\end{array}
$$

Figure 2.3: *Rank 3 SVD of a Matrix*

to be scalable in the presence of large number of users and items.

Dimensionality Reduction and Latent Factor Models

The objective of dimensionality reduction is to decompose the ratings matrix into a lower dimensional matrix. By identifying a set of n-topics the ratings matrix can be represented as a combination of the user's interest in the topics and the relevance of an item to a topic. Latent semantic analysis (LSA)[26], singular value decomposition (SVD)[27], principal component analysis (PCA)[28], and probabilistic latent semantic analysis (PLSA)[29] are some of the matrix factorisation methods used for dimensionality reduction for collaborative filtering[19 20]. Fig. 2.3 gives an illustration of SVD by decomposing a ratings matrix to Users and Items matrices of Rank 3. Using this approach, missing values in any cell of the original matrix can be calculated by obtaining the dot product of the corresponding row vector of the first matrix and column vector of the second matrix. This decomposition has a rank of 3, i.e., the number of latent concepts of the new representation of the original matrix is 3. Since SVD is undefined when there are missing values within a matrix, stochastic gradient descent (SGD) is usually used to obtain the matrices in latent space[19;20]. Proper care must be taken to avoid overfitting while using SGD for matrix factorisation. The SVD problem can be converted to an optimisation problem so as to solve it using SGD. Alternating Least Squares (ALS)[30] approaches are a family of optimization based matrix factorization approaches that try to decompose the preference matrix into a latent space.

2.2.3 Content-based Recommender Systems

Apart from user-item preference information, the items in the domain can also be associated with content and side information. This content could be in the form of title, description, meta-data, tags, user reviews etc. Content-based recommender systems aim to leverage this item specific information to provide recommendations[31]. In order to match a user to an item-based on item content, these recommender systems also maintain a profile of user's interests. Thus there are three main aspects to content-based recommender systems: 1) Representing items in the system based on item properties and item content 2) Representing users in the form of a user profile that captures the user interests 3) Matching the item representations with the user profile so as to recommend items to the users[31]. CF approaches that depend on preference information will fail to provide good recommendations in a cold-start setting i.e., when there is not much user preference information associated with the item. This leads to low exploration and coverage of items within the system. Content-based CF approaches mitigate this problem by considering the content of the item. These approaches have shown to be effective on news recommendation applications where new news articles get published frequently and user-item preference information is not available[32][33]. It is important to note here that CF approaches that are based on user-item preference information suffer from cold-start issues when a new user or a new item are introduced into the system. Content-based CF approaches face cold-start issues when a new user is introduced into the system[34]. Mooney *et al.*[14] show the effectiveness of using item information to provide recommendations in a book recommendation setting. In music recommendation and automatic playlist continuation applications, acoustic properties of a track are extracted from a audio track and these properties are used for recommending new audio tracks to the users[35].

2.2.4 Hybrid Recommender Systems

Hybrid recommender systems are the class of recommendation approaches that take advantage of content as well as user-item preference information, effectively combining content-based and collaborative filtering approaches[36]. Hybrid recommendation algorithms can be broadly classified into 1) ensemble-based approaches 2) monolithic design 3) mixed systems[37]. In ensemble systems, the results of various approaches are combined into a single system. In monolithic approaches, joint modeling over various data sources is performed. In mixed systems, the results of various algorithms are shown next to each other. Mixed systems are a presentation level approach rather than an algorithm level recommendation approach.

2.2.5 Domain Adaptation and Cross-Domain Collaborative Filtering

In this section we begin by introducing the need for domain adaptation in CF, then define the problem and requirements for cross-domain adaptation, and then review some of the previous efforts for cross-domain collaborative filtering. In supervised machine learning we first train a model based on data that is available for a problem that we are interested in; this data is known as training data. This model is then used for predicting the behavior in the target domain by using the testing data. Machine learning methods are usually based on the assumption that the training and the testing datasets have common features and distribution and belong to the same domain, whereas in transfer learning, we do not assume that the training and testing data sets have the same distribution and domain. The purpose of domain adaptation and transfer learning in machine learning is to improve a learning task in a target domain by using knowledge transferred from a source domain in which a related task is known. Thus, using transfer learning, if we have insufficient data in the testing domain then we leverage data from a related domain with sufficient data to transfer relevant knowledge and make predictions in the testing domain.

In real-world applications, there may be dependencies and correlations between user behavior across multiple domains. While recommender systems provide personalised recommendations based on just the host domain, domain adaptation and cross-domain CF aims to exploit knowledge gathered about user preferences from one domain and using it in the target domain. In cross-domain CF, we have a source and a target domain and the assumption is that there is a natural relationship between the source and the target domains and that source domain has an active recommender system whereas the target domain suffers from the sparsity bottleneck. For example the books recommender and a movie recommender can be assumed to have a natural relationship, and the assumption here is that a user interested in a particular genre of books (say romance) will also be interested in a similar genre of movies and vice-versa. These recommender system domains can be considered to be related and useful knowledge can be transferred from the source recommender to the target recommender. The rationale behind the domain adaptation approach to overcoming sparsity is that related domains have a common latent space.

Requirements for Cross-domain Adaptation: A cross-domain approach should be able to infer user preferences and suggest recommendations to the user in domain **T** based on user ratings in domain **S**, where **T** is the *target domain* and **S** is the *source domain.* Thus a cross-domain recommender algorithm must be able to recommend books to a user, based on other user ratings over movies. The performance improvement in the system using cross-domain CF can be evaluated on the rating prediction problem, in which the ground truth exists as a set of ratings of items by users, and the objective is to predict the rating of a user for a new item. Transferring knowledge between domains is a non-trivial task, as it cannot be guaranteed that the knowledge obtained from one domain can be useful within another domain. Transfer learning in CF depends on a wide variety of factors like the dependence between domains and if the data is inherently related, the data characteristics and the availability of side information, and whether the domains have a common subset of resources such as items or users.

Background

The easiest and the simplest way to achieve cross-domain CF is by importing relevant data from a source domain and then combining the transferred knowledge with the target data. This aggregation will be simple when the domains share information on the same users. Berkovsky *et al.*[38] refer to this problem as cross-domain mediation and introduce several methodologies for importing relevant data from target domain when there is an explicit overlap between the users of the domains. Li[39] presents a survey of cross-domain collaborative filtering approaches in. Li outlines three types of domains around which cross-domain CF methodologies are structured namely, system domain, data domain, and temporal domain. In approaches oriented towards system domain, the cross-domain methodologies are structured around the target and source datasets within which transfer learning is performed. A system domain is further decomposed into two sub-domains: the user domain and the item domain. Data domain approaches for cross-domain recommendations revolve around the different representations of user preferences. Data domain representations can be implicit (e.g. clicks, purchases) or explicit (e.g. ratings, likes) . In temporal domain-oriented approaches, the domains are subsets formed by splitting the dataset based on timestamps and the objective is to perform transfer learning across these time stamps.

Winoto and Tang proposed one of the first frameworks for domain adaptation in CF[40], whereby, they identified four primary areas of research in CDCF 1) determining the existence of cross-domain dependence and correlation in user preferences across domains, 2) designing methodologies that perform cross-domain collaborative filtering in order to reduce problems arising out of sparsity in the target domain, 3) designing approaches for increasing the diversity of recommendations in target domain by using knowledge aggregated from other domains 4) designing evaluation methodologies for recommendations based on cross-domain collaborative filtering. As part of this work, the authors utilise a *k-nearest neighbors (k-NN)* model to predict ratings across domains. As part of verifying the existence of dependence among domains, they proposed to verify dependence across group level and individual level. In this work, Winoto and Tang come to the conclusion that CDCF does not increase accuracy of recommendations over single domain CF; however, it has an added advantage of increasing diversity of recommendations (increasing user engagement in the domain), and addressing the cold-start and sparsity problems.

Li *et al.*[41] introduce the notion of a *bridge* between domains to transfer knowledge. Thus the cross-domain CF problem can be reduced to the problem of constructing a bridge between domains. The bridge approach to cross-domain CF has been subsequently used in number of methods[42–46]. Li *et al.* introduce the method of *Codebook-Based knowledge Transfer (CBT)*[41] for recommender systems, which transfers useful knowledge from the source domain auxiliary rating matrix to remedy the sparsity of the rating matrix in a target domain. The knowledge is transferred in the form of a codebook, which is learned from an auxiliary rating matrix by compressing the cluster-level user-item rating patterns into a low level representation. Li *et al.* introduce a *Rating-Matrix Generative Model (RMGM)*[42] to learn the shared latent concept-level cluster rating patterns, these learnt common concept patterns act as bridge across domains and can be used to transfer knowledge across domains to reduce sparsity. In subsequent work, Li *et al.* apply their cross-domain collaboration framework RMGM, to capture transfer learning over time[47]. Li *et al.* try to capture user interest drift over items across temporal domains by splitting user behavior into time slices. Moreno *et al.*[48] enhance the CBT framework proposed by Li *et al.*[41], such that knowledge can be transferred from multiple source domains to reduce sparsity in the target domain. This new model accounts for the interactions amongst the source domains and the varying degree of relatedness amongst the multiple source and target domains. The general framework for CBT and RMGM is to come up with a rating pattern sharing matrix. RMGM uses the rating pattern sharing concept into a probabilistic model that addresses multi-task learning in collaborative filtering. The idea is to construct a shared codebook that encodes group level User-Item sharing patterns across domains. In the rating-pattern sharing approaches given a set of rating matrices $X^{(d)}$ over d-domains, the matrices are factorized such that $X^{(d)} = U^{(d)}B[I^{(d)}]^T$, where $U^{(d)}$ is a user-group membership matrix and $I^{(d)}$ is an item-group membership matrix. B is the group-level codebook shared by the domains. Thus B acts as a bridge over which ratings can be transferred across domains. This approach assumes that the users are shared across domains and the rating scales are uniform across domains.

Pan *et al.*[43] look at ways to transform knowledge from domains which have heterogeneous forms of user feedback. This is especially useful when there is mismatch in user feedback across domains. For example one domain can have user feedback in terms of explicit ratings while another domain can have user feedback in terms of implicit preference such as clicks or dwell time. Pan *et al.* address this issue by discovering the principle coordinates of both users and items in the auxiliary data matrices of source domains, and transfer them to the target domain. To learn the principal coordinates, the authors propose the use of CST (coordinate system transfer) to adapt latent features learnt from source domain to aid recommendations in the target domain. In other work Pan *et al.*[45] use a *Transfer by Collective Factorization (TCF)* concept to transfer binary ratings from source domain matrix to a target numerical rating matrix. Pan *et al.* explore ways in which knowledge about disparate rating representations can be transferred across domains. This is an extension of their previous work in that they take advantage of implicit user feedback in the source domain to predict in target domain. Similarly Zhang *et al.*[49] propose a probabilistic approach that uses *Probabilistic Matrix Factorization (PMF)* to model the ratings prediction problem in source and target domains and adaptively transfers knowledge across the domains by learning the correlation between domains. While Zhang *et al.* use explicit feedback to transfer knowledge, Tang *et al.*[50] use implicit feedback to transfer knowledge across domains by using Bayesian Personalized Ranking (BPR) criterion, which uses collective matrix factorisation to optimize their cost function. In these approaches, CF is performed across domains by latent feature sharing where given a set of rating matrices $X^{(d)}$ over d-domains, the matrices are factorized such that $X^{(d)} = US^{(d)}I^T$, where U and I are the User and Item latent feature matrices and S is a scaling and rotation matrix. Here the knowledge is transferred across the domains by using U and V as bridges across domains.

Shi *et al.*[51] propose a tag-induced cross-domain collaborative ltering framework TagCDCF, that uses common item tags as bridges of domain transfer for improving CDCF across domains. Shi *et al.*[52] improve upon their TagCDCF framework by addressing some of the drawbacks of TagCDCF. Shi *et al.* propose a GTagCDCF framework that takes into consideration user-item, user-tag and item-tag interactions across domains to provide recommendations. Wang *et al.*[53] propose a *Tag Transfer Learning (TTL)* approach to transfer knowledge from a source domain with dense tags to a target domain with sparse tags. As in the earlier works mentioned above, TTL tries to use tags as bridges for transferring knowledge across domains. In TTL, latent topics are obtained by clustering the tags and then recommendations are performed based on these transferred topics. In this work, Wang *et al.*[53] also provide preliminary insights into quantitative analysis of when to transfer tag information across domains.

One major drawback of existing work is that the proposed approaches can only be applied in the cases where the source and the target domain share the same set of users or items. In many real-world applications such shared data does not exist. This drawback of current approaches was pointed out by Fernández-Tobías *et al.*[54]. In practical real-world circumstances, it would make sense to learn from source domain when no explicit overlap exists with the target domain in terms of users or items. Apart from the above drawback, approaches based on codebook-based knowledge transfer[41] and rating-matrix generative model[42] require that different domains should have the same rating scale. This is not a practical restriction in a real-world application, as different domains have diverse ratings scales and varying user feedback mechanisms.

Taxonomy of Domain Adaptation Approaches in Recommender Systems

Cross-domain collaborative filtering raises two important questions: 1) is transfer of knowledge feasible between source and target domains and how do we determine the closeness and dependence between domains, 2) how can knowledge be transferred from source to target domain under varying assumptions? The first challenge is a learning problem and deals with identifying the bounds of transfer learning in recommender systems. The second problem deals with identifying and developing methodologies for cross-domain collaborative filtering based on target and source characteristics.

Various strategies for domain adaptation in recommender systems can be categorised as:

1) Explicit ratings-based domain adaptation (uses CF) : Here, the user set does not overlap between the source and the target domain. The objective would be to transfer knowledge from the source domain to the sparse target domain using the user interests in the target domain. The user rating pattern in the source can be used to enhance User-User collaborative filtering in the target domain. This approach is useful when some kind of user profile information is available in the source and target domains and the idea is to use this

profile information to extract knowledge from source domain and provide recommendations in target domain.

2) Implicit ratings-based domain adaptation (uses CF) : Here the user sets and item sets are considered to be disjoint but an implicit connection can be established across items in the domains. For example if we consider the books and movie domains, a book can have a movie adaptation and this information can be used to enhance Item-Item collaborative filtering in the target domain.

3) Content-based domain adaptation (hybrid approach): This approach can be considered a hybrid content-based approach to domain adaptation. Here we can take the semantic content associated with the items in the source domain and provide recommendations in the target domain. The rationale behind using content-based recommendations is that related domains have a common set of latent topics and topics that have a correlation in source domain will have a similar correlation in the target domain. This type of content-based recommendation can be used to enhance both Item-Item and User-User collaborative filtering in the target domain.

4) Social network-based domain adaptation: The intuition behind social network-based domain adaptation is similar to the rationale behind User-User CF. Here we have a more constrained assumption that in a social network friends with similar tastes in one domain will have similar tastes in other related domains. For example if two friends have similar tastes in movies, then the assumption is that they have similar preferences over TV shows which is related domain.

2.2.6 Deep Learning Approaches for Cross-Domain Recommendations

Deep learning approaches have been shown to be effective for transfer learning applications. Elkahky et al.[55] propose a multi-view deep learning model[56] for cross-domain recommendations. The multi-view model jointly learns from item features and user features over multiple domains, and represents items and users in the same semantic space. Recent applications such as Neural Collaborative Filtering (NCF) aim to learn the interactions, and the non-linear relationships, between users and items, using Deep learning models[57]. Wang et al.[58] extend the NCF model to a cross-domain social recommendations setting, where, user-item interactions in an ecommerce domain, and user-user interactions in social network domains are bridged to provide cross-domain social recommendations. This work leverages the bridge user, i.e., the common users across social network and ecommerce domain, to make recommendations across domains using a neural social collaborative ranking model. Cross-domain Content-boosted Collaborative Filtering neural Network (CCCFNet) proposed by Lian et al.[59] is a dual model that combines collaborative filtering and content-based filtering. CCCFNet learns representations of content information, by splitting each network of the dual net into two components: 1) a collaborative filtering factor, that captures the user and item latent factors and 2)a content information factor that captures the user preferences over item features. Cross-domain recommendations are performed by building a multi-view neural framework on top of the dual model.

2.3 Recommender Systems Evaluation, Metrics, and Measures

In order to measure the effectiveness of various aspects of recommender approaches, a diverse set of metrics have been developed in past work. Recommender systems can be evaluated in an online setting and an offline setting. In an online setting, a recommender system is trained on historical data and effectiveness measure on live traffic. In an offline setting, a test data set is created from historical data and the recommender system is evaluated on the test data set. The test data set is considered as a proxy for future user behavior. In practice, recommender systems are evaluated first in an offline setting and then in an online setting.

Offline Evaluation - Data Preparation: Offline data is sampled from historical user-item interactions, this data that is used to train and evaluate machine learning algorithms is colloquially known as *gold data* or *ground truth*. Offline data is then usually split into three sets: 1) the training set 2) the validation set 3) the testing set. The machine learning models are trained over the training data, the hyper parameter tuning and model selection is performed over the validation set and final offline metrics are measured over the test dataset. When there is no hyper-parameter tuning and model selection, we do not need a validation set.

Preparing ground truth splits: Usually ground truth data is uniformly sampled and proportioned in a pre-determined way into the three data splits. An alternative and more practical way is to split the data temporally. This simulates the real-world scenario of when the models are learnt over historical data and used in the application on a continuation of the data[60]. Temporal splitting of data has been now widely accepted as a standard practice in the recommender systems community.

2.4 Taxonomy of Recommender Systems Metrics

Table 2.1 contains some of the wide variety of metrics used to measure the effectiveness of recommender systems approaches. The metrics can be broadly classified into: 1) error 2) accuracy 3) beyond accuracy metrics. Error metrics are basically rating prediction approaches that are trying to minimize the predicted user-item preference error. Accuracy metrics are used to measure item recommendation approaches, where we are trying to measure the efficacy of top-K recommended items in the recommended list. Some accuracy metrics such as Precision, Recall are trying to measure the ability of Recommender approaches to identify relevant items for the users, whereas other metrics such as normalized discounted cumulative gain (NDCG), mean average precision (MAP)[61] take the ranked ordering of items into consideration. *Beyond accuracy measures* are macro metrics that are trying to measure aspects such as novelty, serendipity, diversity etc., which indicate to some extent the health of the

Measure	Abbreviation	Type	Ranking-aware	Task
Mean absolute error	MAE	error/accuracy	No	Rating Prediction
Root mean square error	RMSE	error/accuracy	No	Rating Prediction
Precision at K	P@K	accuracy	No	Recommendation
R-Precision	R-Prec	accuracy	No	Recommendation
Recall at K	R@K	accuracy	No	Recommendation
Mean average precision at K	MAP@K	accuracy	Yes	Recommendation
Clicks at Increment K	Clicks%K	accuracy	No	Recommendation
Normalized discounted cumulative gain	NDCG	accuracy	Yes	Recommendation
Half-life utility	HLU	accuracy	Yes	Recommendation
Mean percentile rank	MPR	accuracy	Yes	Recommendation
Spread	—	beyond accuracy	No	—
Coverage	—	beyond accuracy	No	—
Novelty	—	beyond accuracy	No	—
Serendipity	—	beyond accuracy	No	—
Diversity	—	beyond accuracy	No	—

Table 2.1: *Popular Recommender Systems Evaluation Metrics.*

recommender system.

2.4.1 Error and Accuracy metrics

Mean absolute error (MAE) is used to measure the rating prediction accuracy of Recommender approaches. MAE computes the average absolute deviation between the predicted ratings and the actual ratings[62]. MAE is computed as follows:

$$MAE = \frac{1}{|T|} \sum_{r_{u,i} \in T} |r_{u,i} - \hat{r}_{u,i}| \tag{2.2}$$

where $r_{u,i}$ and $\hat{r}_{u,i}$ are the actual and the predicted ratings of item i for user u. MAE sums over the absolute prediction errors for all ratings in a test set T. MAE assumes that there is a linear cost associated with rating prediction error. For example, a rating prediction of 3 for an item when ground truth is 4 is twice as bad as a 3.5 prediction. A Recommendation approach with a lower MAE score is better when comparing MAE scores of various approaches, and corresponds to a superior recommendation performance.

Root mean square error (RMSE) is similar MAE and is computed as:

$$RMSE = \sqrt{\frac{1}{|T|} \sum_{r_{u,i} \in T} (r_{u,i} - \hat{r}_{u,i})^2} \tag{2.3}$$

RMSE fixes the linear cost drawback of MAE by squaring the prediction error, this penalizes larger differences between predicted and true ratings more than smaller ones. A Recommendation approach with a lower RMSE score is better when comparing RMSE scores of various approaches, and correspond to a superior recommendation performance.

The major drawback with MAE and RMSE is that they treat all prediction errors equally. For example, a true rating of 3 and predicted rating of 2 is treated the same way as true rating of 4 and predicted rating of 5. From a practical perspective, this is not the case as rating prediction accuracy matters only for items that are relevant to the user. Accuracy metrics try to address this drawback of error metrics.

Precision at** K **(P@K) is a widely used metric in the information retrieval domain that measures the accuracy of the system in identifying relevant items[61]. $P@K$ is a metric designed for binary relevance judgments, i.e. an item is either relevant or not relevant to the user. If the user-item preference information is available in the form of ratings, this is binarized, for example on a 5 point rating scale, ratings greater than or equal to 4 are considered to be relevant.

For each user u, $P_u@K$ is computed as follows:

$$P_u@K = \frac{|L_u \cap \hat{L}_u|}{|\hat{L}_u|} \tag{2.4}$$

where L_u is the set of relevant items for user u in the test set T and $\hat{L}_u$ is the recommended set containing the K items in T with the highest predicted ratings for the user u. The overall $P@K$ is then computed by averaging $P_u@K$ values for all users in the test set. A Recommendation approach with a higher P@K score is better when comparing P@K scores of various approaches, and corresponds to a superior recommendation performance.

R-Precision (R-Prec) is defined as the number of retrieved relevant items divided by the number of known relevant items[63]. For a user u, $R - Prec_u$ is defined as:

$$R - Prec_u = \frac{|L_u \cap R_{1:|L_u|}|}{|L_u|} \tag{2.5}$$

L_u is the set of relevant items of user u in the test set T and $R_{1:|L_u|}$ is the set of top retrieved items, where the retrieved list length is the same as L_u. R-Prec metric is averaged across all users. This metric rewards total number of retrieved relevant items (regardless of order) . R-Prec is a variation of P@K where the precision is measured over the size of the ground truth recall set and not a fixed K. R-Prec thus captures the precision when size of the relevant set of items varies. A Recommendation approach with a higher R-Prec score is better when comparing R-prec scores of various approaches, and corresponds to a superior recommendation performance.

Mean average precision at K (MAP@K) is a rank-based metric that computes the overall precision of a Recommender algorithm at various lengths of recommendation lists[61]. MAP is defined as the arithmetic mean of the average precision over the entire set of users in the test set. Average precision for the top K recommendations ($AP@K$) is defined as follows:

$$AP@K = \frac{1}{N} \sum_{i=1}^{K} P@i \cdot rel(i) \tag{2.6}$$

where $rel(i)$ is an indicator signaling if the i^{th} recommended item is relevant, i.e. $rel(i) = 1$, or not, i.e. $rel(i) = 0$; N is the total number of relevant items. For two lists with same recall MAP provides a higher recall to the list with relevant items higher in the ranked set of items, thus addressing a drawback of P@K. MAP implicitly incorporates recall, because it also considers the relevant items not in the recommendation list. A Recommendation approach with a higher MAP score is better when comparing MAP scores of various approaches, and corresponds to a superior recommendation performance.

In the above definition of AP@K, if the length of the list to be recommended is too small when compared to the number of relevant items, the average precision scores in instances with more relevant items are affected. Thus the previous definition of AP@K is biased towards instances with smaller relevant items[64]. This is fixed in a modified version of Average precision as:

$$AP@K = \frac{1}{\min(K, N)} \sum_{i=1}^{K} P@i \cdot rel(k) \tag{2.7}$$

where N is the total number of relevant items and K is the size of recommendation list.

Recall at K ***(R@K)*** Recall is defined as the algorithm's ability to identify relevant items. A good recommender system should ideally have high precision and high recall, i.e., it identifies more relevant items and places relevant items at the top of the list. For most recommender applications, Recall is not an important metric, because in most cases the system is not trying to recommend an exhaustive set of items, but a small set of highly relevant items. For example in the case of web search, most users never scroll past the first page, so the objective here is to primarily identify a small set of web pages that meet the user needs and place that at the top of the search results. For certain domains such as Legal, Patent law where exploratory search is of essence, recall is an important metric. In domains such as movie recommendation, book recommendation etc. where the user has a limited bandwidth, Recall is not an important metric. For a user u, $R_u@K$ is defined as:

$$R_u@K = \frac{|L_u \cap \hat{L}_u|}{|L_u|} \tag{2.8}$$

where L_u is the set of relevant items of user u in the test set T and $\hat{L}_u$ denotes the recommended set containing the K items in T with the highest predicted ratings for the user u. The overall $R@K$ is calculated by averaging $R_u@K$ values for all the users in the test set. A Recommendation approach with a higher Recall score is better when comparing recall scores of various approaches, and corresponds to a superior recommendation performance.

Clicks at Increment K (Clicks%K) is a user interface driven metric, where a user needs to refresh a recommended set of items in order to view a fresh set of items. The user need is satisfied once a user identifies a relevant item. Thus the number of *refresh/clicks* needed before the user need is satisfied defined as *clicks*. Clicks%K is defined as:

$$Clicks\%K = min\{\lfloor(r-1)/K\rfloor, D\} \tag{2.9}$$

where K is the size of the set of recommended items that can be refreshed, D is the maximum number of refreshes allowed by the interface.

Normalized discounted cumulative gain (NDCG) is an Information Retrieval measure for capturing the ranking quality of search results[65]. NDCG nowadays is also used for evaluating the ranking quality of recommender systems[66]. For rating predictions task, assuming that the recommendations are sorted by predicted rating values for each user u. The Discounted cumulative gain for u (DCG_u) is defined as:

$$DCG_u = \sum_{i=1}^{N} \frac{r_{u,i}}{log_2(i+1)} \tag{2.10}$$

where $r_{u,i}$ is the true rating (as found in test set T) for the item ranked at position i for user u, and N is the length of the recommendation list.

Since the rating distribution depends on the user's behavior, the DCG values for different users are not directly comparable, the cumulative gain for each user is normalized. This is captured by computing the ideal DCG for user u ($IDCG_u$) , which is the DCG_u value for the best possible ranking, obtained by ordering the items by true ratings in descending order.

Normalized discounted cumulative gain for user u is then calculated as:

$$NDCG_u = \frac{DCG_u}{IDCG_u} \tag{2.11}$$

The overall normalized discounted cumulative gain *NDCG* is computed by averaging $NDCG_u$ over the entire set of users. A Recommendation approach with a higher NDCG score is better when comparing NDCG scores of various approaches, and corresponds to a superior recommendation performance, and correspond to a superior recommendation performance.

Half-life utility (HLU) : In most recommender applications, the utility of an item in a recommendation list decays as we go lower in the list i.e. in a ranked list, higher ranked positions have a higher value when compared to a lower ranked positions. So it is important to have higher ranked items at the top. HLU measures the utility of a list for a user u with the assumption that the likelihood of selecting a recommended item in the list exponentially decays with the item's position in the ranking[67]. HLU for user u is defined as:

$$HLU_u = \sum_{i=1}^{N} \frac{\max\left(r_{u,i} - d, 0\right)}{2^{(rank_{u,i}-1)/(h-1)}} \tag{2.12}$$

where $r_{u,i}$ and $rank_{u,i}$ are the rating and the rank of item i for user u, respectively, where the recommendation list is of length N; d represents a default rating (e.g., average rating) and h is the half-time, calculated as the rank of an item in the list, such that the user can eventually consume it/view it with a 50% chance. As in the case of NDCG, HLU_u can be further normalized by the maximum utility, and the final HLU is the average over the half-time utilities obtained for all users in the test set. A Recommendation approach with a higher HLU score is better when comparing HLU scores of various approaches, and corresponds to a superior recommendation performance.

Mean percentile rank (MPR) is computed as the average of the *percentile rank* for each test item within the ranked list of recommended items for each user[68]. The percentile rank of an item is the percentage of items whose position in the recommendation list is equal to or lower than the position of the item itself. The percentile rank PR_u for user u is defined as:

$$PR_u = \frac{\sum\limits_{r_{u,i} \in T} r_{u,i} \cdot rank_{u,i}}{\sum\limits_{r_{u,i} \in T} r_{u,i}} \tag{2.13}$$

where $r_{u,i}$ is the true rating of the item in the test set T for item i rated by user u and $rank_{u,i}$ is the percentile rank of item i within the ordered list of recommendations for user u. MPR is calculated over all the users as the arithmetic mean of the individual PR_u values. A randomly ordered recommendation list has an expected MPR value of 50%. A Recommendation approach with a smaller MPR score is better when comparing MPR scores of various approaches, and corresponds to a superior recommendation performance.

2.4.2 Beyond Accuracy Metrics

Beyond accuracy metrics are *macro-level* quantitative evaluation metrics and are intended to capture the *health* of a recommender system[69]. One draw back of accuracy metrics oriented recommender systems is their proclivity to recommend popular items. This leads to low exploration of the item set by the user and leading to *filter bubbles*[7].

Spread is defined as the entropy of the distribution of the items recommended to the users in the test set. Spread is intended to capture how well the recommender algorithm can spread its attention across a larger set of items[70] and is defined as:

$$spread = -\sum_{i \in I} P(i) \log P(i) \tag{2.14}$$

where I represents the entirety of items in the data set and $P(i) = count(i)/\sum_{i' \in I} count(i')$, such that $count(i)$ denotes the frequency of item i showed in the recommendation lists. Thus more popular items have a higher $P(i)$. Spread tends to measure the bias of a system towards popular items.

Coverage of a recommender system is defined as the proportion of items over which the system is capable of generating accurate recommendations[62]:

$$coverage = \frac{|\hat{T}|}{|T|} \tag{2.15}$$

where $|T|$ is the size of the test set and $|\hat{T}|$ is the number of ratings in T for which the system can predict a value. In the scenarios of item cold-start, if a recommender system has low coverage, then new items are never surfaced to the user. Coverage measures the ability of a user to discover new items in a system.

Serendipity takes *relevant and surprising* recommendations into consideration while evaluating recommender systems. There isn't an agreement on how to measure the serendipity of a recommender system or how to define surprising recommendations.

Serendipity of a recommendation list L_u provided to a user u can be defined as:

$$serendipity(L_u) = \frac{\left|L_u^{unexp} \cap L_u^{useful}\right|}{|L_u|} \tag{2.16}$$

where L_u^{unexp} and L_u^{useful} denotes subsets of L that are unexpected and useful to the user. Usefulness of an item is obtained by explicitly asking users or taking user ratings as proxy[69]. The unexpectedness of an item is a measure of distance from expected items or the items already rated by the user.

Diversity measures the extent to which recommended items are different from each other, where difference can be based on any explicit or implicit features associated with the item content, side information. Similar to serendipity, diversity can be defined in several ways and there is no general consensus on what constitutes diversity. One way to measure of diversity is to compute pairwise distance between all items in a recommendation set[71]. The diversity of a recommendation list L is calculated as follows:

$$diversity(L) = \frac{\sum_{i \in L} \sum_{j \in L \setminus i} dist_{i,j}}{|L| \cdot (|L| - 1)} \tag{2.17}$$

where $dist_{i,j}$ is a distance function defined between pairs of items.

Chapter 3

Representation Learning in Networks

"You shall know a word by the company it keeps."

- J. R. Firth, *A synopsis of linguistic theory*

The goal of representation learning in networks is to *embed* nodes or sub-graphs of a network by learning a mapping to a lower-dimensional vector space while simultaneously preserving some properties of the nodes within the network. Representation learning can be viewed as a feature engineering approach over information networks, so that they can be used to construct features for machine learning algorithms.

3.1 Embeddings and Distributed Representations

In natural language processing (NLP), word embeddings/embeddings refer to a set of language modeling and feature learning techniques where words in the vocabulary are mapped to vectors of real numbers. Here, a sparse representation of words in a higher dimension is converted to a low-dimensional latent representation. Conceptually, an embedding can be viewed as a mapping from a space with one dimension per word to a continuous vector space with a much lower dimension such that the mapping preserves the linguistic properties of the words. The term *embeddings* is usually used to refer to dense representations of words or entities in a low-dimensional vector space. Other interchangeable terms for entity embeddings are entity vectors and *distributed representations*.

The distributional hypothesis in linguistics states that words that occur in the same contexts tend to have similar meanings[72] and is the basis for statistical semantics. The main idea behind this hypothesis is that there is a correlation between distributional similarity of words and meaning of words, and knowing about the former helps us to estimate the latter. Weaver contends that word sense disambiguation (WSD) in machine translation should be based on the co-occurrence frequency of the context words near a given target word[73]. Most modern approaches in linguistics, and machine translation, such as language modelling[1], words sense disambiguation, are based on the distributional hypothesis.

In NLP, vector space models have been used in distributional semantics to capture the linguistic association of words in a vocabulary[74]. Bengio et al. combined a neural network and a statistical language model to learn representations that they coined as word embeddings[1]. Collobert and Weston[75] showed the effectiveness of pre-trained word embeddings on NLP tasks. They refined the neural network architecture and proved that word embeddings can improve downstream NLP tasks such as part-of-speech tagging, chunking, named entity recognition (NER) , semantic role labeling (SRL) , identifying semantically similar words, and language modeling. This work demonstrated the utility of word embeddings in NLP tasks and showed how representation learning can be performed using neural network architectures. *Word2Vec*[4], a language modelling approach by Mikolov et al., popularized the word embedding models and led to the wide usage of distributional representations in a wide variety of machine learning applications. The structure of the neural network in *Word2Vec* made it viable to compute embeddings by consuming large amounts of data. *Word2vec* proposes two approaches to learn distributed representations: 1)continuous bag-of-words (CBOW), and 2) Skipgram. Pennigton and Socher propose *Global Vectors* (*GloVe*)[76] model, an alternative approach to calculate word representations, by reformulating *Word2Vec* as a special case of factorization of word co-occurence matrices.

[1]A statistical language model is a probability distribution over sequences of words. Given such a sequence, say of length T, the language model assigns a probability to the whole sequence.

The concept of embeddings has been extended beyond word representations and NLP to other areas such as web search, information retrieval and recommender systems. Just as one can train word embeddings by treating a sequence of words in a sentence as context, one can do the same by treating sequence of user actions as context. Embedding approaches have been successfully used to learn representations of search queries, recommended items, items that were clicked or purchased and ads that were clicked[77;78].

3.2 Language Modelling

In NLP, language models compute the probability of a word w_t given its previous $n-1$ words, i.e. $p(w_t \mid w_{t-1}, \cdots w_{t-n+1})$. Applying the chain rule, we can approximate the probability of a whole sentence or document by the product of the probabilities of each word given its n previous words as:

$$p(w_1, \cdots, w_T) = \prod_i p(w_i \mid w_{i-1}, \cdots, w_{i-n+1}) \tag{3.1}$$

where w_i is word at index i in the document. In n-gram-based language models, we can calculate a word's probability based on the frequencies of its constituent n-grams as:

$$p(w_t \mid w_{t-1}, \cdots, w_{t-n+1}) = \frac{count(w_{t-n+1}, \cdots, w_{t-1}, w_t)}{count(w_{t-n+1}, \cdots, w_{t-1})} \tag{3.2}$$

3.2.1 Neural Network-Based Language Models

Bengio et al. propose using *softmax*[1] to build a neural network-based language model as:

$$p(w_t \mid w_{t-1}, \cdots, w_{t-n+1}) = \frac{\exp(h^\top v'_{w_t})}{\sum_{w_i \in V} \exp(h^\top v'_{w_i})} \tag{3.3}$$

where h is the output vector of the penultimate network layer - the hidden layer, while v'_w is the output embedding of word w. Bengio et al. in this work introduce the concept of word embedding - a real-valued word feature vector in $\mathbb{R}$. This model maximizes the following

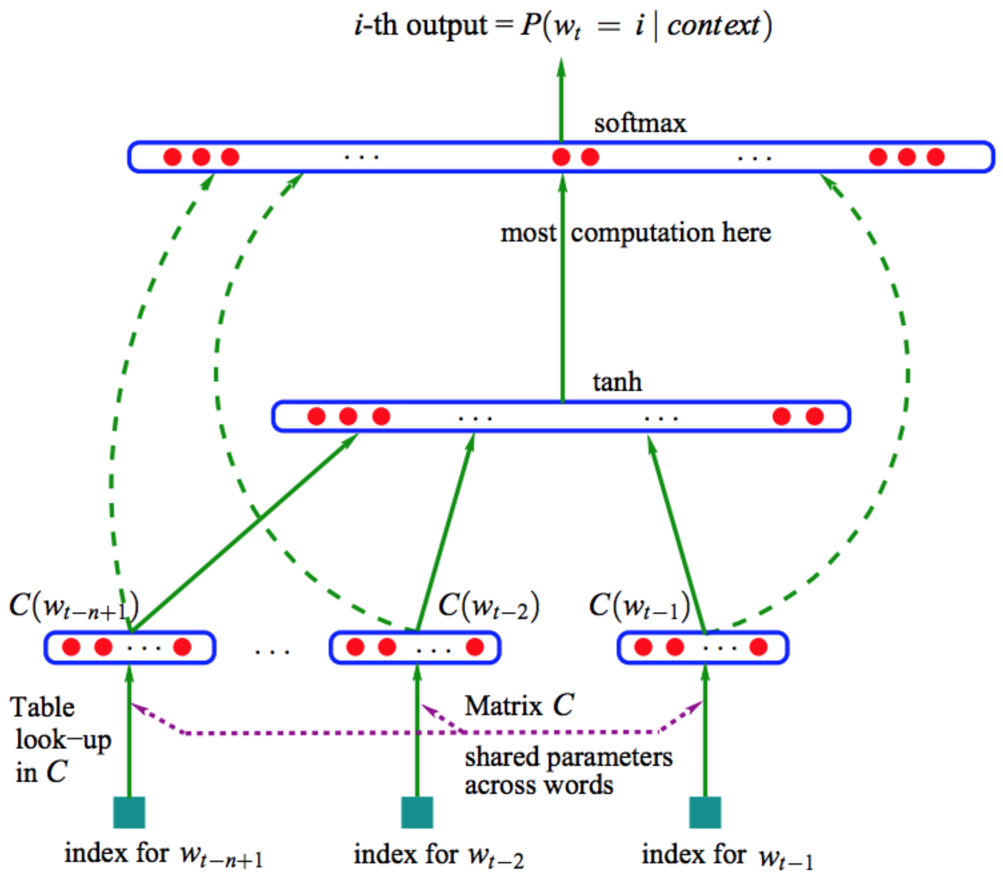


Figure 3.1: *Neural Language Model, Bengio et al.*[1]

objective:

$$J_\theta = \frac{1}{T}\sum_{t=1}^{T} \log f(w_t, w_{t-1}, \cdots, w_{t-n+1}) \tag{3.4}$$

where $f(w_t, w_{t-1}, \cdots, w_{t-n+1})$ is the output of the model, i.e. softmax layer computes the probability $p(w_t \mid w_{t-1}, \cdots, w_{t-n+1})$, where n is the number of previous words fed into the model.

The main components of this neural model are:

1. Embedding Layer: a layer that generates word embeddings by multiplying an index vector with a word embedding matrix

2. Intermediate Layer (s) : one or more layers that produce an intermediate representation of the input

3. Softmax Layer: the final layer that produces a probability distribution over words in vocabulary V

The authors note that the neural language model proposed in this approach has high cost of computation,and identify the bottleneck as the softmax layer. The cost of computing the softmax is proportional to the number of words in the vocabulary V.

Collobert and Weston showed that the word embeddings trained on a sufficiently large dataset using a neural language model captures syntactic and semantic meaning[75]. They also showed how these embeddings can be used to improve the performance of popular NLP tasks. In this model the authors try to avoid the computational bottleneck encountered in the Neural language model proposed by Bengio et al. Instead of computing the expensive softmax as their objective function, Collobert and Weston train a neural network to output a higher score f_θ for a correct word sequence than for an incorrect one. This work use the following pairwise ranking criterion as their objective:

$$J_\theta = \sum_{x \in X} \sum_{w \in V} \max\{0, 1 - f_\theta(x) + f_\theta(x^{(w)})\} \tag{3.5}$$

Because of the absence of softmax, this model is faster to compute than the neural language model proposed by Bengio et al. But the fully connected hidden layer acts as a computational bottleneck.

3.2.2 Word2Vec

In *Word2Vec* Mikolov et al. propose two different architectures and learning strategies for learning word embeddings that are computationally less expensive than previous models[4]. In a follow up paper, Mikolov et al. improve upon the previous *Word2Vec* models by employing additional strategies such as negative sampling, noise contrastive estimation and adaptive sampling to improve training speed, efficiency and accuracy of the neural language model[79].

The architecture of the *Word2Vec* model offers two main advantages over the previously discussed neural language models:

1. *Word2Vec* does not have a hidden layer. As this was a computationally expensive layer in the Bengio and Collobert models, not having this in the model makes training efficient.

2. The *Word2Vec* model allows the language model to take additional context into account. More the context available during the training process, the better the accuracy of the language model.

Continuous Bag-Of-Words (CBOW)

During the training process of traditional language models and neural language models, the model is supplied only the previous words in a sentence and is evaluated on the model's ability to predict the next word in the sentence, Mikolov et al. realized that in order to generate accurate word embeddings, the model does not have to limit itself to this set up and should be able to use words that precede and succeed a target word. They call this version of the *Word2Vec* model as the continuous bag-of-words (CBOW) model. In the CBOW model, the order of the words in the context window is not important, hence the reference to bag-of-words.

The objective function of CBOW is:

$$J_\theta = \frac{1}{T}\sum_{t=1}^{T} \log p(w_t \mid w_{t-n}, \cdots, w_{t-1}, w_{t+1}, \cdots, w_{t+n}) \tag{3.6}$$

This is quite similar to the traditional language model objective. Here the model gets a context window of words around the target word w_t as the input and the model is judged on the ability to predict the target word.

Skipgram

In CBOW we use the surrounding words, i.e., the context of a target word to predict the target word. Skipgram inverts this criteria by specifying the objective as, given a target word, we try to predict the context/surrounding words of the target word. Thus the Skipgram

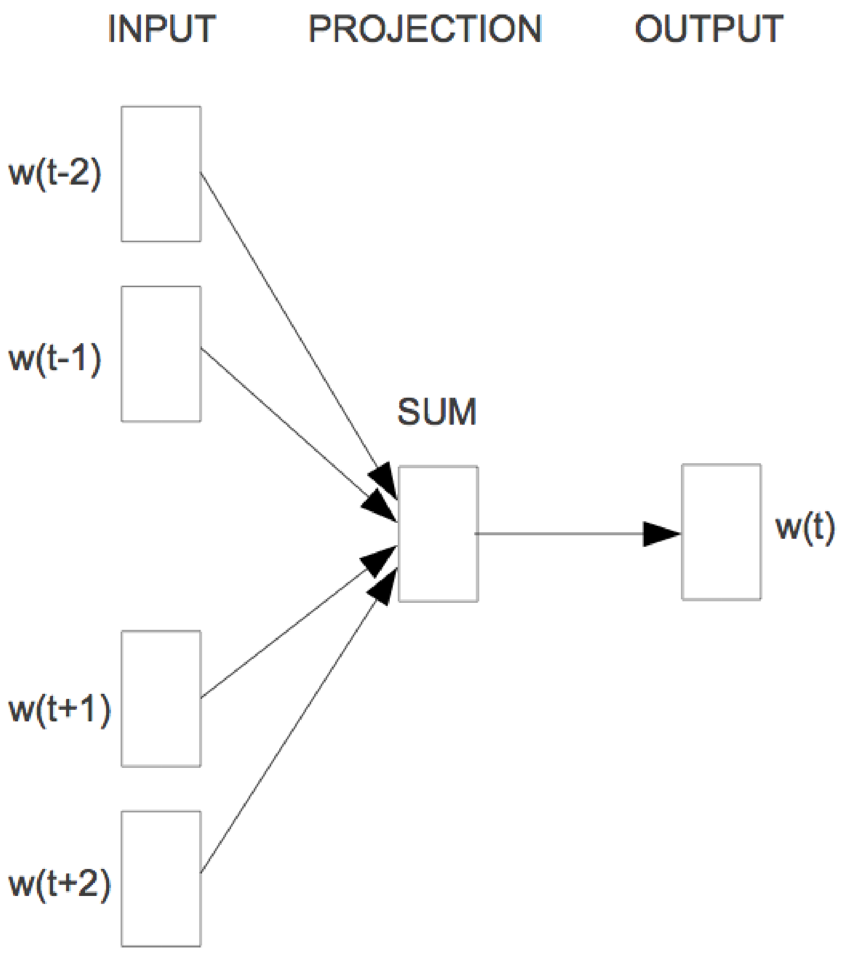


Figure 3.2: *Continuous Bag-Of-Words (CBOW) architecture*

objective sums the log probabilities of the surrounding n words to the left and to the right of the target word w_t in the context window as:

$$J_\theta = \frac{1}{T}\sum_{t=1}^{T}\sum_{-n\leq j\leq n,\neq 0} \log p(w_{t+j} \mid w_t) \tag{3.7}$$

where w_{t+j} is a word surrounding the middle word/target word in the context window. Softmax is used in Skipgram to compute the probabilities.

As there is no hidden layer in the Skipgram neural model, two different embeddings matrices are maintained for input embeddings and output embeddings. The softmax function

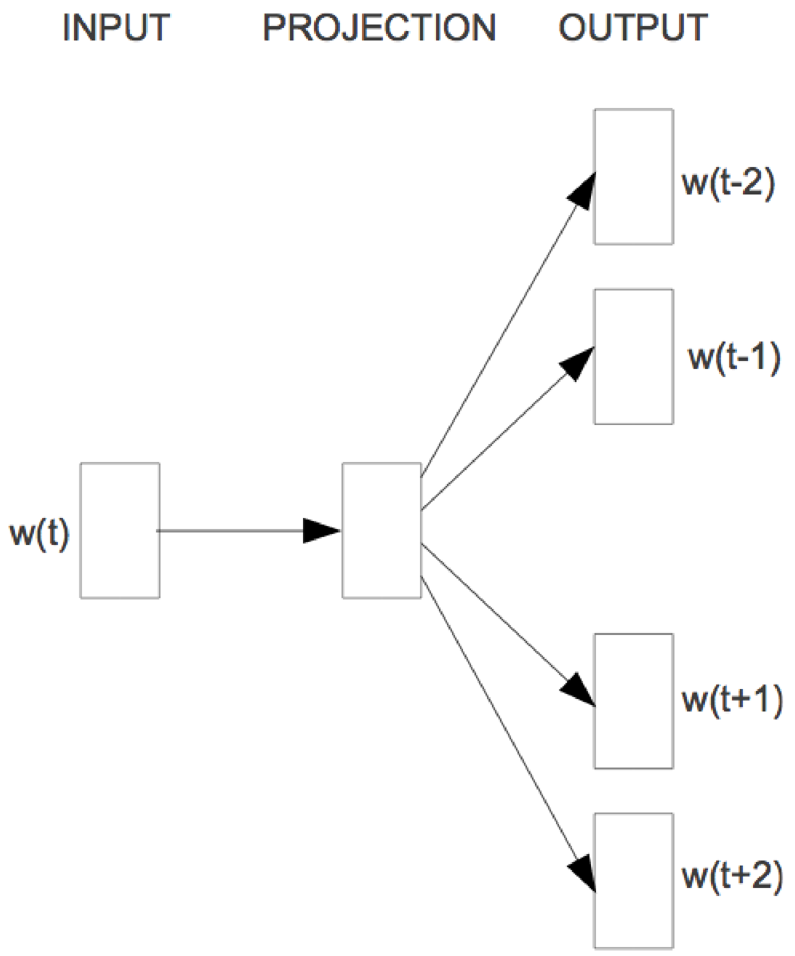


Figure 3.3: *Skipgram architecture*

for Skipgram is given as:

$$p(w_{t+j} \mid w_t) = \frac{\exp(v_{w_t}^{\top} v'_{w_{t+j}})}{\sum_{w_i \in V} \exp(v_{w_t}^{\top} v'_{w_i})} \tag{3.8}$$

where v_w is the input embedding of word w and v'_w is the output embedding of the word w. As calculating the softmax is expensive, efficient approaches such as hierarchical softmax and differentiated softmax have been used to make the Skipgram model efficient.

3.3 Representation Learning in Networks

Representation learning in networks aims to learn latent, low-dimensional, semantic representations of nodes in an information network, while preserving properties such as network topology structure, vertex content, node neighborhood, etc. Once representations are learnt over the network, network analytic tasks such as link prediction, predicting missing properties of nodes can be efficiently carried out by using ML algorithms in the new semantic space. An information network can consist of various types of nodes and multiple types of relations between the nodes. The edges in the information network can be weighted/unweighted, and directed/un-directed. In such a setting, the main challenge of representation learning in networks is that there is no straightforward way to encode the high-dimensional, non-euclidean information about graph structure into a feature vector.

An Information Network is a graph defined as $G = (V, E, X, Y)$, where V denotes a set of vertices, and $|V|$ denotes the number of vertices in network G. $E \subseteq (V \times V)$ denotes a set of edges connecting the vertices. $X \in \mathbb{R}^{|V| \times m}$ is the vertex attribute matrix, where m is the number of attributes, and the element X_{ij} is the value of the ith vertex on the j-th attribute. $Y \in \mathbb{R}^{|V| \times |\mathcal{Y}|}$ is the vertex label matrix with $\mathcal{Y}$ being a set of labels. If the ith vertex has the kth label, the element $Y_{ik} = 1$; otherwise, $Y_{ik} = -1$.

First-order Proximity in an information network captures the local pairwise proximity between two connected vertices. It is designed to capture neighborhood of nodes in an information network.

Second-order Proximity in an information network is the number of common neighbors between pairs of nodes and is designed to capture the local structure of the network. **High-order proximity** is similar to second order proximity, and is defined as number of common nodes between two pairs of nodes within a k-step distance from the nodes and captures global structure of the network.

Representation learning in networks in an information network $G = (V, E, X, Y)$, is defined as learning a mapping function $f : v \longmapsto r_v \in \mathbb{R}^d$, where r_v is the learned vector representation of vertex v, and d is the dimension of the learned representation. The transformation f preserves some original property of the network.

Based on the graph structure, input data, and application, representation learning approaches can be broadly classified into two categories: 1) Unsupervised representation learning 2) Semi-supervised representation learning.

Unsupervised representation learning: In this setting, the vertices and the edges do not have any labels associated with them. The goal is to learn a representation over the network and the embeddings are used subsequently in down stream tasks.

Semi-supervised/supervised representation learning: In this setting, some of the vertices in the network are labeled. Semi-supervised representation learning is used to take advantage of vertex labels to generate better embeddings as more information regarding the network is available via node labels.

Based on the techniques, representation learning methods can be broadly classified into (1) Random walk-based methods, and (2) Edge modeling-based methods.

Random walk-based methods: Random walk-based approaches have shown to be highly effective for representation learning. In these methods, sequences of truncated random walks are generated over the information network and these sequences are then used to generate embeddings of nodes in the network. As in language modeling approaches to learning word representations[79;80], vertex representations are learnt by using random walks. DeepWalk[81] was one of the first approaches to demonstrate the random walk-based representation learning. Node2vec[82] further extends this by adopting a biased random walk strategy to capture more flexible network structure.

Edge modeling-based methods: Edge modeling-based methods directly learn vertex representations from the edges between the vertices. LINE[83] models a joint probability distribution and a conditional probability distribution, on connected vertices, thus having the ability to preserve first-order and second-order proximity. Linked Document Embedding (LDE)[?] learns the representations of linked documents by modeling the document-document relationships by maximizing the conditional probability between connected documents. LDE combines link and label information with content information to learn document representations for classification. GraphGAN[84] adopts Generative Adversarial Nets (GAN)[85] to accurately model the vertex connectivity probability and learn distributed representations of vertices in the network.

3.3.1 DeepWalk

DeepWalk[81] utilizes the idea of the Skipgram model[79;80] of Word2Vec. In the Skipgram approach for language modeling, we learn latent representations of words by using context windows of words in sentences; we can apply the same reasoning to sequence of nodes in the network, where the sequences are obtained by performing random walks over the network. Given a random walk sequence with length L, $\{v_1, v_2, \cdots, v_L\}$, similar as in Skipgram, DeepWalk learns the representation of vertex v_i by using it to predict its context vertices in the random walk window:

$$\min_f \; -\log \Pr(\{v_{i-t}, \cdots, v_{i+t}\} \setminus v_i | f(v_i)), \tag{3.9}$$

where $\{v_{i-t}, \cdots, v_{i+t}\} \setminus v_i$ are the context vertices of vertex v_i within t window size. Making conditional independence assumption, the probability $\Pr(\{v_{i-t}, \cdots, v_{i+t}\} \setminus v_i | f(v_i))$ is approximated as

$$\Pr\left(\{v_{i-t}, \cdots, v_{i+t}\} \setminus v_i | f(v_i)\right) = \prod_{j=i-t, j\neq i}^{i+t} \Pr(v_j | f(v_i)). \tag{3.10}$$

DeepWalk uses the hierarchical softmax technique to compute the normalizing factor, where a binary-tree structure is used to accelerate the computation of the softmax. Because of the way random walks are generated, DeepWalk captures node embedding such that the local neighborhood of the vertices are preserved. Because context vertices in random walk sequences describe neighborhood structure, vertices sharing similar neighbors will be closer to each other in the embedding space, i.e., the second-order and higher-order proximity are preserved (depending on length of random walks and the size of the context window) .

3.3.2 Large-scale Information Network Embedding (LINE)

Unlike DeepWalk, which is based on random walks, LINE[83] learns vertex representations by explicitly modeling the first-order and second-order proximity. To preserve the first-order proximity between nodes, LINE minimizes the following objective:

$$O_1 = d(\hat{p}_1(\cdot,\cdot), p_1(\cdot,\cdot)). \tag{3.11}$$

For each vertex pair v_i and v_j with $(v_i, v_j) \in E$, $p_1(\cdot,\cdot)$ is the joint distribution modeled by their latent embeddings r_{v_i} and r_{v_j}. $\hat{p}_1(v_i, v_j)$ is the empirical distribution between them. $d(\cdot,\cdot)$ is the distance between two distributions.

To preserve the second-order proximity, LINE minimizes the following objective:

$$O_2 = \sum_{v_i \in V} \lambda_i d(\hat{p}_2(\cdot|v_i), p_2(\cdot|v_i)), \tag{3.12}$$

where $p_2(\cdot|v_i)$ is the context conditional distribution for each $v_i \in V$ modeled by vertex embeddings, $\hat{p}_2(\cdot|v_i)$ is the empirical conditional distribution and λ_i is the prestige of vertex v_i. Here, vertex context is determined by its neighbors, i.e., for each v_j, v_j is v_i's context, if and only if $(v_i, v_j) \in E$.

By minimizing these two objectives, LINE learns two kinds of vertex representations that preserve the first-order and second-order proximity, and takes their concatenation as the final vertex representation. Both the first-order and second-order objectives are optimized using loss functions derived from the KL-divergence metric. Thus, LINE explicitly factorizes first- and second-order similarities, instead of combining them in fixed-length random walks.

3.3.3 Node2vec

Node2vec[82] uses biased random walks as a neighborhood sampling strategy to generate sequences of nodes.This strategy smoothly interpolates between two extreme sampling strategies, Breadth-first Sampling (BFS) and Depth-first Sampling (DFS) . The biased random walk approach used in Node2vec has shown to better preserve both the second-order and high-order proximity.

Using the Skipgram architecture, and given the set of neighbor vertices $N(v_i)$ generated by biased random walk as the input, Node2vec learns the vertex representation $f(v_i)$ by optimizing the occurrence probability of neighbor vertices $N(v_i)$ conditioned on the representation of vertex v_i, $f(v_i)$:

$$\max_f \sum_{v_i \in V} \log \Pr(N(v_i)|f(v_i)). \tag{3.13}$$

Instead of normalizing over the full vertex set as in DeepWalk, Node2vec approximates the normalizing factor using a set of random negative samples. This is known as Skipgram with negative sampling[80]. The main difference between DeepWalk and Node2vec is that, while DeepWalk uses simple unbiased random walks over the graph to obtain vertex embeddings, Node2vec allows for a flexible definition of random walks.

The node embeddings learnt over the information networks can now be used as features in downstream machine learning and data mining tasks such as vertex classification, link prediction, clustering, and recommendation.

3.3.4 Deep Learning Approaches for Representation Learning in Information Networks

This section summarizes some of the deep learning-based approaches for representation learning in information networks. Deep learning-based methods have the ability to capture non-linear associations in information networks. Deep learning techniques such as encoder-decoder architectures, auto-encoders[86] have been used to learn vertex representations, and network representation learning. Cao et al. propose *Deep Networks for Graph Representations*[87] by applying *stacked denoising autoencoders* (SDAE)[86] on the high-dimensional matrix representations to learn node embeddings. This uses the random surfing model from PageRank[88] to capture contextual relatedness between each pair of nodes, and represents them as a $|V|$-dimensional vertex representation X. Then a stacked denoising autoencoders (SDAE) is used to convert the node representations X to low-dimensional embeddings.

Structural Deep Network Embedding (SDNE)[89] applies a semi-supervised deep auto-encoder[90], where the unsupervised approach learns the second-order proximity to retain the global network structure, while the supervised approach uses the first-order proximity as supervised information to preserve the local network structure. SDNE learns the second-order proximity by reconstructing the $|V|$-dimensional adjacency matrix, which tries to minimize the following objective:

$$\mathcal{L}_{2nd} = \sum_{i=1}^{|V|} \|(r_{v_i}^{(0)} - \hat{r}_{v_i}^{(0)}) \odot b_i\|_2^2 \tag{3.14}$$

where $r_{v_i}^{(0)} = S_{i:}$ is the input representation, and $\hat{r}_{v_i}^{(0)}$ is the reconstructed representation. b_i is a weight vector used to minimize construction errors. This constitutes the unsupervised component of SDNE.

SDNE learns the first-order proximity by penalizing the distance between connected nodes in the embedding space using the following objective:

$$\mathcal{L}_{1st} = \sum_{i,j=1}^{|V|} S_{ij} \|r_{v_i}^{(K)} - r_{v_j}^{(K)}\|_2^2, \tag{3.15}$$

where K represents the number of hidden layers, and $r_{v_i}^{(K)}$ is the K-th layer representation of node v_i.

SDNE minimizes the joint objective function:

$$\mathcal{L} = \mathcal{L}_{2nd} + \alpha\mathcal{L}_{1st} + \nu\mathcal{L}_{reg}, \tag{3.16}$$

where $\mathcal{L}_{reg}$ is a regularization term to avoid overfitting.

Recently Generative Adversarial Networks (GAN)[85] have been used for representation learning in IN. GraphGAN[84] learns node embeddings by modeling the connectivity behavior using an adversarial learning approach. Similar to classical GANs, GraphGAN has two components: (1) A Generator $G(v|v_c)$, which fits the distribution of the vertices connected to v_c across V and generates the likely connected vertices, and (2) A Discriminator $D(v, v_c)$, which outputs a connecting probability for the vertex pair (v, v_c), to differentiate the vertex pairs generated by $G(v|v_c)$ from the ground truth. $G(v|v_c)$ and $D(v, v_c)$ are adversarial to each other, i.e., $G(v|v_c)$ learns to generates fake connected vertex pairs to fool $D(v, v_c)$, while $D(v, v_c)$ learns to increase its ability to distinguish the vertex pairs generated by $G(v|v_c)$ from the ground truth.

3.3.5 Pre-trained Embeddings and their Applications in Downstream Discovery Tasks

Collobert et al.[75;91] demonstrated the effectiveness of pre-trained embeddings learnt from a neural language model over large amounts of unlabeled data can be used to improve the performance of a wide array of supervised, and semi-supervised NLP tasks such as part of speech tagging, semantic role labeling, word sense disambiguation, named entity recognition, etc. The authors argue that neural networks are able to encode hidden representations that capture semantic and lexical features, which in turn can be used to improve down stream tasks. More recently Qi et al.[92] show how pre-trained embeddings are useful for machine translation tasks.

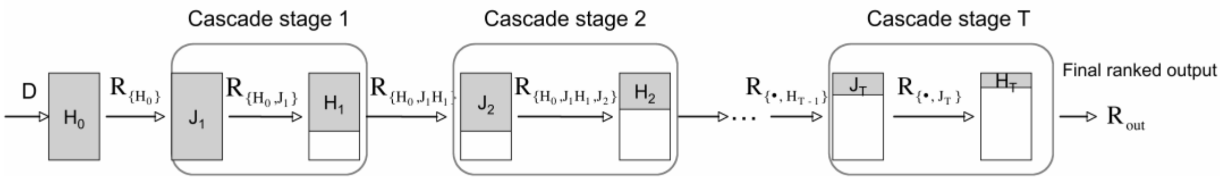


Figure 3.4: *A search engine architecture - iterative refinement of results*[2]*. An example cascade architecture. After an initial ranking function H_0, each stage consists of two sequential operations: J_t prunes the input ranked documents, then a local ranking function H_t refines the rank order of the retained documents. The new ranked list is passed to the next stage. The size of the shaded area denotes the size of the candidate documents. Subscripts for each ranked list denotes the sequence of actions applied.*

Using pre-trained embeddings in search

Fig. 3.4 illustrates a typical cascade architecture on large-scale search engines. Most modern search engines perform search in two stages. When a user enters a query, in the first stage, a large subset of documents that match the input query are obtained by a simple scoring function such as tf-idf or BM25[93]. In the second phase, the top-k (by score) of these matching documents are "re-ranked" using a complex ranking function that uses a richer set of features. The candidate item generation phase is similar to the first stage of a search engine, where give a ordered/unordered set of entities, we retrieve related entities that could be part of the list. Then the candidate items are ranked by the "context fit" with respect to the items in the list as well as the metadata associated with the list in order to obtain a high quality set of candidates. The second stage of search engine can have a *cascade/telescoping* architecture[2], where, the items in the ranked lists are iteratively refined multiple times. Thus, the first stage is tuned towards recall, while, the second stage is tuned towards ranking and personalization of search results. These stages have various machine learning models constructed with different objectives in mind. The machine learning models take a varying set of features into consideration, for tasks such as ranking, retrieval, personalization, query completion, ambiguous query resolution, etc. Thus, pre-trained embeddings can be used at various stages, and the significance of these embedding features can change at each model level. Zamani and Croft[94] learn embeddings of query words using unsupervised relevance-based language models, and show how these pre-trained embeddings can be used for tasks such as query classification, and query expansion. Mitra et al.[95] show how pre-trained embeddins in IR, can be further tuned while training specific supervised and un-supervised tasks , using deep learning models.

Using pre-trained embeddings in recommender systems

Fig. 3.5 illustrates the architecture of Netflix recommender system [2]. The authors divide the architecture into three components: 1) offline 2)nearline, and 3) online. Offline refers to the set of algorithms, and tasks that can be computed as a batch process. Recommender systems algorithms that need to be updated periodically as more data comes in can be considered to be part of the offline architecture. Overtime, user behavior shifts, and more signals are captured. Offline algorithms account for this shift, and change in behavior by

[2]https://medium.com/netflix-techblog/system-architectures-for-personalization-and-recommendation-e081aa94b5d8

re-training or incrementally training on new data. Nearline algorithms do not have real-time constraints, but need to be updated constantly as new user data comes in. Online algorithms are time sensitive processes that need to respond to user actions in real-time. For example if a user stops watching a movie abruptly, then the online algorithms need to reason with this behavior and provide appropriate alternatives. Thus, in this recommendation flow there are various kinds of components that consume user behavior data, and are tailored for different scenarios.

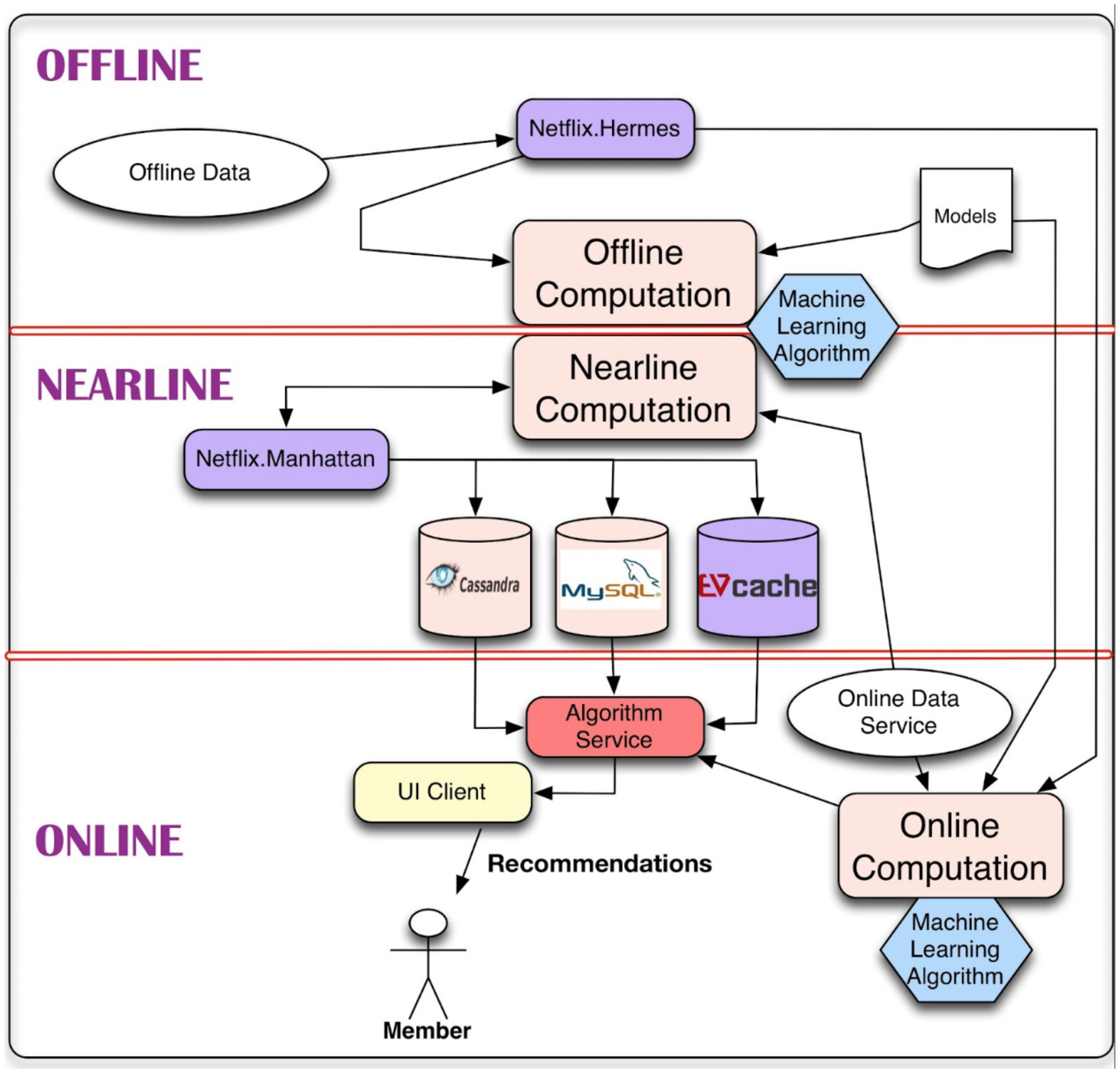


Figure 3.5: *A recommender system architecture*

In Liu et al.[96], the Pinterest recommender system, that serves *pins* to users, based on their current selection is described. This paper describes the *Pin2Vec* neural network model, that learns embeddings of items in the Pinterest domain. The embeddings that are learnt are used in real-time recommendations. This work also explains the architecture for visual search, where visual embeddings on images are used to identify similar items. Grbovic and Cheng[97] detail the use of *listing embeddings*, and *user embeddings*, for the downstream tasks of search ranking, and recommendations on AirBnb.

Chapter 4

Representation Learning and Recommendations in Heterogeneous Networks

In this chapter we discuss how representation learning can be heterogeneous information networks (HIN) using constrained random walks. Apart from users, items, and user feedback in the form of ratings, most domains also contain typed attributes associated with these entities. Information network embeddings approaches have the ability to project the network entities into low dimensional space while preserving the network structure. These approaches have been used for applications such as link prediction and entity disambiguation. In this chapter, we propose a rating aware semi-supervised network embedding approach that takes the entity side information into consideration to generate low dimensional embeddings of HIN. Our empirical evaluation shows that metapath-constrained embeddings perform better than popular network embedding approaches for recommendations.

Projecting all the entities within the HIN into the same space allows the users to express their information needs in the form of queries, where the user query is comprised of various entities in the HIN. This opens up some interesting interaction models for the user to explore the entity space. In this chapter, we also show how graph embeddings can be used to perform query-based recommendations.

4.1 Introduction

Linked open data sources such as *DBPedia* and *WikiData* have become great sources for generating knowledge graphs for online entities. Knowledge graphs with typed entities and relationships can be viewed as instances of a heterogeneous information network (HIN). Heterogeneous information networks are graphs wherein vertices and edges belong to one or more types $t \in \mathcal{T}$. Although most information networks can be viewed as heterogeneous, researchers and practitioners typically ignore the type designations when performing various analytical tasks. One such task is the *recommendation of entities in heterogeneous information networks*, which we define as follows: given a heterogeneous information network $\{V, E\} = G$ and a preference matrix $\mathcal{W}$, we aim to predict the top k vertices $< v_1, v_2, \ldots, v_k >$ of type v associated with some input vertex u. As current network embedding generation approaches assume that the networks are homogeneous these approaches do not take into consideration the ”type” of the entities and interaction patterns between these entities while generating the low dimensional representations. In our approach we capture these interaction patterns between typed entities by specifying domain aware metapaths. Thus to obtain distributed representations of entities in heterogeneous networks, we first model the interactions across the typed entities in the network as metapaths, then we use these domain aware metapaths as blue prints to generate path constrained random walks within the network.

We evaluate our approach by performing extensive experiments using 2 real world heterogeneous datasets. We perform experiments on the *DBLP* citation network and IMDB network. We use the Precision@K metric to evaluate the author recommendations obtained in the DBLP citation network. We then empirically evaluate the recommendations generated by these low dimensional representations. Our experiments show that *MetaWalk* outperforms popular graph embedding approaches.

Personalized PageRank[98], SimRank[99], Deepwalk[100], LINE[101] and other models have been

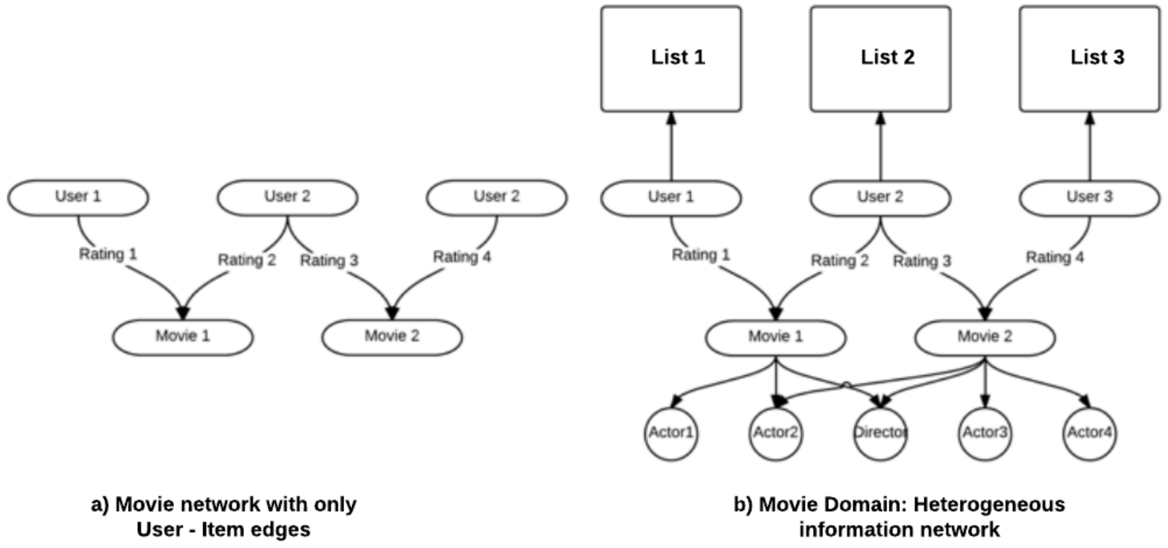


Figure 4.1: *Movie domain interaction between entities*

applied to the recommendations task in homogeneous information networks with good results; however, the type signal provided by heterogeneous information networks allows the user to specify *type* constraints on information to return. Recent work in hybrid recommender systems have demonstrated the efficacy of using knowledge graphs to improve recommendations[102]. Musto et al. in[102;103] use linked open data and PageRank-style features to improve the performance of graph-oriented recommender systems. Path-based approaches that are aware of node type in a graph were proposed in[104] for personalized recommendations. Graph embedding on random walks over graphs and using embedding similarities from *Node2vec*[105] was proposed in *Entity2rec*[106], this approach learns a user, item relatedness in the context of a intermediate property.

4.1.1 User-driven Interactive Recommendations

In addition to typed recommendations, the presence of multiple types of entities within a heterogeneous information network, opens up new modalities for specifying user intent and retrieving recommendations to the users. Query-based information retrieval is one of the primary ways in which meaningful nuggets of information are retrieved from large amounts of data. Here the query specifies a user's information need. In a homogeneous network, in the absence of type and contextual side information, the retrieval context for a user reduces to the user's preferences over observed items. In a heterogeneous setting, information regarding entity types and preference context is available. Thus, query-based contextual recommendations are possible in a heterogeneous network. The contextual query could be type-based (e.g. directors, actors, movies, books etc.) or value-based (e.g. based on tag values, genre values such as "Comedy", "Romance") or a combination of types and values. Thus a user can express not only queries such as *"Generate movies that I like"*, but also queries such as ***"Generate movies that I like" + "similar to the movie 'Saving Private Ryan'" + "war movie" + "drama movie" - "comedy movie"***.

Existing Metapath-based approaches such as PathSim[107] have explored the task of Top-K similarity search in HIN. One limitation of these approaches is that the approaches only allow for obtaining pair wise similarity of entities and cannot support combination queries that involve multiple types of entities. In our proposed approach, all the typed entities within the HIN are represented in the same space, we have the ability to retrieve recommendations for complex contextual queries.

We evaluate the effectiveness of query-based recommendations using network embeddings by applying them to the list completion task. Experiments conducted on the user curated movie lists in the IMDB data set for set completion problem demonstrate the usefulness of distributed representations. We use the Precision@K metric to compare the user lists generated with the ground truth user lists.

The following is a summary of the contributions documented in this chapter:

1. We propose a metapath-based path constrained random walk framework to capture the interaction patterns between entities in the HIN, and to generate distributed representations of these entities.

2. We propose a query-based user driven interactive recommendation approach using the Metapath-based HIN embeddings. This allows the user to define the context for the

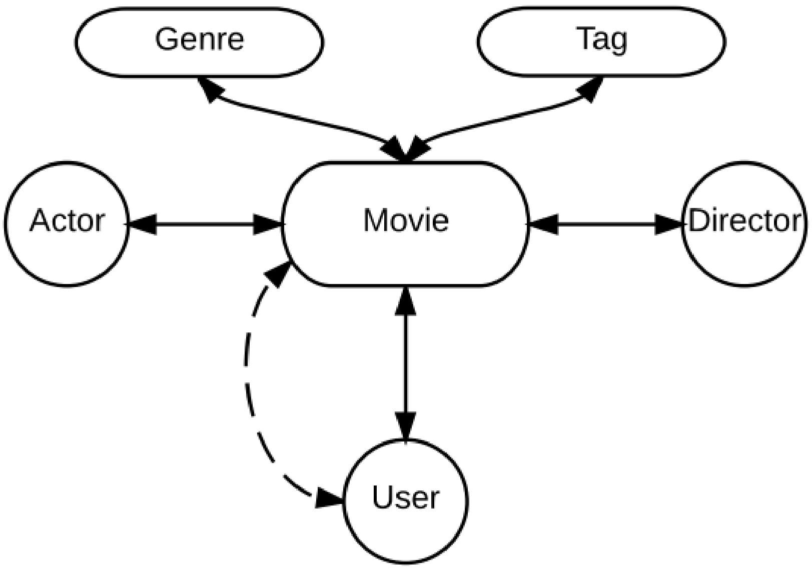


Figure 4.2: *Movie - Knowledge Graph Schema*

recommendations. The query context is expressed as a combination of entities present in the HIN. We demonstrate the efficacy of this approach by using the HIN embeddings for the list completion task. We propose two kinds of user interface modalities for user driven interactive recommendations: 1) exemplar-based recommendations 2) *"less like this/more like this"* style recommendations

4.2 Problem Definition

In this section we provide an overview of of heterogeneous information networks and introduce MetaWalk a metapath embedding approach for recommendations and retrieval in HIN. Figure 4.2 illustrates the various kinds of interactions that can exist with the Movie domain. Traditional recommender systems were restricted to a bipartite network between Movie and User entities with rating edges connecting them.

Heterogeneous Information Network (HIN) : A weighted HIN is defined as a directed graph $G = \{V, E\}$ with an entity mapping function $\phi : \mathbb{V} \rightarrow A$ and a edge type mapping function $\psi : \mathbb{E} \rightarrow R$ where each node v $\in \mathbb{V}$ belongs to one particular entity type

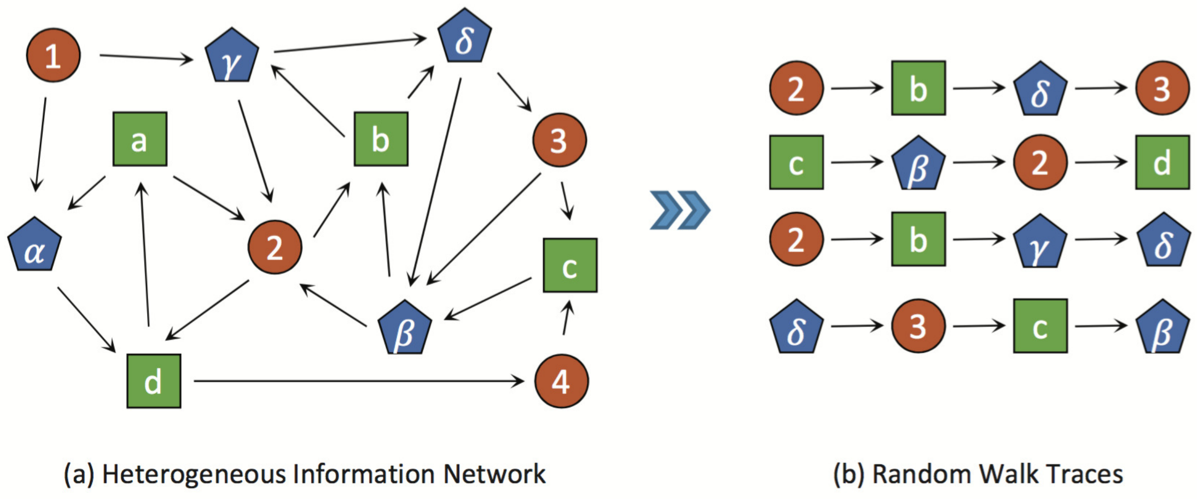


Figure 4.3: *(a) heterogeneous information network, (b) traces of fixed-length random walks*

ϕ(v) $\in \mathcal{A}$ and each edge e $\in \mathbb{E}$ belongs to a relationship type ψ(e) $\in \mathcal{R}$. The edge weights associated between vertices with the relationship context ψ(c) $\in \mathcal{R}$ is captured as a preference matrix $\mathcal{W}_c$.

Metapath: A metapath is a sequence of edges specified over a HIN schema $\mathcal{S}_G = (\mathcal{A}, \mathcal{R})$. The Metapath defines a composite relationship that consists of ordered sequence of edge types specified in the HIN schema.

Figure 4.5 illustrates some of the metapaths derived from the Movie network schema specified in Figure 4.2. The UMU metapath specified in Figure 4.5 (b) illustrates the rating semantic relationship between the entities User and Movie. This relationship is captured in the User-Rating-Movie bipartite network, where the edges exist as rating relationship between users and movies. Similarly the MGM metapath specified in Figure 4.5 (c) illustrates the Movie attribute semantic relationship between the set of movies and the genres associated with these movies.

As we define the HIN as a directed graph and capture the edge weights in a contextual preference matrix $\mathcal{W}_c$, we can handle cases where multiple contexts exist between two entity types.

Weighted Citation network: Citation network is a prime example of a weighted HIN.

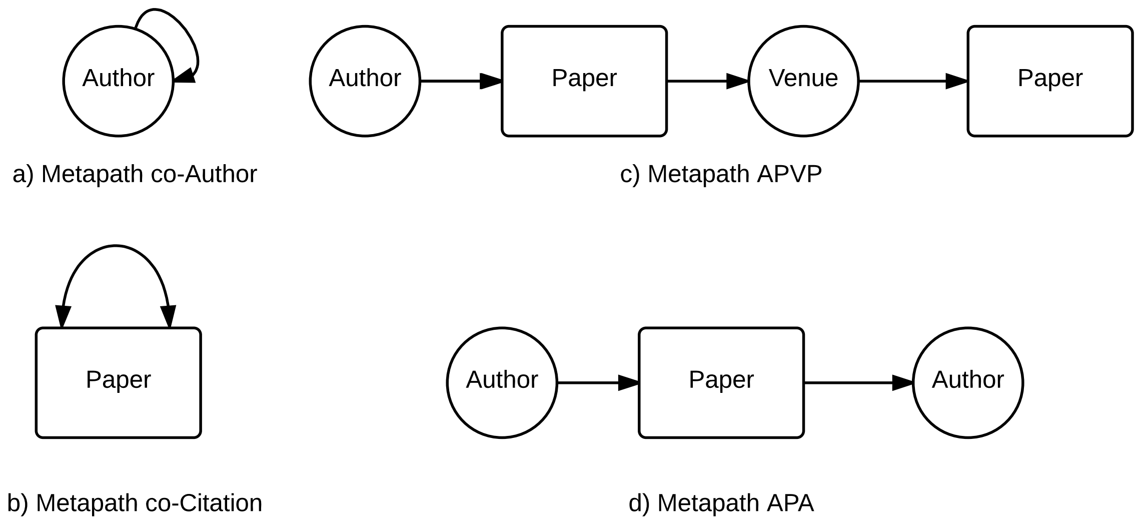


Figure 4.4: *Citation network Metapaths*

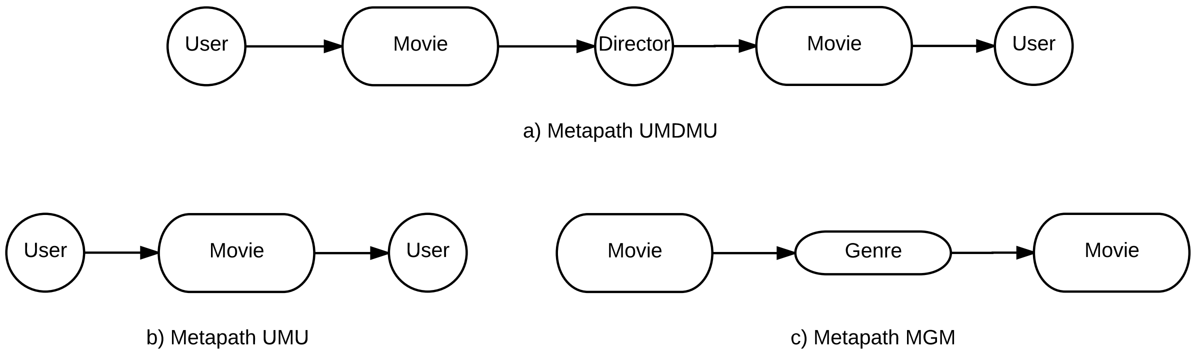


Figure 4.5: *Movie Network Metapaths*

The citation network contains entities of the types: Authors (A) , Papers (P) , Venues (V) and Terms (T) . For each paper the possible set of relations are co-Author (Author - Author) , Author - Paper, Paper - term, Paper - Venue, co-Citation (Citation - Citation) , Paper - Citation relationships. A publication can have two roles, the Paper and Citation roles. The preference matrix among co-Authors and co-Citations is weighted, the preference matrix for these contexts captures the co-occurrence count of pairs of entities at the global data set level. The Author - Paper, Paper - Term, Paper - Venue and Paper - Citation relationships are unweighted.

Movie network with explicit feedback: The movie network contains entities of the types: Movies (M) , Users (U) , Actors (A) , Directors (D) , Genre (G) and Terms (T) . Some of the relationships that exist in this network are: Rating relationship between User and Movie, Acted relationship between Actor and Movie, Directed relationship between Director and Movie, Genre relationship between Movie and Genre, List Curation relationship between User and Movie, Keyword relationship between Movie and Term. Rating relationship between User and Movie is weighted, where the weight associated is the rating, while the List Curation relationship between user and movie is unweighted. The acted, directed, genre and keyword relationships between a movie and its attributes are unweighted.

Network Schema of a Weighted HIN: The network schema is a high level representation of the typed relationships between various entities in a HIN. The network schema of a graph $G = \{V, E\}$ with entity types in $\mathcal{A}$ and and edge types in $\mathcal{A}$ is represented as $\mathcal{S}_G = (\mathcal{A}, \mathcal{R})$. Network schema is nothing but a blue print for the various interactions and the contexts that can exist in a type network. The network schema can be leveraged by domain experts to specify interaction patterns in the HIN. Figure 4.2 provides two example schemata, the citation network schema and the movie network schema. The self edges present in Figure 4.2 (a) illustrates the co-occurrence edges captured by co-citation and co-author relationships. The presence of multiple contextual relationships between two entity types is illustrated by the presence of solid and dotted edges between a User and a Movie in Figure 4.2 (b). These two edges capture the User-Rating-Movie interaction and the user-List-Movie interaction.

Weighted metawalks:

Using the metapaths as references, we perform path constrained random walks on the HIN. As the relationship edges between entities in the graphs have weights associated with them, we cannot perform uniform sampling over the outgoing edges of a vertex. In weighted metawalks we propose performing edge weighted sampling. The weight of the edge between two entities in the graph can be viewed as the strength of association between those two

Table 4.1: *Metapaths captured from IMDB schema*

IMDB Metapaths
user - movie
user - movie - director - movie
user - movie - actor - movie
user - movie - genre - movie
user - movie - language - movie
user - movie - keyword - movie
movie - genre - movie - director
director - movie - actor - movie
director - movie - genre - movie
language - movie
keyword - movie

Table 4.2: *Metapaths captured from Citation Network schema*

Citation Network Metapaths
coAuthor
coReference
paper - reference
author - venue - author - reference
author - paper - coAuthor
venue - paper -reference
venue - reference
author - paper - reference
author - paper - venue - paper

entities. But the transition strength need not be uniform on these entities. For example let's say a user rates a movie with rating 5.0, then the edge weight between this user and movie is 5.0, but, the transition strength from user to movie is not the same as transition strength from movie to user. This directed edge value depends on the number of outgoing edges from the transition source entity as well as the relative strength of this edge over all the outgoing edges.

Table 4.1 enumerates the various metapaths that we specify over the IMDB schema and Table 4.2 enumerates the various metapaths that we specify over Citation network schema. These domain aware metapaths are used as blueprints for performing random walks. We use the metapath-based random walk algorithm described in Algorithm 1 to perform weighted metawalks over a HIN.

Injecting Non-linearities for edge selection on weighted edges:

While specifying the edge selection strategy it is essential to understand the nature of the relationships between the entities and the meaning of weights associated between the entities. For example, if a user U has 2 outgoing edge E_1 and E_2 to movies M_1 and M_2 respectively. If U rates M_1 as a 10 and M_2 as 5, its does not necessarily mean that the strength of association between U and M_1 is twice as much as M_2. To understand the effect of such non-linear associations between entities we perform experiments with both linear and non-linear weightings over the preference matrices.

4.3 Distributed Representations and HIN embeddings

Distributed representations have become a popular and effective way to represent higher-dimensional information. Distributed representations have found a wide degree of applications in the fields of machine learning, natural language processing and information retrieval. In the *MetaWalk* framework we aim to represent HIN into a low dimensional representation while ensuring that the structural and semantic properties of the HIN are preserved. To obtain information network embeddings we first specify the set of metapaths with the network that capture semantic and structural patterns and then generate metapath constrained random walks over the network. Once these random walks are generated, we use the Skip-gram approach specified by Mikolov et al.[4] to generate information network embeddings that capture semantic relationships captured by user curated lists.

4.3.1 Skipgram

The Skipgram approach is a neural network approach for generating distributed representations over contextual windows. Given an entity e, the Skipgram model tries to predict the surrounding context entities that are present within the random walk. The context for an entity in a path constrained random walk is defined by a window around the entity. For example in the User - Movie metapath, one context for users is movies that they prefer and similar users that have shown similar preference for that movie. This metapath essentially captures the collaborative filtering hypothesis of similar users show similar preferences over items, for recommendations generation. For each entity and its context, a Softmax function acts on the output layer activations for each input context entity. This approach tries to maximize the co-occurrence probability of entities that appear within the same context.

4.3.2 Skipgram for Metapaths

In the case of User-Item metapath, this relationship spans the bipartite graph of users and items. We then generate a random walk by traversing this bipartite graph. The semantic meaning of these random walks can be expressed as similar users will have semantically similar movies. For a sequence of items $\{s_i\}_{i=1}^{K} \subseteq S$ we maximize the following term:

$$\frac{1}{K}\sum_{i=1}^{K}\sum_{j\neq i}^{K}\log\Big(\sigma(u_i v_j)\prod_{k=1}^{N}\sigma(-u_i v_k)\Big)$$

where $u_i \in U$ is the latent vectors of target entity and $v_i \in V$ is the latent vectors of other entities in the context window around target entity.

$$\sigma(x) = 1/1 + exp(-x)$$

N is the number of negative samples drawn that would be used to minimize the similarity between the target entity and the item not present in the item context. The negative items are sampled from the item distribution. The latent embeddings of items are estimated by using stochastic gradient descent that maximizes the above criteria. We use *cosine similarity* to calculate similarity between two latent representations. We use additive vector composition (AVC) to come up with a single representation from multiple representations if we want get a single representation for a set of entities.

Skipgram for Homogeneous Paths: When all the nodes in a metapath are of the same *type* we can model the conditional probability $\Pr(v_j|\Phi(v_i))$ to be independent of the *type* of the node v_j, and can be derived by softmax as:

$$\Pr(v_j|\Phi(v_i)) = \frac{\exp(\Psi(v_j) \cdot \Phi(v_i))}{\sum_{u \in V} \exp(\Psi(u) \cdot \Phi(v_i))}. \tag{4.1}$$

Skipgram For Metapaths that assumes the probability $\Pr(v_j|\Phi(v_i))$ is related to the type of node v_j:

$$\Pr(v_j|\Phi(v_i)) = \Pr(v_j|\Phi(v_i), \phi(v_j))\Pr(\phi(v_j)|\Phi(v_i)), \tag{4.2}$$

and can be derived by softmax as:

$$\Pr(v_j|\Phi(v_i), \phi(v_j)) = \frac{\exp(\Psi(v_j) \cdot \Phi(v_i))}{\sum_{u \in V, \phi(u) = \phi(v_j)} \exp(\Psi(u) \cdot \Phi(v_i))}. \tag{4.3}$$

Table 4.3: *IMDb dataset overview*

sparsity	5	10	50	100
user	18414	18414	18414	18414
movie	30408	40837	73571	91070
rating	323979	648847	3249149	6494480
actor	35911	46475	79633	96910
director	13042	16843	27922	33493
genre	26	27	28	28
language	216	220	256	259

Table 4.4: *Citation network dataset overview*

sparsity	paper	author	venue	reference
5	57317	113625	7200	72769
10	111045	190230	7200	118732
50	541551	561721	7200	303968
100	1072190	842822	7200	411723

4.4 Experiments and Results

In this section, we present the empirical analysis of the proposed recommendation framework that leverages the network schema of the heterogeneous domain to generate recommendations. We performed a series of experiments to demonstrate the effectiveness of our MetaWalk model on IMDB.com domain. In this movie domain, we apply the proposed algorithm to recommend a personalised Top-N movie-list for the user.

4.4.1 Dataset

To demonstrate the effectiveness of the proposed recommendation framework, we choose two datasets from different domains (movie and academic citations). We represent the two datasets as heterogeneous network. In the IMDB dataset, we use user ratings and meta-data of movies that captures side information in order to build a heterogeneous information network. The different entities in our graph are users, movies, actors, directors and genre. The HIN graph consists of both weighted and unweighted edges over different typed entities.

The edges between user and movie entities are weighted. The edges about the meta-data of movies are unweighted. The weighted edges represent the degree of affinity between two nodes. For example, if a user rated a movie, we assign a weight to the user-movie edge as a function of user's rating. On the other hand, unweighted edges represent metadata about the movies e.g. a movie-genre edge is created if movie belongs to that particular genre. We filter out all the users who have rated less than 20 movies while building the dataset. The dataset is divided into training and testing set as follows. For each user, we retain 50% user-movie edges in training dataset, and rest is treated as testing dataset. In order to show the effectiveness of our proposed approach, we further divide training set into sparse sets based on number of edges. We create four sets with 5%, 10%, 50% and 100% randomly sampled edges conditioned on user from the training dataset. The four sparse sets are described in Table 4.3.

The second dataset is Citation Network Dataset[108] extracted from DBLP. The citation graph covers all the citations within a dataset of 3,272,991 papers with 8,466,859 edges. Each publication has six attributes: paper title, publication venue, published year, abstract, authors, and references. We extract relations with respect to authors, references and venues. While building the dataset, we filter out authors with less than 5 venues. We do this since we are predicting the next venues for an author. Train and test set is created pivoted on the venues. For each venue, we retain 50% edges in train dataset, and rest of the edges fall in test dataset. Similar to the experiments in IMDB dataset, to measure the effectiveness of our approach at various levels of sparsity, we create sparse sets from the train dataset. The sparse partition of training data is described in Table 4.4.

Degrees of freedom: For our experiments we consider the following while evaluating various approaches: 1) sparsity of dataset to compare effectiveness of various approaches across different sparsities 2) dimensionality of the learnt user/item embedding 3) significance of using side information meta data in heterogeneous networks 4) significance of injecting non-linearities while performing random walks over weighted edges.

Table 4.5: *Results for Precision@K with varying sparsity*

	10% Data		50% Data		100% Data	
	P@10	P@20	P@10	P@20	P@10	P@20
NCF(Baseline)	0.034	0.059	0.127	0.178	0.235	0.316
LINE	0.082	0.132	0.083	0.133	0.089	0.143
DeepWalk	0.118	0.187	0.141	0.218	0.169	0.246
Node2Vec	0.081	0.132	0.082	0.133	0.083	0.134
MetaWalkU	0.113	0.182	0.116	0.183	0.122	0.191
MetaWalkW	0.159	0.238	0.18	0.262	0.185	0.271

Table 4.6: *Results for Kendall's τ, for ranked-order comparison*

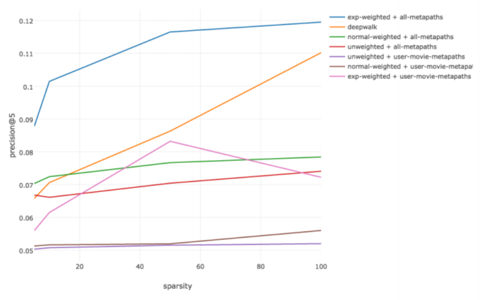
	5% Data	10% Data	50% Data	100% Data
LINE	0.151	0.170	0.187	0.202
DeepWalk	0.259	0.292	0.347	0.364
Node2Vec	0.296	0.313	0.333	0.340
MetaWalkU	0.270	0.279	0.284	0.292
MetaWalkW	0.359	0.400	0.462	0.477

4.4.2 Evaluation Metrics

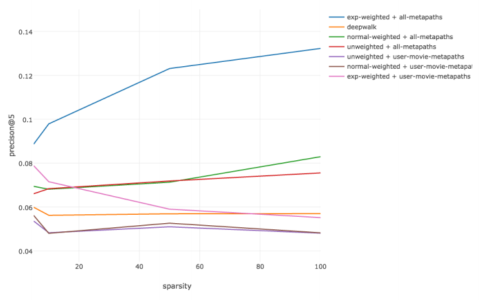
To measure the efficacy of top-K item recommendations using various proposed approaches, we use the Precision@K. In the case of IMDB data set, we treat the recommendation task as recommending unseen movies to the users. We treat this as a Top-K recommendation task and not as rating completion task.

4.4.3 Results

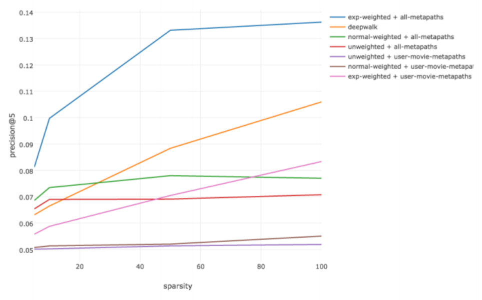
Table 4.6 shows the empirical results for Precision@K in the Imdb.com domain. A MetaWalk approach that leverages domain information in the HIN schema outperforms our baseline approach. MetaWalkU - Unweighted MetaWalk generates random walks without considering the user-item ratings. MetaWalkW - Weighted MetaWalk is the approach that generates random walks while taking into account the user-item ratings.



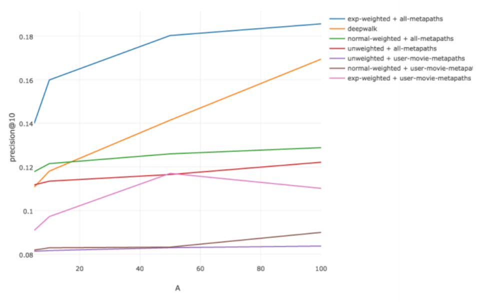
(a) embedding dim = 64



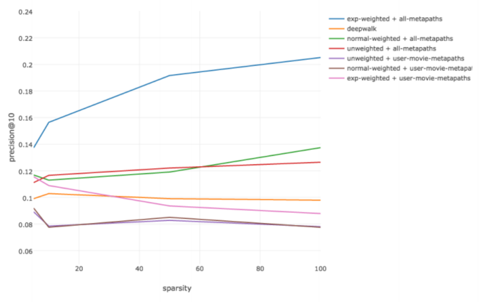
(b) embedding dim = 100



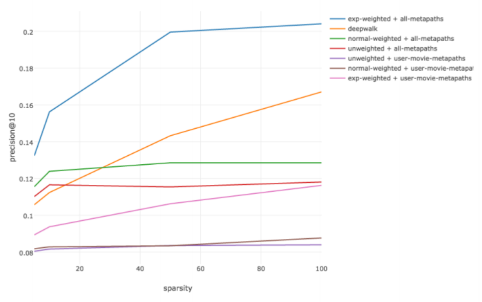
(c) embedding dim = 300



(d) embedding dim = 64

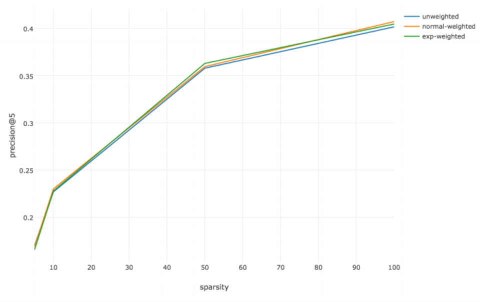


(e) embedding dim = 100

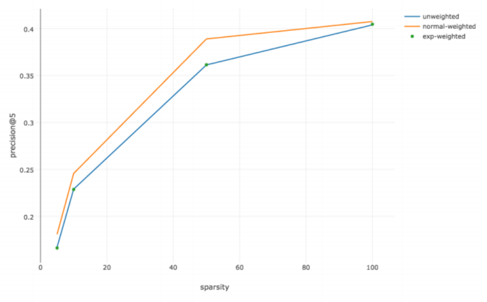


(f) embedding dim = 300

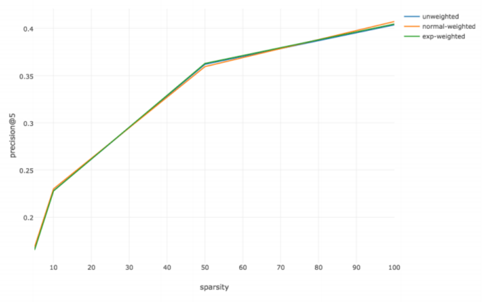
Figure 4.6: *precision@5 and precision@10 for varying sparsity in IMDb domain*



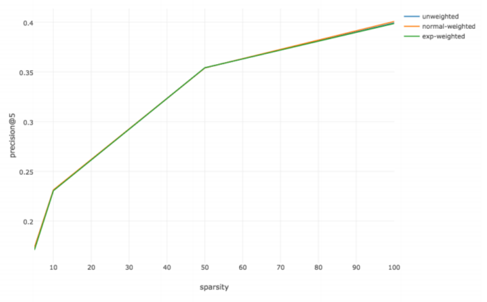
(a) embedding dim = 64



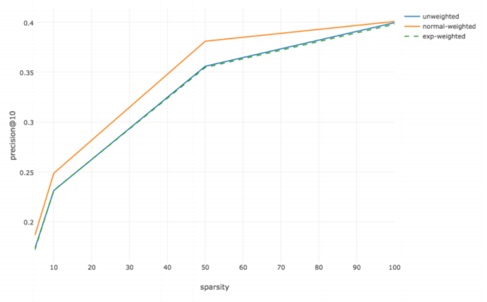
(b) embedding dim = 100



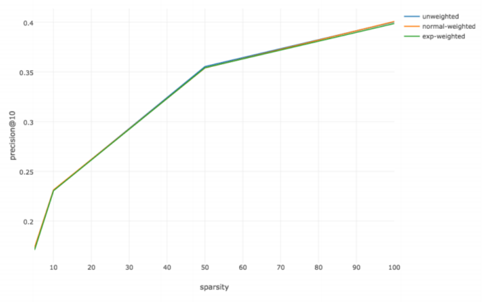
(c) embedding dim = 300



(d) embedding dim = 64



(e) embedding dim = 100



(f) embedding dim = 300

Figure 4.7: *precision@5 and precision@10 for varying sparsity in citation network*

Table 4.7: *Exemplar-based user list generation for "Bond Movies" (Left) and "Film Noir" (Right) Movie titles in bold constitute the query set*

Dr. No	**Maltese Falcon**
Goldfinger	Double Indemnity
You Only Live Twice	The Big Sleep
Thunderball	Key Largo
On Her Majesty's Secret Service	The Third Man
From Russia with Love	Laura
Moonraker	Out of the Past
Diamonds Are Forever	To Have and Have Not
The Man with the Golden Gun	White Heat
For Your Eyes Only	Dark Passage
The Living Daylights	The Lady from Shanghai

4.5 Interactive Recommendations

The semantic embeddings can be used to refine a user information need by providing exemplars. This opens up new modalities of interactions for the users to gravitate towards their requirements.

4.5.1 Exemplar-based Recommendations

Table 4.7 shows the result of exemplar-based list completion for the user information need "Bond" movies and "Film Noir" movies, as the community structure of these movies is tight knit, our list generation approach generates a good set of movies with query set of size 1. This interactive modality can be viewed as a list refinement task, where the user is providing examples and refining the result set of items based on a tacit intent that they can more easily express. Table 4.8 shows the result of exemplar-based list completion for the user information need "new romantic comedy" movies, by adding more examples into the query set the user can tweak the membership of the list generated. The rank of the candidate items changes as the query set membership is modified.

Table 4.8: *Exemplar-based user refinement for the intent "newer romantic comedies", Movie titles in bold constitute the query set*

The 40-Year-Old Virgin	**The 40-Year-Old Virgin**
Wedding Crashers	**Knocked Up**
Waiting...	Superbad
Anchorman...	Wedding Crashers
Step Brothers	The Heartbreak Kid
EuroTrip	Anchorman...
Harold & Kumar...	EuroTrip
The Ringer	Sex Drive
The 40-Year-Old Virgin	**The 40-Year-Old Virgin**
Knocked Up	**Knocked Up**
Anchorman...	**Anchorman...**
Superbad	**Step Brothers**
Wedding Crashers	Superbad
EuroTrip	Wedding Crashers
Harold & Kumar...	Role Models
Step Brothers	Harold & Kumar...

4.5.2 *Less Like this and More Like this*

As all the entities (users, items, item attributes) are represented in the same space and as the user embeddings are personalized, we can retrieve the resultant entities that match the query expressed by the user. Notice here that entities need not be associated with the attributes such as "war", "comedy" etc. to qualify for this as they are represented semantically. We can also express queries such as *"more like this"* and *"less like this"*. *e.g.* ***"generate movies more like 'Casino Royale' and less like 'Golden Eye' "***. The presence of positive and negative interactions between the users and the entities can also be leveraged to provide *Less Like this and More Like this* interactive interfaces (where high ratings/thumbs up can be considered as a positive interaction) . We believe that interactive search and recommendations interfaces can be highly effective where users are in an *exploratory* phase and are not familiar with the domain. In these cases, these novel modalities of interaction can prove to be highly effective. This is a less researched area and more work needs to be done in this field, especially with respect to metrics to identify effectiveness of such interface modalities.

Algorithm 1 Metapath-based Random Walk generation for weighted graphs

```
Data: initial node $n$ of type $T$, metapath $P$, length of random walk $L$
/* metapath $P$ is a Circular Queue data structure */
Result: weighted random walk $w$
$w$ ={}
/* random walk $w$ is a Queue data structure */
Enqueue ($w$,$n$)
while $iter \leq L$ do
    $nextType$=Next ($P$)
    /* getting the next edge type in metapath */
    if isWeighted (nextType) then
        /* if next Edge in the metapath is a weighted edge */
        $WNeighborhood$= GetNeighbors ($n$,$nextType$)
        /* Get the Typed neighbors From the current Node */
        $n$ = WeightedSampleNeighborhood ($WNeighborhood$) /* Perform weighted
        sampling in the Typed Neighborhood */
        Enqueue ($w$,$n$)
    else
        $Neighborhood$= GetNeighbors ($n$,$nextType$)
        /* Get the Typed neighbors From the current Node */
        $n$ = UniformSampleNeighborhood ($WNeighborhood$)
        /* Perform uniform sampling in the Typed Neighborhood */
        Enqueue ($w$,$n$)
    end
end
```

Chapter 5

Semantic Diversity in Top-N Recommender Systems

5.1 Introduction

Modern recommender systems use interaction data between users and items - either in the form of explicit or implicit feedback - to predict user preference of unobserved items. These approaches tend to use relevance metrics such as RMSE and ranking metrics to predict a user's proclivity towards items, but focusing solely on these relevance metrics leads to recommending highly similar homogeneous set of items that exhibit low diversity[109]. The drawback of such an approach is that the user is constrained to a low entropy result set resulting in lower user satisfaction. This will also result in reduced coverage of the item set and lower exploration opportunities for the user to discover novel and serendipitous items[110]. Improving diversity of recommender systems has become an important research topic in order to increase the discoverability of items. Nguyen et al.[7] note that lack of diversification of results lead to *"filter bubbles"* and over time recommender systems expose users to a narrowing set of items. Vargas et al. use various latent user preferences over items to improve quality of recommendations[111]. The authors propose identifying user sub profiles by creating subsets of user interests from the set of user preferences over items. In this work, first the user profiles are partitioned into pre-defined categories over the items and then use these partitioned user sub-profiles to generate partition specific recommendations. The recommendations from various partitions are then aggregated to generate a diverse set of recommendations.

User-curated lists span a wide range of domains and usually contain items that users view to be of a coherent topic. These lists of items can be videos belonging to a particular topic on YouTube, movies on the Internet Movie Database (IMDb) , books on GoodReads domain, lists of users and accounts on Twitter, playlists on music platforms such as Spotify and wish lists on e-commerce domains such as Amazon. On domains such as Spotify, most user-item preference activity happens predominantly through *"list activity"*. On IMDb, a movie domain and GoodReads, a books domain the items in lists that are curated by users typically share some common attribute such as genre, tag, director, actor, author, etc. These lists capture semantically meaningful items at various granularities across various dimensions of user interests. For example in the list titled "TOP WAR MOVIES"[1] on IMDb, the user lists a set of "war" movies. A user list titled "Great Old Movies (pre-1960) "[2] deals with pre-1960 movies. In these examples, the items in the lists co-exist in different contexts such as Genre and Temporal similarity, respectively. While a 'tag' such as "western" exists on imdb.com, a user identifies "Modern western" movies in the user list "Modern Westerns:The Ultimate List."[3] On goodreads.com, within lists tagged with the tag "love,"[4] we can see a wide variety of contexts within which items can be categorized semantically. In this chapter we propose to leverage the semantic context that exists within lists to provide a more diverse set of recommendations. Nguyen et al.[7] use information encoded in user-generated tags to measure the content diversity of items recommended by recommender systems. The user-generated tags in this instance capture a context association of a user to an item.

Determinantal Point Processes (DPPs)[112] have been succesfully used in machine learning tasks such as information retrieval, diverse subset sampling, and document summarization.

[1] https://www.imdb.com/list/ls026329851/
[2] https://www.imdb.com/list/ls000000580/
[3] https://www.imdb.com/list/ls055895628/
[4] https://www.goodreads.com/list/tag/love

Determinantal Point Processes have the ability to model the balance between quality and diversity of sets as they model *repulsion*. In this exploratory work, we propose to improve the diversity of Top-N recommender systems, by using DPPs over user-created lists. We crawled the imdb.com[5] domain to generate a lists dataset, the semantic similarity measure from these user-generated lists is used by the DPP to model the diversity of the items. We use an average dissimilarity metric to measure the diversity of the resulting re-ranked list. Our early results on the MovieLens 1-million ratings dataset[113] show that incorporating semantic similarity expressed in user lists as a diversity proxy results in a more diverse set of recommendations. The contributions in this chapter are the following:

– We propose leveraging user-curated lists for improving the intra list diversity of Top-N recommender systems using Determinantal Point Processes (DPPs) . We empirically show that the intra-list diversity score of Top-N systems can be improved by DPP re-ranking.

– We show how latent representations – learnt over HIN by performing metawalk-based random walks over – can be used to inject semantic diversity in top-n recommender systems.

– We argue for using more diversity metrics apart from the “popularity-biased” co-occurrence similarity that is popular in the recommender systems literature. We observe that to improve our understanding of this field, we need objective metrics that would inform us of the utility of various diversity metrics.

5.2 Background

5.2.1 Determinantal Point Processes (DPPs)

A Determinantal Point Process (DPP) is useful in models and applications where repulsive effects or diversity are important. For example, creating a model with the assumption of a

[5]https://www.imdb.com/

uniform spatial distribution (e.g., particles[114], cluster centers[115], etc.) may be unwarranted, and using a DPP may be a better choice. In recommender systems and information retrieval settings, it may be desirable to return a more diverse collection of items, and a DPP can be used to incorporate this preference for diversity.

Given a set Y of cardinality N, a DPP can be thought of as a probability distribution on the subsets of Y, where the probabilities are proportional to the determinant of some matrix. Such distributions are important because the determinant of a matrix captures the volume of the parallelepiped spanned by its columns, which provides a useful measure of diversity amongst the column vectors. In principle, one can encode relevant information into an N-by-N matrix, each row and column being indexed by an item in Y. The determinants of interest are those corresponding to the principal minors of this matrix. Different approaches to DPPs exist, depending on the conditions the N-by-N matrix satisfies. In the *L-ensemble* approach, our N-by-N matrix L is positive semidefinite, and the probability that our randomly selected subset $R \subseteq Y$ is exactly the subset $A \subseteq Y$ is given by the equation

$$P_L(A = R) = \tfrac{\det(L_A)}{\det(L+I)},$$

where I is the $N \times N$ identity matrix and L_A is the principal minor of L whose rows and columns are indexed by the items in A.

The L-ensemble approach is useful when we have a feature space representation for the items in Y. In this case, L is just a Gram matrix associated to these items.

For sufficiently large N, it is infeasible to compute the determinants of the all the principal minors and select a subset of maximal determinant. For this reason, applications rely on sampling algorithms that probabilistically constructs a subset with the probability of constructing A being approximately $P_L(A)$.

5.2.2 k-DPPs

In general, sampling from a DPP involves two random variables: the random subset itself and the size of the subset. For some applications, the desired number of items to select, say k, is already known. (For example, a recommender system may recommend $k = 5$ items for purchase.) In this case, a *k-DPP* is used. Technical details, including the contents of this section, can be found in[116] and[112] by Kulesza and Taskar.

Given an L-ensemble DPP P_L, we may recalculate probabilities to only take into consideration k-item subsets:

$$P_L^k(A = R) = \frac{\det(L_A)}{\sum\limits_{|A'|=k} \det(L_{A'})}. \tag{5.1}$$

Since

$$\sum_{A' \subseteq Y} \det(L_{A'}) = \det(L + I) = \det(L + I) \sum_{A' \subseteq Y} P_L(A'),$$

the denominator in (1) can be rewritten as

$$\sum_{|A'|=k} \det(L_{A'}) = \det(L + I) \sum_{|A'|=k} P_L(A') \tag{5.2}$$

while the numerator in (1) can be rewritten as

$$\det(L_A) = \det(L + I) P_L(A). \tag{5.3}$$

Using results in the theory of symmetric functions, we derive from (2) that

$$\det(L + I) \sum_{|A'|=k} P_L(A') = e_k(\lambda_1, ..., \lambda_N), \tag{5.4}$$

where e_i is the ith elementary symmetric polynomial and the $\{\lambda_j\}$ are eigenvalues of L.

Using (3) and (4) , we have that the atomic probability can be written as

$$P_L^k(A = R) = \tfrac{\det(L+I) P_L(A)}{e_k(\lambda_1, ..., \lambda_i)}.$$

The practical significance of this form is that the elementary symmetric polynomials may be computed in polynomial time, whereas the sum $\sum\limits_{|A'|=k}$ has exponentially many terms, presenting a priori time complexity challenges. Furthermore, it can be shown that the marginal probability is given by

$$P(i \in R) = \frac{\lambda_i e_{k-1}(\lambda_1,\dots,\lambda_{i-1})}{e_k(\lambda_1,\dots,\lambda_i)},$$

yielding a sampling algorithm described by Kulesza and Taskar[116].

Algorithm 2 Procedure for Sampling from k-DPP

```
Input: $k$ Eigenvector/Eigenvalue Pairs $\{(v_n, \lambda_n)\}$
Output: A sampled subset $R$
for $n \leftarrow 1$ to $N$ do
    if $u \sim U[0,1] < \lambda_n \frac{e_{k-1}(\lambda_1,\dots,\lambda_{n-1})}{e_k(\lambda_1,\dots,\lambda_n)}$ then
        $J \leftarrow J \cup \{n\}$
        $k \leftarrow k - 1$
        if $k = 0$ then
            break
        end
    end
end
$V \leftarrow \{v_n\}_{n \in J}$
$R \leftarrow \emptyset$ while $|V| > 0$ do
    $R \leftarrow R \cup \{i\}$ with probability $\frac{\sum_{v \in V} (v^T e_i)^2}{|V|}$
    $V \leftarrow V_\perp$, orthonormal basis of the subspace of $V$ orthogonal to $e_i$
end
```

The sampling process from the k-DPP is described in Algorithm 2. Its input is an eigendecomposition of matrix L, i.e. a collection of eigenvectors of L along with their associated eigenvalues. It outputs a set containing k items, and the probability of outputting any particular k-element set is given by the k-DPP. To use DPP model in the top-N recommendation task, we should construct the similarity matrix. In the context of recommender systems, we might think of L as a similarity matrix between all pairs of items. In Section 3, we provide more details on obtaining L from user-curated lists.

5.3 HIN Embeddings over User Curated Lists

We consider the list of items present in a playlist as a sequence, and employ the popular CBOW model from *Word2Vec*[80] to learn item embeddings on the sequence C. A fixed size window is moved over each list and the model is trained by trying to predict the track in the middle of the window correctly given all the other tracks within the same window. This translates to minimizing the following loss,

$$\mathcal{L}_{\text{CBOW}} = \sum_{\phi \in C} \sum_{i}^{|\phi|} -\log\big(p(\vec{v}_i | \sum_{i-k \leq j \leq i+k, j \neq i} \vec{v}_j)\big) \tag{5.5}$$

Where, $\vec{v}_i$ is the embedding of the i^{th} track in the playlist ϕ. Similar to Mikolov et al.[79], our item embeddings are trained with negative sampling instead of the full softmax over the complete track collection. The IMDB network involving user curated lists contains four different types of entities—lists, items, users, and tags. Alternatively, we can view this dataset as a heterogeneous information network (HIN) . A meta-path defines a composite relationship by an ordered sequence of edge types specified in the HIN schema $\mathcal{S}_G = (\mathcal{A}, \mathcal{R})$. Based on two different meta-path definitions: tag→list→item→tag (TLIT) and item→list→item (ILI) .

In summary, we learn item embeddings based on three different approaches:

1. EMB1: CBOW over lists as sentences and items as terms

2. EMB2: CBOW over the ILI metapath

5.4 Data Set, Experiments and Results

The MovieLens 1-million ratings dataset contains ∼1 million ratings of 3,076 movies provided by 6,040 users.

We obtained the IMDb user lists by performing targeted crawls for lists on the imdb.com domain. We performed various crawls on the imdb.com domain over a period of 4 weeks from 11/24/2017 to 12/24/2017. The crawl was able to capture historical interactions of 74134 users, these interactions were of the form “list activity” and “rating activity”. A user performs list activity when they create a list and the lists has at least 1 item in the lists. These approximately 74k users generated 352543 user-curated lists. These lists contained

13 million interactions between users and items. Of these 350k lists, we pruned generic lists that contain mostly popular items (as these generic lists are associated with no semantic coherence) . This resulted in a pruned set of 155496 lists. From these ∼155k lists we removed items that were not part of the MovieLens dataset.

5.4.1 Experiments and Results

We divide the Movielens dataset into training and test sets, where we randomly select 1 item from the user's item set to be part of the test set and the rest of the items are part of the training set. For each user, using the user-item interaction in the training set as the user profile, we identify the top 20 similar items of each item and the aggregate of all the top 20 similar items of every item in the profile set forms the final candidate set. We then rank the items in the candidate set and items are evaluated for N= 10 and 20. The baseline algorithm aggregates the item-item similarity to generate a top-N list.

The semantic item-item similarity $SemSim_{ij}$ utilized by k-DPP is calculated based on the co-occurrence of the item pairs across the item lists. We can think of $SemSim$ as a Gram matrix by viewing each item as a vector of 0s and 1s, with the m-th coordinate 1 if the item shows up in list m and 0 otherwise. Under this representation, $SemSim_{ij}$ is just the dot product of the vectors for items i and j, and $SemSim$ is a Gram matrix. As a Gram matrix, $SemSim$ is positive semidefinite and therefore able to be used in our k-DPP.

Evaluation metric

We use the average dissimilarity metric to measure the diversity of the resulting set. The similarity of 2 items i, j is given by Sim_{ij} which is the co-occurrence similarity of the items in the Movielens datsets as defined in[117].

$$\text{Diversity(AverageDissimilarity)} =_{i,j \in R_u, i \neq j} (1 - \mathbf{Sim}_{ij})$$

Table 5.1: *Average dissimilarity measure of diversity*

Approach	diversity-Top 10	diversity-Top 20
Top-100 + Random	0.45	0.47
Top-100	0.72	0.74
Top-100 + k-DPP	0.79	**0.86**
Top-100 + k-DPP-Emb1	0.70	0.76
Top-100 + k-DPP-Emb2	**0.81**	0.83

Results

Table 5.1 contains the empirical results for the baseline approaches compared to semantic similarity aware diverse recommendations. Top-100 + Random randomizes the top 100 items of the candidate set, Top-100 just uses the top 100 ranked items as it is, and Top-100 + k-DPP obtains the k-DPP sampling from the top 100 ranked items. As shown in the Table 5.1, applying k-DPP to the Top-100 will improve the diversity of the Top-N recommender systems.

5.5 Diversity Metrics and their Utility

One popular way of measuring diversity of a recommendation list is to look at rating vectors associated with the items and calculate similarity of items based on these rating vectors. One major problem with this approach is that if two movies have similar user rating vectors, that should lead to the conclusion that these are similarly liked, not the conclusion that their content is similar[7]. By this approach, most popular movies would have a high similarity even though from a content and taste perspective they could be highly diverse. Thus, even for prevalent metrics we do not have a consensus on the utility of these metrics. To further complicate things, one can consider a wide array of diversity measures based on content and attributes associated with entities when large amounts of side information is available. We need a structured approach to identify the utility of diversity metrics and such a framework would help us take informed decisions based on a wide range of diversity metrics.

Chapter 6

Music Playlist Completion and Continuation

Music streaming services such as Spotify have more than 2 Billion playlists and 30 million tracks in their catalog. Playlists have become the primary mode of consuming content on online streaming platforms. Thus playlist continuation and completion approaches have gained traction in the Recommender Systems field. In this chapter we show how we can represent a Music playlist data set as a HIN, where the nodes in the network are tracks, title keywords, artists and albums; the edges are the co-occurrence relationship between these heterogeneous nodes. Using this HIN, we can derive various domain driven distributional representations of vertices. We then show how these representations can be used as features in Learning to Rank (LTR) models for the task of music playlist continuation.

The popular approaches to recommendation and *ad hoc* retrieval tasks are largely distinct in the literature. In this work, we argue that many recommendation problems can also be cast as ad hoc retrieval tasks. To demonstrate this, we build a solution for the RecSys 2018 Spotify challenge[1] by combining standard ad hoc retrieval models and using popular retrieval tools sets. We draw a parallel between the playlist continuation task and the task of

[1]https://recsys-challenge.spotify.com/

finding good expansion terms for queries in ad hoc retrieval, and show that standard pseudo-relevance feedback can be effective as a collaborative filtering approach. We also use ad hoc retrieval for content-based recommendation by treating the input playlist title as a query and associating all candidate tracks with meta-descriptions extracted from the background data. The recommendations from these two approaches are further supplemented by a nearest neighbor search based on track embeddings learned by a popular neural model. Our final ranked list of recommendations is produced by a learning to rank model[118]. Our proposed solution using ad hoc retrieval models achieved a competitive performance on the music recommendation task at RecSys 2018 challenge—finishing at rank 7 out of 112 participating teams and at rank 5 out of 31 teams for the main and the creative tracks, respectively.

6.1 Introduction

Recommendation and ad hoc retrieval are two important information retrieval tasks. Given a list of previously viewed items, a recommender system may suggest new items to the user by considering past interactions between all users and all items (*collaborative filtering*[119]) , or it may suggest new items that share similar attributes to the already viewed items (*content-based filtering*[120]) —or it may adopt a *hybrid* approach. In contrast, in ad hoc retrieval[121] the user expresses an explicit information need—typically in the form of a short text query—and the retrieval system responds with a ranked list of relevant information resources (e.g., documents or passages) based on the estimated match between the query and the document text. The popular approaches to recommendation and ad hoc retrieval tasks are largely distinct in the literature, although the two tasks share many similar properties.

The 2018 edition of the RecSys Challenge[122] featured the Spotify automatic playlist continuation task. The goal is to recommend additional tracks for a playlist for which (either or both of) the title and a number of existing tracks are known. A dataset containing one million Spotify playlists[2] is provided. This Million Playlist Dataset (MPD) can be used as background data, as well as for generating training examples and for offline evaluation.

[2]Million Playlist Dataset, official website hosted at https://recsys-challenge.spotify.com/

Looking through the lens of a typical recommender system, we may approach this task as a collaborative filtering problem considering the playlist-track membership matrix derived from the background data. A track may also be described by its own title, the primary artist name, the parent album name, and even the names of the playlists in which it occurs in the background data. These descriptions can be useful for content-based filtering. However, in this work we explore how standard ad hoc retrieval methods and tools can be useful to solve this recommendation task, using similar signals as collaborative filtering and content-based recommendation models.

We generate a collection of pseudo-documents, each of which corresponds to a playlist in the background data. The tracks in the playlist are treated as the terms in the document. We use a standard retrieval system to index these pseudo-documents. An input playlist—for which we should recommend new tracks—is treated as a query with its member tracks as the query terms. Using pseduo-relevance feedback (PRF)[123;124] we generate new expansion tracks for the query and present these as our recommendations for the input playlist. As this approach only considers past track-playlist membership information, we expect this method to recommend tracks similar to the collaborative filtering approach.

The title of the input playlist, if provided, can also be an important relevance signal. For example, if the input playlist title is "running music", then tracks from other playlists titled "running jams" or "running mix" may be good candidates for recommendation. Therefore, we create a second collection where each pseudo-document corresponds to a track in the background data. We concatenate the titles of all the background playlists that contain the track to generate the content for these pseudo-documents. Meta-descriptions about the track—such as, its: title, primary artist name, and parent album name—can be similarly useful for matching against the input playlist title, and be included as part of the pseudo-documents. We index this second collection and produce additional candidates by considering the input playlist title, if available, as a query to an ad hoc retrieval system.

Finally, we learn track embeddings using the popular *Word2Vec* model[80] and generate additional recommendations by a nearest-neighbour search in the learned latent space. The candidates from all three approaches are combined and re-ranked using a LambdaMART model[125]. By using only standard IR tools and methods, we built a solution that is competitive with other top performing submissions at the RecSys 2018 Spotify Challenge.

6.2 The RecSys 2018 Challenge

Spotify—an online music streaming company[3]—co-organized the RecSys 2018 challenge. The goal of this year's challenge was music recommendation—to suggest new tracks for playlist continuation. As part of this challenge, Spotify released a dataset containing one million randomly sampled user generated playlists that are publicly available to any users of the music streaming platform. The dataset includes playlists that were created between January 1, 2010 and November 1, 2017 by users who are at least 13 years old and resident in the United States. Any private user information is excluded from the dataset, and adult and offensive content scrubbed. Additional constraints placed on the inclusion of any playlist in this dataset include:

1. a minimum number other playlists that should contain the same title,
2. a minimum of three distinct artists and two distinct albums in the playlist,
3. at least one follower other than the creator, and
4. no less than five and no more than 250 tracks in the playlist.

The demographic distribution of the users who contributed to the dataset—according to the challenge website[4]—is reproduced in Figure 6.1.

The challenge dataset contains ten thousand playlists. For each playlist $\Phi = \phi_{\text{seed}} \cup \phi_{\text{held}}$, a set of tracks $\phi_{\text{seed}} = \{tr_1, tr_2, \ldots, tr_m\}$ are provided as seed tracks and the remaining tracks $\phi_{\text{held}} = \{tr_1, tr_2, \ldots, tr_n\}$ have been heldout. Optionally, the title T_Φ of the playlist Φ is also provided. The recommendation task involves predicting the heldout tracks in ϕ_{held} given ϕ_{seed} and optionally T_Φ. The number of heldout tracks n for each playlist Φ is known and

[3]https://www.spotify.com/
[4]https://recsys-challenge.spotify.com/dataset

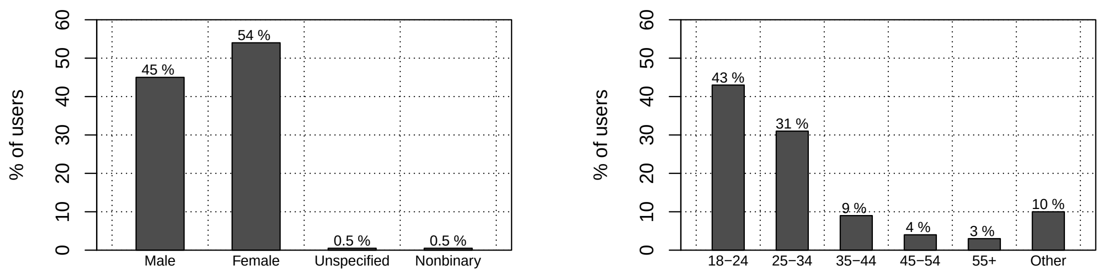


Figure 6.1: *Demographics of users who contributed to the MPD by Left: gender and Right: age.*

each playlist in the challenge set belongs to one of the following ten categories based on the information provided.

(i) the title only,

(ii) the title and the first track,

(iii) the title and the first five tracks,

(iv) the first five tracks only,

(v) the title and the first ten tracks,

(vi) the first ten tracks only,

(vii) the title and the first 25 tracks,

(viii) the title and 25 random tracks,

(ix) the title and the first 100 tracks, and

(x) the title and 100 random tracks.

When track information is provided, each track *tr* is described by:

(i) its position in the playlist,

(ii) the track name,

(iii) the track URI,

(iv) the primary artist name,

(v) the primary artist URI,

(vi) the album name,

(vii) the album URI, and

(viii) its duration.

The challenge set is sampled following the same guidelines as the MPD. For each playlist, the recommender system needs to generate a ranked list of exactly 500 distinct tracks ϕ_{pred} with no overlap with the seed tracks ϕ_{seed} provided as part of the playlist information.

Submissions are accepted under two different tracks—the main track and the creative track. For the main track, participants were allowed to were allowed to use only the MPD dataset. For the creative track, participants are allowed to use external data for making the recommendations. The use of external data, however, is restricted to those that are publicly available to all participants.

Each submission is evaluated based on three different metrics:

1. R-precision[63], with partial credit for artist match even if the track is incorrect
2. Normalized Discounted Cumulative Gain (NDCG)[65]
3. Recommended songs clicks, computed as:

$$clicks = min\{\lfloor (r-1)/10 \rfloor, 51\} \tag{6.1}$$

   where, r is the highest rank of a relevant track, if any.

The challenge leaderboard ranked each participants based on the Borda Count[126] election strategy over all the three specified metrics. The Recsys challenge[5] ran from January-2018 through June 30, 2018. From March-2018 to June 30, 2018 —the submission stage– participating teams were allowed to submit one challenge set-prediction per day for each of the

[5]http://www.recsyschallenge.com/2018/

main and creative tracks. During the submission stage, the leaderboard was calculated and updated daily based on 50% of the challenge dataset. When the challenge was finished, the entire leaderboard was recalculated using the complete challenge dataset and the final submission from each team.

6.3 Our approach

Our proposed solution consists of a candidate generation stage and a re-ranking stage. To recall a diverse set of candidates for ranking, we employ three different candidate generation strategies. Two of these approaches depend on track co-occurrence information, and the other approach models the relationship between tracks and the titles of parent playlists. Two of the approaches are implemented using Indri[6]—a standard ad hoc retrieval system—while the other employs a nearest neighbor-based lookup. We describe all three candidate generation methods and the re-ranking model in the next section.

[6]http://www.lemurproject.org/indri/

Table 6.1: *The full list of features that our learning to rank model considers. The features are categorized based on whether they depend only on the input playlist or the candidate track, or both.*

Features	Types
Input playlist only features	
Is playlist title available	Binary
Number of total tracks	Integer
Number of held out tracks	Integer
Ratio of number of unique albums to number of tracks	Float
Ratio of number of unique artists to number of tracks	Float
Ratio of frequency of most frequent album to number of tracks	Float
Ratio of frequency of most frequent artist to number of tracks	Float
Playlist title contains any of the words: top, best, popular, hot, or hits	Binary
Playlist title contains any of the words: latest, new, or recent	Binary
Playlist title contains any of the words: remix, remixed, or remixes	Binary
Candidate track only features	
Ratio of number of background playlists containing this track to total number of background playlists	Float
Ratio of number of background playlists containing this artist to total number of background playlists	Float
Ratio of number of background playlists containing this album to total number of background playlists	Float
Track title contains any of the words: remix, remixed, or remixes	Binary
Ratio of number of background parent playlists with title containinng any of the words: top, best, popular, hot, or hits to total number of background playlists	Float
Ratio of number of background parent playlists with title containing any of the words: lates, new, or recent to total number of background playlists	Float
Ratio of number of background parent playlists with title containing any of the words: remix, remixed, or remixes to total number of background playlists	Float
Input playlist and candidate track dependent features	
Rank in top 1000 candidates from QE, set to 1001 if not present	Integer
Rank in top 500 candidates from META1, set to 501 if not present	Integer
Rank in top 500 candidates from META2, set to 501 if not present	Integer
Rank in top 250 candidates from EMB1, set to 251 if not present	Integer
Rank in top 250 candidates from EMB2, set to 251 if not present	Integer
Rank in top 250 candidates from EMB3, set to 251 if not present	Integer
Rank in top 250 candidates from EMB4, set to 251 if not present	Integer
Ratio of number of tracks in playlist from same artist to number of tracks in playlist	Float
Ratio of number of tracks in playlist from same album to number of tracks in playlist	Float

6.3.1 Candidate generation

Playlist completion as query expansion (QE) In PRF[123;124], given a query q of m terms $\{t_1, t_2, \ldots, t_m\}$, first a set of k documents $D = \{d_1, d_2, \ldots, d_k\}$ are retrieved and based on these retrieved documents D the query is updated to q'. The translation from q to q' typically involves addition of new terms from D to the original query q. A new round of retrieval is performed using q' and the newly retrieved documents presented to the user.

Let us consider individual tracks as terms and playlists as text—like a document or a query—containing one or more terms. Let us also assume that we have an incomplete playlist ϕ_{seed} which is derived from an original playlist Φ. Let C be the collection of all playlists in the MPD and let $C' = C \cup \{\Phi\}$ be an imaginary collection created by adding Φ to C. Now, say, we want to retrieve Φ from C' but we are only provided ϕ_{seed} as a query. We know that we can obtain a smoother estimate of the unigram distribution of terms (or tracks) in Φ—and hence a better retrieval performance on this retrieval task—by first expanding ϕ_{seed} to $\phi_{\text{exp}} = \phi_{\text{seed}} \cup \phi_{\text{new}}$, where ϕ_{new} is the set of additional "query terms" identified by performing PRF over the collection C. While we do not, in fact, have C' and nor are we interested in retrieving Φ from this imaginary collection, it is interesting to note that PRF over C starting from ϕ_{seed} can help us identify a set of terms (or tracks) that are potentially from Φ but missing in ϕ_{seed}. Estimating ϕ_{new} accurately is similar to our playlist completion task. We note that a similar approach has been previously proposed for collaborative filtering[127;128].

Motivated by this, we use Indri to index a collection of all the the playlists in the MPD, where each playlist is a sequence of track identifiers. Given an incomplete playlist ϕ_{seed}, we retrieve a set of k playlists c from the collection and identify good expansion terms (or tracks) using RM1[129].

$$p(tr|\theta_\Phi) = \sum_{\phi \in c} p(tr|\theta_\phi) \prod_{\bar{tr} \in \phi_{\text{seed}}} p(\bar{tr}|\theta_\phi) \tag{6.2}$$

$$p(tr|\theta_\phi) = \frac{|\phi \cap \{tr\}|}{|\phi|}, \qquad \text{without smoothing} \tag{6.3}$$

The top candidate tracks ranked by $p(tr|\theta_\Phi)$ are considered for recommendation. In the rest of this chapter, we refer to this candidate generation strategy as QE for Query Expansion.

Ad hoc track retrieval using meta descriptions (META) In ad hoc retrieval, a document representation may depend on its own content such as: title; body text; external sources of descriptions; anchor text; clicked queries[130;131]. Similarly, we can describe a track by its own title, the primary artist name, and the parent album name—or by the titles of all the playlists in which it appears. All of this meta information about the track may be useful for our recommendation task. Given an input playlist title T_Φ, we can query a collection of pseudo-documents—where each document contains meta descriptions for a track—using a standard retrieval system, such as Indri. The retrieved ranked list of tracks can be considered as candidates for the playlist completion task. Based on this intuition, we generate two collections—one that describes tracks by their parent playlist titles and another that describes a track by its own title, primary artist name, and album name. The separate sets of candidates retrieved based on each of these two collections are referred to as META1 and META2, respectively, in the rest of this chapter. In our specific implementation, we use BM25[93] as the retrieval model and each document is generated by concatenation of the constituent text descriptions, similar to Robertson et al.[131].

Nearest neighbor search using track embeddings (EMB) Instead of comparing the query and the document text in the term space, some ad hoc retrieval models[132–134], we compute the query and the document representations as a centroid of their term embeddings and estimate their similarity in the latent space. A similar strategy may be useful for the playlist completion task. We experiment with a number of unsupervised approaches to learn the track embeddings that do not require any additional manual annotations.

First, we consider tracks as terms and playlists as documents containing a sequence of tracks. We employ the popular CBOW model from *Word2Vec*[80] to learn track embeddings on this pseduo-document collection C. A fixed size window is moved over each playlist and the model is trained by trying to predict the track in the middle of the window correctly given all the other tracks within the same window. This translates to minimizing the following loss,

$$\mathcal{L}_{\text{CBOW}} = \sum_{\phi \in C} \sum_{i}^{|\phi|} -\log\big(p(\vec{v}_i | \sum_{i-k \leq j \leq i+k, j \neq i} \vec{v}_j)\big) \tag{6.4}$$

where, $\vec{v}_i$ is the embedding of the i^{th} track in the playlist ϕ. Similar to Mikolov et al.[79], our track embeddings are trained with negative sampling instead of the full softmax over the complete track collection.

A playlist representation can be derived from both its member tracks $\{tr_1, tr_2, \ldots, tr_m, \}$ as well as its title T_ϕ. An analogy can be drawn to two collections in two different languages with document aligned across the collections. Vulić and Moens[135] consider a similar scenario in the context of cross-lingual retrieval and propose to learn a shared embedding space for terms from both languages by merging the two versions of each document from respective languages into a single pseudo-document. Motivated by their approach, we generate a collection of playlists where each pseudo-document is constructed by interspersing the member tracks and the playlist title terms. We train a CBOW model on this collection as our second approach to learn track embeddings.

The MPD contains four different types of entities—playlists, tracks, artists, and albums. Alternatively, we can view this dataset as a heterogeneous information network (HIN) . A HIN is defined as a directed graph $G = \{V, E\}$ with an entity mapping function $\xi : \mathbb{V} \rightarrow A$ and a edge type mapping function $\psi : \mathbb{E} \rightarrow R$ where each node v $\in \mathbb{V}$ belongs to one particular entity type $\xi(\text{v}) \in \mathcal{A}$ and each edge e $\in \mathbb{E}$ belongs to a relationship type $\psi(\text{e}) \in \mathcal{R}$. The edge weights associated between vertices with the relationship context $\psi(\text{c}) \in \mathcal{R}$ is captured as a preference matrix $\mathcal{W}_c$. Finally, a metapath defines a composite relationship by an ordered sequence of edge types specified in the HIN schema $\mathcal{S}_G = (\mathcal{A}, \mathcal{R})$. A number of previous studies have explored methods to learn node embeddings in homogeneous[81–83] and heterogeneous[136;137] graphs. In particular, Dong et al.[136] propose metapath-based random walks in heterogeneous networks to generate neighborhood representations that capture semantic relationships between different types of nodes in the graph followed by training a *Word2Vec* model on this neighborhood data to learn node embeddings. We adopt a similar approach based on two different meta-path definitions: artist→track→playlist→artist (ATPA) and track→playlist→track (TPT) . In summary, we learn track embeddings based on four different approaches:

1. EMB1: CBOW over playlists as documents and tracks as terms
2. EMB2: CBOW over interspersed member tracks and title terms for a playlist
3. EMB3: CBOW over the ATPA metapath
4. EMB4: CBOW over the TPT metapath

After training, we represent an input playlist ϕ_{seed} as the average of its member track embeddings $\vec{v}_{\text{seed}}$. New recommendation candidates are identified by finding tracks that have high cosine similarity with $\vec{v}_{\text{seed}}$. The embedding size is fixed to 200 dimensions for all four approaches and the window size for *Word2Vec* at 20 for EMB1 and EMB2 and at 5 for EMB3 and EMB4.

6.3.2 Learning to Rank

We take the union of all the candidates generated by each of the approaches described in Section 6.3.1. More precisely, we take the top 1000 candidates from QE, top 500 candidates each from META1 and META2, and top 250 candidates each from EMB1, EMB2, EMB3, and EMB4. We re-rank these candidates using a learning to rank (LTR)[138] model. We choose LambdaMART[125] with 100 trees and 50 leaves per tree as our model. We use the publicly available implementation in RankLib[7] for our experiments. We train the model with a learning rate of 0.1 and optimize for NDCG@10 for our main track submission and for NDCG@500 for our submission to the creative track. The full list of features used by the LTR model is specified in Table 6.1.

[7] https://sourceforge.net/p/lemur/wiki/RankLib/

Table 6.2: *Offline evaluation results for individual candidate sources and the combined LTR model output. For the combined model, we only measured the metrics at rank 500. The combined model achieves the best performance while QE emerges as the best candidate source. Note that for the clicks metric a lower value indicates a better performance.*

	Recall				RPrec				NDCG				Clicks
Model	@10	@250	@500	@1000	@10	@250	@500	@1000	@10	@250	@500	@1000	@500
QE	0.072	0.392	0.497	0.596	0.063	0.129	0.129	0.129	0.204	0.264	0.303	0.337	05.129
META1	0.033	0.232	0.309	0.393	0.032	0.100	0.100	0.100	0.160	0.181	0.217	0.252	08.839
META2	0.001	0.012	0.016	0.018	0.001	0.003	0.003	0.003	0.003	0.007	0.009	0.010	47.857
EMB1	0.025	0.129	0.174	0.234	0.022	0.038	0.038	0.038	0.065	0.084	0.099	0.118	21.740
EMB2	0.031	0.156	0.200	0.250	0.028	0.049	0.049	0.049	0.087	0.104	0.119	0.135	17.531
EMB3	0.042	0.174	0.214	0.261	0.038	0.065	0.065	0.065	0.116	0.126	0.140	0.155	21.112
EMB4	0.048	0.219	0.268	0.320	0.043	0.078	0.078	0.078	0.138	0.155	0.173	0.190	17.174
All candidate sources + LTR	-	-	0.513	-	-	-	0.134	-	-	-	0.313	-	04.380

Table 6.3: *Recall advantage of embeddings over QE. Recall advantage indicates the additional percentage of tracks retrieved by Embedding approaches over QE*

k	Emb1	Emb2	Emb3	Emb4
10	0.05895757051	0.05864550538	0.05905754329	0.05741188187
25	0.06884732689	0.07195282216	0.064488787	0.07116784649
50	0.07002885564	0.07574552345	0.06740902447	0.0741944282
100	0.06842808039	0.07955881428	0.07140596821	0.07707550985
250	0.06737840437	0.08299675109	0.07712303414	0.08266640182
500	0.06444418757	0.0772983969	0.07842230649	0.08216979349
1000	0.05895158878	0.07073236345	0.078430483	0.08179223353

During the LTR model training, we use 75% of the MPD for candidate generation and feature computation. From the remaining portion we use 50K playlists to train the LTR model and 5K playlists for offline evaluation. For each playlist in both the train and the evaluation, we hold out some of the member tracks—and optionally the playlist title—to generate a dataset with similar distributions as the challenge set. After finalizing the LTR model, we regenerate the candidates and recompute the features using the full MPD for the final challenge submission.

An open source implementation of our framework is available at: `https://github.com/skallumadi/BachPropagate`.

6.4 Results

Table 6.3 shows the offline evaluation results for the individual candidate generation strategies and the final combined output of the LTR model. Among the different candidate sources, QE demonstrates the strongest performance across all four metrics and all rank positions. While META1 shows reasonable performance, META2 achieves modest results most likely because the challenge set is designed such that each playlist containts a diverse set of artists and albums. So matching the input playlist title with the candidate track's title or its album/artist name does not add enough value. EMB4 fares the best among all the track embedding-based approaches. The LTR model that re-ranks a combined set of candidates from all the different sources performs best and shows significant improvement over the strongest individual source-QE.

The final standing on the RecSys 2018 challenge for the main and the creative tracks are shown in Table 6.4. Our submission based on the framework described in this chapter features among the top ten teams out of 112 participants on the main track and among the top five teams out of 31 teams on the creative track. Our submission also ranked among the top five teams based on the clicks metric alone on both tracks. We achieved this competitive performance based on simple applications of standard IR models. Our approach may be improved even further by incorporating more advanced retrieval models, including those based on recent neural and other machine learning-based approaches[95].

6.5 Conclusion

In this chapter, we have argued that ad hoc retrieval models can be useful for recommendation tasks. However, so far we have based our argument solely on retrieval performance. Another important consideration in this debate is the runtime efficiency. Using inverted index and other specialized data structures, typical web scale IR systems can retrieve the relevant results under a second from collections containing more than billions of items[139]. The Recsys 2018 challenge does not consider runtime efficiency. It is likely that our argument for applying ad hoc retrieval models to recommendation tasks may be strengthened if we consider model response times.

Finally, because our main goal in this work was to achieve a competitive performance at this year's RecSys challenge, the current study is focused primarily on empirical results. However, a theoretical comparison of ad hoc retrieval models and recommender systems may reveal more insights and opportunities in the intersection of these two research communities. We conclude by highlighting this as an important direction for future work in this area.

However, at the end of the competition the final ranking was computed based on the full set. For more details, we refer the interested readers to the official rules as listed on the challenge website: `https://recsys-challenge.spotify.com/rules`.

Table 6.4: *The final RecSys 2018 spotify challenge leaderboards. Our submissions are highlighted in bold. Only the top 10 teams from the leaderboards are shown. The total number of participating teams was 112 and 31 for the main and the creative tracks, respectively. For the clicks metric a lower value indicates a better performance.*

[a]

		RPrec		NDCG		Clicks		
#	Team name	Value	Rank	Value	Rank	Value	Rank	Borda
1	vl6	0.224	1	0.395	1	1.784	2	329
2	hello world	0.223	2	0.393	2	1.895	6	323
3	Avito	0.215	6	0.385	4	1.782	1	322
4	Creamy Fireflies	0.220	3	0.386	3	1.934	7	320
4	MIPT_MSU	0.217	4	0.382	5	1.875	4	320
6	HAIR	0.216	5	0.380	6	2.182	13	309
7	KAENEN	0.209	11	0.375	8	2.054	10	304
7	**BachPropagate**	**0.209**	**12**	**0.374**	**12**	**1.883**	**5**	**304**
9	Definitive Turtles	0.209	13	0.375	7	2.078	11	302
10	IN3PD	0.208	14	0.371	14	1.952	8	297

Main track [b]

		RPrec		NDCG		Clicks		
#	Team name	Value	Rank	Value	Rank	Value	Rank	Borda
1	vl6	0.223	1	0.394	1	1.785	1	90
2	Creamy Fireflies	0.220	2	0.385	2	1.925	4	85
3	KAENEN	0.209	3	0.375	3	2.048	6	81
4	cocoplaya	0.202	7	0.366	6	1.838	2	78
5	**BachPropagate**	**0.202**	**6**	**0.366**	**5**	**2.003**	**5**	**77**
6	Trailmix	0.206	4	0.370	4	2.259	9	76
7	teamrozik	0.205	5	0.361	7	2.164	8	73
8	Freshwater Sea	0.195	9	0.350	9	2.130	7	68
9	Team Radboud	0.198	8	0.356	8	2.293	11	66
10	spotif.ai	0.192	10	0.339	11	2.267	10	62

Creative track

Chapter 7

Conclusions and Future work

In this chapter, we summarize the various research contributions and the future work associated with the dissertation.

7.1 Summary and Conclusions

In this work we propose metapath-based approaches for representation learning in HIN, and show how these learnt representations can be used for providing recommendations in a heterogeneous information network. The metapath-based path-constrained random walk framework, which we use to capture the interaction patterns between entities in the HIN, generates distributed representations of these entities. We show that using ratings as strength of association between the entities and by using non-linear associations we can perform recommendations in a HIN with weighted edges. We also highlight the utility of using the distributional representations for exploratory search, information retrieval, and recommendations. We propose two query-oriented search mechanisms that help the user become an active participant instead of a passive recipient of recommendation algorithms. Diversity of results generated by machine learning algorithms has recently become a topic of importance because of the prevalence of filter bubbles[7]. By learning semantic relatedness of entities in various contexts, we show that the diversity of recommendation lists can be improved. These learnt representations can further be used in downstream tasks such as improving diversity of the ranked lists, playlist recommendation and playlist continuation in list oriented domains such as Spotify. In the following sections we summarise findings, review research objectives and claims drawn from the results.

7.1.1 Representation Learning in HIN using Constrained Random Walks

We can encode the domain knowledge as a set of constrained random walks, thus capturing the various interaction patterns present within the network. The assumption here is that these interaction patterns, capture a semantic context within which a user expresses preference over an item. Using metapaths as blue prints for these interaction patterns, we can generate random walks. We show how random walks can be performed over weighted and unweighted networks; and how non-linearities can be injected into weighted random walks to make them more or less biased over the ratings. We apply the metapath approach over IMDB network and citation network domains. We show the effectiveness of weighted and unweighted metawalks over these diverse set of domains. Metawalk approaches perform better than approaches such as DeepWalk[81], LINE[83] and Node2vec[82] as they do not take the type information into account. Further more, we show how metawalk approaches compare to other representation learning approaches with varying degrees of sparsity. Type-aware metawalk approaches have a clear advantage over other approaches in the presence of high sparsity.

7.1.2 New Modalities for Interaction for Interactive Search and Recommendations

We show the utility of using the distributed representations for interactive search and recommendations. The semantic embeddings generated from heterogeneous knowledge sources combined with user preferences can be used to refine a user's information needs. This representational modeling of users, entities and their associated properties opens up new modalities of interactions for the users to gravitate towards their requirements. We propose the use of semantic embeddings for two kinds of interactive recommendation modalities: 1) exemplar-based recommendations 2) *"less like this/more like this"* style recommendations. In our opinion providing these modalities will boost the expressive power of exploratory search and recommender systems. With exemplar-based recommendations, the user provides their information need in the form of a few examples; the system then captures the context within which these examples are related and then identifies more items that fit this context. The user can iteratively interact with the system to edit the recommendations to fine tune the results. Thus, a co-operative and interactive feedback mechanism can be established to explore the item space.

7.1.3 Using Representation Learning to Improve Semantic Diversity of Ranked Lists

Top-N Recommender Systems usually suffer from intra-list diversity as they are tailored for relevance and predicted rating accuracy. This problem is magnified in the case of cold-start setting - resulting in users being restricted to popular set of items and can result in a *"rich getting richer eco-system"*. As a result, in recent years, more attention has been paid to improving the diversity of recommender system results. List creation has become a popular way for users to express preferences over items on online platforms such as imdb.com and goodreads.com. These user-curated lists tend to contain a coherent semantic representation of the domain the list of items belong to. List curation can be seen as a way to capture fine grained topic-specific item-lists by users. Understanding and modeling user preferences expressed in these curated lists can help with diverse set of applications such as recommendations, user modeling, session understanding etc. As part of this dissertation, I propose an approach to improve the diversity of results generated by Top-N recommender systems, by using Determinantal Point Processes (DPPs) over user curated lists in the movie domain and incorporating them to re-rank the top-N recommender systems. We further show the utility of distributed representations learnt over user curated lists to improve the diversity of results, the two domain-driven metapaths capture different interaction patterns over the lists and thus can be used for two different contexts of diversity. In this work, we use the user curated lists in the imdb.com domain. We evaluate our approach over the *MovieLens* 1-Million dataset and compare the results with other baseline approaches. Our results show that incorporating semantic similarity expressed in user lists as a diversity proxy, results in a more diverse set of recommendations.

7.1.4 Using Semantic Embeddings for Recommendation Tasks, with Application to Playlist Completion and Continuation

One of the goals for representation learning, as stated in Chapter 3, is to use the embeddings learnt over information networks as features for machine learning algorithms. We use the distributed representations of tracks present in playlists to improve the metrics for the music playlist continuation task. We learn four diverse set of semantic embeddings over the MPD dataset. Using a metapath approach over HIN, as defined in Chapter 4, we learn track embeddings based on four different semantics:

1. EMB1: CBOW over playlists as documents and tracks as terms
2. EMB2: CBOW over interspersed member tracks and title terms for a playlist
3. EMB3: CBOW over the ATPA metapath
4. EMB4: CBOW over the TPT metapath

Each of these embeddings capture different interaction patterns over the music playlist network. We then use a LTR model over the various feature spaces to identify candidate tracks for partial playlists. While META1 shows reasonable performance, META2 achieves modest results most likely because the challenge set is designed such that each playlist contains a diverse set of artists and albums. EMB4 fares the best among all the track embedding-based approaches. The LTR model that re-ranks a combined set of candidates from all the different sources performs best and shows significant improvement over the strongest individual source QE. Thus demonstrating the utility of using metapath track representations for downstream recommendation tasks.

In conclusion, we have shown how item and user representations can be learnt over diverse domains such as movies, citation networks and music. We have also seen how these representations can be used to capture the semantic structure of networks. These representations can be used as features to improve the performance of machine learning algorithms for tasks such as user modeling, personalization and recommendations.

7.2 Future work

7.2.1 Deep Semantic Models that Utilize Semantic Embeddings for Information Retrieval and Discovery

In information retrieval and web search, Deep Structured Semantic Models (DSSM) have been shown to be quite effective in meeting the user's information needs[140]. Inspired by these models we propose using the entity embeddings learnt from HIN as features for the DSSM style model shown in Figure 7.1. The intuition behind using the embeddings generated over HIN is to supply semantic information in addition to co-occurrence information to the deep learning model and then use fully connected layers to capture the interactions between the co-occurrence patterns and the semantic patterns.

7.2.2 High Accuracy Recall to Improve Performance of Search Systems

Most modern search engines perform search in two stages. In the first stage, a large subset of documents that match the input query are obtained by a simple scoring function such as tf-idf or BM25[93]. In the second phase, the top-k (by score) of these matching documents

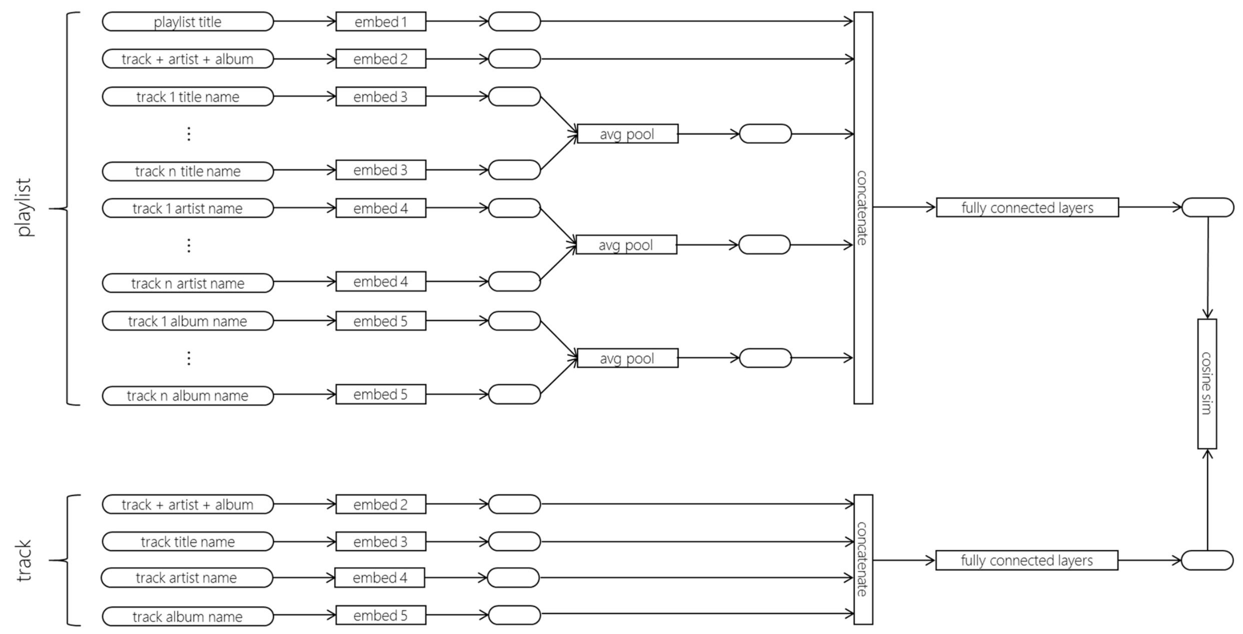


Figure 7.1: *DSSM style model for playlist continuation task*

are "re-ranked" using a complex ranking function that uses a richer set of features. The candidate item generation phase is similar to the first stage of a search engine, where give a ordered/unordered set of entities, we retrieve related entities that could be part of the list. Then the candidate items are ranked by the "context fit" with respect to the items in the list as well as the metadata associated with the list in order to obtain a high quality set of candidates. For ambiguous queries and low information queries, the semantic information associated with the user's information need can be used to identify relevant documents and items, thus improving the efficiency of the candidate generation phase. Performing candidate generation and retrieving a high quality candidate list in an efficient manner is a significant aspect of list completion recommender systems as it affects user experience and semantic embeddings can be used to improve the quality of first phase of retrieval systems.

7.2.3 Cross-Domain Recommendations using Domain-Aware Meta-paths

One way of reducing the effects of sparsity in recommender systems is to perform cross-domain CF. In CDCF we can have multiple source domains and a target domain and the objective is to transfer knowledge in the form of user preferences from the source domains to the target domain. One central assumption to CDCF is that there should be an inherent relationship between the domains between which learning is performed. We can consider news, browsing activity, search and ads as closely-related domains, where the type of news we consume, the online articles we browse, the queries we search for, and the ads we click on can be considered to be inherently related. Thus knowledge from one or more of these domains could be used as side information to drive recommendations within another domain. By performing metawalks over connected domains — where the bridges can be common users, semantically similar tags, similar items etc — we can transfer knowledge across related domains. This can improve the performance of recommender systems when there is not enough preference information available regarding the items.

7.2.4 Retraining and Incremental Training for New Data, Users, and Items

As more user-item interaction data comes in, the distributed representations associated with items, users, and other entities need to be updated. Over time, there will also be a drift in user preferences. For example a user who starts off preferring "Romantic" books might slowly drift to other "Genres". A good recommender system, should take this drift in user interests over time into consideration while recommending items. Also new users, and new items are constantly added to the system. These users and items will have a smaller amount of preference, and behavior data associated with them. Taking all these variabilities into account, the embedding models and entity models would either need to be retrained, or incrementally trained. For example, in a news recommender system, user preferences are heavily biased towards their recent activity. Thus, in such a scenario, incremental training might not be a good strategy. This raises the following research problems: 1) How often should a system be retrained? 2) What aspects/which embeddings can be incrementally trained without loss of accuracy? 3) Can the embedding components be split in such a way that, some of the item embeddings can be incrementally trained and other embeddings need to be re-trained? And, can a joint model utilize both these incremental, and re-trained embeddings to provide recommendations?

Bibliography

[1] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. A neural probabilistic language model. *J. Mach. Learn. Res.*, 3:1137–1155, March 2003. ISSN 1532-4435. URL http://dl.acm.org/citation.cfm?id=944919.944966.

[2] Lidan Wang, Jimmy Lin, and Donald Metzler. A cascade ranking model for efficient ranked retrieval. In *Proc. SIGIR*, pages 105–114. ACM, 2011.

[3] Paul Resnick and Hal R. Varian. Recommender systems. *Commun. ACM*, 40(3):56–58, March 1997. ISSN 0001-0782. doi: 10.1145/245108.245121. URL http://doi.acm.org/10.1145/245108.245121.

[4] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781, 2013. URL http://arxiv.org/abs/1301.3781.

[5] Francesco Ricci, Lior Rokach, Bracha Shapira, and Paul B. Kantor. *Recommender Systems Handbook*. Springer-Verlag, Berlin, Heidelberg, 1st edition, 2010. ISBN 0387858199, 9780387858197.

[6] Asela Gunawardana and Christopher Meek. A unified approach to building hybrid recommender systems. In *Proceedings of the Third ACM Conference on Recommender Systems*, RecSys '09, pages 117–124, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-435-5. doi: 10.1145/1639714.1639735. URL http://doi.acm.org/10.1145/1639714.1639735.

[7] Tien T. Nguyen, Pik-Mai Hui, F. Maxwell Harper, Loren Terveen, and Joseph A. Konstan. Exploring the filter bubble: The effect of using recommender systems on content diversity. In *Proceedings of the 23rd International Conference on World Wide*

Web, WWW '14, pages 677–686, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-2744-2. doi: 10.1145/2566486.2568012. URL http://doi.acm.org/10.1145/2566486.2568012.

[8] Harald Steck, Roelof van Zwol, and Chris Johnson. Interactive recommender systems: Tutorial. In *Proceedings of the 9th ACM Conference on Recommender Systems*, RecSys '15, pages 359–360, New York, NY, USA, 2015. ACM. ISBN 978-1-4503-3692-5. doi: 10.1145/2792838.2792840. URL http://doi.acm.org/10.1145/2792838.2792840.

[9] Chen He, Denis Parra, and Katrien Verbert. Interactive recommender systems: a survey of the state of the art and future research challenges and opportunities. *Expert Systems with Applications*, 2016. doi: 10.1016/j.eswa.2016.02.013. URL http://web.ing.puc.cl/~dparra/pdfs/pre-print_ESWA_He_Parra_Verbert_2016.pdf.

[10] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*, WWW '01, pages 285–295, New York, NY, USA, 2001. ACM. ISBN 1-58113-348-0. doi: 10.1145/371920.372071. URL http://doi.acm.org/10.1145/371920.372071.

[11] RA Likert. A technique for measurement of attitudes. 22:1–, 01 1932.

[12] Xuan Nhat Lam, Thuc Vu, Trong Duc Le, and Anh Duc Duong. Addressing cold-start problem in recommendation systems. In *Proceedings of the 2nd international conference on Ubiquitous information management and communication*, ICUIMC '08, pages 208–211, New York, NY, USA, 2008. ACM. ISBN 978-1-59593-993-7. doi: 10.1145/1352793.1352837. URL http://doi.acm.org/10.1145/1352793.1352837.

[13] Manos Papagelis, Dimitris Plexousakis, and Themistoklis Kutsuras. Alleviating the sparsity problem of collaborative filtering using trust inferences. In *iTrust*, pages 224–239, 2005.

[14] Raymond J. Mooney and Loriene Roy. Content-based book recommending using learning for text categorization. In *Proceedings of the Fifth ACM Conference on Digital Libraries*, DL '00, pages 195–204, New York, NY, USA, 2000. ACM. ISBN 1-58113-231-X. doi: 10.1145/336597.336662. URL http://doi.acm.org/10.1145/336597.336662.

[15] Prem Melville, Raymod J. Mooney, and Ramadass Nagarajan. Content-boosted collaborative filtering for improved recommendations. In *Eighteenth national conference on Artificial intelligence*, pages 187–192, Menlo Park, CA, USA, 2002. American Association for Artificial Intelligence. ISBN 0-262-51129-0. URL http://dl.acm.org/citation.cfm?id=777092.777124.

[16] Justin Basilico Basilico and Thomas Hofmann. Unifying collaborative and content-based filtering. In *In ICML*, pages 65–72. ACM Press, 2004.

[17] Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. Grouplens: an open architecture for collaborative filtering of netnews. ACM, 1994. URL http://doi.acm.org/10.1145/192844.192905.

[18] Thomas Hofmann and Jan Puzicha. Latent class models for collaborative filtering. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, IJCAI '99, pages 688–693, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc. ISBN 1-55860-613-0. URL http://dl.acm.org/citation.cfm?id=646307.687583.

[19] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems, 2009.

[20] Yehuda Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '08, pages 426–434, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-193-4. doi: 10.1145/1401890.1401944. URL http://doi.acm.org/10.1145/1401890.1401944.

[21] Jon Herlocker, Joseph A. Konstan, and John Riedl. An empirical analysis of design choices in neighborhood-based collaborative filtering algorithms. *Inf. Retr.*, 5(4):287–310, October 2002. ISSN 1386-4564. doi: 10.1023/A:1020443909834. URL http://dx.doi.org/10.1023/A:1020443909834.

[22] Greg Linden, Brent Smith, and Jeremy York. Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing*, 7(1):76–80, January 2003. ISSN 1089-7801. doi: 10.1109/MIC.2003.1167344. URL http://dx.doi.org/10.1109/MIC.2003.1167344.

[23] Joseph A. Konstan, Bradley N. Miller, David Maltz, Jonathan L. Herlocker, Lee R. Gordon, and John Riedl. Grouplens: Applying collaborative filtering to usenet news. *Commun. ACM*, 40(3):77–87, March 1997. ISSN 0001-0782. doi: 10.1145/245108.245126. URL http://doi.acm.org/10.1145/245108.245126.

[24] Shumeet Baluja, Rohan Seth, D. Sivakumar, Yushi Jing, Jay Yagnik, Shankar Kumar, Deepak Ravichandran, and Mohamed Aly. Video suggestion and discovery for youtube: Taking random walks through the view graph. In *Proceedings of the 17th International Conference on World Wide Web*, WWW '08, pages 895–904, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-085-2. doi: 10.1145/1367497.1367618. URL http://doi.acm.org/10.1145/1367497.1367618.

[25] Xiaojin Zhu and Zoubin Ghahramani. Learning from labeled and unlabeled data with label propagation. 2002.

[26] M. W. Berry, S.T. Dumais, G.W. O'Brien, Michael W. Berry, Susan T. Dumais, and Gavin. Using linear algebra for intelligent information retrieval. *SIAM Review*, 1995.

[27] Daniel Billsus and Michael J. Pazzani. Learning collaborative information filters. In *Proceedings of the Fifteenth International Conference on Machine Learning*, ICML '98, pages 46–54, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc. ISBN 1-55860-556-8. URL http://dl.acm.org/citation.cfm?id=645527.657311.

[28] Ken Goldberg, Theresa Roeder, Dhruv Gupta, and Chris Perkins. Eigentaste: A constant time collaborative filtering algorithm. *Inf. Retr.*, 4(2):133–151, July 2001. ISSN 1386-4564. doi: 10.1023/A:1011419012209. URL http://dx.doi.org/10.1023/A:1011419012209.

[29] Thomas Hofmann. Latent semantic models for collaborative filtering. *ACM Trans. Inf. Syst.*, 22(1):89–115, January 2004. ISSN 1046-8188. doi: 10.1145/963770.963774. URL http://doi.acm.org/10.1145/963770.963774.

[30] Hyunsoo Kim and Haesun Park. Nonnegative matrix factorization based on alternating nonnegativity constrained least squares and active set method. *SIAM journal on matrix analysis and applications*, 30(2):713–730, 2008.

[31] Michael J. Pazzani and Daniel Billsus. The adaptive web. chapter Content-based Recommendation Systems, pages 325–341. Springer-Verlag, Berlin, Heidelberg, 2007. ISBN 978-3-540-72078-2. URL http://dl.acm.org/citation.cfm?id=1768197.1768209.

[32] Michal Kompan and Mária Bieliková. Content-based news recommendation. In Francesco Buccafurri and Giovanni Semeraro, editors, *E-Commerce and Web Technologies*, pages 61–72, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg. ISBN 978-3-642-15208-5.

[33] Owen Phelan, Kevin McCarthy, Mike Bennett, and Barry Smyth. Terms of a feather: Content-based news recommendation and discovery using twitter. In *ECIR*, 2011.

[34] Charu C Aggarwal. Content-based recommender systems. In *Recommender systems*, pages 139–166. Springer, 2016.

[35] Aaron van den Oord, Sander Dieleman, and Benjamin Schrauwen. Deep content-based music recommendation. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 2643–2651. Curran Associates, Inc., 2013. URL http://papers.nips.cc/paper/5004-deep-content-based-music-recommendation.pdf.

[36] Gediminas Adomavicius and Alexander Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Trans. on Knowl. and Data Eng.*, 17(6):734–749, June 2005. ISSN 1041-4347. doi: 10.1109/TKDE.2005.99. URL https://doi.org/10.1109/TKDE.2005.99.

[37] Charu C Aggarwal. Ensemble-based and hybrid recommender systems. In *Recommender Systems*, pages 199–224. Springer, 2016.

[38] Shlomo Berkovsky, Tsvi Kuflik, and Francesco Ricci. Cross-domain mediation in collaborative filtering. In *Proceedings of the 11th international conference on User Modeling*, UM '07, pages 355–359, Berlin, Heidelberg, 2007. Springer-Verlag. ISBN 978-3-540-73077-4. doi: 10.1007/978-3-540-73078-1_44. URL http://dx.doi.org/10.1007/978-3-540-73078-1_44.

[39] Bin Li. Cross-domain collaborative filtering: A brief survey. In *Proceedings of the 2011 IEEE 23rd International Conference on Tools with Artificial Intelligence*, ICTAI '11, pages 1085–1086, Washington, DC, USA, 2011. IEEE Computer Society. ISBN 978-0-7695-4596-7. doi: 10.1109/ICTAI.2011.184. URL http://dx.doi.org/10.1109/ICTAI.2011.184.

[40] Pinata Winoto and Tiffany Tang. If you like the devil wears prada the book, will you also enjoy the devil wears prada the movie? a study of cross-domain recommendations. *New Generation Computing*, 26:209–225, 2008. ISSN 0288-3635. URL http://dx.doi.org/10.1007/s00354-008-0041-0. 10.1007/s00354-008-0041-0.

[41] Bin Li, Qiang Yang, and Xiangyang Xue. Can movies and books collaborate?: cross-domain collaborative filtering for sparsity reduction. In *Proceedings of the 21st international jont conference on Artifical intelligence*, IJCAI'09, pages 2052–2057, San Francisco, CA, USA, 2009. Morgan Kaufmann Publishers Inc. URL http://dl.acm.org/citation.cfm?id=1661445.1661773.

[42] Bin Li, Qiang Yang, and Xiangyang Xue. Transfer learning for collaborative filtering

via a rating-matrix generative model. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 617–624, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-516-1. doi: 10.1145/1553374.1553454. URL http://doi.acm.org/10.1145/1553374.1553454.

[43] Weike Pan, Evan Wei Xiang, Nathan Nan Liu, and Qiang Yang. Transfer learning in collaborative filtering for sparsity reduction. In Maria Fox and David Poole, editors, *AAAI*. AAAI Press, 2010.

[44] Zhongqi Lu, ErHeng Zhong, Lili Zhao, Evan Wei Xiang, Weike Pan, and Qiang Yang. Selective transfer learning for cross domain recommendation. *CoRR*, abs/1210.7056, 2012.

[45] Weike Pan, Nathan Nan Liu, Evan Wei Xiang, and Qiang Yang. Transfer learning to predict missing ratings via heterogeneous user feedbacks. In Toby Walsh, editor, *IJCAI*, pages 2318–2323. IJCAI/AAAI, 2011. ISBN 978-1-57735-516-8.

[46] Weike Pan, Evan Wei Xiang, and Qiang Yang. Transfer learning in collaborative filtering with uncertain ratings. In Jörg Hoffmann and Bart Selman, editors, *AAAI*. AAAI Press, 2012.

[47] Bin Li, Xingquan Zhu, Ruijiang Li, Chengqi Zhang, Xiangyang Xue, and Xindong Wu. Cross-domain collaborative filtering over time. In *Proceedings of the Twenty-Second international joint conference on Artificial Intelligence - Volume Volume Three*, IJCAI'11, pages 2293–2298. AAAI Press, 2011. ISBN 978-1-57735-515-1. doi: 10.5591/978-1-57735-516-8/IJCAI11-382. URL http://dx.doi.org/10.5591/978-1-57735-516-8/IJCAI11-382.

[48] Orly Moreno, Bracha Shapira, Lior Rokach, and Guy Shani. Talmud: transfer learning for multiple domains. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, CIKM '12, pages 425–434, New York, NY,

USA, 2012. ACM. ISBN 978-1-4503-1156-4. doi: 10.1145/2396761.2396817. URL http://doi.acm.org/10.1145/2396761.2396817.

[49] Yu Zhang, Bin Cao, and Dit-Yan Yeung. Multi-domain collaborative filtering. *CoRR*, abs/1203.3535, 2012.

[50] Jian Tang, Jun Yan, Lei Ji, Ming Zhang, Shaodan Guo, Ning Liu, Xianfang Wang, and Zheng Chen. Collaborative users' brand preference mining across multiple domains from implicit feedbacks. In *AAAI'11*, pages –1–1, 2011.

[51] Yue Shi, Martha Larson, and Alan Hanjalic. Tags as bridges between domains: improving recommendation with tag-induced cross-domain collaborative filtering. In *Proceedings of the 19th international conference on User modeling, adaption, and personalization*, UMAP'11, pages 305–316, Berlin, Heidelberg, 2011. Springer-Verlag. ISBN 978-3-642-22361-7. URL http://dl.acm.org/citation.cfm?id=2021855.2021882.

[52] Yue Shi, Martha Larson, and Alan Hanjalic. Generalized tag-induced cross-domain collaborative filtering. *CoRR*, abs/1302.4888, 2013.

[53] Weiqing Wang, Zhenyu Chen, Jia Liu, Qi Qi, and Zhihong Zhao. User-based collaborative filtering on cross domain by tag transfer learning. In *Proceedings of the 1st International Workshop on Cross Domain Knowledge Discovery in Web and Social Network Mining*, CDKD '12, pages 10–17, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1555-5. doi: 10.1145/2351333.2351335. URL http://doi.acm.org/10.1145/2351333.2351335.

[54] Ignacio Fernández-Tobías, Iván Cantador, Marius Kaminskas, and Francesco Ricci. Cross-domain recommender systems: A survey of the state of the art.

[55] Ali Mamdouh Elkahky, Yang Song, and Xiaodong He. A multi-view deep learning approach for cross domain user modeling in recommendation systems. In *Proceedings of the 24th International Conference on World Wide Web*, WWW '15, pages 278–288, Republic and Canton of Geneva, Switzerland, 2015. International World Wide Web

Conferences Steering Committee. ISBN 978-1-4503-3469-3. doi: 10.1145/2736277.2741667. URL https://doi.org/10.1145/2736277.2741667.

[56] Jing Zhao, Xijiong Xie, Xin Xu, and Shiliang Sun. Multi-view learning overview: Recent progress and new challenges. *Information Fusion*, 38:43–54, 2017.

[57] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. Neural collaborative filtering. In *Proceedings of the 26th International Conference on World Wide Web*, WWW '17, pages 173–182, Republic and Canton of Geneva, Switzerland, 2017. International World Wide Web Conferences Steering Committee. ISBN 978-1-4503-4913-0. doi: 10.1145/3038912.3052569. URL https://doi.org/10.1145/3038912.3052569.

[58] Xiang Wang, Xiangnan He, Liqiang Nie, and Tat-Seng Chua. Item silk road: Recommending items from information domains to social users. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '17, pages 185–194, New York, NY, USA, 2017. ACM. ISBN 978-1-4503-5022-8. doi: 10.1145/3077136.3080771. URL http://doi.acm.org/10.1145/3077136.3080771.

[59] Jianxun Lian, Fuzheng Zhang, Xing Xie, and Guangzhong Sun. Cccfnet: A content-boosted collaborative filtering neural network for cross domain recommender systems. In *Proceedings of the 26th International Conference on World Wide Web Companion*, WWW '17 Companion, pages 817–818, Republic and Canton of Geneva, Switzerland, 2017. International World Wide Web Conferences Steering Committee. ISBN 978-1-4503-4914-7. doi: 10.1145/3041021.3054207. URL https://doi.org/10.1145/3041021.3054207.

[60] Chris Chatfield. *The analysis of time series: an introduction*. CRC Press, Florida, US, 6th edition, 2004.

[61] W Bruce Croft, Donald Metzler, and Trevor Strohman. *Search engines: Information retrieval in practice*, volume 283. Addison-Wesley Reading, 2010.

[62] Jonathan L. Herlocker, Joseph A. Konstan, Loren G. Terveen, and John T. Riedl. Evaluating collaborative filtering recommender systems. *ACM Trans. Inf. Syst.*, 22 (1):5–53, January 2004. ISSN 1046-8188. doi: 10.1145/963770.963772. URL http://doi.acm.org/10.1145/963770.963772.

[63] Ellen M Voorhees and Donna Harman. Common evaluation measures. In *The twelfth text retrieval conference (TREC 2003)*, pages 500–255, 2003.

[64] Brian McFee, Thierry Bertin-Mahieux, Daniel P.W. Ellis, and Gert R.G. Lanckriet. The million song dataset challenge. In *Proceedings of the 21st International Conference on World Wide Web*, WWW '12 Companion, pages 909–916, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1230-1. doi: 10.1145/2187980.2188222. URL http://doi.acm.org/10.1145/2187980.2188222.

[65] Kalervo Järvelin and Jaana Kekäläinen. Cumulated gain-based evaluation of ir techniques. *ACM Transactions on Information Systems (TOIS)*, 20(4):422–446, 2002.

[66] Nathan N. Liu and Qiang Yang. Eigenrank: A ranking-oriented approach to collaborative filtering. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '08, pages 83–90, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-164-4. doi: 10.1145/1390334.1390351. URL http://doi.acm.org/10.1145/1390334.1390351.

[67] John S. Breese, David Heckerman, and Carl Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, UAI'98, pages 43–52, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc. ISBN 1-55860-555-X. URL http://dl.acm.org/citation.cfm?id=2074094.2074100.

[68] Yifan Hu, Yehuda Koren, and Chris Volinsky. Collaborative filtering for implicit feedback datasets. In *Proceedings of the 2008 Eighth IEEE International Conference on Data Mining*, ICDM '08, pages 263–272, Washington, DC, USA, 2008. IEEE Computer Society. ISBN 978-0-7695-3502-9. doi: 10.1109/ICDM.2008.22. URL http://dx.doi.org/10.1109/ICDM.2008.22.

[69] Marius Kaminskas and Derek Bridge. Diversity, serendipity, novelty, and coverage: A survey and empirical analysis of beyond-accuracy objectives in recommender systems. *ACM Trans. Interact. Intell. Syst.*, 7(1):2:1–2:42, December 2016. ISSN 2160-6455. doi: 10.1145/2926720. URL http://doi.acm.org/10.1145/2926720.

[70] Daniel Kluver and Joseph A. Konstan. Evaluating recommender behavior for new users. In *Proceedings of the 8th ACM Conference on Recommender Systems*, RecSys '14, pages 121–128, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-2668-1. doi: 10.1145/2645710.2645742. URL http://doi.acm.org/10.1145/2645710.2645742.

[71] Cai-Nicolas Ziegler, Sean M. McNee, Joseph A. Konstan, and Georg Lausen. Improving recommendation lists through topic diversification. In *Proceedings of the 14th International Conference on World Wide Web*, WWW '05, pages 22–32, New York, NY, USA, 2005. ACM. ISBN 1-59593-046-9. doi: 10.1145/1060745.1060754. URL http://doi.acm.org/10.1145/1060745.1060754.

[72] Zellig Harris. Distributional structure. *Word*, 10(23):146–162, 1954.

[73] Warren Weaver. Translation. In William N. Locke and A. Donald Boothe, editors, *Machine Translation of Languages*, pages 15–23. MIT Press, Cambridge, MA, 1949/1955. Reprinted from a memorandum written by Weaver in 1949.

[74] Hinrich Schtze. Word space. In *Advances in Neural Information Processing Systems 5*, pages 895–902. Morgan Kaufmann, 1993.

[75] Ronan Collobert and Jason Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th*

International Conference on Machine Learning, ICML '08, pages 160–167, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-205-4. doi: 10.1145/1390156.1390177. URL http://doi.acm.org/10.1145/1390156.1390177.

[76] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1532–1543, 2014. URL http://aclweb.org/anthology/D/D14/D14-1162.pdf.

[77] Mihajlo Grbovic. Search ranking and personalization at airbnb. In *Proceedings of the Eleventh ACM Conference on Recommender Systems*, RecSys '17, pages 339–340, New York, NY, USA, 2017. ACM. ISBN 978-1-4503-4652-8. doi: 10.1145/3109859.3109920. URL http://doi.acm.org/10.1145/3109859.3109920.

[78] Chantat Eksombatchai, Pranav Jindal, Jerry Zitao Liu, Yuchen Liu, Rahul Sharma, Charles Sugnet, Mark Ulrich, and Jure Leskovec. Pixie: A system for recommending 3+ billion items to 200+ million users in real-time. In *Proceedings of the 2018 World Wide Web Conference*, WWW '18, pages 1775–1784, Republic and Canton of Geneva, Switzerland, 2018. International World Wide Web Conferences Steering Committee. ISBN 978-1-4503-5639-8. doi: 10.1145/3178876.3186183. URL https://doi.org/10.1145/3178876.3186183.

[79] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[80] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[81] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social

representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 701–710. ACM, 2014.

[82] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864. ACM, 2016.

[83] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. Line: Large-scale information network embedding. In *Proceedings of the 24th International Conference on World Wide Web*, pages 1067–1077. International World Wide Web Conferences Steering Committee, 2015.

[84] Hongwei Wang, Jia Wang, Jialin Wang, Miao Zhao, Weinan Zhang, Fuzheng Zhang, Xing Xie, and Minyi Guo. Graphgan: Graph representation learning with generative adversarial nets. *CoRR*, abs/1711.08267, 2017. URL http://arxiv.org/abs/1711.08267.

[85] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, pages 2672–2680, Cambridge, MA, USA, 2014. MIT Press. URL http://dl.acm.org/citation.cfm?id=2969033.2969125.

[86] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *J. Mach. Learn. Res.*, 11:3371–3408, December 2010. ISSN 1532-4435. URL http://dl.acm.org/citation.cfm?id=1756006.1953039.

[87] Shaosheng Cao, Wei Lu, and Qiongkai Xu. Deep neural networks for learning graph representations. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intel-*

ligence, AAAI'16, pages 1145–1152. AAAI Press, 2016. URL http://dl.acm.org/citation.cfm?id=3015812.3015982.

[88] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab, 1999.

[89] Daixin Wang, Peng Cui, and Wenwu Zhu. Structural deep network embedding. In *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 1225–1234, New York, NY, USA, 2016. ACM. ISBN 978-1-4503-4232-2. doi: 10.1145/2939672.2939753. URL http://doi.acm.org/10.1145/2939672.2939753.

[90] Ruslan Salakhutdinov and Geoffrey Hinton. Semantic hashing. *Int. J. Approx. Reasoning*, 50(7):969–978, July 2009. ISSN 0888-613X. doi: 10.1016/j.ijar.2008.11.006. URL http://dx.doi.org/10.1016/j.ijar.2008.11.006.

[91] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural language processing (almost) from scratch. *J. Mach. Learn. Res.*, 12:2493–2537, November 2011. ISSN 1532-4435. URL http://dl.acm.org/citation.cfm?id=1953048.2078186.

[92] Ye Qi, Devendra Singh Sachan, Matthieu Felix, Sarguna Janani Padmanabhan, and Graham Neubig. When and why are pre-trained word embeddings useful for neural machine translation? *CoRR*, abs/1804.06323, 2018. URL http://arxiv.org/abs/1804.06323.

[93] Stephen Robertson, Hugo Zaragoza, et al. The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends® in Information Retrieval*, 3(4):333–389, 2009.

[94] Hamed Zamani and W. Bruce Croft. Relevance-based word embedding. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '17, pages 505–514, New York, NY, USA, 2017. ACM.

ISBN 978-1-4503-5022-8. doi: 10.1145/3077136.3080831. URL http://doi.acm.org/10.1145/3077136.3080831.

[95] Bhaskar Mitra and Nick Craswell. An introduction to neural information retrieval. *Foundations and Trends® in Information Retrieval (to appear)*, 2018.

[96] David C. Liu, Stephanie Rogers, Raymond Shiau, Dmitry Kislyuk, Kevin C. Ma, Zhigang Zhong, Jenny Liu, and Yushi Jing. Related pins at pinterest: The evolution of a real-world recommender system. In *Proceedings of the 26th International Conference on World Wide Web Companion*, WWW '17 Companion, pages 583–592, Republic and Canton of Geneva, Switzerland, 2017. International World Wide Web Conferences Steering Committee. ISBN 978-1-4503-4914-7. doi: 10.1145/3041021.3054202. URL https://doi.org/10.1145/3041021.3054202.

[97] Mihajlo Grbovic and Haibin Cheng. Real-time personalization using embeddings for search ranking at airbnb. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery &#38; Data Mining*, KDD '18, pages 311–320, New York, NY, USA, 2018. ACM. ISBN 978-1-4503-5552-0. doi: 10.1145/3219819.3219885. URL http://doi.acm.org/10.1145/3219819.3219885.

[98] Hanghang Tong, Christos Faloutsos, and Jia-Yu Pan. Fast random walk with restart and its applications. In *Proceedings of the Sixth International Conference on Data Mining*, pages 613–622, Washington, DC, USA, 2006. IEEE. ISBN 0-7695-2701-9. doi: 10.1109/ICDM.2006.70. URL http://dx.doi.org/10.1109/ICDM.2006.70.

[99] Glen Jeh and Jennifer Widom. Simrank: A measure of structural-context similarity. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 538–543, New York, NY, USA, 2002. ACM. ISBN 1-58113-567-X. doi: 10.1145/775047.775126. URL http://doi.acm.org/10.1145/775047.775126.

[100] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social

representations. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '14, pages 701–710, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-2956-9. doi: 10.1145/2623330.2623732. URL http://doi.acm.org/10.1145/2623330.2623732.

[101] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. Line: Large-scale information network embedding. In *Proceedings of the 24th International Conference on World Wide Web*, WWW '15, pages 1067–1077, New York, NY, USA, 2015. ACM. ISBN 978-1-4503-3469-3. doi: 10.1145/2736277.2741093. URL http://doi.acm.org/10.1145/2736277.2741093.

[102] Cataldo Musto, Pierpaolo Basile, Marco Degemmis, Pasquale Lops, Giovanni Semeraro, and Simone Rutigliano. Automatic selection of linked open data features in graph-based recommender systems. In *CBRecSys@RecSys*, 2015.

[103] Cataldo Musto, Pierpaolo Basile, Pasquale Lops, Marco de Gemmis, and Giovanni Semeraro. Introducing linked open data in graph-based recommender systems. *Inf. Process. Manage.*, 53(2):405–435, March 2017. ISSN 0306-4573. doi: 10.1016/j.ipm.2016.12.003. URL https://doi.org/10.1016/j.ipm.2016.12.003.

[104] Xiao Yu, Xiang Ren, Yizhou Sun, Quanquan Gu, Bradley Sturt, Urvashi Khandelwal, Brandon Norick, and Jiawei Han. Personalized entity recommendation: A heterogeneous information network approach. In *Proceedings of the 7th ACM International Conference on Web Search and Data Mining*, WSDM '14, pages 283–292, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-2351-2. doi: 10.1145/2556195.2556259. URL http://doi.acm.org/10.1145/2556195.2556259.

[105] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. *CoRR*, abs/1607.00653, 2016. URL http://arxiv.org/abs/1607.00653.

[106] Enrico Palumbo, Giuseppe Rizzo, and Raphaël Troncy. Entity2rec: Learning user-item relatedness from knowledge graphs for top-n item recommendation. In *Proceedings of*

the Eleventh ACM Conference on Recommender Systems, RecSys '17, pages 32–36, New York, NY, USA, 2017. ACM. ISBN 978-1-4503-4652-8. doi: 10.1145/3109859.3109889. URL http://doi.acm.org/10.1145/3109859.3109889.

[107] Yizhou Sun, Jiawei Han, Xifeng Yan, Philip S. Yu, and Tianyi Wu. Pathsim: Meta path-based top-k similarity search in heterogeneous information networks. In *In VLDB 11*, 2011.

[108] Jie Tang, Jing Zhang, Limin Yao, Juanzi Li, Li Zhang, and Zhong Su. Arnetminer: Extraction and mining of academic social networks. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '08, pages 990–998, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-193-4. doi: 10.1145/1401890.1402008. URL http://doi.acm.org/10.1145/1401890.1402008.

[109] Neil Hurley and Mi Zhang. Novelty and diversity in top-n recommendation – analysis and evaluation. *ACM Trans. Internet Technol.*, 10(4):14:1–14:30, March 2011. ISSN 1533-5399. doi: 10.1145/1944339.1944341. URL http://doi.acm.org/10.1145/1944339.1944341.

[110] Cai-Nicolas Ziegler, Sean M. McNee, Joseph A. Konstan, and Georg Lausen. Improving recommendation lists through topic diversification. In *Proceedings of the 14th International Conference on World Wide Web*, WWW '05, pages 22–32, New York, NY, USA, 2005. ACM. ISBN 1-59593-046-9. doi: 10.1145/1060745.1060754. URL http://doi.acm.org/10.1145/1060745.1060754.

[111] Saúl Vargas and Pablo Castells. Exploiting the diversity of user preferences for recommendation. In *Proceedings of the 10th Conference on Open Research Areas in Information Retrieval*, OAIR '13, pages 129–136, Paris, France, France, 2013. LE CENTRE DE HAUTES ETUDES INTERNATIONALES D'INFORMATIQUE DOCUMENTAIRE. ISBN 978-2-905450-09-8. URL http://dl.acm.org/citation.cfm?id=2491748.2491776.

[112] Alex Kulesza and Ben Taskar. *Determinantal Point Processes for Machine Learning*. Now Publishers Inc., Hanover, MA, USA, 2012. ISBN 1601986289, 9781601986283.

[113] F. Maxwell Harper and Joseph A. Konstan. The movielens datasets: History and context. *ACM Trans. Interact. Intell. Syst.*, 5(4):19:1–19:19, December 2015. ISSN 2160-6455. doi: 10.1145/2827872. URL http://doi.acm.org/10.1145/2827872.

[114] Odile Macchi. The coincidence approach to stochastic point processes. 7:83–122, 03 1975.

[115] Yanxun Xu, Peter Muller, and Donatello Telesca. Bayesian inference for latent biologic structure with determinantal point processes (dpp). *Biometrics*, 2016. ISSN 0006-341X.

[116] Alex Kulesza and Ben Taskar. k-dpps: Fixed-size determinantal point processes. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ICML'11, pages 1193–1200, USA, 2011. Omnipress. ISBN 978-1-4503-0619-5. URL http://dl.acm.org/citation.cfm?id=3104482.3104632.

[117] George Karypis. Evaluation of item-based top-n recommendation algorithms. In *Proceedings of the Tenth International Conference on Information and Knowledge Management*, CIKM '01, pages 247–254, New York, NY, USA, 2001. ACM. ISBN 1-58113-436-3. doi: 10.1145/502585.502627. URL http://doi.acm.org/10.1145/502585.502627.

[118] Tie-Yan Liu. Learning to rank for information retrieval. *Found. Trends Inf. Retr.*, 3(3):225–331, March 2009. ISSN 1554-0669. doi: 10.1561/1500000016. URL http://dx.doi.org/10.1561/1500000016.

[119] John S Breese, David Heckerman, and Carl Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence*, pages 43–52. Morgan Kaufmann Publishers Inc., 1998.

[120] Peter Brusilovsky, Alfred Kobsa, and Wolfgang Nejdl. *The adaptive web*. Springer-Verlag Berlin Heidelberg, 2007.

[121] Ellen M Voorhees, Donna K Harman, et al. *TREC: Experiment and evaluation in information retrieval*, volume 1. MIT press Cambridge, 2005.

[122] Alan Said. A short history of the recsys challenge. *AI Magazine*, 37(4), 2016.

[123] Victor Lavrenko. *A generative theory of relevance*, volume 26. Springer Science & Business Media, 2008.

[124] Victor Lavrenko and W Bruce Croft. Relevance based language models. pages 120–127. ACM, 2001.

[125] Qiang Wu, Christopher JC Burges, Krysta M Svore, and Jianfeng Gao. Adapting boosting for information retrieval measures. *Information Retrieval*, 13(3):254–270, 2010.

[126] Jean C de Borda. Mémoire sur les élections au scrutin. 1781.

[127] Javier Parapar, Alejandro BellogíN, Pablo Castells, and Álvaro Barreiro. Relevance-based language modelling for recommender systems. *Information Processing & Management*, 49(4):966–980, 2013.

[128] Daniel Valcarce, Javier Parapar, and Álvaro Barreiro. Efficient pseudo-relevance feedback methods for collaborative filtering recommendation. In *European Conference on Information Retrieval*, pages 602–613. Springer, 2016.

[129] Nasreen Abdul-Jaleel, James Allan, W Bruce Croft, Fernando Diaz, Leah Larkey, Xiaoyan Li, Mark D Smucker, and Courtney Wade. Umass at trec 2004: Novelty and hard. *Computer Science Department Faculty Publication Series*, page 189, 2004.

[130] Hamed Zamani, Bhaskar Mitra, Xia Song, Nick Craswell, and Saurabh Tiwary. Neural ranking models with multiple document fields. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, pages 700–708. ACM, 2018.

[131] Stephen Robertson, Hugo Zaragoza, and Michael Taylor. Simple bm25 extension to multiple weighted fields. In *Proc. CIKM*, pages 42–49. ACM, 2004.

[132] Quoc V Le and Tomas Mikolov. Distributed representations of sentences and documents. In *ICML*, volume 14, pages 1188–1196, 2014.

[133] Bhaskar Mitra, Eric Nalisnick, Nick Craswell, and Rich Caruana. A dual embedding space model for document ranking. *arXiv preprint arXiv:1602.01137*, 2016.

[134] Hamed Zamani and W Bruce Croft. Estimating embedding vectors for queries. pages 123–132. ACM, 2016.

[135] Ivan Vulić and Marie-Francine Moens. Monolingual and cross-lingual information retrieval models based on (bilingual) word embeddings. In *Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval*, pages 363–372. ACM, 2015.

[136] Yuxiao Dong, Nitesh V Chawla, and Ananthram Swami. metapath2vec: Scalable representation learning for heterogeneous networks. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 135–144. ACM, 2017.

[137] Jian Tang, Meng Qu, and Qiaozhu Mei. Pte: Predictive text embedding through large-scale heterogeneous text networks. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1165–1174. ACM, 2015.

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

[139] Jaime Teevan, Kevyn Collins-Thompson, Ryen W White, Susan T Dumais, and Yubin Kim. Slow search: Information retrieval without time constraints. In *Proc. HCIR*, page 1. ACM, 2013.

[140] *Learning Deep Structured Semantic Models for Web Search using Clickthrough Data*, October 2013. ACM International Conference on Information and Knowledge Management (CIKM).