Doktori értekezés

Doctoral dissertation

Frederick Ayala Gómez

2019

Context-Aware Recommendations for Groups in Location-Based Social Networks and Academic Social Networks

Frederick Ayala Gómez

Supervisor: András Benczúr, Ph.D.

Eötvös Loránd University
Faculty of Informatics
Department of Information Systems

Doctoral School of Informatics
Erzsébet Csuhaj-Varjú, D.Sc.

Ph.D. Program of "Foundations and Methodology of Informatics"
Zoltán Horvth, Ph.D.

A dissertation submitted for the degree of
Philosophiae Doctor (Ph.D.)

Budapest, 2019.

DOI: 10.15476/ELTE.2019.018

EÖTVÖS LORÁND TUDOMÁNYEGYETEM

DECLARATION FORM for disclosure of a doctoral thesis

I. **The data of the doctoral thesis:**

Name of the author: Frederick Ayala Gómez

MTMT-identifier: 10060256

Title and subtitle of the doctoral thesis: Context- Aware Recommendations for Groups in Location- Based Social Networks and Academic Social Networks

DOI-identifier[1]: 10.15476/ELTE.2019.018

Name of the doctoral school: Doctoral School of Informatics

Name of the doctoral programme: Foundations and Methodology of Informatics

Name and scientific degree of the supervisor: András Benczúr, PhD

Workplace of the supervisor: MTA SZTAKI

II. Declarations

1. As the author of the doctoral thesis,[2]

a) I agree to public disclosure of my doctoral thesis after obtaining a doctoral degree in the storage of ELTE Digital Institutional Repository. I authorize Adina Kulcsár, the administrator of the ELTE Doctoral School of Informatics to upload the thesis and the abstract to ELTE Digital Institutional Repository, and I authorize the administrator to fill all the declarations that are required in this procedure.

b) I request to defer public disclosure to the University Library and the ELTE Digital Institutional Repository until the date of announcement of the patent or protection. For details, see the attached application form;[3]

c) I request in case the doctoral thesis contains qualified data pertaining to national security, to disclose the doctoral thesis publicly to the University Library and the ELTE Digital Institutional

[1] Filled by the administrator of the faculty offices.
[2] The relevant part shall be underlined.
[3] Submitting the doctoral thesis to the Disciplinary Doctoral Council, the patent or protection application form and the request for deferment of public disclosure shall also be attached.

Repository ensuing the lapse of the period of the qualification process.;[4]

d) I request to defer public disclosure to the University Library and the ELTE Digital Institutional Repository, in case there is a publishing contract concluded during the doctoral procedure or up until the award of the degree. However, the bibliographical data of the work shall be accessible to the public. If the publication of the doctoral thesis will not be carried out within a year from the award of the degree subject to the publishing contract, I agree to the public disclosure of the doctoral thesis and abstract to the University Library and the ELTE Digital Institutional Repository.[5]

2. As the author of the doctoral thesis, I declare that
a) the doctoral thesis and abstract uploaded to the ELTE Digital Institutional Repository are entirely the result of my own intellectual work and as far as I know, I did not infringe anyone's intellectual property rights.;
b) the printed version of the doctoral thesis and the abstract are identical with the doctoral thesis files (texts and diagrams) submitted on electronic device.

3. As the author of the doctoral thesis, I agree to the inspection of the thesis and the abstract by uploading them to a plagiarism checker software.

Budapest, 28 January 2019.

Signature of thesis author

[4] Submitting the doctoral thesis, the notarial deed pertaining to the qualified data shall also be attached.
[5] Submitting the doctoral thesis, the publishing contract shall also be attached.

Acknowledgments

I express my sincere gratitude to my supervisor András Benczúr, whose advice, encouragement, and patience guided my doctoral studies. It was very valuable to be a visitor in the Data Mining and Search Group at the Institute for Computer Science and Control of the Hungarian Academy of Sciences (MTA SZTAKI). This opportunity gave me the chance to meet and collaborate with great people. I always felt welcome and supported by the group. In particular by Bálint Daróczy, who significantly helped me with explanations, guidance, and brainstorming. Also, I thank Róbert Pálovics for introducing me to the topic of recommender systems.

Special thanks to Aristides Gionnis and Michael Mathioudakis for accepting me as a visitor at the Data Mining group of the Computer Science Department of Aalto University. I learned a lot from both of you. I also would like to express my gratitude to Pinar Karagoz whose guidance helped me complete our research on itinerary recommendations. Finally, to Baris Kenis for his contributions and great ideas.

Also, thanks to professor Zoltán Horváth, Dr. Zoltán Istenes, and to the staff of the ELTE Faculty of Informatics for their guidance and administrative support.

I appreciate the support that México gave me via the "Consejo Nacional de Ciencia y Tecnologia" (CONACYT). Without your funding I could not have finished my doctoral studies.

Last but not least, I am grateful to my family for their unconditional support during my doctoral studies. It has been a long journey and I always felt your support, even from afar.

Contents

Acronyms

AIR Average Individual Ratings

ALM Average Least Misery

ALS Alternating Least Squares

AWM Average Without Misery

CGAR Collaborative Group Activity Recommender

CTR Click-Through Rate

DCG Discounted Cummulative Gain

ECP Empirical Conditional Probability

EIR Euclidean Item Recommender

FC Fisher Conditional Score

FD Fisher Distance

FMFMGM Fused Matrix Factorization Framework with the Multi-center Gaussian Model

GGR Geo-Group-Recommender

HTTP Hypertext Transfer Protocol

iALS Implicit Alternating Least Squares

KDE Kernel Density Estimation

LBSN Location-Based Social Network

LOD Linking Open Data

LTR Learning to Rank

MAG Microsoft Academic Graph

MCTS Monte-Carlo Tree Search

MRF Markov Random Field
nDCG Normalized Discounted Cummulative Gain
NER Named-Entity Recognition
OWL Web Ontology Language
POI Point of Interest
RDF Resource Description Framework
RNN Recurrent Neural Networks
SGD Stochastic Gradient Descent
SSE Sum of Squared Errors
tf-idf Term Frequency–Inverse Document Frequency
UCT Upper Confidence Bound for Trees
URI Uniform Resource Identifier
W3C World Wide Web Consortium
XML Extensible Markup Language

List of Figures

List of Tables

List of Algorithms

Introduction

Motivation

The impact of the World Wide Web goes beyond communicating between computers. The web changed the way we interact with organizations, people, documents, and data. It moved from a technological artifact to an integral part of human activity [Hall and Tiropanis, 2012]. Social media is one of the new buzzwords of the web. It is a comparatively new term that covers a wide range of online applications, platforms, and media that support online interactions, collaboration, and the sharing of content. It includes all that constitutes human interaction in online platforms. Social media has had grown explosively in a short time and has a profound impact on advertising, publicity, marketing, public opinion, entertainment, software services, and decision-making.

In the web one may find several social media providers offering an extensive amount of content to their users. Usually, social media providers have a unique value proposition to their users. For instance, *facebook.com* allows their users to create a social network of friends and share text, images, videos, links, pages, and user profiles; *youtube.com* offers an extensive collection of videos from different categories (e.g., art, reviews, news, sports, games); *twitter.com* allows users to post short texts and re-share posts to the user ego-network; *instagram.com* allows users to post images, and videos and *foursquare.com* allows users to share the Point of Interest (POI) they visit.

The content offered in social networks is exciting to users. However, it is challenging, as users need to spend time and effort finding content that might be of their interest. Navigating such an extensive collection of items becomes difficult if there are no tools to help the users. To overcome this problem, social media relies on data mining solutions such as recommender

Research Question	Description
RQ1.1	How do preferences change when users are alone vs. when they are in a group?
RQ1.2	How to recommend a list of POIs for groups of users in LBSN in the areas that a group prefers?
RQ1.3	How to recommend a sequence of POIs for groups of users in LBSNs?

Table 1: Research questions for POI recommendation in Location-Based Social Network.

systems and information retrieval systems. In particular, recommender systems have found applications in many domains including movies, music, videos, news, books, and products in general. Depending on the context they produce a list of recommended items using a variety of techniques.

The goal of this dissertation is to research new approaches in recommender systems that help to satisfy the needs of users in social media. Specifically, we examine the role of recommender systems for users in two types of social media, Location-Based Social Networks (LBSN) and academic social networks. LBSNs enable their users to share the places they go to and with whom they are. Academic social networks help modern researchers organize their scientific libraries and discover relevant papers to their research. For both types of social media most of the existing work focuses on recommendations for individual users. The proposed approaches are distinguishable from others in that we focus on providing recommendations to a group of users, rather than to individuals. Moreover, we investigate the importance of item-to-item recommendations and propose a new method for recommending on infrequent items.

Research Questions

This dissertation aims to contribute to answering the following research questions. For clarity, we separate these into three groups. Table 1 contains the research questions for POI recommendations in Location-Based Social Network, Table 2 for global citation recommendations in academic social networks. Table 3 for item-to-item recommendations.

We contributed to the scientific literature with the publication of the following articles. To answer **RQ1.1** and **RQ1.2**, in [Ayala-Gómez et al., 2017], we investigated the behavior of groups in LBSNs and experimented

Research Question	Description
RQ2.1	How to expand sematic features of scientific abstracts using knowledge graphs?
RQ2.2	Given an abstract of a new paper mapped to a knowledge graph, how to find a set of candidate papers to cite?
RQ2.3	Given an abstract of a new paper mapped to a knowledge graph, and a set of candidate papers, how to build *top-k recommendations* of papers to cite?

Table 2: Research questions for global citation recommendations in academic social networks.

Research Question	Description
RQ3.1	How to use the similarity information in the item-item transition conditional probability for a given infrequent-item?
RQ3.2	How to rank the next item by its distance from an infrequent item?
RQ3.3	How to handle different similarity metrics between the items to improve the infrequent item-to-item recommendations?

Table 3: Research questions for item-to-item recommendations.

with recommender systems to suggest POIs near the areas that a group frequents. In [Ayala-Gómez et al., 2018] we proposed an approach for constructing itineraries for groups of users in LBSNs to answer **RQ1.3**. In [Ayala-Gómez et al., 2018] we used knowledge graphs to build global citation recommendations for new scientific articles to answer **RQ2.1**–**3**. Finally, in [Daróczy et al., 2018] we analyze infrequent item recommendations to solve question **RQ3.1**–**3**.

Additionally, I contributed to the work presented in [Pálovics et al., 2014] where we proposed an ensemble of binary classifiers and matrix factorization for recommending movie rating tweets.

Thesis Outline

Chapter 1 provides the necessary theoretical background for our research. We review the concepts of graphs, centrality measures, knowledge graphs, and Kernel Density Estimation (KDE). Also, the different types of recommender systems such as non-personalized recommenders, content-based recommenders, nearest neighbor collaborative filtering, and matrix factorization. Additionally, we present learning to rank models and group aggregation strategies. The chapter ends with a review of the metrics for evaluating the performance of recommender systems.

Chapter 2 contains our proposed approaches for recommending a list of points of interest (POI) for groups of LBSNs in areas that the group frequents and our work on building itineraries of POIs for groups given a starting POI. Chapter 3 presents our approach to building global citation recommendations using knowledge graphs, where the authors submit the abstract of a new article as an input to the recommender system. Chapter 4 presents our work on item-to-item recommendations. Chapters 2, 3, and 4 provide details on the problem definition, setting, working data set, data analysis and exploration, proposed approach, evaluation methodology, and experimental results of our research in different settings.

Finally, Chapter 5 concludes this dissertation by presenting the findings, summarizing the chapters, our contributions, implications, and future work.

Chapter 1

Theoretical Background

This chapter contains relevant definitions to the proposed approaches for building the recommender systems presented in this dissertation. The chapter starts with an introduction to recommender systems. We review non-personalized recommenders, content-based recommenders, nearest neighbor collaborative filtering, matrix factorization, learning to rank (LTR) models, and group recommendations. We then explain how the performance of recommender systems is measured. Finally, we describe knowledge graphs and Kernel Density Estimation (KDE). The readers may find more information on recommender systems in [Ricci et al., 2015, Aggarwal, 2016] and on knowledge graphs and semantic web in [Szeredi et al., 2014].

1.1 Recommender Systems

The World Wide Web offers an incredible amount of content, especially with the evolution of user generated content termed as the Web 2.0. Web users find relevant content on the Web by spending time and effort navigating the vast collection of items. Websites often include search and recommendation tools to aid users in finding the content of their interests. Recommender systems help users find content that satisfies their interests and needs based on past transactions. In this section we review relevant concepts in recommender systems and present different types of recommender systems from the literature.

The goal of a recommender system is to increase key performance indicators for the website owner (e.g., retention, revenue, session duration). Additionally, there are operational goals focused on satisfying user needs. Table 1.1 summarizes the four operational goals of a recommender system

Goal	Explanation
Relevance	Relevant to the user.
Novelty	Items that the user has not seen before.
Serendipity	Items that are not just novel but also surprising.
Diversity	Items that are not similar to each other.

Table 1.1: Operational goals of a recommender system.

[Aggarwal, 2016]. The motivation of these goals is the following. Relevant items are more likely to be consumed by users. Novelty prevents the recommender system from suggesting popular or known items. Serendipity aims to find *hidden gems* that the user has not consumed and are genuinely relevant. Finally, recommending different types of items helps to prevent users getting bored by receiving very similar items as recommendations.

In this dissertation we use recommender systems widely used in research and industry as baseline methods. In the next section we introduce these models, but first, we review some important concepts on recommender systems.

Recommender systems are based on the idea of using past user-item interactions to learn user preferences, and suggest new items based on recommendation models. To do this, recommender systems rely on the set of users U, items I, and the user-item feedback R for modeling. An item is an individual piece of content in the catalog of a content provider[1]. For instance, an item could be a song, a video, a blog or news entry, a post, or a product. A user is a person with specific interests and needs who is willing to consume items. Intuitively, users consume items that are of their interests and needs. There are different ways to consume items. Users may view, click, share, comment, or buy an item. The interaction between users and items is called feedback, and represents the preference of a user for a certain item. The feedback can be explicit or implicit. Implicit feedback is taken from the user actions on an item (i.e., click, view, time spent visiting or watching). Explicit feedback is collected by asking users to rate an item (e.g., 1-5 stars, likes, dislikes).

As an example of explicit feedback, suppose that we have five users and six items. We ask the users to rate the items from 1 to 5 (the higher the number, the better). We can represent this data as a matrix $R_{U \times I}$, called the rating matrix. Table 1.2 presents an example of an explicit feedback rating matrix. Implicit feedback is often represented with binary values.

[1]We refer to content providers to social media platforms, blogs, media services, e-commerce websites, and any other entity that offers content online.

	i_1	i_2	i_3	i_4	i_5	i_6
u_1	5				3	
u_2			4			3
u_3		4		3	5	
u_4	2					
u_5			4		3	5

Table 1.2: An example of explicit ratings matrix. Each row represents a user and each column an item. The entries are the ratings that users gave to each item.

	i_1	i_2	i_3	i_4	i_5	i_6
u_1	1	1		1	1	
u_2			1			1
u_3		1		1	1	
u_4	1		1			1
u_5			1	1	1	1

Table 1.3: An example of an implicit matrix. Note that this matrix is less sparse, as users do not necessarily rate all the things they have watched.

We mentioned that implicit feedback is based on the interaction between users and items (e.g., click, play). Suppose that users do not rate all the items they have consumed. The rating matrix based on implicit feedback can have more non-empty cells, as shown in Table 1.3.

1.1.1 Recommender Systems tasks

Mainstream recommender systems focus on solving two tasks: *rating prediction* and *top-k recommendation*.

In rating prediction the task is to estimate the rating that users will give to unobserved items. That is, finding a model able to predict ratings $\hat{r}_{u,i}$ that approximate the known-ratings in $R_{U\times I}$ and fill the empty entries. Usually recommender systems try to minimize the sum of the squared error (SSE):

$$\min_{\hat{r}_{u,i}\in R_{U\times I}} \sum_{r_{u,i}\in R_{U\times I}} (r_{u,i} - \hat{r}_{u,i})^2. \tag{1.1}$$

Although predicting ratings for all the unobserved items is important, sometimes we want to recommend a small set of *best* items to the user. In this case, the recommended items are those with the highest scores at a par-

ticular cut. This type of recommendation is called *top-k recommendation*[2]. Let us define $\hat{I}$ to be an ordered set of the items based on the recommender system score. The recommendation is $\hat{I}@K = top(\hat{I}, k)$, where *top* is a function that takes the first k elements on the ordered set $\hat{I}$. We will provide more details on this process at the end of the chapter.

There are many models for building recommender systems, each of which is suitable for different task settings and each with their own strengths and weaknesses. The next section presents relevant models for this dissertation.

1.1.2 Non-personalized

A widely used non-personalized recommender system is the popularity based model. The popularity of an item represents a certain summary of user feedback on the item. An example of item popularity is the average rating or the total number of plays. The recommendation is built by sorting the items in decreasing order and taking the top items. A rating for an item in a popularity recommender system is described as:

$$\hat{r}(i) = f(R_{*i}), \tag{1.2}$$

where f is an aggregation function such as *count*, *sum*, *mean*. Note that the scoring function does not depend on the user.

The popularity based model is simple to compute and provides an overview of the items to the users. For example, if we recommend videos, the recommendation can look like *"Top-k popular songs"* or *"Best k rated songs"*.

1.1.3 Content-Based Recommendation

Recommender systems can be modeled based on the attributes of the items and users. These attributes are called *content*. For instance, a movie could have a genre, year, or director, and users may have an age, country, or academic degree. The fundamental idea of a content-based recommender is to model items by their relevant attributes. Users can also be modeled by the attributes of the items they consumed. Thus, recommendations can be based on matching the item and the user models.

Item and user content often include text, tags, or categorical values. Before we can input these features to a recommender system we must pre-process the data. There are different approaches to pre-processing textual features [Manning et al., 2008]. The most common steps for transforming

[2]In what follows, we use the word recommendation meaning *top-k* recommendation.

text to a numeric feature vector are the following. The first step is to extract the words from the document and separated by using white spaces and punctuation. The second step removes the punctuation and the common words (stop words) such as articles and connectors of the language. Lastly, a process called stemming normalizes the words (e.g., fishing, fished, and fisher are normalized to the word fish).

After pre-processing we compute numeric features for each term such as the term frequency, or the number of items containing the term. The most widely used statistic for the terms is the term frequency–inverse document frequency (tf-idf) [Luhn, 1957]. The idea is to assign a weight to each term that represents how relevant the term is for a particular item. It is defined as:

$$w_{t,d} = (1 + \log f_{t,d}) \times \log_{10}(\frac{N}{df_t}), \tag{1.3}$$

where N is the number of documents (items), $f_{t,d}$ is the frequency of the term in the items and df_t is the number of items containing t. In our case, d refers to an item. In information retrieval, given a query q represented as terms, documents are scored by adding the tf-idf of the terms contained in the query:

$$\hat{r}(i,q) = \sum_{t \in i \cap q} tf.idf_{t,q}. \tag{1.4}$$

In a recommender system the role of the query q is substituted with the set of relevant terms for a user; usually, the top-k tf-idf terms from the previously consumed items of the user. An alternative approach to scoring the items is to compute certain similarity between the items rated by the user and the candidate items. By doing so, we can also use features other than tf-idf and use a variety of similarity functions (e.g., Pearson, cosine, Jaccard).

1.1.4 Collaborative Filtering

Collaborative filtering is the class of basic models for building recommendations based on feedback. The key idea of these methods is to leverage the ratings that multiple users gave to the items. There are two kinds of collaborative filtering methods related to our work that we describe in the next two paragraphs.

User–user collaborative filtering

The main idea of user based collaborative filtering is to find similar users based on the previous rated items that a user gave. Recommendations are

built based on similar users' ratings of items that a user has not consumed yet [Herlocker et al., 1999]. To score an item we first need to find the K most similar users for the given user u, which can for example be done by using the similarity of the set of items consumed previously. The scoring of an item is given by:

$$\hat{r}(u,i) = \bar{r_u} + \frac{\sum_{k \in K} w_{u,k}(r_{k_i} - \bar{r_k})}{\sum_{k \in K} w_{u,k}}, \tag{1.5}$$

where $\bar{r}$ is the average rating, and $w_{u,k}$ is a similarity function or a weight between the user and the similar users (e.g., Pearson correlation).

Item–item collaborative filtering

In item–item collaborative filtering [Sarwar et al., 2001a] the motivation is that items have more ratings than what users give on average. Thus, it is more stable to compute item similarities than user similarities. Item–item collaborative filtering has two steps, where the first is to calculate the similarity between items. This is usually done by first normalizing the user ratings and then computing the cosine similarity as:

$$sim(i,j) = \frac{\sum_{u \in U_i \cap U_j} r_{u,i} r_{u,j}}{\sqrt{\sum_{r_{u,i}}} \sqrt{\sum_{r_{u,j}}}}. \tag{1.6}$$

The second step is to score the items for a particular user by finding a set of nearest neighbors N (e.g., users who rated similar items) that the user has rated and then scoring using:

$$\hat{r}(u,i) = \bar{r_i} + \frac{\sum_{j \in N} w_{i,j}(r_{u_j} - \bar{r_j})}{\sum_{j \in N} |w_{i,j}|}, \tag{1.7}$$

and for implicit feedback:

$$\hat{r}(u,i) = \sum_{j \in N} w_{i,j}. \tag{1.8}$$

Matrix Factorization

Between 2006 and 2009, Netflix organized a challenge called the *Netflix Prize* [Bennett et al., 2007a, Koren, 2009]. This competition boosted the research on recommender systems. One of the best performing techniques is called biased matrix factorization [Koren et al., 2009]. This model reduces

the matrix R into two matrices of dimensions $P_{U,f}$ and $Q_{f,I}$, where f is the size of the lower dimension. The model also learns a bias for each user and item. The idea of matrix factorization is to approximate each known entry in R as:

$$\hat{r_{ui}} = \mu + b_i + b_u + p_u q_i^T \tag{1.9}$$

The lower dimensional matrices are learned by optimizing the squared error function:

$$\underset{p,q}{\text{minimize}} \quad \sum_{(u,i)\in X} (r_{ui} - \mu - b_u - b_i - p_u q_i^T)^2 + \lambda(||q_i||^2 + ||p_u||^2), \tag{1.10}$$

where X denotes the set of pairs (u, i) with known ratings $r_{u,i}$. The model can, for example, be optimized using stochastic gradient descent (SGD) or alternating least squares (ALS). In [Koren et al., 2009] a detailed explanation on the optimization process is presented.

The implicit alternating least squares (iALS) matrix factorization method is presented in [Hu et al., 2008]. The input to the model is a set of binary variables $p_{u,i}$ such that:

$$p_{u,i} = \begin{cases} 1 & \text{if } u \text{ interacted with } i \\ 0 & \text{otherwise} \end{cases} \tag{1.11}$$

Additionally, each $p_{u,i}$ is associated to a confidence variable $c_{u,i}$ defined as:

$$c_{u,i} = 1 + \alpha p_{u,i}, \tag{1.12}$$

where α is a constant parameter that weighs the importance of the ratings.

Following the same idea of the matrix factorization, the task is to estimate $p_{u,i} = x_u^T y_i$ as an inner product of the learned vectors x_u for users and y_i for items. By adding the confidence levels, the optimization problem is defined as:

$$\underset{x*,y*}{\text{minimize}} \quad \sum_{u,i} c_{u,i}(p_{ui} - x_u^T y_i)^2 + \lambda(\sum_u ||x_u||^2 + \sum_i ||y_i||^2). \tag{1.13}$$

Note that the error is computed over all the pairs of users and items rather than on the known ratings. The model is optimized via ALS. [Hu et al., 2008] presents a detailed description of the optimization process.

1.1.5 Factorization Machines

Context-aware recommender systems use the context (e.g., time, popularity, content) of the ratings with collaborative filtering. A widely used context-aware recommender system is the Factorization Machine [Rendle, 2010].

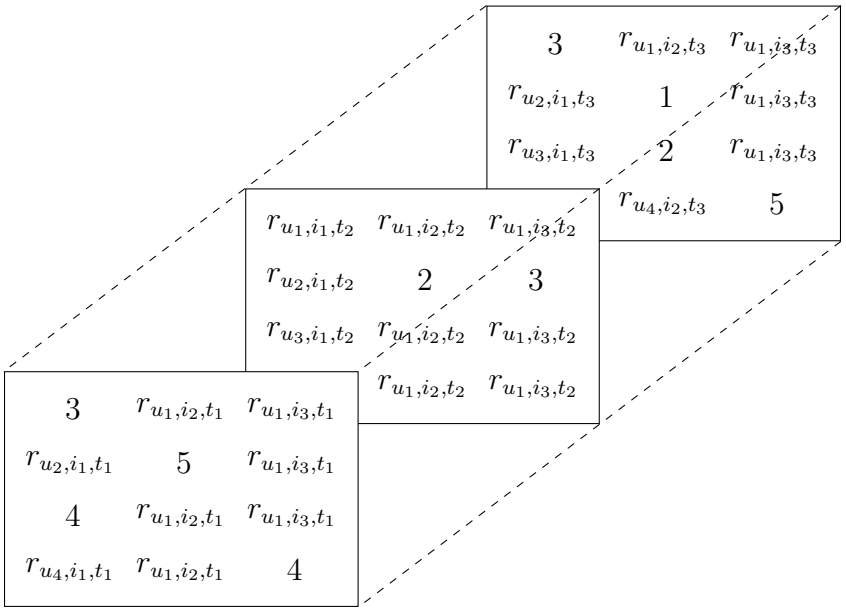

Figure 1.1: A three-dimensional feedback matrix where time is being represented as the 3rd dimension. For example, by calculating a categorical feature that represents the hour when the rating was given.

This model is able to learn from different features of the user, item, and context ratings using feature engineering. The general idea is to model the ratings as a multidimensional tensor that contains the context associated to the rating. Figure 1.1 presents a rating matrix that includes time as context.

The input to the Factorization Machine is the flattened version of the tensor, a sparse matrix X of N features with columns corresponding to the one-hot encoding of users, items, and additional contextual dimensions. Each row of the matrix X is associated with the rating given by the user to the item i. Table 1.4 is the sparse matrix representation of the tensor in Figure 1.1.

The Factorization Machine builds the following model:

$$\hat{r(x)} := w_0 + \sum_{n=1}^{N} w_n x_n + \sum_{n=1}^{N} \sum_{m=n+1}^{N} \langle v_n, v_m \rangle x_n x_m, \tag{1.14}$$

where the model parameters are $w_0 \in \mathbb{R}$, $w \in \mathbb{R}^n$, $V \in \mathbb{R}^{n \times k}$, and $< \cdot, \cdot >$ is the dot product of two vectors of size k:

$$< v_n, v_m >= \sum_{f=1}^{k} v_{n,f} \cdot v_{m,f}, \tag{1.15}$$

u_1	u_2	u_3	u_4	i_1	i_2	i_3	t_1	t_2	t_3	r
1				1			1			3
	1				1		1			5
		1		1			1			4
			1			1	1			4
	1				1			1		2
		1			1			1		3
1				1					1	3
	1				1				1	1
		1			1				1	2
			1			1			1	5

Table 1.4: A sparse matrix representing the contextual feedback from Figure 1.1.

where k is the dimension of the two matrices that factorize V. In the model, w_0 is the global bias, w_n is the weight of the *n-th* feature, and $< v_n, v_m >$ models the interaction between the n^{th} and m^{th} feature.

In this thesis we use the Factorization Machines implementation of Apple Turi Create[3] as baseline. They optimize the model using SGD.

While Factorization Machine is the most widely used model; there are additional models for building context-aware recommendations, for instance, the Pairwise Interaction Tensor Factorization [Rendle and Schmidt-Thieme, 2010] and the General Factorization Framework [Hidasi and Tikk, 2016].

1.2 Learning to Rank

Models like biased matrix factorization are optimized using SGD on the sum of squared error to minimize the difference between the predicted rating and the actual rating. In contrast, learning-to-rank (LTR) models optimize the order of the recommended items rather than the predicted score. Given a query q and a set of candidates C, a LTR model ranks the candidates by their relevance for a given query:

$$\{((q_i, c_1), r_1), ((q_i, c_2), r_2), \ldots, ((q_i, c_n), r_n)\}, \tag{1.16}$$

where $c_* \in R^N, r_* \in \{0, \ldots, L\}$. Note that the relevance r_* could be expressed using different labels (e.g., 0: bad, 1: good). In our case, q is a user

[3] `https://github.com/apple/turicreate`

and C are the possible items to recommend. A LTR model should rank relevant items higher than irrelevant ones.

There are three general approaches to building LTR models [Burges, 2010]. Point-wise LTR handles each instance independently by directly optimizing the rank (e.g., by regression). Pair-wise LTR optimizes the order of pairs of instances independently (e.g., [Joachims, 2002]). Finally, list-wise LTR optimizes the order of the entire list of items at once.

1.2.1 Ranking Factorization Machines

Factorization Machines can be optimized for ranking by modifying the loss function. The Apple Turi Create implements a pair-wise LTR model by optimizing ranking as:

$$\begin{aligned}\min_{w,a,b,V,U} \frac{1}{|X|} \sum_{(i,j,r_{ij})\in X} (\hat{r_{ui}} - r_{ui})^2 + \lambda_1(||w||_2^2 + ||a||_2^2 + ||b_2^2||) + \\ \lambda_2(||U||_2^2 + ||V||_2^2) + \frac{\lambda_{rr}}{\text{const} \cdot |N|} \sum_{(i,j)\in N} (\hat{r_{ui}} - r_{nr})^2,\end{aligned} \tag{1.17}$$

where λrr is the hyper-parameter for ranking regularization, N is a set of unobserved items for the user, and r_{nr} is the predicted rating for the unobserved item according to the model. This normalization helps for sorting the scores for the user. For implicit feedback the logistic loss function is used instead of the squared error (SSE), and the unobserved pairs are only weighted by λrr.

1.2.2 LambdaRank and LambdaMART

LambdaRank [Burges et al., 2006] is a list-wise LTR method for optimizing nDCG (to be defined in equation (1.27)). This technique requires the model (e.g., neural networks, gradient boosted trees) to be differentiable in its parameters. We consider each user who request a recommendation separately. For a user let the relevance of an item i be r_i. For each user we optimize nDCG, which is based on the relevance r_i. In an earlier result [Burges et al., 2005] the authors of LambdaRank explained that for optimizing nDCG we do not require to compute the scores of the model, only its gradients. In each step of an SGD procedure, LambdaRank identifies the relevant items that are wrongly positioned and computes the gradient of the loss function as a pointer that tells the direction to update the parameters. The gradient is weighted by the relative gain on nDCG of swapping items i and j. This

is done using a symmetric cost function C defined as:

$$C = \frac{1}{2}(1 - S_{ij})\sigma(s_i - s_j) + \log(1 + e^{-\sigma(s_i - s_j)}), \tag{1.18}$$

where s_i and s_j are the predicted scores from the model, the parameter σ determines the shape of the sigmoid, and

$$S_{ij} \in 0, \pm 1 = \begin{cases} 1 & \text{if } r_i > r_j \\ -1 & \text{if } r_j > r_i \\ 0 & r_i = r_j \end{cases}, \tag{1.19}$$

keeps the function symmetric. We use the cost function C to define a variable λ_{ij} that encapsulates the relative gain of swapping the incorrectly ranked items as:

$$\lambda_{ij} = \frac{\delta C}{\delta s_i} = \frac{-\sigma}{1 + \exp^{\sigma(s_i - s_j)}} |\Delta \text{nDCG}|. \tag{1.20}$$

λ_{ij} will help us to optimize the model parameters using:

$$w_k \to w_k - \eta \frac{\delta C}{\delta w_k}, \tag{1.21}$$

where η is the learning rate and w_k are the model parameters.

LambdaMART [Burges, 2010] is the application of LambdaRank on MART trees [Friedman, 2001], a type of gradient boosted trees. LambdaMART has shown state-of-the-art performance in recent LTR competitions including the *Yahoo! Learning to Rank Challenge* [Chapelle and Chang, 2010].

1.3 Group Recommendations

So far we have presented models that recommend items to an individual user (i.e., one person). A fundamental part of this dissertation is the investigation of models that build recommendations for groups (i.e., more than one person).

Recommendations for groups are surveyed in [Masthoff, 2015]. They characterize group recommender systems by considering the following dimensions: (*i*) individual preferences are known vs. developed over time; (*ii*) recommended items are experienced by the group vs. presented as options; (*iii*) the group is passive (e.g., users are not voting) vs. active (e.g., the system helps to create consensus); and (*iv*) recommending a single item vs. a set. Authors in [Campos et al., 2009] present two more options for group recommender systems: they may collect individual preferences, or aggregate profiles.

1.3.1 Group Aggregation Strategies

To estimate a rating for the group G, there are three widely used methods to aggregate individual recommendations for users $u \in G$.

Average Individual Ratings (AIR) scores the item as the average rating that each user gave:

$$\hat{r}(G, i) = \frac{\sum_{u \in G} r_{u,i}}{|G|}. \tag{1.22}$$

Average Without Misery (AWM) scores the item as the average of individual ratings without including items rated under a threshold s:

$$\hat{r}(G, i; s) = \frac{\sum_{u \in G; r_{u,i} > s} r_{u,i}}{|G|}. \tag{1.23}$$

Average Least Misery (ALM) considers the set of low ranked items below certain rank cut r, and compute the average score over these lowest ranked items:

$$\hat{r}(G, i) = \frac{\sum_{u \in G; rank(r_{u,i}) < r} r_{u,i}}{|G|}. \tag{1.24}$$

1.4 Evaluation and Metrics

After we train a recommendation model by fitting to known data, we may test based on a withheld sample or fresh data. Testing involves two crucial steps: deciding how to split the data and defining the metrics to measure the quality of the recommendations. In this section we review the difference between offline and online evaluations, different approaches for splitting the data, and the metrics for evaluating a recommender system.

1.4.1 Offline and Online Evaluation

In offline evaluation the set of items, users, and ratings are fixed. Offline evaluation is the most widely used approach for evaluating a recommender system. For instance, the MovieLens Data set [Harper and Konstan, 2015] and Netflix Data set [Bennett et al., 2007a] are open data sets that one can download and use for experimentation. A drawback of offline evaluation is that researchers cannot experiment with providing recommendations for users in real-time.

Online evaluation is based on the idea of evaluating recommender systems in real-time. This is often done by designing A/B tests by creating different groups of users, showing recommendations built from different models, and comparing them with online metrics like clickthrough rate (CTR) defined as clicks divided by impressions.

One observation of offline evaluation is that metrics measured offline often do not correlate with online metrics like CTR [Beel et al., 2013a]. However, it is rare in academia that researchers have access to production systems for online experimentation.

1.4.2 Train-Test Split

For training and evaluating a recommender system we need to split the data in three parts; training, validation, and test sets. The training set contains the ratings that we will use to fit the model. The algorithms that train recommender systems use supervised learning, which requires a label for each instance (i.e., a data point). In our case, the label of each instance is the user-item feedback. In practice, each instance contains the user, item, rating, and recommendation metadata (e.g., time of the recommendation, context). After the recommender system is fit using the training set, we use the model to build recommendations for the user-item pairs of a validation set. The validation set helps us to tune the hyper-parameters of the model (e.g., regularization, learning rate) and prevent over-fitting the training set. Finally, once the model is tuned, recommendations are generated for the users in the testing set, and quality metrics are computed based on the test user-item pairs.

There are several ways to split the data. Our analysis will use random split and time based split. Random split is based on a random function that associates each instance with a number. The instances are filtered based on specific thresholds to the training, validation, and testing sets. A commonly used split is allocating 80% of the data to the training set, 20% to the testing set, and 10% of the training set as the validation set. This approach works well when the items are time independent, for example, when all the items exist at the same time. However, if the items follow a sequence and at a given point in time some items do not exist (e.g., research papers) it is better to split by time [Campos et al., 2014]. This is done by measuring a cumulative distribution over the time and finding a time point to split the instances.

1.4.3 Metrics

Recommender systems can be evaluated by traditional data mining metrics such as precision and recall. In this dissertation we work mainly with implicit feedback and "top-k" recommendations. Evaluating the performance of "top-k recommendations" using precision and recall is done at the different cutoffs of K (e.g. 5, 10, 20), defined as:

$$\text{Precision@}K = \frac{\text{Consumed Items} \cap \text{Recommended Items}}{\text{Recommended Items}}., \tag{1.25}$$

and:

$$\text{Recall@}K = \frac{\text{Consumed Items} \cap \text{Recommended Items}}{\text{Consumed Items}}. \tag{1.26}$$

In top-k recommendations the goal is to preserve the order of relevant items, keeping those most relevant at the top and less relevant items at the bottom. That is, we would like to find relevant items on top of the recommendations and penalize if relevant items appear lower in the rank. nDCG is a metric that follows this behavior and it is defined as:

$$nDCG@k = \frac{\sum_{i}^{10} \frac{2^{rel_i}-1}{log_2(i+1)}}{IDCG}., \tag{1.27}$$

where rel_i is the relevance of the item.

The $IDCG$ is the ideal ordering of the ranking defined as:

$$IDCG = \sum_{i}^{|REL|} \frac{2^{rel_i}-1}{log_2(i+1)}., \tag{1.28}$$

where $|REL|$ represents the list of items rated by the user.

1.5 Knowledge Graphs

Knowledge graphs are closely related to ontologies. [Gruber, 1995] defined an ontology in the field of computer science as the description of concepts and relationships that exist in a particular domain. In an ontology, the concepts are described using types, properties and relationships. There are several languages that enable us to define ontologies that computers can process. The W3C defined the Web Ontology Language (OWL) [McGuinness et al., 2004] to facilitate machine interpretability of knowledge using

web technologies such as the Extensible Markup Language (XML) and the Resource Description Framework (RDF) [Beckett and McBride, 2004].

RDF is a general-purpose language for describing resources on the Web that machines can read and understand. To accomplish this, RDF uses triplets of resource, property, and property value. A resource can be anything and it must be identified using an URI [Masinter et al., 2005]. A property represents a fact about the resource using a name and a value. The RDF triplets represent a statement and its elements are referred to as *(subject, predicate, object)*. For example, *(`http://dbpedia.org/resource/Brooklyn`, isPartOf, `http://dbpedia.org/resource/New_York_City`)*, states that *Brooklyn is a part of New York City*. The URI enables making multiple statements about the resources and to easily retrieve all the statements. Using RDF triplets one can define a knowledge graph using labeled-nodes and labeled-edges to study the structure of the concepts.

A practical example of knowledge graphs is the Linked Data initiative [Bizer et al., 2009]. The motivation of Link Data is to provide best practices for publishing and connecting structured data in the Web. Link Data uses RDF and the main idea is to use HTTP URIs to enable computers to retrieve useful information on resources and links to other resources. By doing this, different organizations are able to share statements about resources in a standardized technological manner (i.e., URI, HTTP, RDF).

The Linking Open Data (LOD) Organization gathers different projects from the Web [Abele et al., 2017]. Figure 1.2 presents a graph of Linked Data projects and the connections between them[4]. Each of the circles in Figure 1.2 is a data set that shares statements about resources.

A main Linked Data project is DBpedia [Auer et al., 2007]. It contains structured data from the topics and facts from Wikipedia, "the largest and most popular general reference work on the Internet"[5]. As of 2017, the english version of DBpedia has approximately 4.2 M resources, including 1.4 M persons, 735 K places, 411 K creative works, 240 K organizations, 251 K species, and 6 K diseases.

1.5.1 Finding DBpedia Resources in Text

A well-known task in information extraction is Named Entity Recognition (NER), where the goal is to annotate (i.e., locate) and classify named entities in a text. The annotated entities belong to a category (e.g., organizations, places, creative works).

[4] An interactive visualization is available in `http://lod-cloud.net/versions/2017-08-22/lod.svg`

[5] `https://en.wikipedia.org/wiki/Wikipedia`

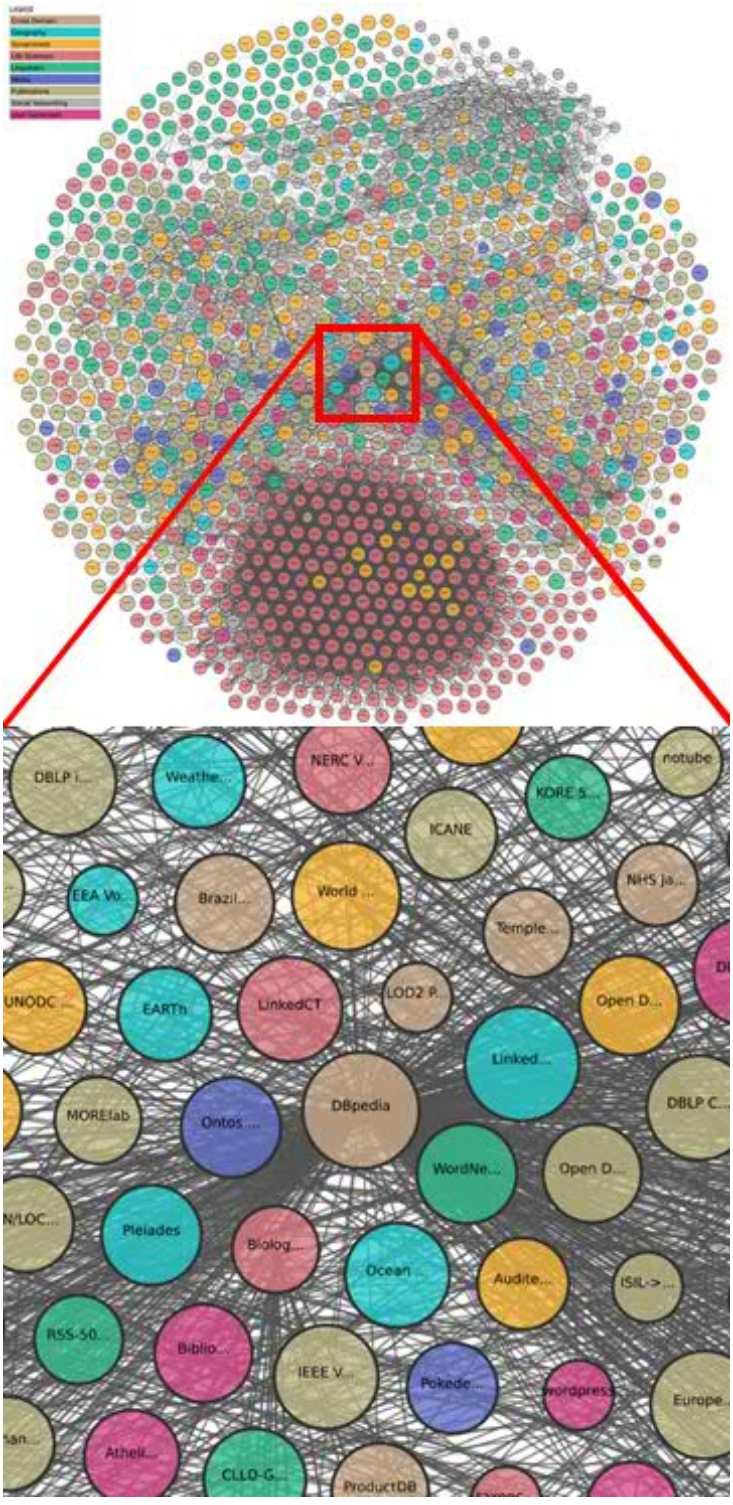

Figure 1.2: Linking Open Data cloud diagram (2017).

Text	DBPedia Resource
social networking	http://dbpedia.org/resource/Social_networking_service
Foursquare	http://dbpedia.org/resource/Foursquare
Hybrid	http://dbpedia.org/resource/Hybrid_(biology)
Recommender Systems	http://dbpedia.org/resource/Recommender_system
Density Estimation	http://dbpedia.org/resource/Density_estimation

Table 1.5: Annotated DBpedia resources.

The application of NER for annotating DBpedia resources in a text is done in DBpedia Spotlight [Mendes et al., 2011]. The following is an example of using DBpedia Spotlight to find DBpedia resources in a paragraph taken from [Ayala-Gómez et al., 2017]:

"Our data consists of activity on Swarm, a *social networking* app by *Foursquare*, and our results demonstrate that our proposed Geo-Group-Recommender (GGR), a class of *hybrid recommender systems* that combine the group geographical preferences using Kernel *Density Estimation*."

Table 1.5 shows the URIs associated to the annotated entities. These URIs allows us to retrieve the RDF triplets of the DBPedia resource. Figure 1.3 shows some triplets of the *Social networking* entity. Nevertheless, the task of NER is challenging and NER tools often incorrectly annotate some entities.

1.6 Kernel Density Estimation

In statistics and probability each of the observed events in a data set is considered a sample from an underlying probability function. Density estimation [Silverman, 1986] refers to the estimation of a density function f from the observed data, and it is done using parametric and non-parametric methods. Parametric methods assume that the data is drawn from a known parametric family of distributions. For example, we could assume that data follows normal distribution, which has two parameters, the mean μ and the variance σ^2. Thus, the task is to find a μ and σ^2 that best fits the data set. Non-parametric estimation does not constraint the data set to a particular parametric distribution. Instead, a function is used to approximate the density function f using the data itself.

Kernel density estimation (KDE) [Silverman, 1986] is a non-parametric approach for density estimation. Given a set of samples $x_1, x_2, \ldots, x_n$ in the data set X, we use a kernel K and a smoothing parameter h called

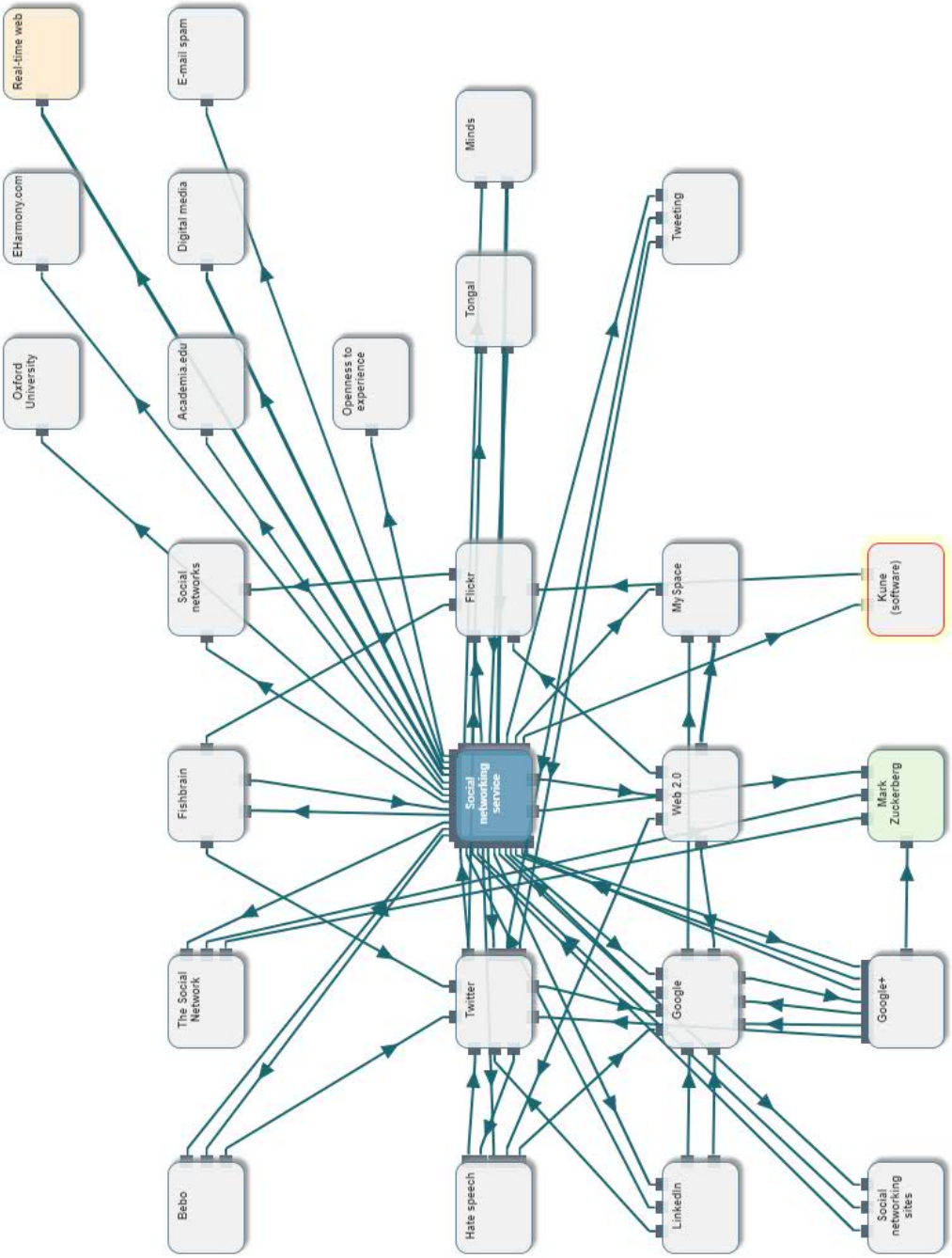

Figure 1.3: Using MTA SZTAKI LODmilla tool [Micsik et al., 2015] this figure presents a sub-graph of the associated to the Social_networking_service entity via the *wikiPageWikiLink* property. The edges represent connections between the nodes via any of their properties.

bandwidth to estimate the probability density function:

$$f(x) = \sum_{i=1}^{n} K(x, x_i; h). \tag{1.29}$$

The kernel K satisfies the condition:

$$\int_{-\infty}^{\infty} K(x)dx = 1. \tag{1.30}$$

In our experiments, K is the Gaussian Kernel defined as:

$$K(x, x_i; h) \sim \exp\left(-\frac{(x - x_i)^2}{2h^2}\right). \tag{1.31}$$

KDE is used in several location based recommender systems [Bao et al., 2015, Zhang and Chow, 2015] to find the areas where users move based on the observed location patterns. In this case, the data set X consists of latitude and longitude of user check-ins. In our research, we project the latitude and longitude to the three-dimensional euclidean space (x, y, z). Further details are presented in Chapter 2.

Chapter 2

Location-Based Social Networks

Cities around the world, and in particular large metropolises, have numerous points of interests (POI) to visit. People go to places either for culture, leisure, tourism, or entertainment. Location based social networks (LBSN) enable their users to share with their friends the places they go to and whom they go with. Given the number of possible places to go, finding a relevant POI is a time-consuming process as it requires people to search for POIs that satisfy their preferences and needs. To overcome this problem LBSN often recommend POIs that might be relevant to their users. Users of LBSNs find POI recommendations useful because it saves them time.

The task of recommending POIs is different from that of recommending other types of items (e.g., products, videos). As geography comes into play, Tobler's first law of geography states that "everything is related to everything else, but near things are more related than distant things" [Tobler, 1970]. The task of recommending POIs in LBSN to users is actively researched. Most of the research focuses on finding relevant POIs for individual users; techniques to provide recommendations to groups of users are scarce. We use the term 'group' to refer to sets of at least two (2) users, and the term 'individual' refers to a singleton set (one user). Figure 2.1 is an example of five users and their membership to different groups.

This chapter summarizes the research presented in [Ayala-Gómez et al., 2017], and addresses the problem of recommending a list of POIs to a group of users in the areas that the group prefers and contribute to answering the research questions **RQ1.1**, and **RQ1.2**. Additionally, we propose an approach to recommending a sequence of POIs to a group of users [Ayala-Gómez et al., 2018] contributing to answer **RQ1.3**. For each of these papers we briefly present the problem setting, working data set, proposed approach, evaluation methodology, experimental results, and conclusions.

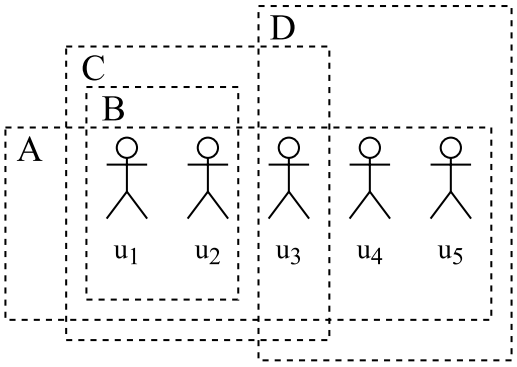

Figure 2.1: An example of the definition of a group. Groups have at least two members. A user might be a member of many groups. Groups are considered to be different if any of their members is added or removed. For example, the difference between groups B and C is the user u_3. This could be the case of a couple going out with a friend. Intuitively the preferences of a couple (i.e., u_1, u_2) might be different when they are just the two of them, versus when they go out with u_3.

My contributions to our research in [Ayala-Gómez et al., 2017] and [Ayala-Gómez et al., 2018] were on the problem formulation, data collection, data exploration, proposed approach, recommender systems implementation, and experimentation.

2.1 POI recommendation for groups

The focus of this section is on *recommending new POIs to a group of users*. This task differs from recommending venues to individual users. Consider that when users visit a venue with somebody else (e.g., partner, friends) the type of venue can generally differ if they do so alone. For example, someone who is a big fan of tacos going out with friends who prefer pizzas. Such conflict of interest is challenging for building recommendation for groups and raises some questions: *can we quantify the differences between individual preferences and group preferences?* and *what POIs to recommend for a group of users if the individual preferences differ?*. We restrict ourselves in recommending *new POIs* (i.e., venues that the group has not visited in the past), since new recommendations are more interesting than already visited POIs.

To analyze and experiment with the recommendations, we collected a

data set from Swarm[1], a LBSN developed by Foursquare[2]. The data cover activities in three major cities: Istanbul, Izmir and Mexico City. The use case of the POI recommendations is a group of users who plan to meet and look for recommendations for a new place to try. For this purpose, as a first step, the group selects an area among the ones they have been to in the past; then, the system provides a list of unvisited POIs as recommendations for the selected area. We experiment with numerous techniques drawn from the literature; and present the Geo-Group-Recommender (GGR), a context-aware recommender system that combines collaborative and content filtering together with a geographical Kernel Density Estimation (KDE, Section 1.6). Our results show that the proposed recommender system outperforms existing systems and other baselines.

2.1.1 Literature Review

A recent survey of recommending venues for individual users in LBSNs can be found in [Bao et al., 2015]. State-of-the-art models like FMFMGM [Cheng et al., 2012] and GeoSoCa [Zhang and Chow, 2015] exploit users' geographical and social information. The idea of including the location preferences in the collaborative filtering learning is presented in GeoMF [Lian et al., 2014]. However, research on recommending venues to groups is still scarce but emerging and promising.

There is scarce research work specialized in POI recommendation for groups. The behavior of groups in LBSNs has been researched for different tasks. [Liao et al., 2016] studies the case when a person wants to visit a certain POI and would like to know which of her friends could be interested in joining her. [Anagnostopoulos et al., 2016] focuses on recommending an itinerary for tourist groups visiting a city.

A study that compares users and groups behavior in LSBN is investigated in [Brown et al., 2014]. For this purpose, the authors use data from Foursquare (i.e. Swarm) and Telecommunications networks. They show that the venue type and its location affects the groups meeting at a certain place, and states that this behavior could affect the POI recommendation task. Our results complement [Brown et al., 2014] in several ways: we analyze check-ins that explicitly mention friends that are together instead of co-located within an hour; study the behavior of Swarm users from other cities than New York; quantify the category preferences of users and groups using Kendall-tau as a ranking correlation metric, and study time

[1] `https://www.swarmapp.com`

[2] `https://www.foursquare.com`

Model	Categories	Geography	Social	Group	top-k @	Prioritize POIs
GGR (Ours)	Yes	Yes	No	Yes	5-50	Group Geo Density
CGAR [Purushotham et al., 2014]	Yes	No	Yes	Yes	50-1000	No
GeoMF [Lian et al., 2014]	No	Yes	No	No	5-100	User Geo Density, POI influence
GeoSoCa [Zhang and Chow, 2015]	Yes	Yes	Yes	No	2-50	User Social Network
FMFMGM [Cheng et al., 2012]	No	Yes	Yes	No	5-10	No

Table 2.1: Differences between our proposed approach to building POI recommendations compared to recent LBSN recommenders. The headers represent the features that are used in the model: *categories* are the POI types (e.g., museum, restaurant, park), *geography* is the POI location, *social* is the social network, and *group* show if the model is used to build POI recommendations for groups.

and distance between check-ins. Our data utilizes all of the group and users' check-ins instead of just the top POIs for users and groups. Finally, we present empirical results on the performance of recommender systems for groups in LBSN.

Recommending POIs for groups is studied in [Purushotham et al., 2014]. The authors identify groups using connected components based on time, location, and network of friends in the Gowalla data set. Their model, called Collaborative Group Activity Recommender (CGAR), represents group and location activities as topic models that are combined using collaborative filtering techniques. Their model includes latent variables for activity preference and community influence that express whether an activity at a location is more interesting for one group than to another, as well as how user communities influence the preference of locations. They highlight that preferences between users and groups differ and show that their model personalizes category preferences better than regular strategies to combine individual recommendations (i.e., aggregating by average).

Table 2.1 highlights the differences of our approach, the Geo-Group-Recommender (GGR), in comparison with recent LBSN recommender systems.

The authors of [Masthoff, 2015] highlight that the use case of the recom-

Dimension	Use Case
Group preferences [Masthoff, 2015]	Preferences are known vs. developed over time
Item consumption [Masthoff, 2015]	Recommended items are experienced by the group vs. presented as options in a list
Item selection [Masthoff, 2015]	The group is passive (e.g., users are not voting) vs. active (e.g., the system helps create consensus)
Number of recommended items [Masthoff, 2015]	Recommending a single item vs. a sequence or a set
Preferences collection (Feedback) [Campos et al., 2009]	Explicit vs. Implicit
Group Identification [Campos et al., 2009]	Group profile vs. aggregation of individual preferences

Table 2.2: Characteristic of group recommender systems as defined by [Masthoff, 2015] and [Campos et al., 2009]. Characteristics that apply to our setting are underlined.

mender system greatly affects its design. They characterize group recommender systems as described in Table 2.2. In our case, the user preferences are known, recommendations are presented as options in a list, the group is passive, the recommendation is a single item, the preference collection (i.e., feedback) is implicit, and we build recommendations using group profiles.

2.1.2 Working Data set

To compare the behavior of users when they are alone and when they are in a group we require a data set from a LBSN containing individual and group check-in information. Most of the publicly available LBSN data sets [Bao et al., 2012, Cho et al., 2011, Gao et al., 2013, Brown et al., 2014, Noulas et al., 2011] have information about the activity of individual users and their friends, but no group information and some lack detailed venue information. To overcome this problem we collected data from Swarm, a popular LBSN platform that enables users to indicate the venue they are *checking-in*. On Swarm, *users are able to mention with whom they visit a venue* and share their check-ins publicly in other social networks, like Twitter[3]. The following is an example of a publicly available check-in on Twitter where *@user_1* shares the Tweet: "I'm at *POI* w/ *@user_2, @user_3* `https://www.swarmapp.com/c/123abc`", *POI* is the venue where

[3] `https://www.twitter.com`

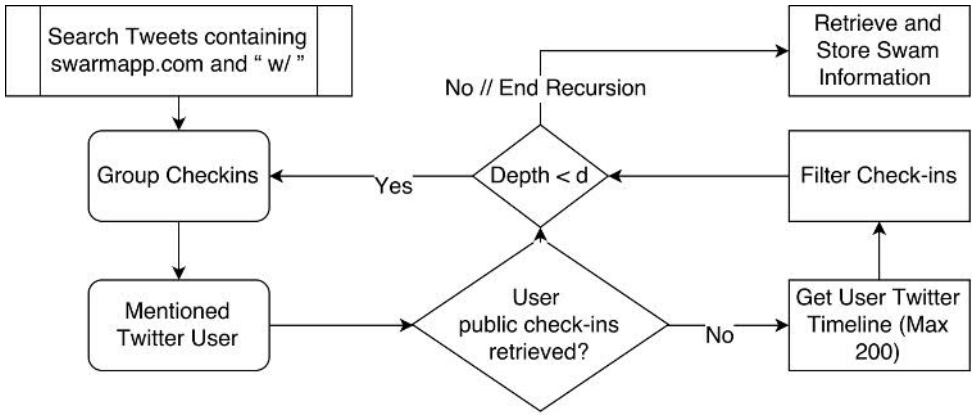

Figure 2.2: The crawling process of Swarm check-ins. Additionally, the lookup endpoint of the Twitter API allow us to constrain the search to a specific location by defining the geographical center and a radius.

the group (i.e., the three users) is at, *@user_2, @user_3* are users mentioned by *@user_1*, and `https://www.swarmapp.com/c/123abc` provides more details about the check-in. This gives us the opportunity to collect data related to both individual and group activity.

To collect the data we deploy a crawler that uses the Twitter API[4] to search for public tweets that contain group check-ins. For each of the group check-ins we perform snowball sampling [Goodman, 1961]. We use snowball sampling to gather information not only about a user, but also about their acquaintances. This will help us to analyze the preferences of a user between the groups that she is a member of. We do this recursively for all the member of the groups as this will increase the chance of observing the users in many groups. The following is an explanation of the process. We collect the latest 200 tweets from each of the users that are mentioned in a group check-in, and extract the users' public check-ins. Figure 2.2 is a visualization of this recursive process. We do this recursion until a maximum depth d from the original group check-in: We assign depth 0 for the first pass, depth 1 for users who are mentioned in a group check-in from depth 0, and so on.

Snowball sampling resulted in a global data set with approximately 143 K users, 522 K venues, 780 categories, 453 K groups, 5.6 M check-ins, and 1 M group check-ins. The Top-5 cities from the collection are presented in Table 2.3. The names of the cities were obtained by assigning to

[4] `https://developer.twitter.com/en/docs`

City	Total Check-ins	Total Venues	Total Cat.	Group Check-ins	Group Venues	Group Cat.
Istanbul	483 K	25 K	402	43 K	8 K	297
Izmir	369 K	16 K	378	37 K	4 K	263
Mexico City	95 K	15 K	354	12 K	4 K	271
Kuala Lumpur	69 K	12 K	359	3 K	1 K	203
Bursa	59 K	4 K	283	5 K	1 K	164

Table 2.3: Top-5 cities in the data collection.

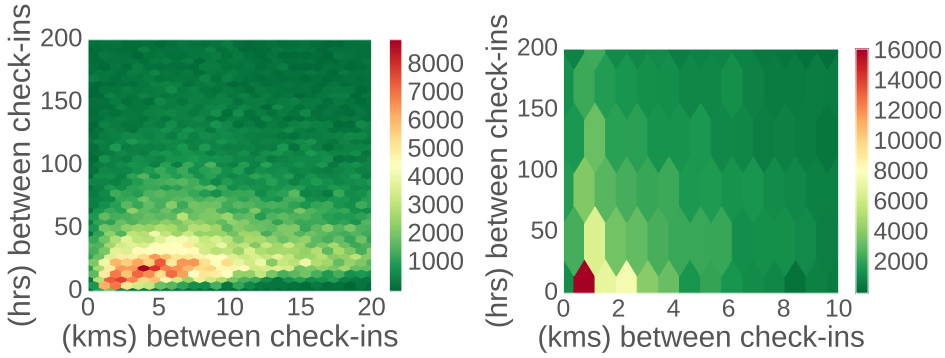

Figure 2.3: Frequency of time and distance between check-ins for users (left) and groups (right).

each check-in the closest city from the Geonames database[5]. For proximity search, we used R-Trees [Guttman, 1984].

We observed that there were very active users, which are likely bots. These users have a big geographical dispersion in their check-ins. We removed these users by computing quartiles of the standard deviation of their geographical mobility, and removed the fourth quartile. Moreover, in our experiments, we consider groups with a maximum of twelve members and removed the last quartile of inter-time and inter-distance of the groups. Finally, we removed approximately 2.3 M check-ins with categories irrelevant for our research (i.e., *Residence*, *States & Municipalities*, *Professional & Other Places* and *Event*, *College & University*, *Travel & Transport*). Figure 2.3 presents the time and distance between check-ins for users and groups. We removed the last quartile of time and distance. After cleaning the data set, it contained approximately 143 K users, 311 K venues, 597 categories, 67 K groups, 2.2 M check-ins, and 221 K group check-ins. The data is available for research purposes.

[5] `http://download.geonames.org/export/dump/`

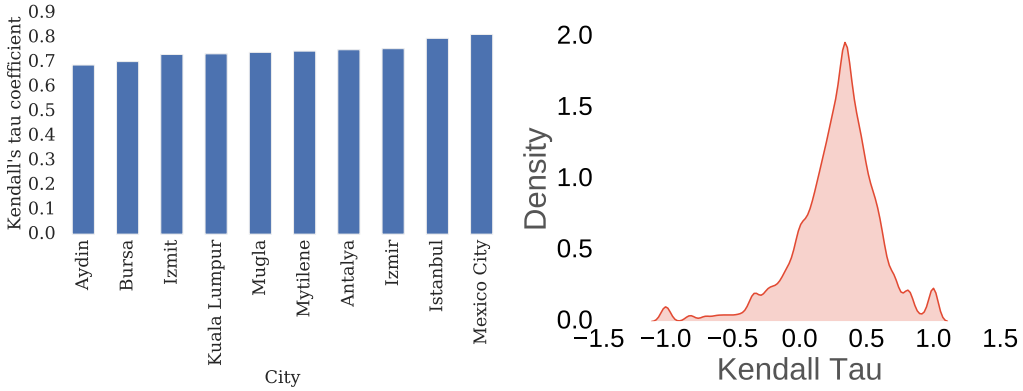

Figure 2.4: Kendall-tau for user and group category preferences. Left: average by city; Right: distribution on users.

2.1.3 Proposed Approach

Category Preferences

To measure the difference between group and individual preferences, we construct a ranked list of the preferred categories for individuals and groups by check-in count. We measure the rank correlation using the Kendall-tau rank correlation coefficient [Kendall, 1945]. Specifically, we use Scipy Python implementation of Kendall-tau rank correlation[6] defined as:

$$tau = \frac{P - Q}{\sqrt{(P + Q + T) * (P + Q + U)}}, \tag{2.1}$$

where P is the number of concordant pairs, Q the number of discordant pairs, T the number of ties only in the group's rank, and U the number of ties only in user's rank. If a tie occurs for the same pair in both group's rank and user's rank, it is not added to either T or U. Note that the values of Kendall-tau are between $[-1, 1]$, where values closer to -1 indicate a strong disagreement, and values closer to one indicate a strong agreement.

We measured the Kendall-tau rank correlation at three levels: for all the cities, per city, and per user. For all the cities, Kendall-tau was 0.82. Figure 2.4 presents the Kendall-tau at the city and user level. In Figure 2.5, we give examples of a city and a user by parallel coordinates.

[6] `https://docs.scipy.org/doc/scipy-0.15.1/reference/generated/scipy.stats.kendalltau.html`

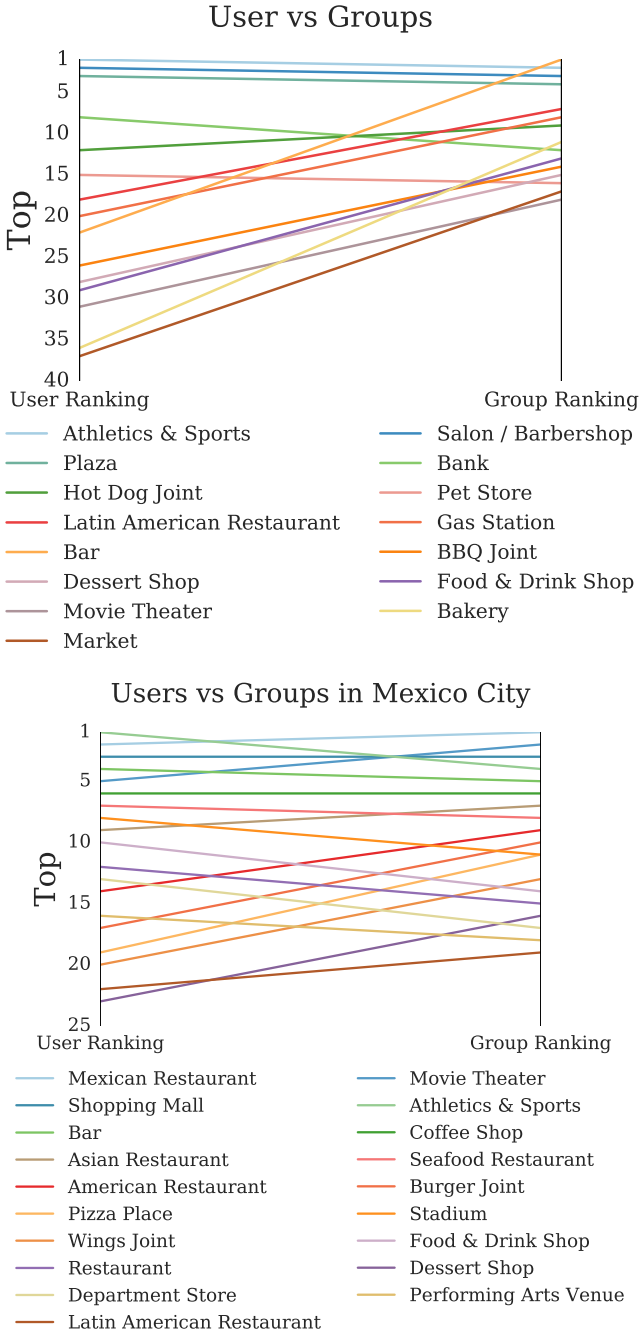

Figure 2.5: The comparison between the preferences of an individual and her groups. As they are very different the Kendall-tau is 0.1 (top). Total check-in count for all the users and groups in Mexico City. The Kendall-tau is 0.8(bottom).

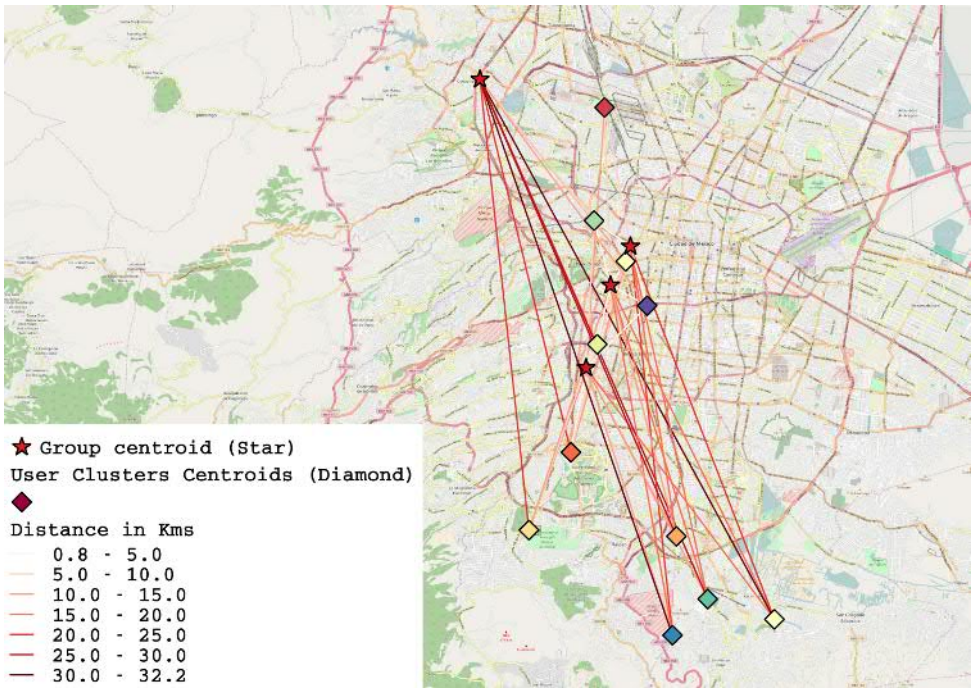

Figure 2.6: A map of Mexico City with an example of a user and her group location preferences. The *diamond* shapes are the centroids of the user check-in clusters. The *star* shapes are the centroids of the group cluster.

Location Preferences

To compare the location preferences of individuals and groups we first identify the areas that users and groups visit, then, we compute a weighted average of movement of the user to the groups. To do this, we use DBSCAN [Ester et al., 1996] and *Vincenty distance* to find clusters based on the user check-ins and on the group check-ins. An example of this is shown in Figure 2.6.

For each user and her groups we compute the weighted average distance between the user clusters and group clusters. Given the user check-in cluster centroids c_u, total user check-ins per cluster w_u, group check-in cluster centroids c_g, and total check-ins per cluster w_g, the weighted average distance is computed as:

$$d(c_u, c_g) = \frac{\sum_{c_{ui}} \sum_{c_{gj}} w_{ui} w_{gj} d_{vinc.}(c_{ui}, c_{gj})}{\sum_{c_{ui}} \sum_{c_{gj}} w_{ui} w_{gj}}. \tag{2.2}$$

We want to allow clusters to form where the POIs are at maximum 1.5 km (i.e., about 10 blocks) away from each other – and, if a POI is too

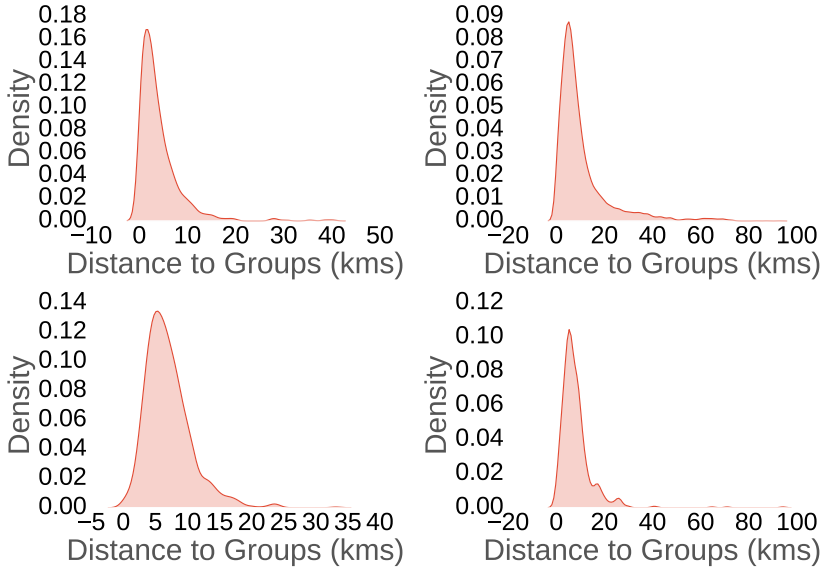

Figure 2.7: The KDE of users individual location preference vs. group location preference in four major cities. Lower means that the user had to travel less to meet the groups. From left to right and top to bottom: Istanbul, Izmir, Mexico City and Kuala Lumpur.

far away, we consider it as an independent cluster. We used an *epsilon* of 1.5 km and *minimum points* of 1 as the parameters for the DBSCAN clustering.

Figure 2.7 shows the Kernel Density Estimation (KDE) for the weighted average traveling distance that users need to move to meet with the groups. We observed that the KDE of the average weighted distance saturates between 5-10 kms. This means that users are willing to move from the areas that they frequent to meet with the groups.

The Geo-Group-Recommender (GGR)

Our proposed approach to building recommendations of unvisited POIs to groups in known areas to the groups, combines recommender systems from the literature with: *i*) POI pre-filtering, *ii*) category and location feature engineering, *iii*) model training based on group check-ins.

The model is based on several observations. From Figure 2.3 we observe that groups do not travel much between their check-ins. Therefore, it is natural to geographically narrow down the set of possible POIs. We do this by fitting KDE as described in Section 1.6. KDE helps us to differentiate the dense areas for a group based on the geographical check-in distribution, as

Model	Acronym
iALS Matrix Factorization (1.1.4)	iALS
KDE ∩ iALS Matrix Factorization	KDE iALS
SGD Matrix Factorization (1.1.4)	SGD
KDE ∩ SGD Matrix Factorization	KDE SGD
Item To Item (1.1.4)	ITEM-ITEM
KDE ∩ Item To Item	KDE ITEM-ITEM
Popularity Recommender (1.1.2)	POP
KDE ∩ Popularity Recommender	KDE POP
Content Based Recommender (1.1.3)	CB
KDE ∩ Content Based Recommender	KDE CB

Table 2.4: Models used to generate recommendations.

shown in Figure 2.8 for one group. Specifically, we fit a KDE for each group using the check-ins in the training data set. Subsequently, we compute a density score at the location of each venue, and keep as candidates only the venues in the highest (most dense) quartile. These POIs are then passed on to the recommender system for ranking.

In our experiments the variants that include POI pre-filtering contain the keyword *KDE* in their name. The second type of variant also includes category and/or geolocation features. These variants are denoted with *GEO* and *CAT* in their acronym. The *GEO* and *CAT* features are included as side-information in the recommender system. Among the third type of variants we distinguish if the recommendations are built specifically for the group or aggregated using individual recommendations. Specifically, the recommender systems generate a rating $r_{u,i}$ for each pair of user u and candidate POI i that are combined with one of the strategies described in Section 1.3.1. The variants that include the POI pre-filtering, category features, location features, and are trained on the group check-ins are collectively referred to as GGR (e.g., KDE IALS, KDE POP CAT, KDE POP, KDE POP GEO).

Table 2.4 includes the names of the recommender systems that differ along the first two dimensions. Recommender systems that are trained on individual user check-ins have one of the three acronyms (i.e. AIR, AWM, ALM) appended to their name.

2.1.4 Evaluation Methodology

Figure 2.9 gives an overview of the evaluation methodology. For the three largest cities in our data set (Istanbul, Izmir and Mexico City), we split the

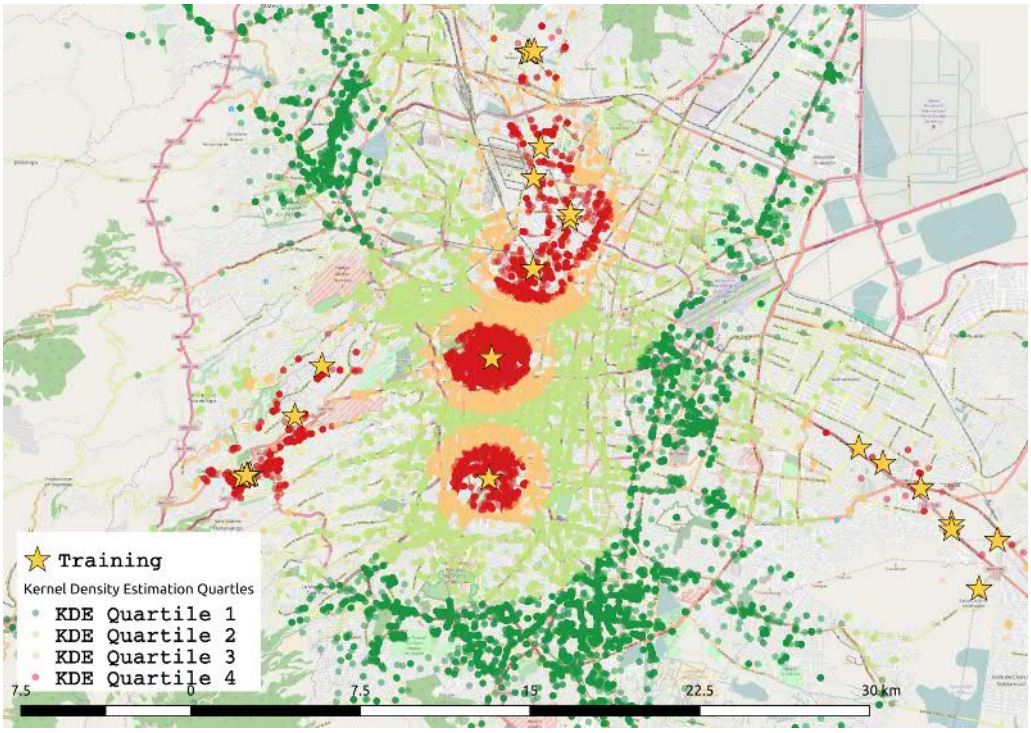

Figure 2.8: Example of pre-processing of geographical information to calculate the KDE for a particular group. The map shows the POIs that could be recommended in Mexico City colored by the KDE score for a particular group. The *stars* shapes represent the check-ins used to fit the KDE. In our experiments we picked venues at the 4^{th} quartile (i.e., venues near the most visited areas by the group) as POI candidates.

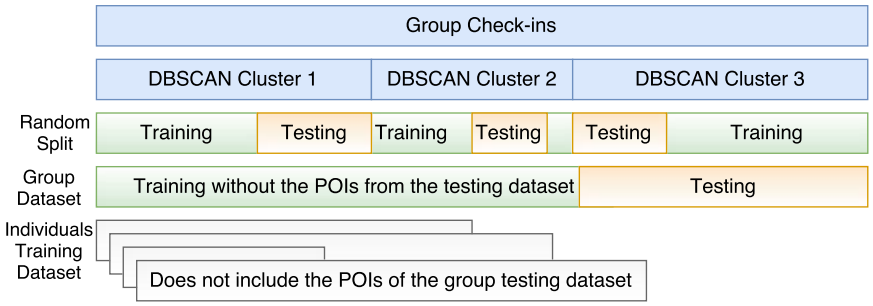

Figure 2.9: Random split per group and cluster.

check-ins at the group cluster level (i.e., clusters detected by DBSCAN [Ester et al., 1996]) to create two data sets. The training set contains approximately 70% of the check-ins of the group cluster, and the remaining data comprises the testing set. Group clusters with size lower than the median were added to the training set. We combined all of the group cluster splits to create one training and one testing data set. Figure 2.10 gives an example of how we split a group check-ins into training and testing.

We used the Apple Turi Create[7] implementation of the recommender systems. The model hyper-parameters were fine-tuned using grid search on a validation set (i.e., 5% of the training set). The experiments were conducted in a single machine with 40 cores and 200 GB of RAM and ran for a day.

For performance metrics we use precision and recall at different cutoffs K (i.e. 5, 10, 20, 30, 40, 50), as described in Section 1.4.

2.1.5 Experimental Results

Our main result is the comparison of recommender system algorithms in different large cities. First, we compared the performance of recommender systems trained on group profiles versus aggregating user recommendations. Figure 2.11 shows the comparison for iALS. We observed that using the group profile to build the recommender system performed better than the aggregation of individual recommenedations. Thus, the following results use the group profiles to build the recommendations. GGR models are top performers and the types of recommender systems perform differently among cities. Figure 2.12 shows the results for Istanbul, where KDE iALS performs best for both precision and recall. Figure 2.13 shows the results for Izmir, where in contrast, KDE SGD GEO performs best for both pre-

[7] `https://github.com/apple/turicreate`

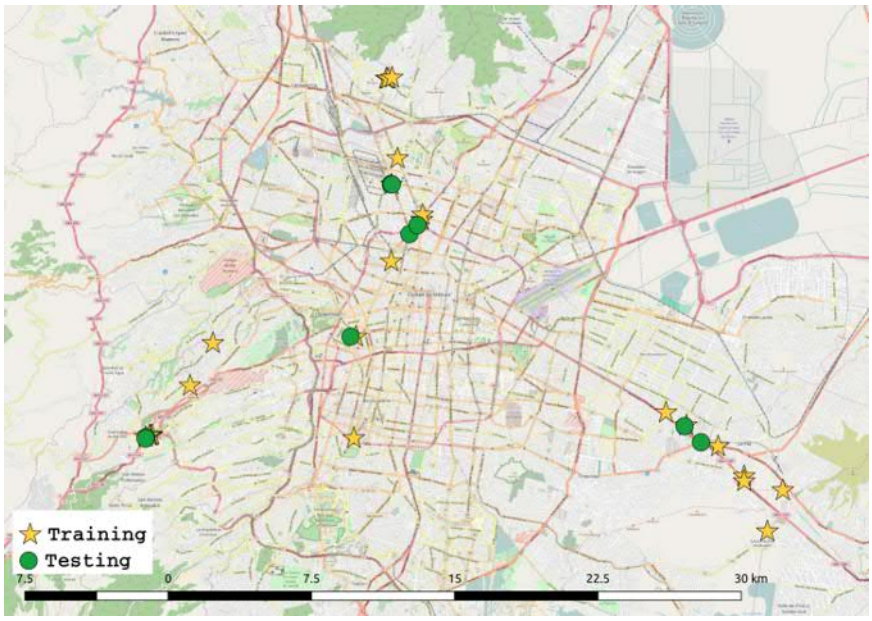

Figure 2.10: An example of the group check-ins split.

cision and recall. Finally, for Mexico City (Figure 2.14), KDE iALS again performs best for recall@5-10 while KDE SGD CAT for recall@20-50. Best performing methods are the same for precision as well.

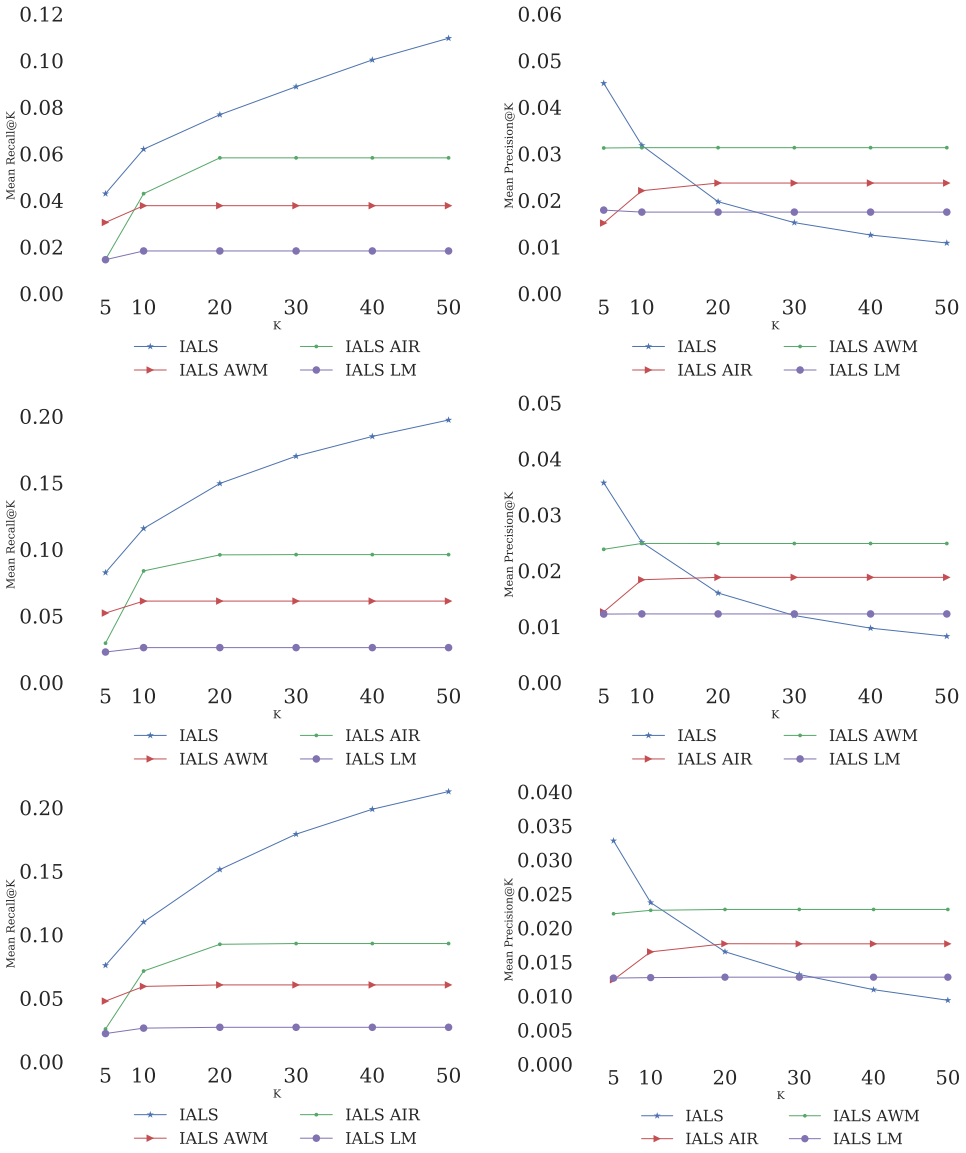

Figure 2.11: Combining individual iALS recommendations by equations(1.22–1.24) under-perform the group model. From top to bottom: Mexico City; Istanbul; Izmir. Left: Recall; Right: Precision.

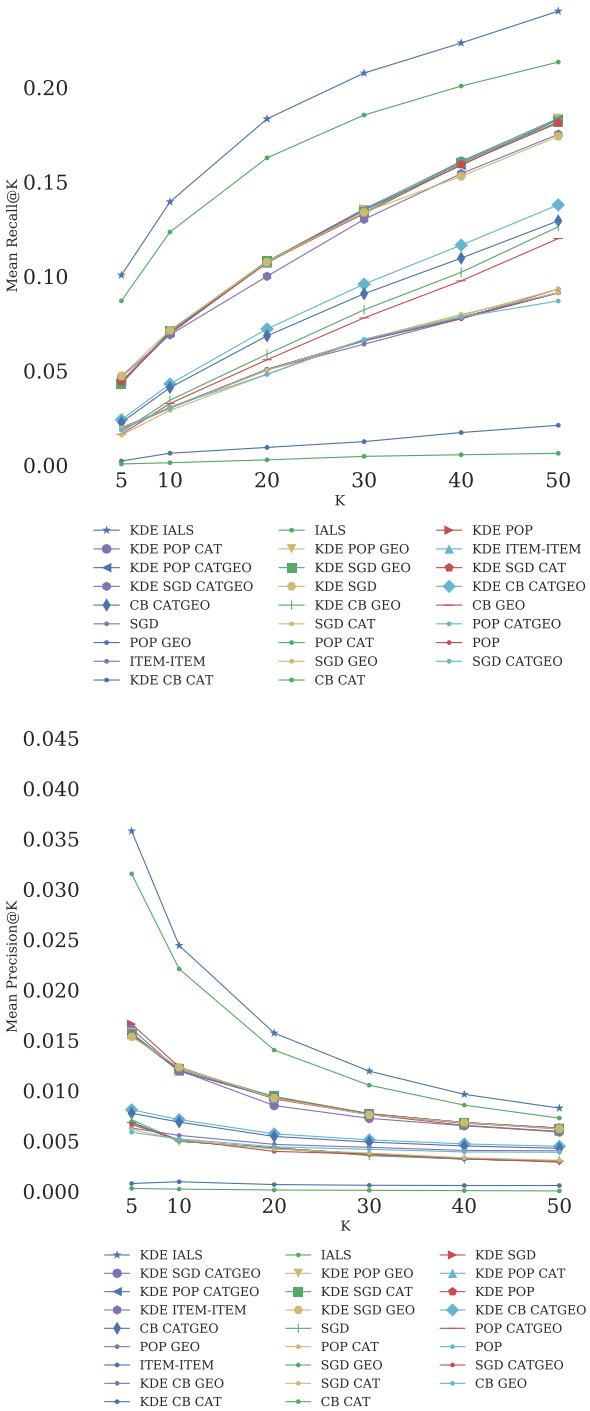

Figure 2.12: Istanbul Recall@K(top) and Precision@K(bottom). The legends are sorted by the best performance models from left to right and top to bottom.

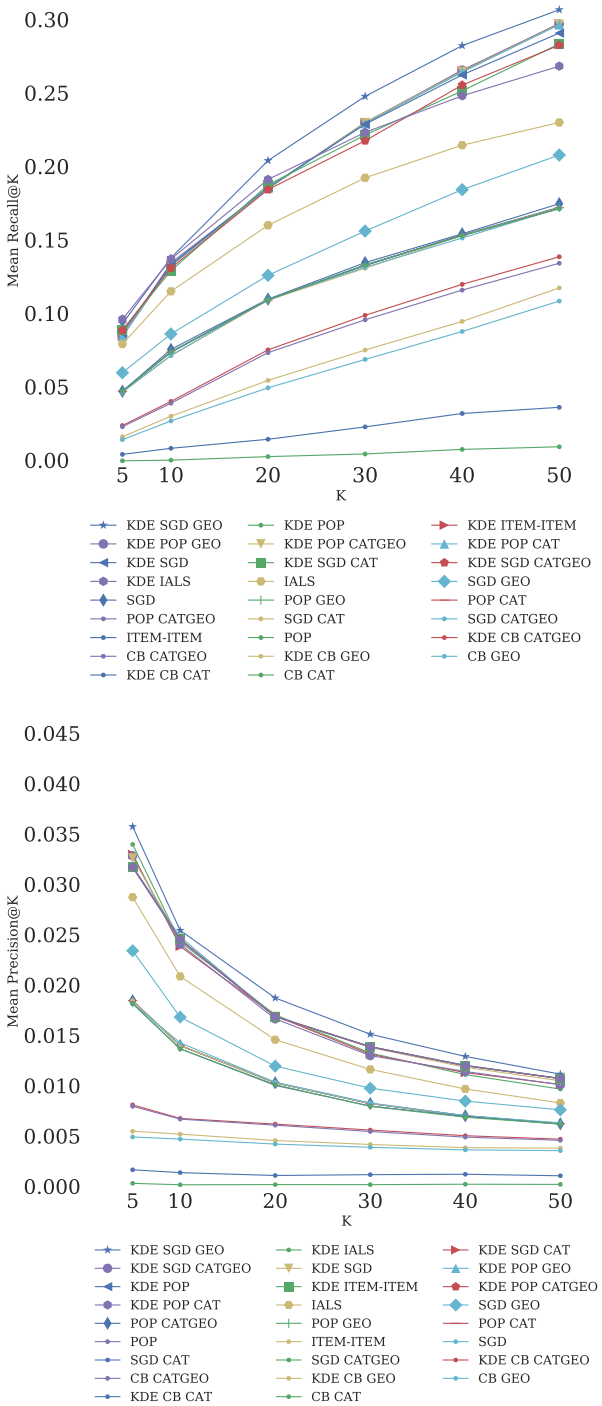

Figure 2.13: Izmir Recall@K(top) and Precision@K(bottom). The legends are sorted by the best performance models from left to right and top to bottom.

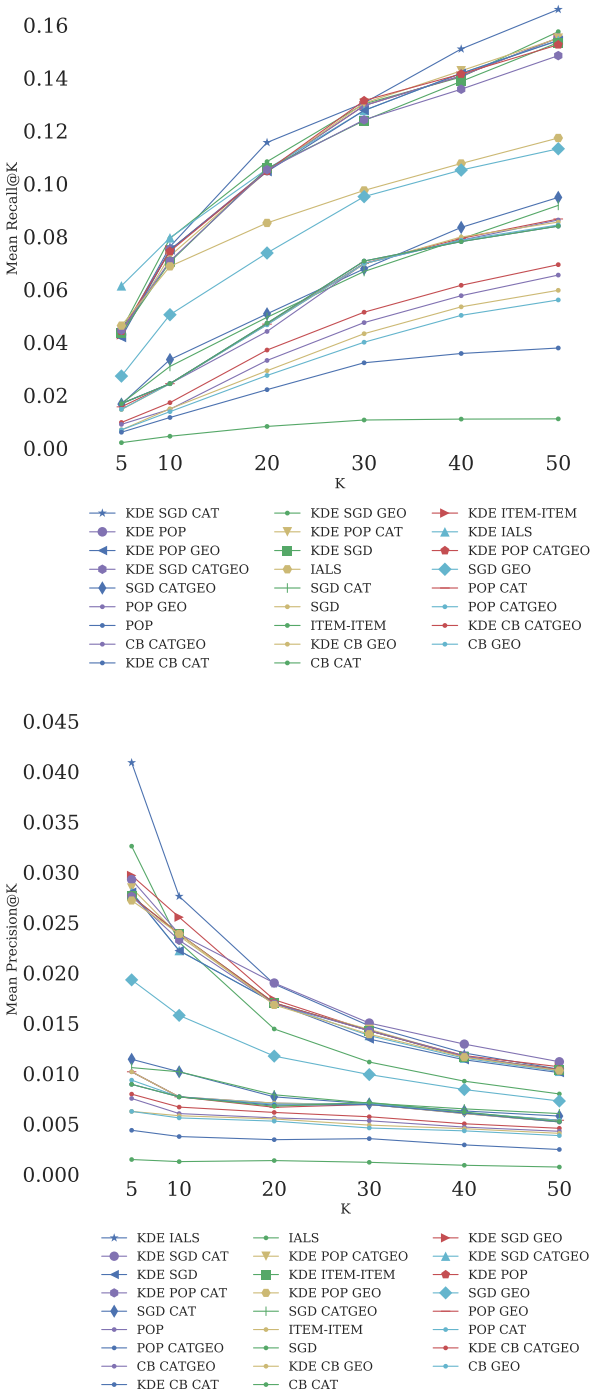

Figure 2.14: Mexico City Recall@K(top) and Precision@K(bottom). The legends are sorted by the best performance models from left to right and top to bottom.

2.2 POI Itinerary recommendation for groups

In the previous section we presented a model that generates POI recommendation for groups. In this section we focus on building recommendations for groups who are interested in visiting more than one place during their gathering in a given order (e.g., a restaurant, a historical site, and a coffee shop). This task is more challenging, as additional constraints (e.g., time) must be satisfied. This task is known to be the orienteering under several constraints (e.g., time, distance, type ordering). In [Ayala-Gómez et al., 2018] we studied the problem *recommending top-k sequences of POIs for a known group, given a starting POI, time constraint, and contextual information (e.g., popularity of the venue at time of arrival).* Recent work on group orienteering [Fan et al., 2017, Anagnostopoulos et al., 2017] is done in two steps. The first step is to train a recommender system, and the second step uses the scores given by the recommender system to build the itinerary. A common way for building group recommendations is to aggregate the group members' individual preferences. However, in our work we model group preference on the basis of observed group behavior; as we previously reported, recommendations based on group preferences perform better than aggregations of individual preferences. We model group preferences not only from the POI check-ins but also from the venue popularity at the time of arrival, its characteristics, and the observed POI transitions for each city in our data set.

In this section our goal is to design methods to recommend top-k sequences of POIs for a known group given a starting POI, time constraint, and contextual information (e.g., popularity of the venue at time of arrival). We conducted the experiments on a real data set of check-ins collected from Swarm by Foursquare and compared the recommender systems' quality using nDCG defined in Equation (1.27). For each city we used the best recommender system models with tree variants of Monte-Carlo Tree Search, and a greedy approach for itinerary construction. Our results show that the proposed approach improves the performance in comparison to baseline techniques.

2.2.1 Literature Review

Research on itinerary recommendation mostly focuses on building recommendations for individuals. In [De Choudhury et al., 2010] the authors investigate the usage of geo-temporal traces (i.e., breadcrumbs) left by travelers in social media platforms to generate itineraries. They collected data

from Flickr[8] and mapped the photos taken by users to POIs. The resulting sequences of photos are mined under several constraints to construct POI paths. The authors in [Gionis et al., 2014] also use geo-temporal traces of travelers on LBSNs. They propose two algorithms to construct itinerary under several constraints including those on venue types and distance. In [Roy et al., 2011] the authors include user feedback for each selected POI in the itinerary. User feedback is used to pick the next steps in the itinerary. The work presented in [Chen et al., 2014] focuses on generating itineraries involving several days. The authors generate all possible single-day itineraries and then combine them selectively for an optimal multi-day itinerary. In [Lim et al., 2017b] the user preference for a POI is weighted by visit duration and recency of past visits rather than visit frequency. In a follow-up study [Lim et al., 2017a] the authors studied the itinerary planning problem under queuing time consideration. The goal then is to maximize the user preference to the POIs while minimizing the queuing time.

Itinerary recommendation for groups has recently attracted attention as a research problem in the literature. In [Fan et al., 2017] the proposed algorithm aggregates individual preferences by considering agreement and disagreement among the group members. The work presented in [Anagnostopoulos et al., 2017] considers the task to be the *orienteering* problem, where the goal is to find a path in a graph within a time budget that has the best value for the group. The itinerary value is an aggregation of all of the individual preferences to the POIs in the path. They do this in two steps, where the first step is to use a content-based recommender system to build recommendations for individuals. The second step generates itineraries by maximizing different objective functions: *i) TourGroupSUM*, adds the individual POI recommendations; *ii) TourGroupMIN*, assigns the lowest individual score to the POI; *iii) TourGroupFAIR*, averages the value of the individual scores minus a weighted standard deviation as POI score. They found the highest valued itineraries using ant colony optimization, although the scores did not vary much compared to the greedy baseline.

One of the basic differences in our work from the two previously summarized approaches is that rather than aggregating group member preferences we use the known group preferences. In that sense, we have a common approach with the study in [Ayala-Gómez et al., 2017]. We focus on recommending a list of top-k itineraries instead of a single itinerary. In terms of the methods used to solve the orienteering task we use the Monte-Carlo Tree Search (MCTS), and added contextual features during the itinerary planning such as time and transition probabilities.

[8] `https://www.flickr.com/`

City	Check-ins	Venues	Categories	Group Check-ins	Group Venues	Group Categories
Istanbul	763K	32K	423	205K	18	362
Izmir	262K	14K	373	46K	5K	271
Mexico City	179K	19K	370	71K	12K	340
Tokyo	108K	19K	392	7K	3K	264

Table 2.5: Statistics for the Top-4 cities in our working data set.

2.2.2 Setting

We define the problem of itinerary planning for a group of users as follows: Given a group G of LBSN users, a set of POIs (venues) V, and a POI v_0 as a starting point, the aim is to construct a path $I =< v_0, \ldots, v_n >$, where $v_i \in V$, such that I fulfills given budget and POI type constraints. The budget constraints are total time and distance.

We assume that some history of check-ins for the group and its members is available, and that POI metadata (e.g., popularity, tips, description, ratings) is also available. Another assumption is that each POI has a type (category).

Following the classification shown in Table 2.2, we can characterize methods for building an itinerary of POIs for groups of users in LBSN as follows: Preferences are known, recommendations are presented as options in a list, the group is passive, we recommend a sequence of items (POIs), feedback is implicit, and we use the group profile.

2.2.3 Working Data set

For the itinerary recommendation task, we extended the data set of Section 2.1. In this new data set, we narrow down the search on specific cities. We collected a data set that has approximately 2.7 M individual check-ins, 1 M group check-ins, 200 K users, 400 K groups, 116 K venues, and 547 categories. We followed the cleaning steps described in Section 2.1.2. Table 2.5 presents statistics of the groups, users, venues, and categories for the Top-4 cities after cleaning the data.

An important aspect of our work is using LBSN contextual features such as time and location. Time-wise, we observed seasonal effects in the time of the day, day of the week and month: the popularity of venues varies by week, hour, and season. For example, bars are more popular during the weekends and evenings, while museums are more popular during mornings and afternoons. Figure 2.15 presents examples of the effects that the weekday, time, and venue type have on the venue's popularity.

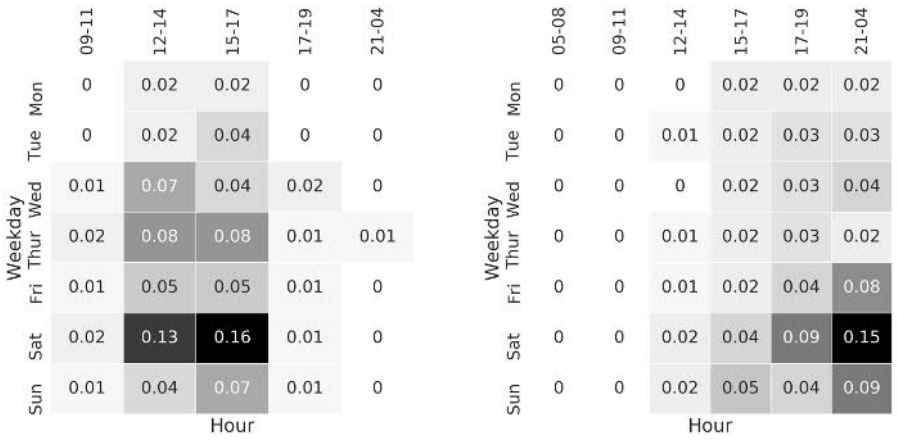

Figure 2.15: Distribution of check-ins per weekday and hour for museums (left) and bars (right) in Istanbul. The rows are the parts of the day, and the columns are days of the week. A darker color represents higher distribution of check-ins.

We sorted the group check-ins by time to study the transitions between venues. These POI transitions help us to understand popular venue type transitions in a city. Figure 2.16 shows an extract from the transition matrix for Istanbul.

2.2.4 Feature engineering

The contextual information in the data set is represented using different types of features. We pre-process content, transition, and temporal features to describe different aspects of the venues, groups, users, and context.

Content Features: Foursquare provides textual information that gives more information about the venues. POIs may have *tips* left by other users, *phrases* that describe the venue, and *attributes* (e.g., romantic, cozy). We performed conventional pre-processing steps as described in Section 1.1.3 on the tips, phrases, attributes, and on the venue category tree[9]. We also included a categorical feature provided by Foursquare that represetnts the price tier (e.g., cheap, moderate, expensive). Finally, we add as features the venue rating and the number of check-ins. In our experiments these features are used by the content-based recommender system.

[9]`https://developer.foursquare.com/docs/resources/categories`

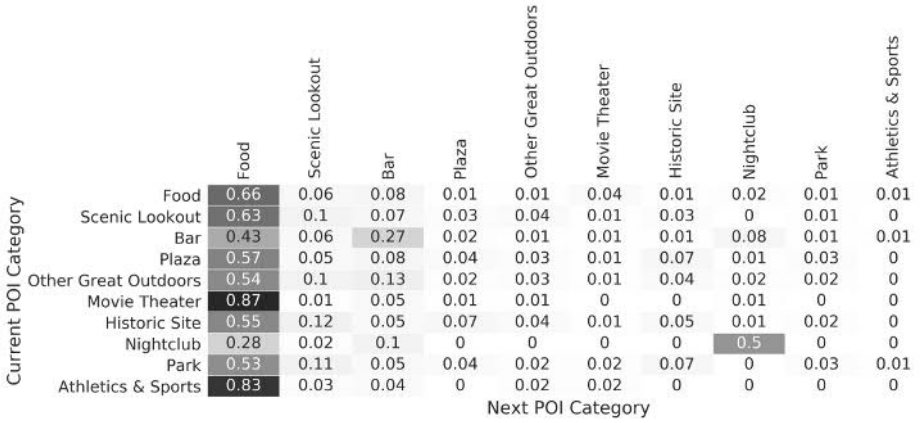

Figure 2.16: Part of the transition matrix for activity types in Istanbul. The numbers represent the transition probability between two categories and show that moving from a venue of certain category conditions the category of the next venue. For example, if a group is in a park it is more likely that they will go to a food establishment than to a nightclub. If the group is in a nightclub; they are more likely to go to a food establishment afterwards than to a park.

Transition Features: Given a current POI category c_i and a possible next POI category c_j, we compute the empirical conditional probability as $ecp(c_j|c_i) = \frac{n_{ij}}{n_i+1}$, where n_{ij} is the number of observed transitions between c_i and c_j, and c_i is the number of check-ins at c_i.

Temporal Features: We split the day into 6 segments: 05-08 (Early Morning), 09-11 (Late Morning), 12-14 (Early Afternoon), 15-17 (Late Afternoon), 17-19 (Early Evening), and 21-04 (Night). For each of these segments, we count how many check-ins are in the venue and venue type.

2.2.5 The Itinerary Graph

For each group in our training set, we construct a weighted graph where nodes are the candidate POIs that could be included in the itinerary. The edges have two weights; one that represents the value, and the other that represents the cost. The value depends on the contextual information and the score given by the recommender system. The cost of the edge is the time required for visiting the POI, calculated based on stay time and commute time. For stay time, we use the average stay time for the category of the end node. We determine the average stay time for a venue using the

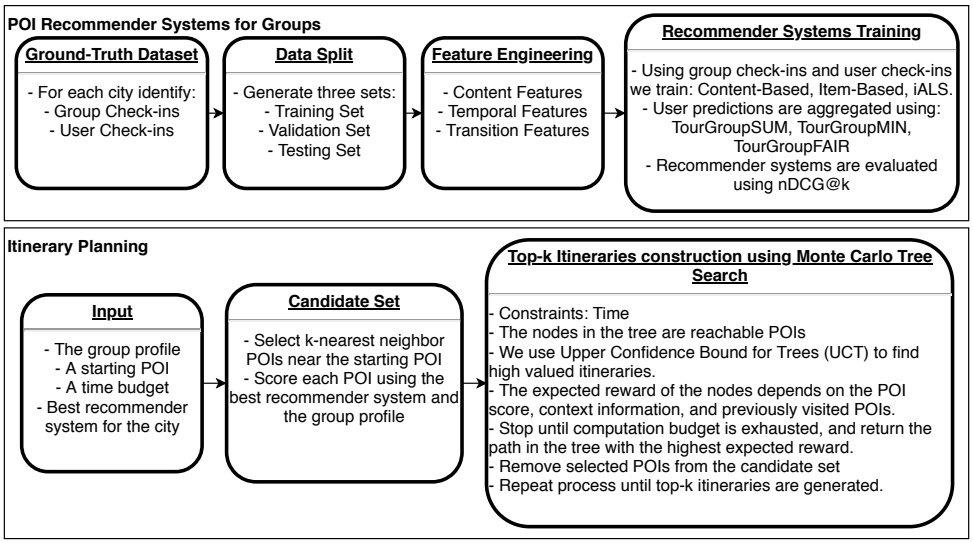

Figure 2.17: Recall at different candidate set sizes for the cities.

time difference between the transitions in the data set. Commute time is calculated by dividing the Vincenty distance by the average human walking speed 4.9 km/h. We represent the itinerary as a path T containing POIs. The cost and value of the itinerary are defined as:

$$\text{cost}(T) = \sum_{v \in T} \text{cost}(v), \tag{2.3}$$

$$\text{value}(T) = \sum_{v \in T} \text{value}(v). \tag{2.4}$$

2.2.6 Proposed Approach

Figure 2.17 presents an overview of the proposed solution. Our proposed solution consists of two steps. First, we train a recommender system using the known check-ins from a ground-truth data set. Then, we find a set of K nearest POIs to starting POI and use it as the candidate set. We score the POI with the recommender system. These ranked candidates are used as input to solve the orienteering problem with Monte Carlo Tree Search.

2.2.7 Candidate Sets

In [Ayala-Gómez et al., 2017], we observed that groups do not travel long distances between check-ins. Based on this behavior, we limit the number of candidate POIs to be used in the itinerary construction by always picking

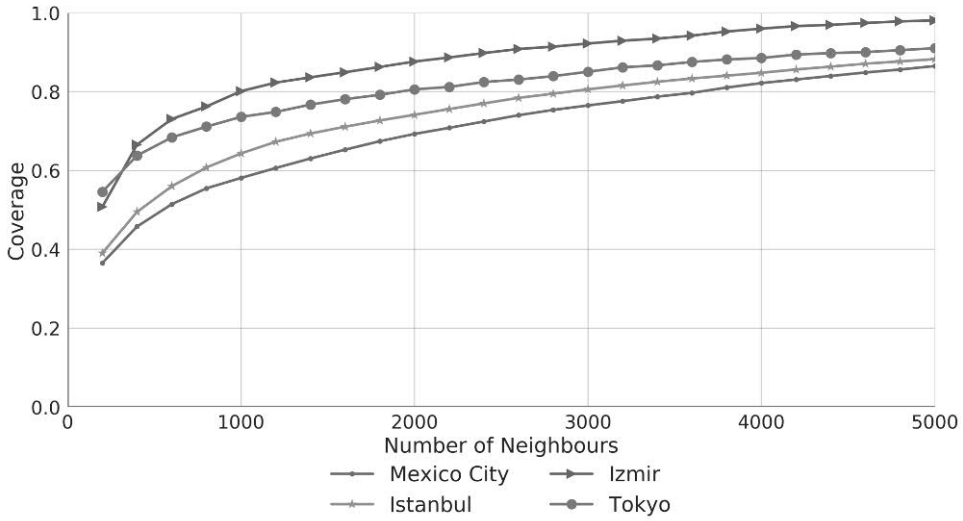

Figure 2.18: Recall at different candidate set sizes for the cities.

venues near the *current* venue of the group. For a given location, we consider the C closest venues for some constant C. We measure the quality of this candidate set C by measuring if the *next* venue that the group goes is in the candidate set. Figure 2.18 shows the recall of the candidate set.

2.2.8 Recommender Systems

We model the group preference for each of the POIs in the candidate set using well-known recommender systems from the literature. We experiment with content-based recommender system [Ricci et al., 2015] based on the content features described in Section 2.2.4, item-based [Linden et al., 2003], and collaborative filtering model for implicit feedback (iALS) [Hu et al., 2008].

We build recommendations for groups using two approaches. The first approach is to create a profile for each of the groups using the observed check-ins in the ground-truth data set. The second approach is to build a profile for each user using the known check-ins in the ground-truth data set, and aggregating the predicted scores for each of the users in the group. Similar to [Anagnostopoulos et al., 2017], we use three different aggregation functions: *i) IndividualSUM* is the sum of the group member recommendations, *ii) IndividualMIN* the min value of the group member recommendations, and *iii) IndividualFAIR* assigns the mean value of the scores minus one standard deviation.

2.2.9 Itinerary Construction

Our proposed solution for the orienteering problem uses Monte-Carlo Tree Search (MCTS) to efficiently find a promising itinerary. MCTS is built on the idea of combining tree search with random sampling, and finds optimal decision paths [Browne et al., 2012]. We use Upper Confidence Bound for Trees (UCT), "the most popular algorithm in the MCTS family" [Browne et al., 2012]. UCT recursively builds a tree by expanding nodes in each iteration, selecting the most promising node, estimating the expected reward of the selected node, and back-propagating the reward to the parents. This process ends when a certain computational budget is reached. The best path is found by traversing the tree from the root and picking the child node with highest expected rewards until a terminal node is reached. In our case a tree node represents a POI.

```
UCTSearch(v_0, variant, Cp)
// The root node v_0 is the starting POI.
// variant is a MCTS variant (i.e., MCTS_Simple,
   MCTS_Score, MCTS_Context)
while within computational budget do
    v_l ← TreePolicty(v_0)
    Δ ← DefaultPolicy(v_l, variant)
    UpPropagate(v_l, Δ)
end
itinerary = [v_0]
v ← BestChild(v_0, 0)
while v is non-terminal do
    itinerary.append(v)
    v ← BestChild(v, 0)
end
return itinerary
```

Algorithm 1: The main function of the UCT Algorithm for finding an itinerary.

We modified the UCT algorithm described in [Browne et al., 2012] as presented in Algorithm 1. The additional functions used in UCT are described in Algorithm 2.

Tree policy controls the trade between exploration and exploitation. It *expand*s a node if it is not fully expanded or terminal, otherwise it picks *BestChild*, which is the node that requires more attention at the current iteration.

```
TreePolicy(v, Cp)
while v is non-terminal do
    if v not fully expanded then
        return Expand(v)
    else
        v ← BestChild(v, Cp)
    end
end

Expand(v)
// Choose a set of reachable POIs from v
forall v_j ∈ v.reachable_pois do
    v.add_child(v_j)
end
return v

BestChild(v, Cp)
return
```

$$\underset{v_j \in v.children}{\operatorname{argmax}} \frac{Q(v_j)}{N(v_j)} + 2Cp\sqrt{\frac{2\ln N(v_j)}{N(v_j)}}$$

```
DefaultPolicy(v, variant)
reward = 0
while v is non-terminal do
    v ← pick_child_by_variant(v.children)
    reward += v.score
end
return reward

UpPropagate(v, Δ)
while v is not null do
    N(v) ← N(v) + 1
    Q(v) ← Q(v) + Δ
    v ← v.parent
end
```

Algorithm 2: Auxiliary functions of the UCT Algorithm.

Default policy simulates an expected reward if the node is non-terminal. A terminal node is the case when the time budget is exhausted and no more POIs can be visited. In the simplest case, the algorithm picks uniformly random unseen reachable nodes until a terminal node is found. The expected reward is the average score of the nodes visited during the simulation. When a node is terminal, its expected reward is the score of the node.

UpPropagate updates the expected reward of node v_j's parents up to the root node. Each parent node keeps track of how many simulations has been run from their children. The expected reward of a parent is the sum of its children simulations divided by the number of visited nodes in the simulations.

BestChild returns the best child by following the Upper Confidence Bound. The Cp parameter controls the exploration-exploitation trade-off. For expected rewards between $[0, 1]$ a value of $\frac{1}{\sqrt{2}}$ for Cp is recommended [Kocsis et al., 2006].

To generates top-k itineraries, the UCT function is called k times. At the end of each run the best itinerary is returned and the selected nodes are removed from the candidate set.

We experiment with three variations of the node reward x_j. Let $\hat{r_j}$ represent the recommendation score, f_{c_j} the number of occurrences of the venue category, $popularity(v_j, t_j)$ the proportion of check-ins observed in p_j at time of arrival t_j, $ecp(c_j|c_i)$ the empirical conditional probability of visiting a venue of type c_j after a venue of type c_i (e.g., going to a coffee shop after a park), and f_{c_j} the number of occurrences of the venue category. Also, let a parameter $\lambda \geq 1$ define a penalization value for repeating venue categories in the itineraries. Larger values of λ have a higher penalty for picking venues of the same category in the itinerary. For example, a value of $\lambda = 2$ would split the node's reward in half if there exists a venue of the same category in the itinerary. The three proposed variations of the node's reward are MCTS_Simple, MCTS_Score, and MCTS_Context.

MCTS_Simple selects uniformly random the POIs. The reward of node v_j is defined in Equation 2.5 as

$$x_j = \frac{\hat{r_j}}{f_{c_j} \cdot \lambda + 1}. \tag{2.5}$$

MCTS_Score replaces the uniformly random sampling with a greedy search using the score of the recommender system. The reward of node v_j is the same definition as in Equation 2.5.

MCTS_Context also replaces the uniformly random sampling with a greedy search using the score of the recommender system and contextual information. The reward of a node v_j is defined in Equation 2.6 as

$$x_j = \frac{\hat{r_j} \cdot popularity(p_j, t_j) \cdot ecp(c_j|c_i)}{f_{c_j} \cdot \lambda + 1}. \tag{2.6}$$

Greedy Algorithm

A basic approach to solve the itinerary construction problem is to use a greedy approach. Just like the MCTS, our greedy algorithm also utilizes the recommender system to score the POIs. The algorithm starts from a POI and asks for top-k candidates from the recommender system, then filters out the ones that are already included or violate some constraint, and chooses the one that gives the highest score according to the recommender system. The algorithm continues with a next POI selection until the budget is consumed.

2.2.10 Evaluation Methodology

The data set is divided by time into three data sets for each of the cities presented in Table 2.5. The training data set contains 80% of the data. From the training set we withhold 5% as validation set. We test over the remaining 20% of the data. An example of the data split is shown in Figure 2.19.

We detect itineraries by looking for check-in sequences of the same group on the same day. These sequences are used as ground truth to evaluate the POI recommendation and the itineraries generated by the proposed algorithms. Figure 2.19 shows the sequence length distribution in the testing set. Note that most of the sequences have only two POIs increasing the difficulty for measuring the quality of the itineraries, as for those cases the task is to recommend the next POI that the group could visit.

We used nDCG (see Section 1.27) as a metric to evaluate the quality of the recommender systems. The itinerary scheduler is evaluated on how well the constrains are satisfied while maximizing the predicted preference of the group for the candidate POI.

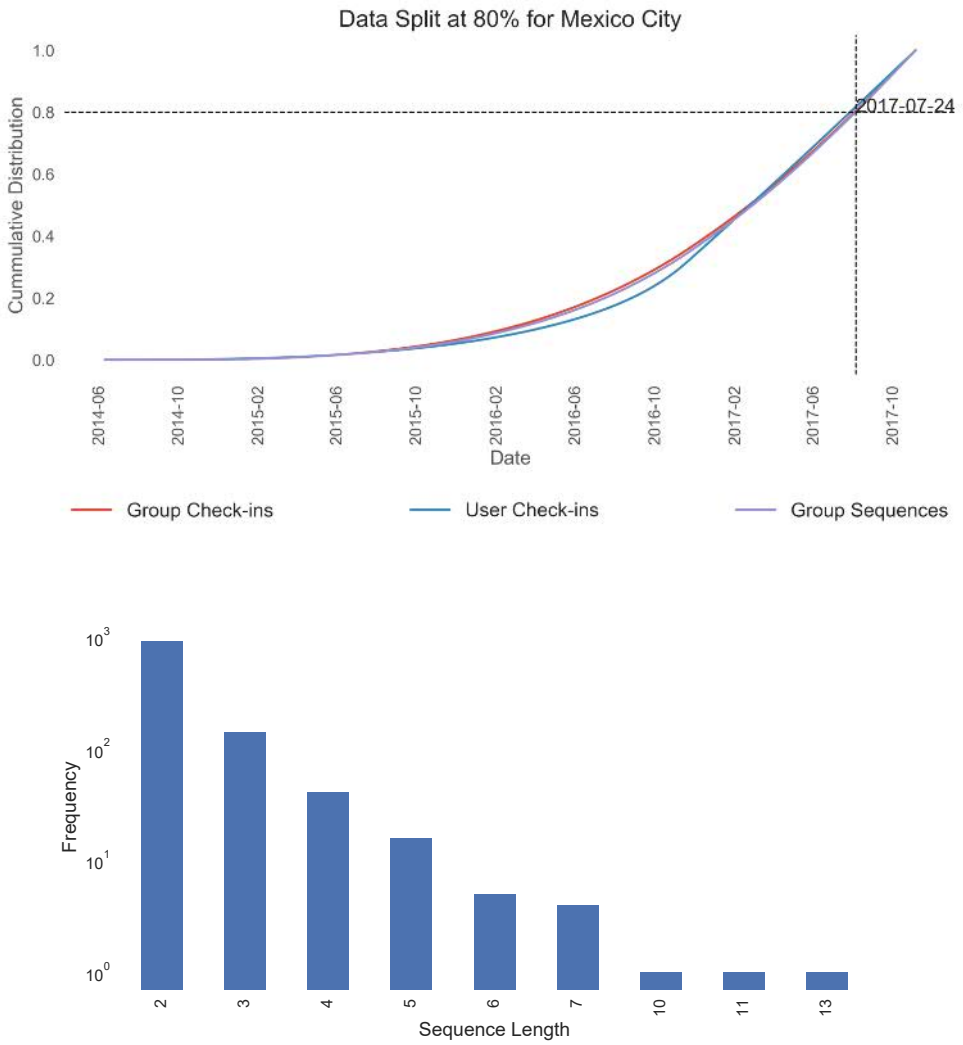

Figure 2.19: An example of the data set split by time for Mexico City (top). The length distribution of the itineraries in the testing set for all the cities (bottom).

2.2.11 Experimental Results

2.2.12 Recommender Systems Evaluation Results

We used the Apple Turi Create[10] recommender module for the implementation of recommender systems. We fine-tuned the parameters of the recommender systems using grid search on the validation set. The code for generating the candidate sets, training, and predictions, was executed on a single machine with 16 GB of RAM, 16 cores.

Table 2.6 shows the results of the different recommender systems. We observe that recommender systems trained with group history perform better than aggregating individual recommendations. However, aggregating individual recommendations could be useful for cold start groups (i.e., groups without check-in history). For that purpose, *summation* of individual recommendation scores provide the highest performance. We noticed that predicting the next POI using a fixed category type (i.e., using next category) improves the quality of the recommender systems considerably. In our experiments iALS gave the highest performance for three of the four cities. The item-to-item recommender system lead to a better performance in the city of Tokyo. This could be due to the characteristics of the cities and the data sets.

2.2.13 Itinerary Construction Evaluation Results

We evaluate our algorithms by requesting the top-10 itinerary recommendations for each of the group tours in the testing set. For each city we used the scores of the best single POI recommender in Table 2.6 as input. For all the itineraries we used a budget of 8 hours, and assumed that the group walks to commute between the POIs at a speed of 1.4 m/s. We took the first POI as the first check-in of the group sequence in our testing set, the starting POI is excluded from the evaluations. We experimented with λ values of 1, 2, 3, and 4. The best λ values for each city are 4 for Istanbul, 3 for Izmir, 3 for Mexico City, and 2 for Tokyo.

Figure 2.20 presents an example of the top-10 itineraries generated for a group in Istanbul.

The results of comparing the greedy baseline with the MCTS models are presented in Table 2.7 for average precision and Table 2.8 for average recall. The MCTS methods outperform the greedy baseline in terms of precision, and had similar performance in terms of recall.

[10] `https://github.com/apple/turicreate`

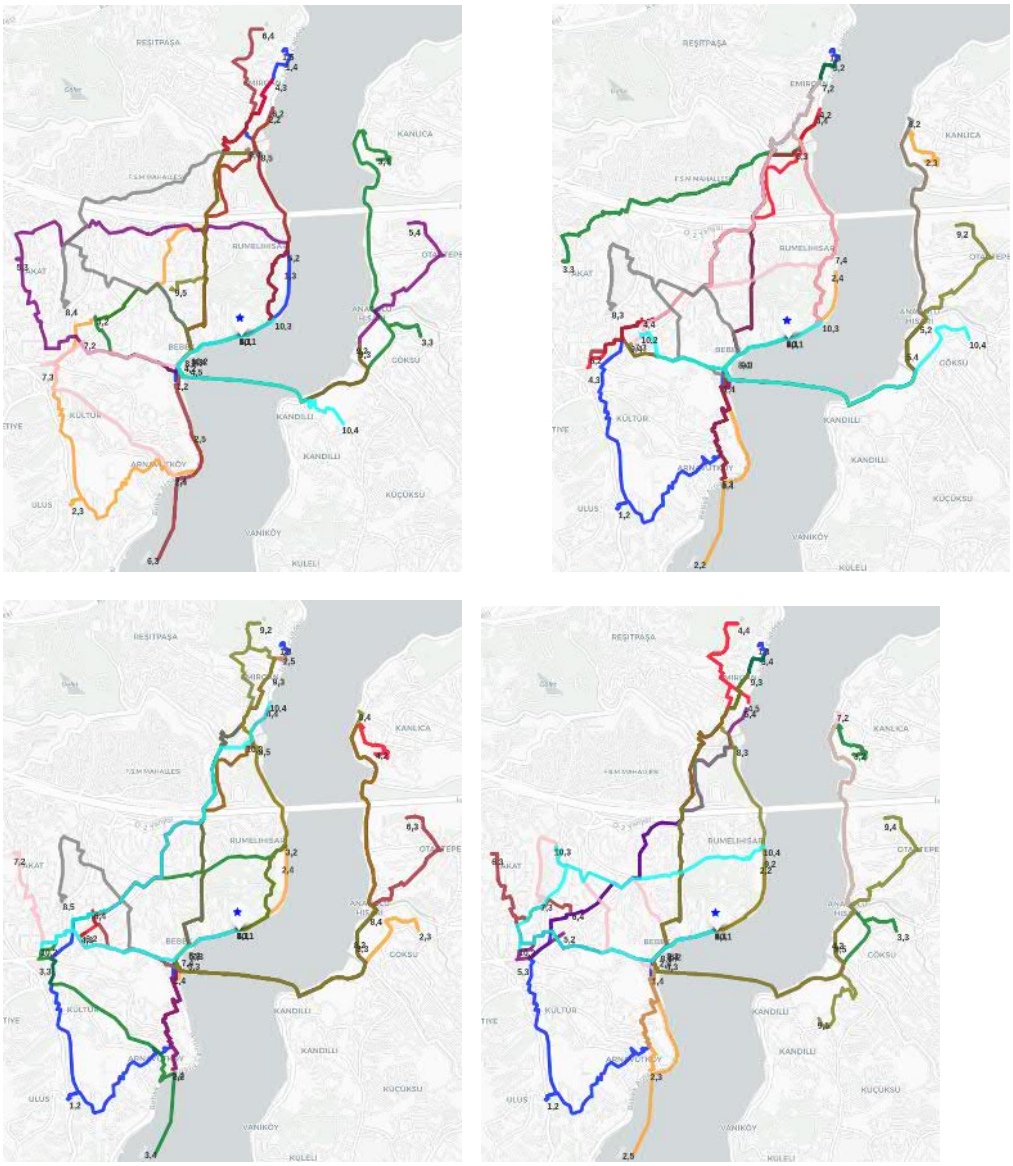

Figure 2.20: Examples of itineraries generated by different algorithms for a group in Istanbul. From top to bottom and left to right: Greedy, MCTS_Simple, MCTS_Context, and MCTS_Score. Each line in the map represents one of the top-10 recommended itineraries.

	Istanbul		Izmir		Mexico City		Tokyo	
Using Next Category	w/o	w/	w/o	w/	w/o	w/	w/o	w/
iALS - Group	**0.546**	**0.700**	**0.550**	**0.705**	**0.531**	**0.716**	0.460	0.666
iALS - IndividualSum	0.395	0.567	0.454	0.643	0.508	0.692	0.455	0.695
Item-to-Item - IndividualSum	0.364	0.532	0.344	0.531	0.418	0.615	0.428	0.655
Item-to-Item - Group	0.502	0.650	0.271	0.441	0.204	0.395	**0.500**	**0.753**
Content - Group	0.269	0.434	0.271	0.441	0.204	0.396	0.282	0.494
Content - IndividualFair	0.246	0.414	0.261	0.430	0.186	0.384	0.145	0.350
Content - IndividualMin	0.246	0.414	0.261	0.430	0.186	0.384	0.145	0.350
Content - IndividualSum	0.246	0.414	0.261	0.430	0.186	0.384	0.145	0.350
Item-to-Item - IndividualMin	0.175	0.321	0.229	0.390	0.212	0.387	0.146	0.363
Item-to-Item - IndividualFair	0.152	0.292	0.132	0.272	0.183	0.338	0.169	0.378
iALS - IndividualMin	0.057	0.206	0.088	0.242	0.134	0.293	0.239	0.409
iALS - IndividualFair	0.034	0.163	0.060	0.163	0.075	0.215	0.112	0.326

Table 2.6: Results of evaluating the *top-100* recommendations of the next POI to visit. The scores are the mean nDCG@100. The highest values of each city are in bold. The abbreviation *w/* stands for recommendations filtered to a specific *next* POI category (i.e., top-100 nearest POI of the given category), and *w/o* when the next category is not specified.

	Greedy	MCTS_Context	MCTS_Score	MCTS_Simple
Istanbul	0.008	**0.009**	**0.009**	**0.009**
Izmir	0.005	**0.007**	**0.007**	**0.007**
Mexico City	0.024	0.024	0.023	**0.026**
Tokyo	0.044	**0.054**	0.048	0.048

Table 2.7: Mean precision of the *top-10* itineraries recommendation. The highest values of each city are in bold.

The *MCTS_Simple*, *MCTS_Score*, and *MCTS_Context* variants performed similarly. The benefits of adding contextual information were not observed in both of the performance metrics. Itineraries built with contextual information had higher precision, and itineraries built with the recommender score and category penalization had higher recall. This could be because of the data sparsity in the datasets. For instance, in Mexico City the check-in sequences were shorter than in Tokyo. In shorter itineraries (e.g., two POIs) the contextual information does not add value, as a simple reward function is sufficient.

	Greedy	MCTS_Context	MCTS_Score	MCTS_Simple
Istanbul	**0.011**	**0.011**	**0.011**	0.010
Izmir	**0.013**	**0.013**	0.012	0.012
Mexico City	0.024	0.023	0.023	**0.025**
Tokyo	**0.042**	0.041	0.038	0.037

Table 2.8: Mean recall of the *top-10* itineraries recommendation. The highest values of each city are in bold.

Chapter 3

Global Citation Recommendation

The number of scientific articles published every year has exhibited an exponential increase and it is likely that this trend will continue [Mabe, 2003, Larsen and Von Ins, 2010]. The vastness of information poses a challenge to researchers as it increases the time that they must spend on finding relevant papers to their specific research projects. This is particularly true in areas of study where they have little knowledge of the related work. Scholarly search engines, reference management tools, and academic social networks provide recommendations for scientific publications that might be of their interest.

In this chapter we summarize the research presented in [Ayala-Gómez et al., 2018] where we consider the problem of recommending a list of relevant papers given an abstract as an input. Our work seeks to answer the research questions **RQ2.1**, **RQ2.2**, and **RQ2.3**. This chapter is organized as follows: introduction to the problem, setting, working data set, proposed approach, evaluation methodology, experimental results, and conclusions.

My contributions to our research in [Ayala-Gómez et al., 2018] was in the problem formulation, data collection, data exploration, proposed approach, recommender systems implementation, and experimentation.

3.1 Global Citation Recommendation

Citation recommendation is the task of recommending citations given a corpus. [He et al., 2010] identified two types of research paper recommendations: *Global Recommendations* and *Local Recommendations*. In the global citation recommendation task the goal is to find relevant articles to a given corpus as a whole. In the local citation recommendation task the goal is

to find related papers in the literature for *local contexts* (e.g., paragraphs, sentences).

The state-of-the-art research on global citation recommendation focuses on different settings, including cases where rich user metadata is available (e.g., user profile, publications, citations, reads). Modeling recommender systems using profiles requires users to create a profile, associate it with the published articles, and extract the citations from the papers. This is troublesome, as the whole corpus is often not publicly available and may be difficult to parse (e.g., PDF). Furthermore, previous publications might be irrelevant to a current topic of research, for example, when a researcher works on a new topic, or collaborates with other researchers, or has no history at all. We can avoid these problems by ignoring the user profile, and instead, asking the researchers for an abstract of their new paper that we consider as the description of the information need.

This section describes methods for expanding semantic features from the abstract of a scientific article to train a top-k recommender system for publications. We investigate a variant of the global recommendations task: *finding papers that are relevant to a short description (abstract) of a new paper, provided as input.* By solving this task we help researchers who work in a new area with a certain idea about the problem and approach, but have limited knowledge of related literature.

An essential contribution of our proposed approach is to expand the semantic features of the given abstract using knowledge graphs (Section 1.5). We show that by using the semantic features we improve the quality of the candidate set generation and the top-10 recommendation measured by nDCG@10. New elements of our method include (1) expanding the semantic features with knowledge graphs by using the Lucene search engine for candidate generation, and (2) using learning-to-rank over the abstract-candidate features.

3.2 Literature Review

A recent survey of citation recommendation is presented in [Beel et al., 2016]. 70% of the surveyed methods used *tf-idf* [He et al., 2010, Huang et al., 2014, Gollub et al., 2013]. Enriching the content of the paper is done in [Middleton et al., 2001, Paraschiv et al., 2016, Beel et al., 2013b]. Graph based approaches compute research paper rankings to find the most central or popular papers using a citation or author graph [Baez et al., 2011, Hess, 2006, Liang et al., 2011, Arnold and Cohen, 2009, Jack, 2012, Zhou et al., 2008, He et al., 2010, Strohman et al., 2007, Bethard and

Jurafsky, 2010]. Current research on using knowledge graphs to expand the semantic features of the research papers is scarce. Authors in [Wu et al., 2016] research the extraction of semantic entities from the text using *Wikifier* [Cheng and Roth, 2013, Ratinov et al., 2011] but do not study the global citation recommendation task. The authors of [Alzoghbi et al., 2016] propose a model that computes pairwise features between users and papers, and then train a learning to rank model. This model learns the user preference over the candidate papers.

Our work is different from the previously mentioned results in several aspects. We use the abstract to specify the information need and exploit the semantic added value of knowledge graphs for building global citation recommendations. We propose an approach that uses knowledge graph features in the following steps. First, we use *DBpedia Spotlight* to map the terms in the text to DBpedia entities URIs, and we extract the entities and properties. Second, we propose generating a candidate set using *Lucene's More Like This* method. This method queries for relevant papers using the DBpedia entities and terms in the abstract. Third, from a ground-truth data set, we compute abstract-candidate pairwise features, and mark the pairs where the candidate was cited by the paper as relevant. We use LambdaMART (Section 1.2.2), a learning to rank model that uses the pairwise features, to predict the labeled relevance. We show that our proposed approach improves the quality of the candidate set, measured by recall, and the *top-10 recommendations*, measured by nDCG@10.

3.3 Notations

We define the global citation recommendation task as a learning to rank task. Let X be the training set of triplets in our ground-truth data set defined as:

$$X = \bigcup_{i \in I} \{< a_i, c, r(c|a_i) >\}, \tag{3.1}$$

where a_i is the abstract of paper i in our ground-truth data set I for which we know its citations C_i, c is a candidate paper from the set of candidates CS_i that could be recommended, and $r(c|a_i)$ is the relevance score, defined as:

$$r(c|a_i) = \begin{cases} 1 & \text{if } c \in C_i \\ 0 & \text{otherwise.} \end{cases} \tag{3.2}$$

The goal is to learn a scoring function f that approximates the relevance $\hat{r}(c|a_i) = f(X)$ for $c \in CS_i$. The resulting $\hat{r}$ is used to sort the elements in CS_i and build top-k recommendations for a_i.

3.4 Working Data set

We compile a ground-truth data set of papers that is used to train and evaluate our global citation (or, in other words, reference) recommender.

We combined data from two large publicly available data sets. The first is CiteSeerX [Li et al., 2006], an open-access repository of about 10 M papers from which we use the title and abstract. The second data set is the MAG [Sinha et al., 2015], a publicly available set of 127 M scientific papers, from which we use the metadata, and the list of citations. We match the papers in the two data sets by the edit distance of their lower-case title. By merging the two data sets we obtained a collection of approximately 2.2 M research papers, 6.7 M authors, and 24 M citations. We only consider the 4.7 M citations to papers in our data set. We include a cleaning phase to remove outliers (e.g., papers with more than 2.9K authors, 9.6K citations). The cleaning removes papers published before 1973, papers with more than 20 authors and 100 references (i.e., citing more than 100 papers), and entities with a frequency below the median are ignored. After cleaning, the total number of papers is 385 K.

We extract semantic features for each paper in the ground-truth data set by mapping the abstracts to DBpedia [Auer et al., 2007] (Section 1.5). The mapping is done using DBpedia Spotlight [Mendes et al., 2011]. The output of this process is a set of DBpedia resources in the title and abstract. We refer to them as **entities**. For each paper we also collect the union of RDF objects associated with its entities and refer to them as its **properties**. An example of the links between papers, entities, and properties is shown in Figure 3.1.

In total we extracted approximately 270 K distinct entities, which account for 25 M annotations and 1.7 M unique properties. To summarize, each paper in the resulting working data set is associated with the following features: title, abstract, set of citations, set of entities, and set of properties. Table 3.1 presents statistics of the clean data set that contains 385 K papers. Tables 3.2 and 3.3 present the Top-10 authors, entities, and properties. Figure 3.2 shows popular co-occurrences for the Top-10 entities of Table 3.2 (i.e., entities mentioned in the same abstract).

3.5 Proposed Approach

We describe the main steps of our proposed approach for building global citation recommendations. In the first step we build a candidate set by finding similar papers using the terms in the given abstract and the an-

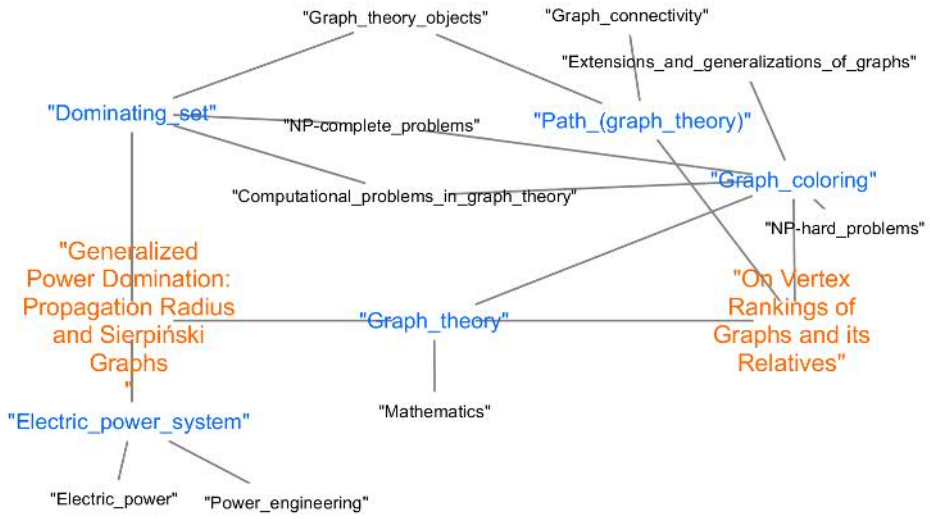

Figure 3.1: An example of two papers and their relation via the `http://purl.org/dc/terms/subject` property. The paper titles are in large orange font, the entities E_i are in blue font. Entity properties P_i are in small black font. In this example, new knowledge is discovered by the link between Dominating_Set and Path_(graph_theory).

	Authors	**Citations**	**Abstract Length**	**Entities**
Mean	3	3	693	9
SD	2	3	198	5
Min.	1	1	0	1
25%	2	1	542	6
50%	3	3	667	9
75%	4	5	832	13
Max.	19	50	3.7K	53

Table 3.1: Statistics of the cleaned data set. The abstract length is in characters.

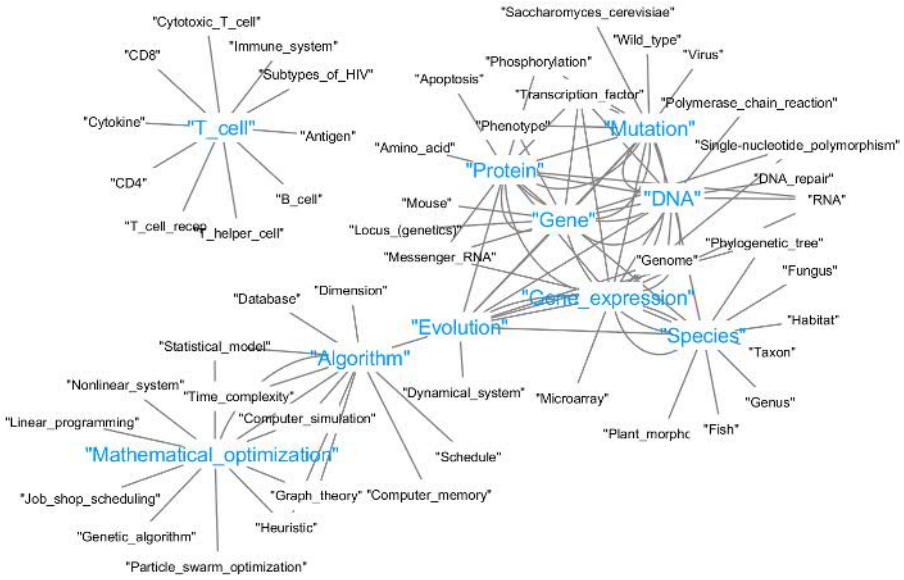

Figure 3.2: Larger text (blue) are the Top-10 entities. Smaller text (black) are entities co-occurrences (i.e., papers mentioning both entities). For simplicity, we show just 10 co-occurrences for each of the top entities.

Top-10 Authors	**Top-10 Entities**
Tania Vu	Algorithm
Jiawei Han	Protein
Nicholas G. Martin	Species
Gonzalo Navarro	Gene_expression
Lei Zhang	DNA
Hanspeter Seidel	Evolution
Terrence J. Sejnowski	Mathematical_optimization
Philip S. Yu	Graph_theory
Moshe Y. Vardi	Mutation
Christos Faloutso	Gene

Table 3.2: Most of the authors and entities are related to the fields of computer science, mathematics, and bio-informatics. Note that in this table the authors and entities are not related per row.

Top-10 Properties
http://purl.org/dc/terms/subject
http://www.w3.org/1999/02/22-rdf-syntax-ns#type
http://www.w3.org/2002/07/owl#Thing
http://dbpedia.org/class/yago/PhysicalEntity100001930
http://dbpedia.org/class/yago/Object100002684
http://dbpedia.org/class/yago/YagoLegalActorGeo
http://dbpedia.org/class/yago/YagoPermanentlyLocatedEntity
http://dbpedia.org/class/yago/Abstraction100002137
http://dbpedia.org/class/yago/Whole100003553
http://dbpedia.org/class/yago/YagoLegalActor

Table 3.3: DBpedia also links to other knowledge graphs like YAGO and other resources (e.g., purl – permanent URLs).

notated entities. For similarity search, we use *Lucene's More Like This* functionality. Figure 3.3 presents the relation between all elements in the ground-truth data set. In the second step we form the training set as defined in Equation 3.1 and compute the pairwise features given in Table 3.4. Finally, we use the training data set to train LambdaMART, a learning to rank method that learns how to score the candidates according to their relevance described in Section 1.2.2. The overview of our proposed approach is presented in Figure 3.4.

3.6 Evaluation Methodology

We split the data set based on the publication date of the papers using two different approaches. *Split random per year* uses 80% of the papers published in a year for training, and holds the remaining 20% for testing to evaluate the performance of the model to recommend citations for papers in the testing set. *Split on next year* measures how well the model predicts citations for the papers of the coming year.

We compared our proposed approach with the pairwise tf-idf score and *Lucene's More Like This* methods as baselines. The quality of the recommendations is measured using nDCG@10. The *LM-all* model had the best performance confirming that the proposed approach provides significant improvements compared to the baselines.

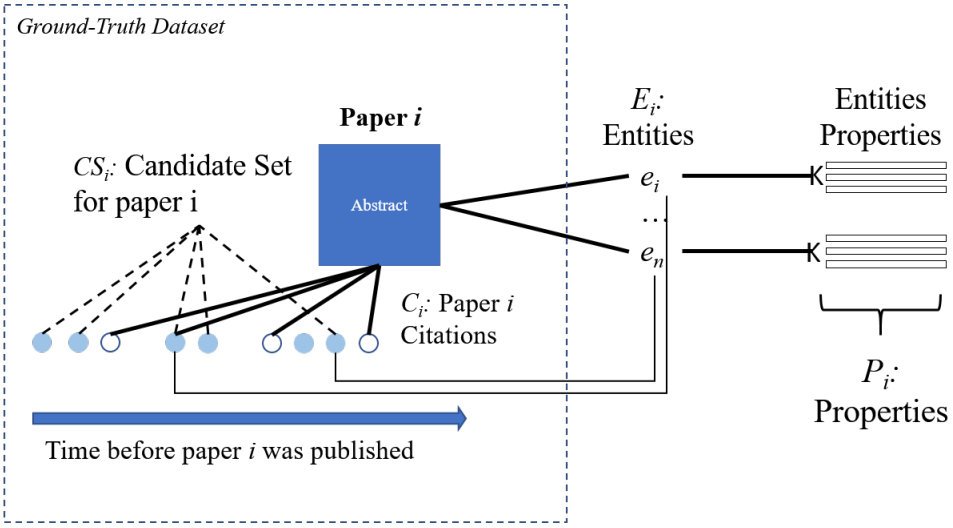

Figure 3.3: Properties of paper i. The paper is associated to DBpedia entities that allow us to connect to other papers with common entities. We use the entities in combination with the abstract's terms for querying Lucene's MLT method to generate a candidate set CS_i for paper i.

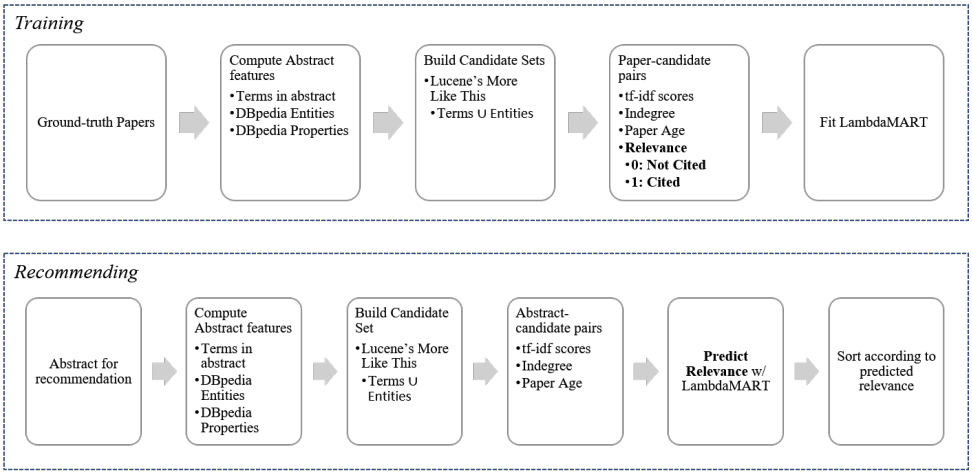

Figure 3.4: Our proposed approach for building global citation recommendations at a glance. Top: the training phase; Bottom: the recommendation phase.

Feature	Description
MLT Score	Lucene's method for scoring candidates based on the most relevant terms and entities.
Entities tf-idf	tf-idf score for the common entities.
Properties tf-idf	tf-idf score for the common properties.
Abstract tf-idf	tf-idf score for the common terms.
Combined tf-id	The sum of the entities tf-idf, properties tf-idf, and abstract tf-idf
Highest tf-idf	Label of the highest tf-idf score
Lowest idf-idf	Label of the lowest tf-idf score
Candidate Age	Years passed since the candidate paper was published. For evaluation, the year used for the candidate age is the split year. For recommendation, it would be the present year.
Indegree	Total number of papers citing the candidate, before the year of training or recommendation.
Indegree w/ Decay	The indegree with a decay according to the candidate age as: $indegree_{decay}(c\|a_i) = \frac{indegree_{c\|a_i}}{\log(age_{c\|a_i})}$.

Table 3.4: Pairwise features between the abstract and the candidates.

Model	Acronym	Features
Lucene's More Like This	MLT	Relevant terms in abstract ∪ entities
Abstract tf-idf Score	ATF	Sum of common tf-idf score
LambaMART No Entities/Properties	LM-w/o	Candidate age, indegree w/ decay, indegree, abstract tf-idf
LambdaMART All	LM-all	All the features described in Table 3.4

Table 3.5: The *LambdaMART* models represents our proposed approach. The score generated by *ATF* and *MLT* is used to build the top-k recommendations.

Features	Recall@10	Recall@20	Recall@50	Recall@100	Recall@500	Recall@1000
Abstract	0.37	0.41	0.47	0.52	0.62	0.66
Entities	0.37	0.40	0.46	0.50	0.59	0.62
Abstract ∪ Entities	**0.39**	**0.43**	**0.49**	**0.54**	**0.64**	**0.67**

Table 3.6: Features used for building the candidate sets.

3.6.1 Empirical Results

Table 3.6 presents the recall of the candidate set approach at different set sizes. We observed an increase in coverage when using the terms in the abstracts and the annotated entities.

Table 3.7 presents the results for the evaluation on *random split per year* and Table 3.8 shows the results for evaluation *on the next year*. In both cases we found that *LM-all* had the best performance. This shows that expanding the semantic features of the given abstract using the entities and properties contributes to a better recommendation.

The feature importance for the models is presented in Figure 3.5 measured by how often the features were used in the MART trees. The *Indegree w/ Decay* had the highest importance, meaning that popular papers are more relevant than unpopular ones. *MLT Score* appeared with high relevance, showing that similar papers are relevant. The *candidate's age* importance indicates that the recency of the papers affects its relevance. The *entities tf-idf* appeared as more relevant than the *abstract tf-idf*, validating that the expanded semantic features added predictive value. Finally, the relevance of the knowledge graph is shown in the *combined tf-idf* and *properties tf-idf*.

Split Year	LM-all	MLT	ATF	LM-w/o
1996	**0.402**	0.344 (+16.9 %)	0.262 (+53.7 %)	0.292 (+37.7 %)
1997	**0.449**	0.409 (+10.0 %)	0.326 (+38.1 %)	0.307 (+46.5 %)
1998	**0.390**	0.390 (+0.0 %)	0.300 (+29.8 %)	0.291 (+34.0 %)
1999	**0.401**	0.364 (+10.2 %)	0.274 (+46.5 %)	0.292 (+37.2 %)
2000	**0.411**	0.391 (+5.0 %)	0.261 (+57.3 %)	0.282 (+45.8 %)
2001	**0.382**	0.347 (+10.2 %)	0.268 (+42.6 %)	0.289 (+32.0 %)
2002	**0.368**	0.330 (+11.5 %)	0.252 (+46.1 %)	0.284 (+29.5 %)
2003	**0.380**	0.347 (+9.5 %)	0.245 (+54.7 %)	0.274 (+38.4 %)
2004	**0.335**	0.308 (+8.8 %)	0.211 (+58.6 %)	0.236 (+42.2 %)
2005	**0.327**	0.297 (+10.1 %)	0.212 (+54.0 %)	0.250 (+30.5 %)
2006	**0.318**	0.293 (+8.6 %)	0.207 (+53.6 %)	0.235 (+35.2 %)
2007	**0.316**	0.290 (+8.9 %)	0.194 (+62.5 %)	0.226 (+40.1 %)
2008	**0.310**	0.286 (+8.6 %)	0.194 (+60.1 %)	0.223 (+39.1 %)
2009	**0.300**	0.265 (+13.2 %)	0.189 (+58.6 %)	0.228 (+31.8 %)
2010	**0.316**	0.286 (+10.6 %)	0.194 (+62.9 %)	0.229 (+38.3 %)
2011	**0.306**	0.281 (+8.9 %)	0.194 (+57.9 %)	0.218 (+40.0 %)
2012	**0.303**	0.268 (+12.9 %)	0.174 (+74.2 %)	0.211 (+43.4 %)
2013	**0.292**	0.252 (+15.8 %)	0.167 (+74.4 %)	0.207 (+41.0 %)
2014	**0.294**	0.251 (+17.2 %)	0.175 (+68.0 %)	0.215 (+36.8 %)
2015	**0.256**	0.212 (+21.0 %)	0.151 (+69.7 %)	0.193 (+33.2 %)

Table 3.7: nDCG@10 for the models evaluated on the year random split. The numbers in parenthesis indicate the gain of the *LM-all* in comparison to the baseline.

Eval. Year	LM-all	MLT	ATF	LM-w/o
1997	**0.458**	0.373 (+22.7 %)	0.287 (+59.7 %)	0.338 (+35.7 %)
1998	**0.420**	0.369 (+13.7 %)	0.260 (+61.2 %)	0.308 (+36.4 %)
1999	**0.427**	0.374 (+13.9 %)	0.280 (+52.2 %)	0.316 (+34.9 %)
2000	**0.417**	0.384 (+8.6 %)	0.271 (+53.8 %)	0.314 (+33.0 %)
2001	**0.391**	0.356 (+10.0 %)	0.263 (+48.6 %)	0.295 (+32.8 %)
2002	**0.378**	0.336 (+12.6 %)	0.253 (+49.7 %)	0.286 (+32.3 %)
2003	**0.365**	0.327 (+11.5 %)	0.238 (+53.4 %)	0.271 (+34.6 %)
2004	**0.346**	0.312 (+10.8 %)	0.222 (+56.1 %)	0.257 (+34.7 %)
2005	**0.334**	0.301 (+11.1 %)	0.218 (+53.7 %)	0.252 (+32.5 %)
2006	**0.325**	0.292 (+11.6 %)	0.209 (+55.6 %)	0.241 (+34.7 %)
2007	**0.324**	0.290 (+11.7 %)	0.202 (+60.4 %)	0.238 (+36.1 %)
2008	**0.313**	0.279 (+12.3 %)	0.196 (+59.7 %)	0.230 (+36.1 %)
2009	**0.312**	0.276 (+13.1 %)	0.192 (+62.4 %)	0.228 (+37.2 %)
2010	**0.318**	0.284 (+12.0 %)	0.198 (+60.5 %)	0.232 (+37.5 %)
2011	**0.306**	0.271 (+12.7 %)	0.188 (+62.3 %)	0.221 (+38.2 %)
2012	**0.300**	0.263 (+14.0 %)	0.177 (+69.4 %)	0.215 (+39.8 %)
2013	**0.296**	0.255 (+16.2 %)	0.176 (+68.5 %)	0.214 (+38.1 %)
2014	**0.295**	0.259 (+14.0 %)	0.176 (+67.7 %)	0.215 (+37.1 %)
2015	**0.281**	0.235 (+19.8 %)	0.168 (+67.2 %)	0.212 (+32.6 %)

Table 3.8: nDCG@10 for the models evaluated on the following year of the training. The numbers in parenthesis indicate the gain of the *LM-all* in comparison to the baseline.

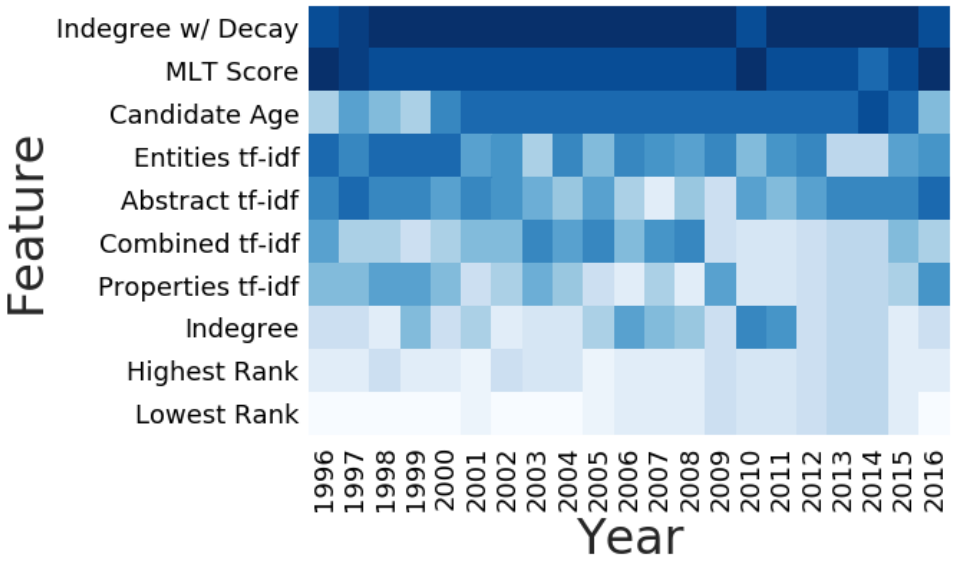

Figure 3.5: The feature importance is measured by how many times each feature was used by LambdaMART in the trees' splits. The plot is sorted from top to bottom. Darker color means a greater contribution to the model.

Chapter 4

Session Based Item-to-Item Recommendation

In previous chapters we showed how Web recommender systems assist users to navigate through possible relevant items. Most of the recommender systems depend on a long history of interactions among users. However, in practice, extensive user history is not always available; for example, when the user browses anonymously without signing in. Moreover, some items (e.g., news, viral content, expensive electronics, and movies) might be relevant only during the current user session. Recommender systems rely on the items recently viewed for such cases. This task is called *session based item-to-item recommendation*.

Generating item-to-item recommendations by "people who viewed this, also viewed" (1.1.4) works well for popular items. However, these methods perform poorly for rare (i.e., infrequent) items with short transaction history.

In this chapter we summarize the research presented in [Daróczy et al., 2018] where we consider the problem of recommending a list of relevant *next* items for a given item in a user session. Our research contributes to answering the questions **RQ3.1**, **RQ3.2**, and **RQ3.3**. This chapter is organized as follows: introduction to the problem, data set, proposed approach, evaluation methodology, and experimental results.

My contribution to our work in [Daróczy et al., 2018] was on the data extraction, data exploration, recommender systems implementation, and experimentation.

4.1 Infrequent Item-to-item Recommendation via Invariant Random Fields

In Section 1.1.4 we presented collaborative filtering methods, and described how to build such models based on user-item interactions. Content based recommender systems (1.1.3) use item properties to build recommendations. Several practitioners [Koenigstein and Koren, 2013, Pilászy et al., 2015] argue that in real-life, most of the recommendations are built using implicit feedback, with short user history, and most are item-to-item based. Our proposed model focuses on solving the task of recommending a list of relevant items to a user based on the items consumed in the current session [Sarwar et al., 2001b, Linden et al., 2003]. An intuitive approach is to count the item pair frequencies that accounts for the times a user has moved from a current item i to a next item i. This works for popular transitions, however, for infrequent items require breaking ties (transitions with the same frequency), and also use a similarity metric to balance the item low support. Moreover, we have to consider that new items will appear (i.e., the cold start case [Schein et al., 2002]).

The main idea of our approach is to utilize the known, recent, or popular items for item-to-item recommendation via multiple representations. As a starting point we compute the item transition pairs. Then, we utilize the entire training data, and not just the item-item conditional probabilities. Our item-to-item model is able to use single or combined similarity measures such as Jaccard or cosine based on collaborative, content, multimedia and metadata information. We evaluate our approach by generating top-k recommendations for a current item, and measure the Recall and DCG on the next item.

Our item-to-item recommender model uses a set of arbitrary item pair similarity measures (e.g., content-based similarity). We use the pairwise similarity values and potentially other model parameters θ to model the item i as a random variable $p(i|\theta)$. From $p(i|\theta)$, we will infer the distance and the conditional probability of pairs of items i and j by using all information in θ.

4.2 Literature Review

Recommender systems are surveyed in [Ricci et al., 2015]. The first item-to-item recommender methods used a similarity metric to find the nearest neighbors or association rules [Sarwar et al., 2001b, Linden et al., 2003, Desrosiers and Karypis, 2011, Davidson et al., 2010]. Both classes of

Data set	25%	50%	75%	Max
Books	1	1	2	1,931
Yahoo! Music	4	9	23	160,514
MovieLens	29	107	300	2,941
Netflix	56	217	1,241	144,817

Table 4.1: Co-occurrence quartiles. We remove pairs above the 75% of the co-occurrence frequency distribution.

methods perform poorly if the last item of the session has a low item transition support (e.g., rare or recent items). [Hidasi and Tikk, 2012] consider transitions (sequentially) as a context, and their approach uses pairwise associations as features in an alternating least squares model. They mention that they face the sparsity problem in setting minimum support, confidence, and lift of the associations, and they used the category of last purchased item as a fallback. Closest to our work is the EIR model [Koenigstein and Koren, 2013] that embeds the item-item transitions in item latent factors used to build recommendations by finding the nearest latent factors for a given item latent factor. Recently, recurrent neural networks [Cho et al., 2014] to build item-to-item models [Quadrana et al., 2017].

4.3 Data sets used in the experiments

We used the following openly available data sets in our experiments: Netflix [Bennett et al., 2007b], MovieLens [Harper and Konstan, 2016], Ziegler's Books [Ziegler et al., 2005], and Yahoo! Music [Dror et al., 2012].

Similar to [Koren et al., 2009] we simulate calculate item transitions from pairs of items consumed by the users in the data sets. For example, if a user consumed items a, b and c we create three pairs: $[(a, b), (b, c), (c, a)]$. We do this for all the users by considering that the items were consumed in one single session, and then we calculate the frequency of each pair. Figure 4.1 and Table 4.1 shows that most of the co-occurrence in the data sets are infrequent. 75% of the pairs have low item support. This observation shows how important is to be able to build recommendations based on infrequent items. We focus only on infrequent items by filtering out the items with high support (i.e., co-occurrence pairs with a frequency above the 75% of the distribution).

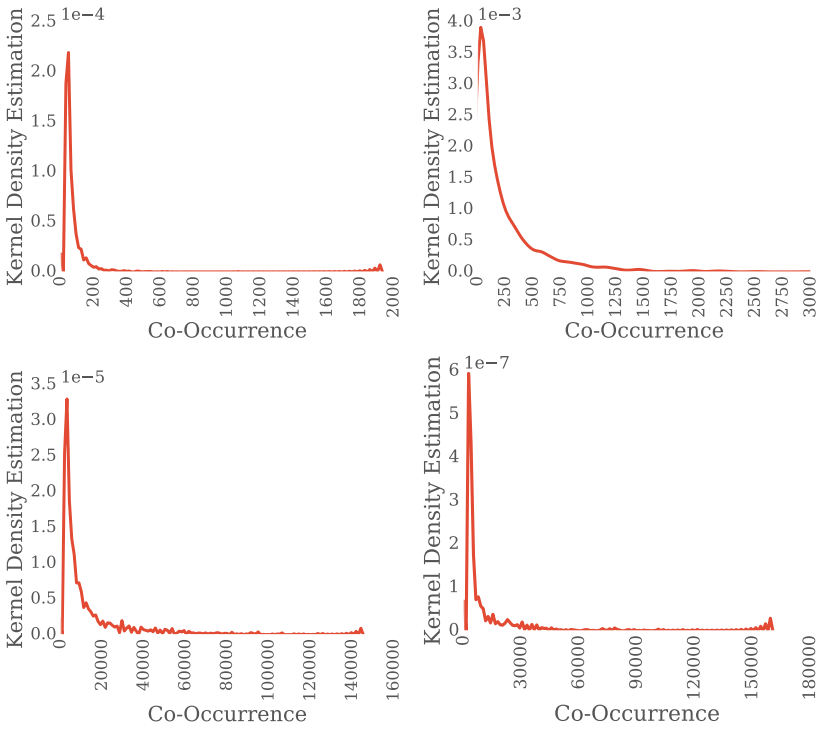

Figure 4.1: The KDE for the item co-occurrence count in the data sets. From left to right and top to bottom: Books, MovieLens, Netflix, Yahoo! Music.

We observed that the KDE saturates in items that are infrequent. That is, most of transition pairs occure rarely in the data sets.

Symbol	Explanation
i	The current item in the user session
j	The next item that followed i in the session
f_*	The frequency in the data set (e.g., f_i is the frequency of item i)
$dist(*,*)$	The distance (similarity) between a pair of items (e.g., $dist(i,j)$ is the distance between items i and j)
θ	The model parameters
$S = \{i_1, i_2, \ldots, i_N\}$	A sample of items
U	Potential function of the similarity graph
$Z(\theta)$	Partition function of U
G_i	The Fisher score of item i
$F(\theta)$	The Fisher information matrix
E	The Expected value
$K(i,j)$	The Fisher Kernel
$\text{dist}_F(i,j)$	The Fisher Distance

Table 4.2: Relevant definitions for our proposed approach for item-to-item recommendations.

4.4 The Proposed Approach

To explain our model, first we explain the concept of similarity graph. Then, we use Fisher Information for computing item similarity.

Table 4.2 contains the definitions used in our proposed approach.

Similarity Graph

We define a similarity graph as a weighted graph. For item i we compute the distance between i to other items $i_n \in S$ and use this similarity as the weight. We consider item i as a random variable generated by a Markov Random Field. Lets assume that we are given a Markov Random Field generative model for $p(i|\theta)$. The Hammersley-Clifford theorem [Hammersley and Clifford, 1971] states that the distribution of $p(i|\theta)$ is a Gibbs distribution, which can be expressed by a potential function U over the maximal cliques as:

$$p(i \mid \theta) = \mathrm{e}^{-U(i|\theta)}/Z(\theta), \tag{4.1}$$

where $U(i \mid \theta)$ is the energy function and

$$Z(\theta) = \sum_i \mathrm{e}^{-U(i|\theta)} \tag{4.2}$$

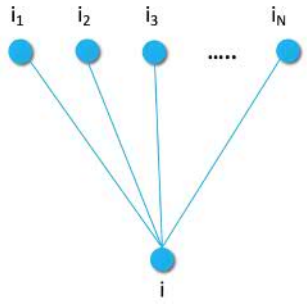

Figure 4.2: Similarity graph of item i with sample items $S = \{i_1, i_2, ..., i_N\}$ of distances $\text{dist}(i, i_n)$ from i.

is the sum of the exponent of the energy function over our generative model.

Figure 4.2 defines the simplest similarity graph and this can be described using the energy function (4.1) as:

$$U(i \mid \theta = \{\alpha_1, .., \alpha_N\}) := \sum_{n=1}^{N} \alpha_n \text{dist}(i, i_n), \tag{4.3}$$

where the hyper-parameter set θ is the weight of the elements in the sample set.

We can add the next item j to the similarity graph as shown in Figure 4.3. In the pairwise similarity graph the maximal clique size increases to three. To capture the joint energy with parameters $\theta = \{\beta_n\}$ we use a heuristic approximation similar to the pseudo-likelihood method [Besag, 1975]. We approximate the joint distribution of each size three clique as the sum of the individual edges as:

$$U(i, j \mid \theta) := \sum_{n=1}^{N} \beta_n (\text{dist}(i, i_n) + \text{dist}(j, i_n) + \text{dist}(i, j)). \tag{4.4}$$

Similarity Measures

There are many distances and divergence measures that can be used in the energy function (4.4). These measures yield the natural baseline methods for item-to-item recommendation. Table 4.3 presents the methods used as similarity for item transitions. Additionally, one can measure the similarity of the items based on their content (e.g., metadata, text, title) using tf-idf or Jensen-Shannon divergence [Fuglede and Topsoe, 2004].

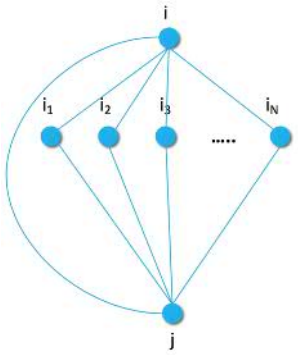

Figure 4.3: Pairwise similarity graph with sample set $S = \{i_1, i_2, ..., i_N\}$ for a pair of items i and j.

Metric	Definition
Cosine similarity (Cos)	$cos(i,j) = \frac{f_{ij}}{\sqrt{f_i f_j}}$
Jaccard similarity (JC)	$JC(i,j) = \frac{f_{ij}}{f_i + f_j - f_{ij}}$
ECP	$ECP(j\|i) = \frac{f_{ij}}{f_i + 1}$, where one is a smoothing parameter

Table 4.3: Distances and divergences used in the similarity graph.

Fisher Information

We are using a similarity graph to encapsulate the relation between the current item in the user's session, a candidate item, and a sample set. However, we need to define a metric that allows us to encapsule how relevant the whole similarity graph is. This metric will help us to rank the candidate items and build the recommendation. We use the Fisher information to measure the amount of information that an observed random variable X carries about an unknown parameter θ of a distribution that models X. The Fisher information is the foundation of our proposed approach for recommending items. In this section, we briefly introduce the Fisher information. More details about theory and applications of the Fisher Information in machine learning can be found in [Daróczy,].

The general parametric class of probability models $p(i|\theta)$, where $\theta \in \Theta \subseteq \mathbb{R}^\ell$, can be viewed as a (statistical) manifold M_Θ, provided that the dependence of the potential on Θ is sufficiently smooth. By [Jost and Jost, 2008] M_Θ can be turned into a Riemann manifold by giving an inner product (kernel) at the tangent space of each point $p(i|\theta) \in M_\Theta$, where the inner product varies smoothly with p. The inner product allows us to define

the so-called Fisher metric on M. According to [Cencov, 2000], the Fisher metric exhibits a unique invariance property under some maps which are quite natural in the context of probability. One can use the Fisher kernel as an attempt to introduce a natural comparison of the items on the basis of the generative model [Jaakkola and Haussler, 1999].

We start defining the Fisher kernel over the manifold M_Θ of probabilities $p(i|\theta)$ as in Equation (4.1) by considering the tangent space. The tangent vector:

$$G_i = \nabla_\theta \log p(i|\theta) = \left(\frac{\partial}{\partial \theta_1} \log p(i|\theta), \ldots, \frac{\partial}{\partial \theta_l} \log p(i|\theta) \right) \tag{4.5}$$

is called the *Fisher score* of item i. The *Fisher information matrix* is a positive semi-definite matrix defined as:

$$F(\theta) := \mathbf{E}_\theta (\nabla_\theta \log p(i|\theta) \nabla_\theta \log p(i|\theta)^T), \tag{4.6}$$

where the expectation is taken over $p(i|\theta)$. In particular, the nm-th entry of $F(\theta)$ is

$$F_{nm} = \sum_i p(i|\theta) \left(\frac{\partial}{\partial \theta_n} \log p(i|\theta) \right) \left(\frac{\partial}{\partial \theta_m} \log p(i|\theta) \right).$$

Thus, to capture the generative process, the gradient space of M_Θ is used to derive the Fisher vector, a mathematically grounded feature representation of item i. The corresponding kernel function:

$$K(i,j) := G_i^T F^{-1} G_j \tag{4.7}$$

is called the *Fisher kernel*. G_i gives the direction where the parameter vector θ should be changed to fit item i the best [Perronnin and Dance, 2007]. We prove a theorem for our kernels on a crucial re-parametrization invariance property that typically holds for Fisher kernels [Janke et al., 2004].

Theorem 1. *For all $\theta = \rho(\mu)$ for a continuously differentiable function ρ, K_θ is identical.*

Proof. The Fisher score is:

$$G_i(\mu) = G_i(\rho(\mu)) \left(\frac{\partial \rho}{\partial \mu} \right) \tag{4.8}$$

and therefore:

$$K_\mu(i,j) = G_i(\mu)F_\mu^{-1}G_j(\mu)$$
$$= G_i(\rho(\mu))\left(\frac{\partial\rho}{\partial\mu}\right)\left(F_{\rho(\mu)}\left(\frac{\partial\rho}{\partial\mu}\right)^2\right)^{-1}G_j(\rho(\mu))\left(\frac{\partial\rho}{\partial\mu}\right)$$
$$= G_i(\rho(\mu))F_{\rho(\mu)}^{-1}G_j(\rho(\mu)) = K_\rho(i,j).\square$$

□

The theorem states that the kernel will not depend on the hyper-parameters θ.

Based on the similarity graphs introduced in Section 4.4 and by taking advantage of the invariance properties of the Fisher metric, we propose two ranking methods for item-item transitions that we describe in Section 4.4.1 and 4.4.2.

4.4.1 Item-Item Fisher Conditional Score (FC)

Our first item-to-item recommender uses the similarity information in the item-item transition conditional probability computation by using Fisher scores as in Equation (4.5). By the Bayes theorem:

$$G_{j|i} = \nabla_\theta \log p(j \mid i;\theta) = \nabla_\theta \log \frac{p(i,j \mid \theta)}{p(i \mid \theta)}$$
$$= \nabla_\theta \log p(i,j \mid \theta) - \nabla_\theta \log p(i \mid \theta), \tag{4.9}$$

we need to determine the joint and the marginal distributions for a particular item pair.

Let us calculate the Fisher score of (4.5) with $p(i|\theta)$ of the single item generative model defined by (4.3):

$$\begin{aligned} G_i^k(\theta) &= \nabla_{\theta_k} \log p(i|\theta) \\ &= \frac{1}{Z(\theta)} \sum_i \mathrm{e}^{-U(i|\theta)} \frac{\partial U(i \mid \theta)}{\partial \theta_k} - \frac{\partial U(i \mid \theta)}{\partial \theta_k} \\ &= \sum_i \frac{\mathrm{e}^{-U(i|\theta)}}{Z(\theta)} \frac{\partial U(i \mid \theta)}{\partial \theta_k} - \frac{\partial U(i \mid \theta)}{\partial \theta_k}. \end{aligned} \tag{4.10}$$

By (4.1), our formula can be simplified as:

$$\begin{aligned} G_i^k(\theta) &= \sum_i p(i \mid \theta) \frac{\partial U(i \mid \theta)}{\partial \theta_k} - \frac{\partial U(i \mid \theta)}{\partial \theta_k} \\ &= \mathbf{E}_\theta\left[\tfrac{\partial(U(i|\theta)}{\partial \theta_k}\right] - \tfrac{\partial U(i|\theta)}{\partial \theta_k}. \end{aligned} \tag{4.11}$$

For an energy function as in equation (4.3), the Fisher score of i has a simple form:

$$G_i^k(\theta) = \mathbf{E}_\theta[\text{dist}(i, i_k)] - \text{dist}(i, i_k), \tag{4.12}$$

and similarly for equation (4.4):

$$\begin{aligned} G_{ij}^k(\theta) = \mathbf{E}_\theta[&\text{dist}(i, i_k) + \text{dist}(j, i_k) + \text{dist}(i, j)] \\ &-(\text{dist}(i, i_k) + \text{dist}(j, i_k) + \text{dist}(i, j)). \end{aligned} \tag{4.13}$$

If we add (4.12) and (4.13) into (4.9), several terms cancel out and the Fisher score becomes:

$$G_{j|i}^k = \mathbf{E}_\theta[\text{dist}(j, i_k) + \text{dist}(i, j)] - (\text{dist}(j, i_k) + \text{dist}(i, j)). \tag{4.14}$$

The distance in Equation (4.14) are available at the moment of computation. The expected values are estimated using the training data via as the average distances from the training data.

The Fisher score resembles how well the model fits the data, thus we can recommend the best fitting next item j^* based on the norm of the Fisher score,

$$j^* = \arg\min_{j \neq i} ||G_{j|i}(\theta)||, \tag{4.15}$$

we use ℓ_2 norm in our experiments.

4.4.2 Item-Item Fisher Distance (FD)

Our second model ranks the next item by its distance from the last one, based on the Fisher metric. Using the Fisher kernel $K(i, j)$, the *Fisher distance* is formulated as:

$$\text{dist}_F(i, j) = \sqrt{K(i, i) - 2K(i, j) + K(j, j)}. \tag{4.16}$$

Thus, we need to compute the Fisher kernel over our generative model as in (4.7). The computational complexity of the Fisher information matrix estimated on the training set is $\mathcal{O}(T|\theta|^2)$, where T is the size of the training set. To reduce the complexity to $\mathcal{O}(T|\theta|)$ we can approximate the Fisher information matrix with the diagonal as suggested in [Jaakkola and Haussler, 1999, Perronnin and Dance, 2007]. Hence, we will only use the diagonal of the Fisher information matrix:

$$\begin{aligned} F_{k,k} &= \mathbf{E}_\theta[\nabla_{\theta_k} \log p(i|\theta)^T \nabla_{\theta_k} \log p(i|\theta)] \\ &= \mathbf{E}_\theta[(\mathbf{E}_\theta[\frac{\partial U(i \mid \theta)}{\partial \theta_k}] - \frac{\partial (U(i \mid \theta)}{\partial \theta_k})^2]. \end{aligned} \tag{4.17}$$

For the energy functions of equations (4.3) and (4.4), the diagonal of the Fisher kernel is the standard deviation of the distances from the sample S. We use the Fisher vector of i (4.3) as:

$$\begin{aligned}\mathcal{G}_i^k &= F^{-\frac{1}{2}} G_i^k \approx F_{kk}^{-\frac{1}{2}} G_i^k \\ &= \frac{\mathbf{E}_\theta[\text{dist}(i,i_k)] - \text{dist}(i,i_k)}{\mathbf{E}_\theta^{\frac{1}{2}}[(\mathbf{E}_\theta[\text{dist}(i,i_k)] - \text{dist}(i,i_k))^2]}.\end{aligned} \tag{4.18}$$

The final kernel function is:

$$\begin{aligned}K(i,j) &= G_i^T F^{-1} G_j \approx G_i^T F_{diag}^{-1} G_j \\ &= G_i^T F_{diag}^{-\frac{1}{2}} F_{diag}^{-\frac{1}{2}} G_j = \sum_k \mathcal{G}_i^k \mathcal{G}_j^k.\end{aligned} \tag{4.19}$$

By substituting into (4.16) the recommended next item after item i will be:

$$j^* = \arg\min_{j \neq i} \text{dist}_F(i,j). \tag{4.20}$$

4.4.3 Multimodal Fisher score and distance

We could also expand the model to handle additional distances with a simple modification to the graph of Figure 4.2. We expand the points of the original graph into new points $R_i = \{r_{i,1}, .., r_{i,|R|}\}$ corresponding to R representations for each item i_n in Figure 4.4. There will be an edge between two item representations $r_{i,\ell}$ and $r_{j,k}$ if they are the same type of representation ($\ell = k$) and the two items are connected in the original graph. This transformation does not affect the maximal clique size and therefore the energy function is a simple addition:

$$U(i \mid \theta) = \sum_{n=1}^{N} \sum_{r=1}^{|R|} \alpha_{nr} \text{dist}_r(i_r, i_{nr}). \tag{4.21}$$

If we expand the joint similarity graph to a multimodal graph the energy function is:

$$\begin{aligned}U(i,j \mid \theta) = \sum_{n=1}^{N} \sum_{r=1}^{|R|} \beta_{nr} (&\text{dist}_r(i_r, i_{nr}) \\ &+ \text{dist}_r(j_r, i_{nr}) + \text{dist}_r(i_r, j_r)).\end{aligned} \tag{4.22}$$

Let G_{ir} be the Fisher score for any distance measure $r \in R$. The multimodal graph is the concatenation of the unimodal Fisher scores as:

$$G_i^{multi} = \{G_{i1}, .., G_{i|R|}\}, \tag{4.23}$$

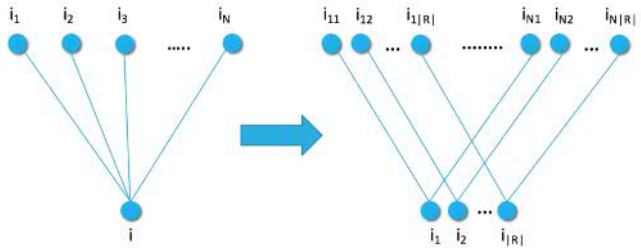

Figure 4.4: The single and multimodal similarity graph with sample set $S = \{i_1, i_2, ..., i_N\}$ and $|R|$ modalities.

Data set	Items	Users	Training pairs	Testing pairs
Netflix	17,749	478,488	7,082,109	127,756
MovieLens	3,683	6,040	670,220	15,425
Yahoo! Music	433,903	497,881	27,629,731	351,344
Books	340,536	103,723	1,017,118	37,403

Table 4.4: Data sets used in the experiments.

and therefore the norm of the multimodal Fisher score is a simple sum over the norms:

$$||G_i^{multi}|| = \sum_{r=1}^{|R|} ||G_{ir}||. \tag{4.24}$$

The calculation is similar for the Fisher kernel of equation (4.18), and the multimodal kernel can be expressed as:

$$K_{multi}(i,j) = \sum_{r=1}^{|R|} K_r(i,j). \tag{4.25}$$

4.5 Evaluation Methodology

To generate the training and testing set we place most of the users in a training set and the rest of the users in the testing set. We generate the co-occurrence pairs as described in Section 4.3. The number of training and testing pairs and the properties of the data sets can be seen in Table 4.4. During testing, the evaluation was performed over a sampled set of 200 items as in [Koenigstein and Koren, 2013] for all three metrics and solved ties arbitrary.

We performed experiments on the four data sets described in Section 4.3. As baseline methods we use the four item-item similarity measures from

Table 4.3, and we also implemented the EIR of [Koenigstein and Koren, 2013]. For content similarity we mapped the movies in the MovieLens data set to DBpedia (1.5), and compute the Jaccard similarity between two items using the nodes connected to the movies.

We conducted experiments by adding 200 sampled items to the testing item to evaluate recommendations. That is, given the current item in a session i and a known co-occurrence j we randomly add 200 items and rank them based on the score of the models. The best models should preserve j on the top of the sorted list. We use Recall and DCG to measure the quality of the recommendations.

4.6 Experimental Results

This section presents different experiments related to the size of the sample set, the modalities used (e.g. implicit, content), the performance on infrequent items, and the overall performance. FC refers to 4.4.1, and FD to Section 4.4.2. In case of multimodal the model use both content and collaborative similarity values.

Sample Set. The similarity graphs are defined via the set of items used as samples as shown in Figure 4.3. We choose the most popular items in the training set as elements for the sample set. Figures 4.5 and 4.6 show that the recommendation quality saturates at a certain sample set size. Interestingly, for *FD Books*, the recall and DCG deteriorate as the sample set increases. Possible reasons are the choice of sample set selection (i.e., most popular items), and the number of items in the sample set compared to the number of items in the data set. For the rest of the experiments, we set the size of the sample set to 20 for the remaining experiments, as the performance does not increase much after 20.

Performance of Similarity Functions. Another relevant parameter of the similarity graphs is the choice of the similarity functions. Table 4.5 presents the performance of the different similarity functions. Overall, Jaccard similarity was the best performing and we used it for the rest of our experiments.

Performance on Infrequent Items. One of the main challenges in the field of recommendation systems is the "cold start" problem, therefore we examine the performance in case of low item support. Figure 4.7 shows the advantage of the Fisher methods for infrequent items. As support increases,

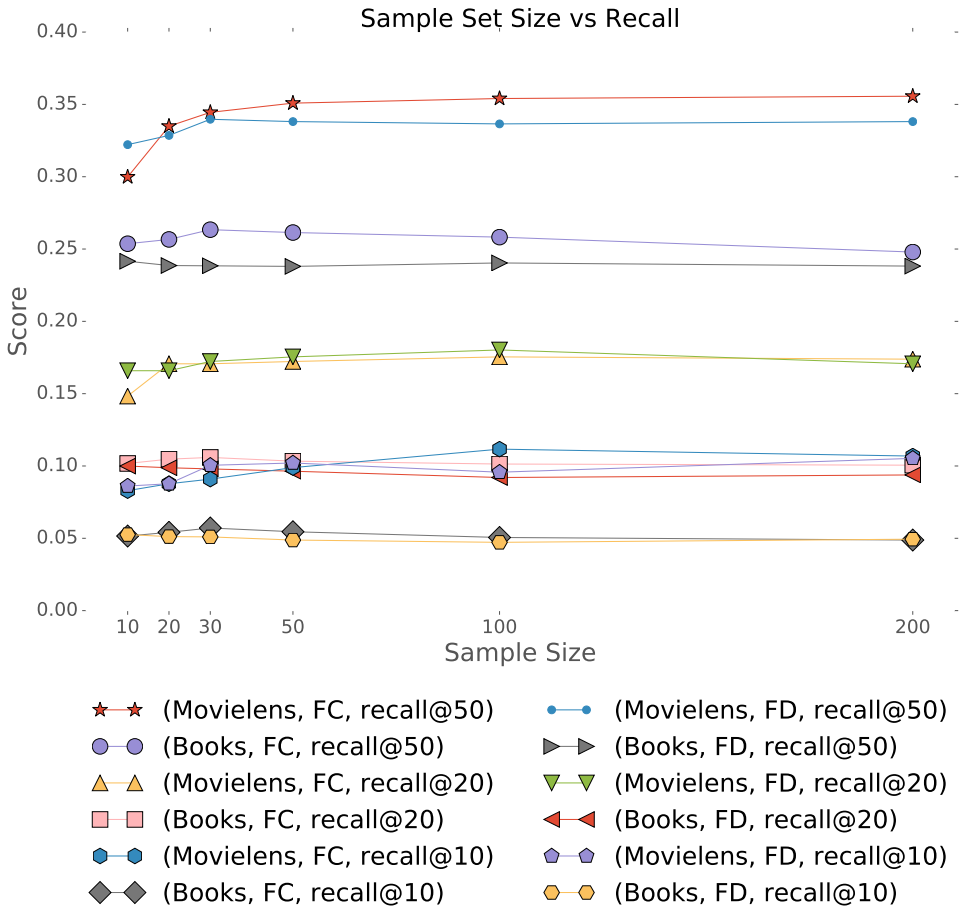

Figure 4.5: For most of the models, the sample set size improves Recall until it saturates at 50. In this example we use Jaccard similarity.

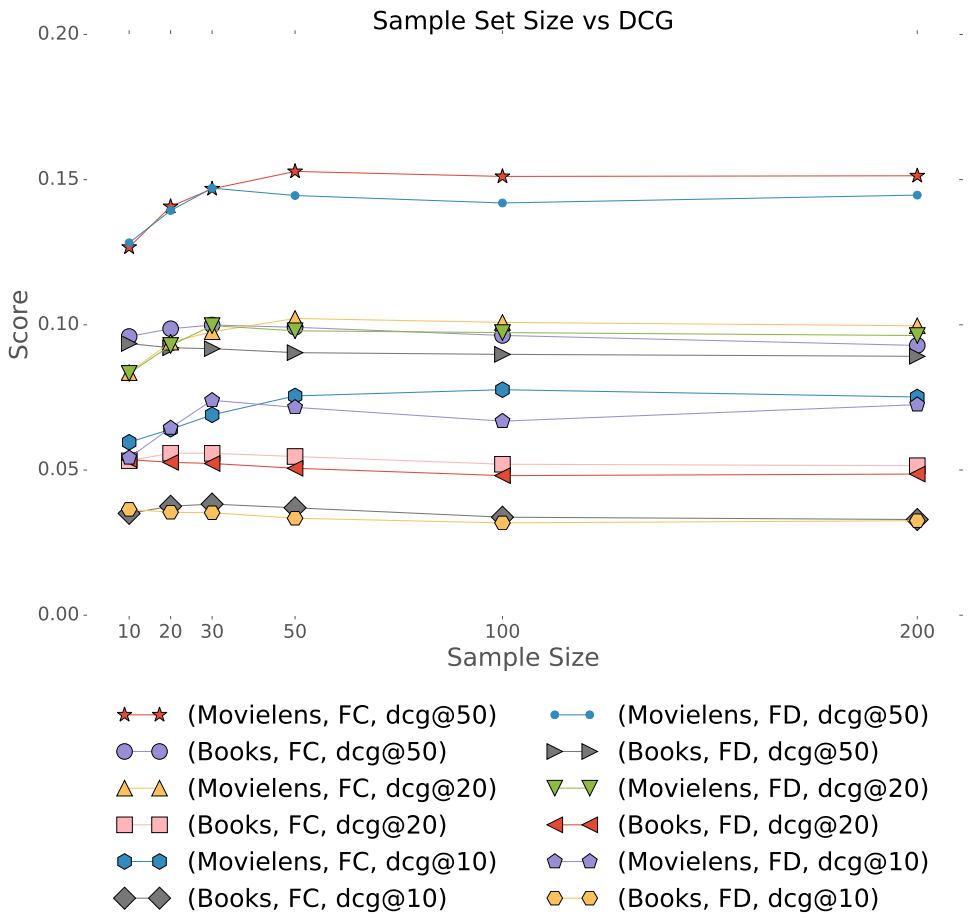

Figure 4.6: For most of the models the sample set size improves DCG until it saturates at 50. In this example we use Jaccard similarity.

Method	Recall@20	DCG@20
Cosine	0.0988	0.0553
Jaccard	0.0988	0.0547
ECP	0.0940	0.0601
EIR	0.1291	0.0344
FC Cosine	0.1020	0.0505
FD Cosine	0.1578	0.0860
FC Jaccard	0.1770	**0.1031**
FD Jaccard	**0.1866**	0.1010
FC ECP	0.0940	0.0444
FD ECP	0.1626	0.0856
FC EIR	0.0861	0.0434
FD EIR	0.1068	0.0560

Table 4.5: Experiments with combination of collaborative filtering for the least frequent (25%) conditional items of the MovieLens data.

Method	Recall@20	DCG@20
Collaborative baseline	0.139	0.057
Content baseline	0.131	0.056
FC content	0.239	0.108
FD content	0.214	0.093
Multimodal	0.275	0.123

Table 4.6: Experiments on MovieLens with DBpedia content using the Jaccard similarity.

best results are reached by blending based on item support. If the current session ends with an item of high support, we may take a robust baseline recommender. If the support is lower, less than around 100, Fisher models can be used to compile the recommendation.

Modalities: Implicit Feedback and Content. Table 4.6 shows our experiments with DBPedia content as a modality on MovieLens. For simplicity, we set the size of the sample set for both Fisher models to 10. The overall best performing model is the multimodal Fisher with Jaccard similarity, while every unimodal Fisher method outperform the baselines. By using equation (4.25), we could blend different modalities such as content and feedback without the need of setting external parameters or applying learning for blending.

Summary of Performance vs Baselines. Tables 4.7–4.5 presents the overall performance of the models. The results in Table 4.7 show that the linear combination of the standard normalized scores of the Fisher methods

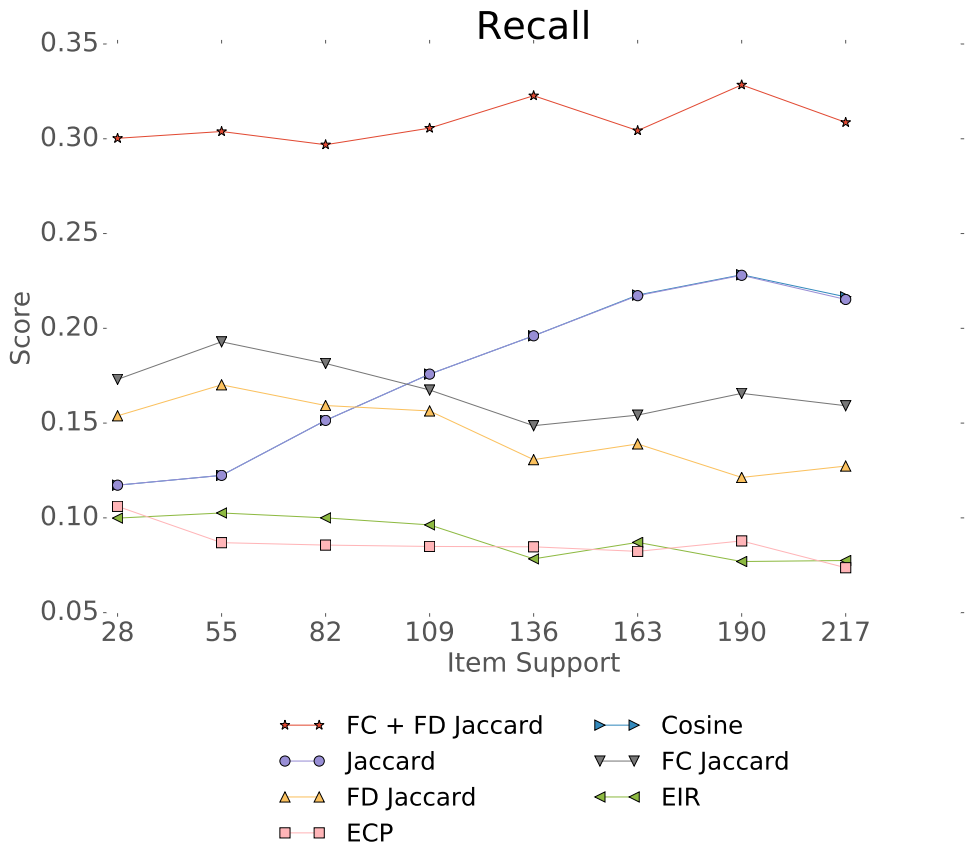

Figure 4.7: Recall@20 as the function of item support for the Netflix data set.

outperforms the best unimodal methods (Fisher with Jaccard) for Netflix and Books, while for MovieLens and Yahoo! Music, Fisher distance with Jaccard performs best.

Metric	Baseline & proposed method	Max. Freq.	MovieLens	Books	Yahoo! Music	Netflix
Recall@20	Jaccard	25%				0.13
		50%				0.18
		75%	0.12*			0.20
	EIR	25%	0.12	0.10	0.13	
		50%	0.11	0.10	0.11	
		75%		0.10	0.12	
	FD Jaccard	25%	**0.18**		**0.23**	
		50%	**0.19**		**0.23**	
		75%	**0.14**		**0.20**	
	FC + FD	25%		**0.14**		**0.30**
		50%		**0.14**		**0.30**
		75%		**0.13**		**0.31**
DCG@20	ECP	25%	0.05			
		50%	0.05			
		75%	0.05			
	EIR	25%		0.06	0.05	0.12
		50%		0.06	0.05	0.12
		75%		0.06	0.05	0.12
	FD Jaccard	25%	**0.10**		**0.11**	
		50%	**0.11**		**0.11**	
		75%	**0.08**		**0.10**	
	FC + FD	25%		**0.08**		**0.17**
		50%		**0.08**		**0.17**
		75%		**0.08**		**0.17**

Table 4.7: Summary of experiments results for the four data sets. Max. Freq. means the maximum frequency that was allowed for the infrequent items to have. For most methods there are (up to rounding errors) two best baseline and two best Fisher models, except for Recall where a third method (Cosine) appears in the cell marked by a star (∗). The best methods are usually FD Jaccard and FC + FD.

Chapter 5

Conclusions

5.1 Location-Based Social Networks

The following is a summary of the contributions from [Ayala-Gómez et al., 2017, Ayala-Gómez et al., 2018] in relation to the research questions of this dissertation.

RQ1.1 - How do preferences change when users are alone vs. when they are in a group? To answer this question we collected and analyzed a data set consisting of check-ins where friends that are together are explicitly mentioned. This differentiates our work to other existing approaches where groups are synthetically formed, for example, by considering users co-located within an hour. With this data set, we compared time, distance, and category preferences using the check-ins for users and groups.

We found the following observations. Groups move less than users and their check-ins are less frequent. Based on the analysis of time and distance between check-ins, we observed that 75% of the user check-ins occur between 2.5 days and within a distance of 10 kms. However, 75% of the groups check-ins happen between 8 days and within 5 kms. 50% of the groups move just 1 km between their check-ins. Groups in LBSNs are small. Most of the check-ins are made by groups of two people. Groups with size greater than twelve people are rare.

We highlighted the difference between categorical and location preferences of users and groups based on their ranking preferences using Kendall-tau rank correlation. We showed that users are willing to move to new locations, as groups prefer areas other than their members. In Figure 2.7 we can observe that users needs to travel to parts of the city that they are

not usually going. The KDE of the average weighted distance saturates between 5-10 kms.

Moreover, groups prefer other types of venues than their members. In Figure 2.4 we observed that top categories of users are different from top categories of their groups. The Kendall-tau most dense part is around 0.4.

RQ1.2 - How to recommend a list of POIs for groups of users in LBSN in the areas that a group prefers? Our proposed approach to building recommendations called GGR includes: *i)* POI pre-filtering, *ii)* category and location feature engineering, and *iii)* recommender systems training based on group check-ins.

Based on our experiments we observed that training recommender systems for groups works better than combining individual recommendations (1.3.1). This was expected, as we previously show that the behavior of users and groups differ. A better approach is to train a model based on group information only. The results for comparing iALS for groups vs. aggregating for individual users in Figure 2.11 show the superiority of group-based over individual recommendations. Moreover, in Figs. 2.12–2.14 we noticed that the geographical KDE improves the performance of the recommendations for all the cities. Finally, both geographical and categorical features are important, as models with either categorical or geographical information performed better than the same model without these features.

RQ1.3 - How to recommend a sequence of POIs for groups of users in LBSNs? In Section 2.2 we show that LBSNs contain contextual features of time and category transitions that affects the itinerary planning. Moreover, the relevance of the context of time and transition between types of POIs changes per venue types.

Our proposed approach is to construct a weighted graph where nodes are the candidate POIs that could be included in the itinerary. The edges have two weights, value and cost. The value is given by a recommender system that ranks the POI, and the cost is calculated by how far the POI is plus the average staying time in the venue type.

By experimenting with well-known recommender systems we show that iALS is a good option for building recommendations based on group profiles. Additionally, the sum of individual recommendations appeared as the best strategy for combining individual scores. Including the category type for selecting the candidates improved the quality of all the recommender systems. The proposed MCTS models for itinerary planning had greater or equal performance than the greedy baseline measured by precision and

recall.

5.2 Global Citation Recommendation

The following is a summary of the contributions from [Ayala-Gómez et al., 2018] in relation to the research questions of this dissertation.

RQ2.1 - How to expand sematic features of scientific abstracts using knowledge graphs? DBpedia Spotlight allowed us to expand the semantic features of the text. From the annotated DBpedia entities we extracted two new features: *entities* and *entities properties*. These features contributed to building top-k global citation recommendations.

RQ2.2 - Given an abstract of a new paper mapped to a knowledge graph, how to find a set of candidate papers to be cited? We used the expanded semantic features to improve the quality of the candidate set generation. Table 3.6 shows that *Lucene's More Like This* method has a higher recall when the relevant terms in the abstract and the entities are combined. This means that the entities help Lucene to formulate a better query to retrieve possible relevant candidates.

RQ2.3 - Given an abstract of a new paper mapped to a knowledge graph, and a set of candidate papers, how to build top-k recommendations of papers to cite? We defined this problem as a list-wise Learning to Rank task. We defined features that characterize the relation of a given abstract and a papers returned by the similarity search functionality of the Lucene search engine. We considered cited papers relevant and others irrelevant. Then, we trained LambdaMART to learn a model that predicts the pairs relevance.

Table 3.7 and 3.8 show that our proposed approach performs well in papers published during the same year of training and also in the next year. These results confirm that the approach of building the candidate set by content similarity and fitting LambdaMART to the features characterizing the relation of the papers leads to an nDCG higher than the baselines.

Tables 3.7 and 3.8 show that the model *LM-all* performs significantly better than *LM-w/o*. Moreover, Figure 3.5 shows that the *Entities tf-idf* is of high importance, as it is among the most frequently chosen by the LambdaMART model. These results show that the expanded semantic features are helping the model to differentiate what makes a candidate to be relevant for the given abstract.

5.3 Session Based Item-to-Item Recommendation

Our proposed approach for the item-to-item recommendation is based on the last item visited in the current user session. We proposed Fisher information based global item-item similarity models for this task and reached significant improvements over existing methods in cases of infrequent item-to-item transitions. The following is a summary of the contributions from [Daróczy et al., 2018] in relation to the research questions of this dissertation.

RQ3.1 - How to use the similarity information in the item-item transition conditional probability for a given infrequent-item? Our first proposed model is called Item-Item Fisher Conditional Score (FC). It estimates the transition probabilities based on the potential function of the simple similarity graph and the pairwise similarity graph. In Table 4.5 we show that the FC model performs the best when it uses the Jaccard similarity and has a better performance than the baselines. We experimented with content-based similarity by mapping the NetFlix dataset to DBPedia, where FC performed better than the other approaches as presented in Table 4.6.

RQ3.2 - How to rank the next item by its distance from an infrequent item? Our second proposed model is called Item-Item Fisher Distance (FD). This model computes the Fisher distance between the given item and a possible next item. Table 4.5 shows that the FD performs the best when it uses the Jaccard similarity, and has a better performance than the baselines. The FD model performs better than the baselines and the FC model as presented in Table 4.7. Furthermore, we combined the scores of the FC and FD models and show that the linear combination performs better than their individual scores.

RQ3.3 - How to handle different similarity metrics between the items to improve the infrequent item-to-item recommendations? We propose a way to combine different similarity metrics by creating multiple nodes and edges in the similarity graph according to the different modalities. We were able to compute content and collaborative similarities for the MovieLens data set. Table 4.6 shows that the multimodal model performed better than the FC and FD models.

5.4 Implications and Future Work

In this section we discuss how our contributions could be used as assumptions, ideas for improvement, and possible future work.

Recommendations for Groups in Location-Based Social Networks. Recommendations for groups in LBSN should be tailored for group preferences. The recommended list of POIs should be in-line with the group preferences in order to be useful. Our findings suggest that this is feasible at least for the known areas that a group prefers and for building an itinerary given a starting POI. However, there are other possible POIs recommendations tasks. For example, recommending new areas or recommending relevant places for new groups. A limitation of our research was the lack of data available for researching groups of users in LBSN. Future work could be to understand the reasons why the models perform different for the different cities, and to try the contextual information with recent developments such as autoencoders and deep neural networks.

Research Paper Recommendations. Researchers could save time by receiving a list of recommended papers related to the subject they are researching in a paper. We presented experimental results on using knowledge graphs to build global citation recommendations. We show that expanding the semantic features of a paper using knowledge graphs is helpful for building the candidate set and training the model.

Given the importance of semantic features, an obvious next step would be to try a similar approach for local citation recommendations. Moreover, it would be interesting to try data sets containing topics from other disciplines (e.g, medicine). Another direction could be building "vertical" recommender systems fit to the semantics of different fields (e.g., DBLP, PubMed). Finally, more advanced weighting methods (e.g., Okapi BM25) could help on balancing the incorrectly annotated entities.

Item-to-Item recommendations. The implications of being able to recommend items even in the lack of user profiles are significant. As both legislation and the users themselves move towards protecting their privacy on the Web, the need of recommender systems able to handle session based recommendations will increase. We could investigate the task of session based item-to-item recommendation task in an offline setting. However, researching the problem in an online setting could help us measure the performance of the models in an environment where new items and transitions

occur. A possible future direction is to combine the Fisher Information together with sequence to sequence learning models (e.g., RNN), and using the session context in the similarity graph to generate a sample set that improves the quality of the recommendations.

Bibliography

[Abele et al., 2017] Abele, A., McCrae, J. P., Buitelaar, P., Jentzsch, A., and Cyganiak, R. (2017). Linking open data cloud diagram (2017).

[Aggarwal, 2016] Aggarwal, C. C. (2016). *Recommender systems*. Springer.

[Alzoghbi et al., 2016] Alzoghbi, A., Ayala, V. A. A., Fischer, P. M., and Lausen, G. (2016). Learning-to-rank in research paper cbf recommendation: Leveraging irrelevant papers. In *CBRecSys@ RecSys*, pages 43–46.

[Anagnostopoulos et al., 2016] Anagnostopoulos, A., Atassi, R., Becchetti, L., Fazzone, A., and Silvestri, F. (2016). Tour recommendation for groups. *Data Mining and Knowledge Discovery*, pages 1–32.

[Anagnostopoulos et al., 2017] Anagnostopoulos, A., Atassi, R., Becchetti, L., Fazzone, A., and Silvestri, F. (2017). Tour recommendation for groups. *Data Mining and Knowledge Discovery*, 31(5):1157–1188.

[Arnold and Cohen, 2009] Arnold, A. and Cohen, W. W. (2009). Information extraction as link prediction: Using curated citation networks to improve gene detection. In *International Conference on Wireless Algorithms, Systems, and Applications*, pages 541–550. Springer.

[Auer et al., 2007] Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., and Ives, Z. (2007). Dbpedia: A nucleus for a web of open data. *The semantic web*, pages 722–735.

[Ayala-Gómez et al., 2018] Ayala-Gómez, F., Daróczy, B., Benczúr, A., Mathioudakis, M., and Gionis, A. (2018). Global citation recommendations using knowledge graphs. *Journal of Intelligent & Fuzzy Systems*.

[Ayala-Gómez et al., 2017] Ayala-Gómez, F., Daróczy, B., Mathioudakis, M., Benczúr, A., and Gionis, A. (2017). Where could we go?: Recommendations for groups in location-based social networks. In *Proceedings of the 2017 ACM on Web Science Conference*, WebSci '17, pages 93–102, New York, NY, USA. ACM.

[Ayala-Gómez et al., 2018] Ayala-Gómez, F., Keniş, B., Karagöz, P., and Benczúr, A. (2018). Top-k context-aware tour recommendations for groups. In *Advances in Computational Intelligence*, pages 176–193, Cham. Springer International Publishing.

[Baez et al., 2011] Baez, M., Mirylenka, D., and Parra, C. (2011). Understanding and supporting search for scholarly knowledge. *Proceeding of the 7th European Computer Science Summit*, pages 1–8.

[Bao et al., 2012] Bao, J., Zheng, Y., and Mokbel, M. F. (2012). Location-based and preference-aware recommendation using sparse geo-social networking data. In Cruz, I. F., Knoblock, C., Kröger, P., Tanin, E., and Widmayer, P., editors, *SIGSPATIAL/GIS*, pages 199–208. ACM.

[Bao et al., 2015] Bao, J., Zheng, Y., Wilkie, D., and Mokbel, M. (2015). Recommendations in location-based social networks: A survey. *Geoinformatica*, 19(3):525–565.

[Beckett and McBride, 2004] Beckett, D. and McBride, B. (2004). Rdf/xml syntax specification (revised). *W3C recommendation*, 10(2.3).

[Beel et al., 2013a] Beel, J., Genzmehr, M., Langer, S., Nürnberger, A., and Gipp, B. (2013a). A comparative analysis of offline and online evaluations and discussion of research paper recommender system evaluation. In *Proceedings of the International Workshop on Reproducibility and Replication in Recommender Systems Evaluation*, RepSys '13, pages 7–14, New York, NY, USA. ACM.

[Beel et al., 2016] Beel, J., Gipp, B., Langer, S., and Breitinger, C. (2016). Research-paper recommender systems: a literature survey. *International Journal on Digital Libraries*, 17(4):305–338.

[Beel et al., 2013b] Beel, J., Langer, S., Genzmehr, M., and Nürnberger, A. (2013b). Introducing docear's research paper recommender system. In *Proceedings of the 13th ACM/IEEE-CS joint conference on Digital libraries*, pages 459–460. ACM.

[Bennett et al., 2007a] Bennett, J., Lanning, S., et al. (2007a). The netflix prize. In *Proceedings of KDD cup and workshop*, volume 2007, page 35. New York, NY, USA.

[Bennett et al., 2007b] Bennett, J., Lanning, S., et al. (2007b). The netflix prize. In *Proceedings of KDD cup and workshop*, volume 2007, page 35. New York, NY, USA.

[Besag, 1975] Besag, J. (1975). Statistical analysis of non-lattice data. *The statistician*, pages 179–195.

[Bethard and Jurafsky, 2010] Bethard, S. and Jurafsky, D. (2010). Who should i cite: learning literature search models from citation behavior. In *Proceedings of the 19th ACM international conference on Information and knowledge management*, pages 609–618. ACM.

[Bizer et al., 2009] Bizer, C., Heath, T., and Berners-Lee, T. (2009). Linked data-the story so far. *International journal on semantic web and information systems*, 5(3):1–22.

[Brown et al., 2014] Brown, C., Lathia, N., Mascolo, C., Noulas, A., and Blondel, V. (2014). Group colocation behavior in technological social networks. *PLOS ONE*, 9(8):1–9.

[Browne et al., 2012] Browne, C. B., Powley, E., Whitehouse, D., Lucas, S. M., Cowling, P. I., Rohlfshagen, P., Tavener, S., Perez, D., Samothrakis, S., and Colton, S. (2012). A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in games*, 4(1):1–43.

[Burges et al., 2005] Burges, C., Shaked, T., Renshaw, E., Lazier, A., Deeds, M., Hamilton, N., and Hullender, G. (2005). Learning to rank using gradient descent. In *Proceedings of the 22nd international conference on Machine learning*, pages 89–96. ACM.

[Burges, 2010] Burges, C. J. (2010). From ranknet to lambdarank to lambdamart: An overview. *Learning*, 11(23-581):81.

[Burges et al., 2006] Burges, C. J. C., Ragno, R., and Le, Q. V. (2006). Learning to rank with nonsmooth cost functions. In *Proceedings of the 19th International Conference on Neural Information Processing Systems*, NIPS'06, pages 193–200, Cambridge, MA, USA. MIT Press.

[Campos et al., 2009] Campos, L. M., Fernández-Luna, J. M., Huete, J. F., and Rueda-Morales, M. A. (2009). Managing uncertainty in group recommending processes. *User Modeling and User-Adapted Interaction*, 19(3):207–242.

[Campos et al., 2014] Campos, P. G., Díez, F., and Cantador, I. (2014). Time-aware recommender systems: a comprehensive survey and analysis of existing evaluation protocols. *User Modeling and User-Adapted Interaction*, 24(1-2):67–119.

[Cencov, 2000] Cencov, N. N. (2000). *Statistical decision rules and optimal inference*. Number 53. American Mathematical Soc.

[Chapelle and Chang, 2010] Chapelle, O. and Chang, Y. (2010). Yahoo! learning to rank challenge overview. In *Proceedings of the 2010 International Conference on Yahoo! Learning to Rank Challenge - Volume 14*, YLRC'10, pages 1–24. JMLR.org.

[Chen et al., 2014] Chen, G., Wu, S., Zhou, J., and Tung, A. K. (2014). Automatic itinerary planning for traveling services. *IEEE transactions on knowledge and data engineering*, 26(3):514–527.

[Cheng et al., 2012] Cheng, C., Yang, H., King, I., and Lyu, M. R. (2012). Fused matrix factorization with geographical and social influence in location-based social networks. In *Aaai*, volume 12, page 1.

[Cheng and Roth, 2013] Cheng, X. and Roth, D. (2013). Relational inference for wikification. *Urbana*, 51(61801):16–58.

[Cho et al., 2011] Cho, E., Myers, S. A., and Leskovec, J. (2011). Friendship and mobility: User movement in location-based social networks. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '11, pages 1082–1090, New York, NY, USA. ACM.

[Cho et al., 2014] Cho, K., van Merrienboer, B., Çaglar Gülçehre, Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *EMNLP*.

[Daróczy,] Daróczy, B. *Link Machine learning methods for multimedia information retrieval*. Eötvös Loránd University.

[Daróczy et al., 2018] Daróczy, B., Ayala-Gómez, F., and Benczúr, A. (2018). Infrequent item-to-item recommendation via invariant random fields. In *Advances in Soft Computing*, pages 257–275, Cham. Springer International Publishing.

[Davidson et al., 2010] Davidson, J., Liebald, B., Liu, J., Nandy, P., Van Vleet, T., Gargi, U., Gupta, S., He, Y., Lambert, M., Livingston, B., et al. (2010). The youtube video recommendation system. In *Proceedings of the fourth ACM conference on Recommender systems*, pages 293–296. ACM.

[De Choudhury et al., 2010] De Choudhury, M., Feldman, M., Amer-Yahia, S., Golbandi, N., Lempel, R., and Yu, C. (2010). Automatic construction of travel itineraries using social breadcrumbs. In *Proceedings of the 21st ACM conference on Hypertext and hypermedia*, pages 35–44. ACM.

[Desrosiers and Karypis, 2011] Desrosiers, C. and Karypis, G. (2011). A comprehensive survey of neighborhood-based recommendation methods. In *Recommender systems handbook*, pages 107–144. Springer.

[Dror et al., 2012] Dror, G., Koenigstein, N., Koren, Y., and Weimer, M. (2012). The yahoo! music dataset and kdd-cup'11. In *Proceedings of KDD Cup 2011*, pages 3–18.

[Ester et al., 1996] Ester, M., Kriegel, H.-P., Sander, J., and Xu, X. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proc. of 2nd International Conference on Knowledge Discovery and Data Mining (KDD-96)*, pages 226–231.

[Fan et al., 2017] Fan, L., Bonomi, L., Shahabi, C., and Xiong, L. (2017). Multi-user itinerary planning for optimal group preference. In *Advances in Spatial and Temporal Databases*, pages 3–23.

[Friedman, 2001] Friedman, J. H. (2001). Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232.

[Fuglede and Topsoe, 2004] Fuglede, B. and Topsoe, F. (2004). Jensen-shannon divergence and hilbert space embedding. In *Information Theory, 2004. ISIT 2004. Proceedings. International Symposium on*, page 31. IEEE.

[Gao et al., 2013] Gao, H., Tang, J., Hu, X., and Liu, H. (2013). Exploring temporal effects for location recommendation on location-based social

networks. In Yang, Q., King, I., Li, Q., Pu, P., and Karypis, G., editors, *RecSys*, pages 93–100. ACM.

[Gionis et al., 2014] Gionis, A., Lappas, T., Pelechrinis, K., and Terzi, E. (2014). Customized tour recommendations in urban areas. In *Proceedings of the 7th ACM International Conference on Web Search and Data Mining*, pages 313–322.

[Gollub et al., 2013] Gollub, T., Hagen, M., Michel, M., and Stein, B. (2013). From keywords to keyqueries: content descriptors for the web. In *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*, pages 981–984. ACM.

[Goodman, 1961] Goodman, L. A. (1961). Snowball sampling. *The annals of mathematical statistics*, pages 148–170.

[Gruber, 1995] Gruber, T. R. (1995). Toward principles for the design of ontologies used for knowledge sharing? *International journal of human-computer studies*, 43(5-6):907–928.

[Guttman, 1984] Guttman, A. (1984). *R-trees: a dynamic index structure for spatial searching*, volume 14. ACM.

[Hall and Tiropanis, 2012] Hall, W. and Tiropanis, T. (2012). Web evolution and web science. *Computer Networks*, 56(18):3859–3865.

[Hammersley and Clifford, 1971] Hammersley, J. M. and Clifford, P. (1971). Markov fields on finite graphs and lattices.

[Harper and Konstan, 2015] Harper, F. M. and Konstan, J. A. (2015). The movielens datasets: History and context. *ACM Trans. Interact. Intell. Syst.*, 5(4):19:1–19:19.

[Harper and Konstan, 2016] Harper, F. M. and Konstan, J. A. (2016). The movielens datasets: History and context. *ACM Transactions on Interactive Intelligent Systems (TiiS)*, 5(4):19.

[He et al., 2010] He, Q., Pei, J., Kifer, D., Mitra, P., and Giles, L. (2010). Context-aware citation recommendation. In *Proceedings of the 19th International Conference on World Wide Web*, WWW '10, pages 421–430, New York, NY, USA. ACM.

[Herlocker et al., 1999] Herlocker, J. L., Konstan, J. A., Borchers, A., and Riedl, J. (1999). An algorithmic framework for performing collaborative

filtering. In *Proceedings of the 22Nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '99, pages 230–237, New York, NY, USA. ACM.

[Hess, 2006] Hess, C. (2006). Trust-based recommendations for publications: A multi-layer network approach. *TCDL Bulletin*, 2(2):190–201.

[Hidasi and Tikk, 2012] Hidasi, B. and Tikk, D. (2012). Fast als-based tensor factorization for context-aware recommendation from implicit feedback. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 67–82. Springer.

[Hidasi and Tikk, 2016] Hidasi, B. and Tikk, D. (2016). General factorization framework for context-aware recommendations. *Data Mining and Knowledge Discovery*, 30(2):342–371.

[Hu et al., 2008] Hu, Y., Koren, Y., and Volinsky, C. (2008). Collaborative filtering for implicit feedback datasets. In *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*, pages 263–272. Ieee.

[Huang et al., 2014] Huang, W., Wu, Z., Mitra, P., and Giles, C. L. (2014). Refseer: A citation recommendation system. In *Proceedings of the 14th ACM/IEEE-CS Joint Conference on Digital Libraries*, pages 371–374. IEEE Press.

[Jaakkola and Haussler, 1999] Jaakkola, T. and Haussler, D. (1999). Exploiting generative models in discriminative classifiers. In *Advances in neural information processing systems*, pages 487–493.

[Jack, 2012] Jack, K. (2012). Mahout becomes a researcher: large scale recommendations at mendeley. In *Presentation at big data week conferences*.

[Janke et al., 2004] Janke, W., Johnston, D., and Kenna, R. (2004). Information geometry and phase transitions. *Physica A: Statistical Mechanics and its Applications*, 336(1-2):181–186.

[Joachims, 2002] Joachims, T. (2002). Optimizing search engines using clickthrough data. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 133–142. ACM.

[Jost and Jost, 2008] Jost, J. and Jost, J. (2008). *Riemannian geometry and geometric analysis*, volume 42005. Springer.

[Kendall, 1945] Kendall, M. G. (1945). The treatment of ties in ranking problems. *Biometrika*, 33(3):239–251.

[Kocsis et al., 2006] Kocsis, L., Szepesvári, C., and Willemson, J. (2006). Improved monte-carlo search. *Univ. Tartu, Estonia, Tech. Rep*, 1.

[Koenigstein and Koren, 2013] Koenigstein, N. and Koren, Y. (2013). Towards scalable and accurate item-oriented recommendations. In *Proceedings of the 7th ACM conference on Recommender systems*, pages 419–422. ACM.

[Koren, 2009] Koren, Y. (2009). The bellkor solution to the netflix grand prize. *Netflix prize documentation*, 81:1–10.

[Koren et al., 2009] Koren, Y., Bell, R., Volinsky, C., et al. (2009). Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37.

[Larsen and Von Ins, 2010] Larsen, P. O. and Von Ins, M. (2010). The rate of growth in scientific publication and the decline in coverage provided by science citation index. *Scientometrics*, 84(3):575–603.

[Li et al., 2006] Li, H., Councill, I., Lee, W.-C., and Giles, C. L. (2006). Citeseerx: an architecture and web service design for an academic document search engine. In *Proceedings of the 15th international conference on World Wide Web*, pages 883–884. ACM.

[Lian et al., 2014] Lian, D., Zhao, C., Xie, X., Sun, G., Chen, E., and Rui, Y. (2014). Geomf: joint geographical modeling and matrix factorization for point-of-interest recommendation. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 831–840. ACM.

[Liang et al., 2011] Liang, Y., Li, Q., and Qian, T. (2011). Finding relevant papers based on citation relations. In *International Conference on Web-Age Information Management*, pages 403–414. Springer.

[Liao et al., 2016] Liao, Y., Lam, W., Jameel, S., Schockaert, S., and Xie, X. (2016). Who wants to join me?: Companion recommendation in location based social networks. In *Proceedings of the 2016 ACM International Conference on the Theory of Information Retrieval*, ICTIR '16, pages 271–280, New York, NY, USA. ACM.

[Lim et al., 2017a] Lim, K. H., Chan, J., Karunasekera, S., and Leckie, C. (2017a). Personalized itinerary recommendation with queuing time awareness. In *Proceedings of the 40th international ACM SIGIR conference on research and development in information retrieval (SIGIR'17) Google Scholar*.

[Lim et al., 2017b] Lim, K. H., Chan, J., Leckie, C., and Karunasekera, S. (2017b). Personalized trip recommendation for tourists based on user interests, points of interest visit durations and visit recency. *Knowledge and Information Systems*, pages 1–32.

[Linden et al., 2003] Linden, G., Smith, B., and York, J. (2003). Amazon. com recommendations: Item-to-item collaborative filtering. *IEEE Internet computing*, 7(1):76–80.

[Luhn, 1957] Luhn, H. P. (1957). A statistical approach to mechanized encoding and searching of literary information. *IBM Journal of research and development*, 1(4):309–317.

[Mabe, 2003] Mabe, M. (2003). The growth and number of journals. *Serials*, 16(2):191–197.

[Manning et al., 2008] Manning, C. D., Raghavan, P., and Schütze, H. (2008). *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA.

[Masinter et al., 2005] Masinter, L., Berners-Lee, T., and Fielding, R. T. (2005). Uniform resource identifier (uri): Generic syntax.

[Masthoff, 2015] Masthoff, J. (2015). *Group Recommender Systems: Aggregation, Satisfaction and Group Attributes*, pages 743–776. Springer US, Boston, MA.

[McGuinness et al., 2004] McGuinness, D. L., Van Harmelen, F., et al. (2004). Owl web ontology language overview. *W3C recommendation*, 10(10):2004.

[Mendes et al., 2011] Mendes, P. N., Jakob, M., García-Silva, A., and Bizer, C. (2011). Dbpedia spotlight: shedding light on the web of documents. In *Proceedings of the 7th international conference on semantic systems*, pages 1–8. ACM.

[Micsik et al., 2015] Micsik, A., Turbucz, S., and Tóth, Z. (2015). Exploring publication metadata graphs with the lodmilla browser and editor. *International Journal on Digital Libraries*, 16(1):15–24.

[Middleton et al., 2001] Middleton, S. E., De Roure, D. C., and Shadbolt, N. R. (2001). Capturing knowledge of user preferences: Ontologies in recommender systems. In *Proceedings of the 1st International Conference on Knowledge Capture*, K-CAP '01, pages 100–107, New York, NY, USA. ACM.

[Noulas et al., 2011] Noulas, A., Scellato, S., Mascolo, C., and Pontil, M. (2011). An empirical study of geographic user activity patterns in foursquare. In Adamic, L. A., Baeza-Yates, R. A., and Counts, S., editors, *ICWSM*. The AAAI Press.

[Pálovics et al., 2014] Pálovics, R., Ayala-Gómez, F., Csikota, B., Daróczy, B., Kocsis, L., Spadacene, D., and Benczúr, A. A. (2014). Recsys challenge 2014: an ensemble of binary classifiers and matrix factorization. In *Proceedings of the 2014 Recommender Systems Challenge*, page 13. ACM.

[Paraschiv et al., 2016] Paraschiv, I. C., Dascalu, M., Dessus, P., Trausan-Matu, S., and McNamara, D. S. (2016). A paper recommendation system with readerbench: the graphical visualization of semantically related papers and concepts. In *State-of-the-Art and Future Directions of Smart Learning*, pages 445–451. Springer.

[Perronnin and Dance, 2007] Perronnin, F. and Dance, C. (2007). Fisher kernels on visual vocabularies for image categorization. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pages 1–8. IEEE.

[Pilászy et al., 2015] Pilászy, I., Serény, A., Dózsa, G., Hidasi, B., Sári, A., and Gub, J. (2015). Neighbor methods vs. matrix factorizationcase studies of real-life recommendations. In *LSRS Workshop at ACM RecSys*.

[Purushotham et al., 2014] Purushotham, S., Kuo, C.-C. J., Shahabdeen, J., and Nachman, L. (2014). Collaborative group-activity recommendation in location-based social networks. In *Proceedings of the 3rd ACM SIGSPATIAL International Workshop on Crowdsourced and Volunteered Geographic Information*, GeoCrowd '14, pages 8–15, New York, NY, USA. ACM.

[Quadrana et al., 2017] Quadrana, M., Karatzoglou, A., Hidasi, B., and Cremonesi, P. (2017). Personalizing session-based recommendations with hierarchical recurrent neural networks. In *Proceedings of the Eleventh*

ACM Conference on Recommender Systems, RecSys '17, pages 130–137, New York, NY, USA. ACM.

[Ratinov et al., 2011] Ratinov, L., Roth, D., Downey, D., and Anderson, M. (2011). Local and global algorithms for disambiguation to wikipedia. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, pages 1375–1384, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Rendle, 2010] Rendle, S. (2010). Factorization machines. In *Data Mining (ICDM), 2010 IEEE 10th International Conference on*, pages 995–1000. IEEE.

[Rendle and Schmidt-Thieme, 2010] Rendle, S. and Schmidt-Thieme, L. (2010). Pairwise interaction tensor factorization for personalized tag recommendation. In *Proceedings of the third ACM international conference on Web search and data mining*, pages 81–90. ACM.

[Ricci et al., 2015] Ricci, F., Rokach, L., and Shapira, B. (2015). *Recommender Systems Handbook*. Springer Publishing Company, Incorporated, 2nd edition.

[Roy et al., 2011] Roy, S. B., Das, G., Amer-Yahia, S., and Yu, C. (2011). Interactive itinerary planning. In *Data Engineering (ICDE), 2011 IEEE 27th International Conference on*, pages 15–26. IEEE.

[Sarwar et al., 2001a] Sarwar, B., Karypis, G., Konstan, J., and Riedl, J. (2001a). Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*, pages 285–295. ACM.

[Sarwar et al., 2001b] Sarwar, B., Karypis, G., Konstan, J., and Riedl, J. (2001b). Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*, pages 285–295. ACM.

[Schein et al., 2002] Schein, A. I., Popescul, A., Ungar, L. H., and Pennock, D. M. (2002). Methods and metrics for cold-start recommendations. In *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 253–260. ACM.

[Silverman, 1986] Silverman, B. W. (1986). *Density estimation for statistics and data analysis*, volume 26. CRC press.

[Sinha et al., 2015] Sinha, A., Shen, Z., Song, Y., Ma, H., Eide, D., Hsu, B.-j. P., and Wang, K. (2015). An overview of microsoft academic service (mas) and applications. In *Proceedings of the 24th international conference on world wide web*, pages 243–246. ACM.

[Strohman et al., 2007] Strohman, T., Croft, W. B., and Jensen, D. (2007). Recommending citations for academic papers. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 705–706. ACM.

[Szeredi et al., 2014] Szeredi, P., Lukcsy, G., and Benk, T. (2014). *The Semantic Web Explained: The Technology and Mathematics Behind Web 3.0*. Cambridge University Press, New York, NY, USA.

[Tobler, 1970] Tobler, W. R. (1970). A computer movie simulating urban growth in the detroit region. *Economic geography*, 46(sup1):234–240.

[Wu et al., 2016] Wu, J., Liang, C., Yang, H., and Giles, C. L. (2016). Citeseerx data: Semanticizing scholarly papers. In *Proceedings of the International Workshop on Semantic Big Data*, SBD '16, pages 2:1–2:6, New York, NY, USA. ACM.

[Zhang and Chow, 2015] Zhang, J.-D. and Chow, C.-Y. (2015). Geosoca: Exploiting geographical, social and categorical correlations for point-of-interest recommendations. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '15, pages 443–452, New York, NY, USA. ACM.

[Zhou et al., 2008] Zhou, D., Zhu, S., Yu, K., Song, X., Tseng, B. L., Zha, H., and Giles, C. L. (2008). Learning multiple graphs for document recommendations. In *Proceedings of the 17th international conference on World Wide Web*, pages 141–150. ACM.

[Ziegler et al., 2005] Ziegler, C.-N., McNee, S. M., Konstan, J. A., and Lausen, G. (2005). Improving recommendation lists through topic diversification. In *Proceedings of the 14th international conference on World Wide Web*, pages 22–32. ACM.