

FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Latent Context-aware Recommender Systems

Maria dos Santos de Abreu



Mestrado Integrado em Engenharia Informática e Computação

Supervisor: Carlos Soares, PhD

Co-supervisor: Tiago Cunha, PhD

February 24, 2020

Latent Context-aware Recommender Systems

Maria dos Santos de Abreu

Mestrado Integrado em Engenharia Informática e Computação

February 24, 2020

Abstract

The relevance of Recommender Systems (RS) grew with the increase in the variety of products, content and activities that users are presented with nowadays. These types of systems are used daily by millions of people on services as diverse as Spotify, Amazon and Netflix. Most approaches use relatively simple and straightforward data, such as user and item properties. As progress in this area was made, it was established that user preferences can change with context. For example, a person running would listen to a different type of music comparatively to a person relaxing at night. Even though context is hard to capture, the ubiquity and interconnectivity of devices present an opportunity to do so. With the widespread usage of smartphones and other smart devices, huge amounts of data are generated daily. Context-aware Recommender Systems (CARS) constitute an approach that aims to make better use of more of the available data. CARS are a subset of RS that use contextual information such as location, time of day, or other relevant variables to improve the quality of the recommendations.

Amongst its main characterisation traits lies the fact that contextual data can be classified by whether it is latent or not. Non-latent variables' meaning is known and well-defined from the start, such as the ones aforementioned. These variables can model context through predetermined variables, such as time and location, for an example. Even though non-latent variables are more prevalent, including these additional contextual variables turns the rating matrix into a tensor, which will be sparser than the matrix. On the other hand, latent variables must be extracted from non-latent context, and their explicit meaning is unknown. Even so, capturing these variables can help model context more accurately, with less variables and consequently, less sparsity. Since latent context is derived from the predetermined, non-latent context, it can reflect hidden relationships or patterns from combinations of the original variables. In the previous example, if time and location were the original non-latent context, we could extract one single variable to model both temporal and positional contexts. Another advantage is that this extraction process can be parameterised, which means it would be possible to decide the number of variables and experiment in order to obtain the best results. Therefore, latent context might be used implicitly to improve the end result. For all these reasons, we infer that it is relevant and important to ascertain the value of this type of context.

Thus, in this dissertation we will develop a method to extract latent contextual variables and empirically evaluate their usefulness for recommendation. The project was composed of several tasks: firstly, a technique to extract latent context variables from contextual variables was employed. Afterwards, this context was integrated in a recommender system. The method developed was evaluated against another state of the art CARS approach. Our method was also compared with the same algorithm not using any type of context, as well as other traditional, popular non-contextual algorithms. All these methods were empirically evaluated in terms of the quality of their recommendations.

The analysis of the results revealed potential in latent contextual approaches. In two out of the three data sets used, the approach developed obtained better results in all the evaluation metrics,

including AUC and NDCG. Furthermore, in the remaining data set, the performance of our method is comparable to the others. Therefore, the results provide evidence in favour of latent context improving recommendations while using less variables.

Resumo

A relevância dos Sistemas de Recomendação tem aumentado com a variedade de produtos, conteúdo e atividades que rodeiam os utilizadores nos dias que correm. Este tipo de sistemas são usados diariamente por milhões de pessoas, em serviços tão diversos como Spotify, Amazon e Netflix. A maior parte destes sistemas usam dados relativamente simples e diretos, tais como as propriedades de cada item e as características do utilizador.

À medida que foi sendo feito progresso nesta área, foi estabelecido que as preferências dos utilizadores podem mudar com o contexto em que estes se encontram. Por exemplo, uma pessoa a correr ouvirá um tipo de música diferente comparativamente a outra que esteja a relaxar à noite. Apesar do contexto ser difícil de capturar, a ubiquidade e interconetividade dos dispositivos apresentam uma oportunidade para o fazer. Com o uso comum de *smartphones* e outros dispositivos inteligentes, são geradas diariamente grandes quantidades de dados. Os Sistemas de Recomendação Contextuais constituem uma abordagem que tem como objetivo aproveitar mais e melhor os dados disponíveis. Os Sistemas de Recomendação Contextuais são um subconjunto dos Sistemas de Recomendação que usa informação contextual, tal como localização, hora do dia, ou outras variáveis relevantes para melhorar a qualidade das recomendações.

Uma das principais formas de caracterizar contexto prende-se ao facto de dados contextuais poderem ser classificados como latentes ou não-latentes. O significado das variáveis não-latentes é conhecido e bem definido desde o início, como nos exemplos mencionados anteriormente. Estas variáveis modelam o contexto através de variáveis predefinidas, como hora do dia e localização, por exemplo. Apesar de variáveis contextuais não-latentes serem mais prevalentes, o seu uso faz com que a matriz de avaliações passe a ser um tensor, que será mais disperso que a matriz original. Por outro lado, as variáveis latentes têm de ser extraídas através de variáveis contextuais não-latentes e o seu significado explícito é desconhecido. As variáveis não-latentes modelam variáveis pré-determinadas, como por exemplo tempo e localização. Ainda assim, capturar estas variáveis pode permitir modelar o contexto mais fielmente, com menos variáveis, e portanto menos dispersão. Dado que o contexto latente é derivado do contexto predeterminado, não-latente, é possível que este reflita relações ou padrões escondidos em combinações das variáveis originais. No exemplo anterior, sendo hora do dia e localização as variáveis contextuais originais não-latentes, poderia extrair-se uma única variável que modelasse tanto o contexto temporal como local. Outra vantagem é que este processo de extração é parametrizável, o que significa que é possível decidir o número de variáveis que são extraídas e experimentar de modo a obter os melhores resultados. Assim, o contexto latente pode ser usado implicitamente para melhorar as recomendações. Por todas as razões apresentadas, depreende-se que é relevante e importante averiguar o valor deste tipo de contexto.

Deste modo, nesta dissertação desenvolveremos um método para extrair contexto latente e avaliaremos empiricamente a sua utilidade em recomendações. O projeto foi composto por várias tarefas: primeiro, uma técnica foi usada para a extração do contexto latente. De seguida, este contexto latente foi integrado num sistema de recomendação. O método de recomendação contextual

latente desenvolvido foi avaliado contra outro sistema de recomendação contextual *state of the art*. O nosso método foi ainda comparado com um algoritmo semelhante que não usa qualquer tipo de contexto, assim como com outros algoritmos populares não-contextuais tradicionais. Todos estes diferentes métodos foram comparados empiricamente em termos da qualidade das recomendações feitas.

A análise dos resultados revelou potencial na abordagem contextual latente. Em dois dos três *data sets* utilizados, o método desenvolvido obteve melhores resultados em todas as métricas de avaliação, incluindo AUC e NDCG. Para além do mais, no *data set* restante os resultados da nossa abordagem são comparáveis aos das outras abordagens. Consequentemente, os resultados apresentam argumentos a favor do uso de contexto latente para melhorar recomendações, usando menos variáveis.

Acknowledgements

Gostaria de começar por expressar a minha gratidão para com os meus orientadores. Pela paciência infinita, por toda a disponibilidade, pelos conselhos e pelo voto de confiança, obrigada.

Queria também deixar uma palavra de apreço a todos os colegas e professores que de alguma forma me ajudaram ao longo deste percurso académico.

A todas as pessoas do laboratório onde desenvolvi esta dissertação, agradeço a simpatia e o acolhimento afável.

A todos os amigos (e vizinhas) que me animaram quando mais precisava, obrigada pelas vossas palavras, companhia e carinho.

Ao Francisco, pelo apoio inabalável, pela calma e pela fé em mim, muito obrigada.

Por último mas não em último, à minha família, sem a qual nada disto seria possível.

Maria dos Santos de Abreu

Este trabalho é financiado parcialmente por Fundos FEDER através do Programa Operacional Competitividade e Internacionalização - COMPETE 2020 e por Fundos Nacionais através da FCT - Fundação para a Ciência e a Tecnologia no âmbito do projeto CONTEXTWA (FCT PTDC/EEI-SCR/6945/2014 - POCI-01-0145-FEDER-016883).

This work is partly funded by the ERDF – European Regional Development Fund through the Operational Programme for Competitiveness and Internationalisation - COMPETE 2020 Programme and by National Funds through the Portuguese funding agency, FCT - Fundação para a Ciência e a Tecnologia within project CONTEXTWA (FCT PTDC/EEI-SCR/6945/2014 - POCI-01-0145-FEDER-016883).

“An answer is invariably the parent of a great family of new questions.”

John Steinbeck

Contents

1	Introduction	1
1.1	Motivation	2
1.2	Goals and Research Questions	2
1.3	Dissertation Structure	3
2	Recommender Systems	5
2.1	Algorithms	5
2.1.1	Content-based Filtering	6
2.1.2	Collaborative Filtering	6
2.1.3	Hybrid Approaches	8
2.2	Common Issues	9
2.3	Evaluation	11
2.4	Contextual Recommender Systems	14
2.4.1	Definition of Context	15
2.4.2	Non-latent Context versus Latent Context	15
2.4.3	Algorithms	16
2.4.4	Literature Review of Latent Approaches	18
2.4.5	Related Work	20
2.5	Summary	21
3	A Latent Context-aware Recommender System Approach	23
3.1	Approach Overview	23
3.1.1	Context-aware Recommender Systems	24
3.1.2	Latent Context Extraction	25
3.2	Empirical Evaluation	26
3.2.1	Latent Context Extraction	26
3.2.2	Latent Context-aware Recommender System	27
3.3	Experimental Setup	28
3.3.1	Data	28
3.3.2	Data Preparation	28
3.3.3	Evaluation	29
3.4	Implementation	32
3.5	Discussion	32
4	Empirical Study	33
4.1	Experimental Setup	33
4.1.1	Data	33
4.1.2	Data Preparation	34

CONTENTS

4.1.3	Evaluation	35
4.2	Implementation	37
4.3	Analysis of the Results	37
4.3.1	DePaulMovie	38
4.3.2	InCarMusic	42
4.3.3	TijuanaRestaurant	45
4.4	Discussion	48
4.5	Research Questions	48
5	Conclusions and Future Work	51
	References	53
A	Preparations	59
B	Results	61

List of Figures

1.1	Illustration of the objectives for the project	3
2.1	Matrix representation of a traditional Recommender System	6
2.2	Amazon collaborative filtering example	7
2.3	Schematic representation of the kNN algorithm	8
2.4	Categorisation of context according to acquisition and latency	15
2.5	Method used in related work [UBSR16]	21
3.1	Tensor representation of contextual data	24
3.2	Matrix approximation of contextual data	25
3.3	High level scheme of the developed method	26
3.4	Process for latent context extraction	27
3.5	Process for generation of recommendations and empirical evaluation	28
3.6	Schematic representation of aggregation of two contextual variables	29
3.7	Evaluation protocol for Recommender Systems	30
4.1	Schematic representation of aggregation of two contextual variables	34
4.2	Evaluation protocol for Recommender Systems	35
4.3	Comparison of P@5 and P@10 in the <i>DePaulMovie</i> data set	39
4.4	Comparison of AUC@5 and AUC@10 in the <i>DePaulMovie</i> data set	40
4.5	Comparison of MAP@5 and MAP@10 in the <i>DePaulMovie</i> data set	40
4.6	Comparison of NDCG@5 and NDCG@10 in the <i>DePaulMovie</i> data set	41
4.7	Comparison of F1@5 and F1@10 in the <i>DePaulMovie</i> data set	41
4.8	Comparison of AUC@5 and AUC@10 in the <i>InCarMusic</i> data set	42
4.9	Comparison of MAP@5 and MAP@10 in the <i>InCarMusic</i> data set	43
4.10	Comparison of NDCG@5 and NDCG@10 in the <i>InCarMusic</i> data set	43
4.11	Comparison of F1@5 and F1@10 in the <i>InCarMusic</i> data set	44
4.12	Comparison of P@5 and P@10 in the <i>InCarMusic</i> data set	44
4.13	Comparison of AUC@5 and AUC@10 in the <i>TijuanaRestaurant</i> data set	45
4.14	Comparison of MAP@5 and MAP@10 in the <i>TijuanaRestaurant</i> data set	46
4.15	Comparison of NDCG@5 and NDCG@10 in the <i>TijuanaRestaurant</i> data set	46
4.16	Comparison of P@5 and P@10 in the <i>TijuanaRestaurant</i> data set	47
4.17	Comparison of F1@5 and F1@10 in the <i>TijuanaRestaurant</i> data set	47
B.1	First worksheet of the results from the <i>DePaulMovie</i> data set	62
B.2	Second worksheet of the results from the <i>DePaulMovie</i> data set	62
B.3	First worksheet of the results from the <i>InCarMusic</i> data set	63
B.4	Second worksheet of the results from the <i>InCarMusic</i> data set	63
B.5	First worksheet of the results from the <i>RestrTijuana</i> data set	64

LIST OF FIGURES

B.6	Second worksheet of the results from the <i>RestrTijuana</i> data set	64
-----	---	----

List of Tables

2.1	Classification of context-aware recommendation algorithms by [LBK17]	16
4.1	Collection of statistics for the used data sets	34
4.2	Set of approaches tested	38
4.3	Non-latent context conditions and latent context conditions extracted per data set	38
4.4	Frequency distribution of data set <i>DePaulMovie</i>	39
4.5	Frequency distribution of data set <i>InCarMusic</i>	42
4.6	Frequency distribution of data set <i>TijuanaRestaurant</i>	45

LIST OF TABLES

Abbreviations

RS	Recommender Systems
CARS	Context-Aware Recommender Systems
ML	Machine Learning
MF	Matrix Factorisation
PCC	Pearson's Correlation Coefficient
JMF	Joint Matrix Factorisation
POI	Points of interest
MAE	Mean Absolute Error
RMSE	Root-mean-square Error
NDCG	Normative Discounted Cumulative Gain
NDPM	Normalised Distance-based Performance Measure
PCA	Principal Component Analysis
GUI	Graphical User Interface
kNN	k Nearest Neighbours
ROC	Receiver Operating Characteristics
PRC	Precision Recall Curve
AUC	Area Under the Curve
MAP	Mean Average Precision
P@K	Precision at K
R@K	Recall at K
LDA	Latent Dirichlet allocation
GSP	Generalized Sequential Pattern
SGD	Stochastic gradient descent

Chapter 1

Introduction

Recommender Systems (RS) have become more prevalent as the variety of products, content and activities increases and it becomes difficult for users to browse an entire catalogue in order to find relevant content. More recently, RS have been evolving alongside the widespread usage of smartphones and other smart devices. With the constant use of these devices, huge amounts of data are generated daily, presenting an opportunity for companies to improve the quality of their recommendations and better match their products to the consumers. In fact, RS are considered the most suitable solution to the information overload problem that plagues consumers nowadays, facilitating their decision of what to consume. The matrix with all the information required can become very large, and therefore Matrix Factorisation (MF) is a common method used for dimensionality reduction. However, it is also common for user preferences to change with context [AT11]. For instance, on weekdays a person might prefer to watch short series, and films on weekends. Therefore, capturing the current circumstances of the user and including them in the recommendation process can improve the quality of the recommendations. Context-Aware Recommender Systems (CARS) are RS that take into account the user context to generate recommendations. The pertinence of CARS can be ascertained by recent literature on the topic [AT11, VSDCT18].

Existing contextual RS show an overall improvement relatively to their non-contextual counterparts [Ado05]. These systems make recommendations by adding a contextual dimension to the traditional User x Item recommendation space. However, this also normally implies using a higher number of variables, in order to represent the user context. This increases the problem of the dimensionality of the attribute space, which negatively affects the computational cost and makes it harder to find patterns. Additionally, a system with more dimensions also requires a considerable amount of data along each one, so the model can learn and generalise for each dimension. However, increasing the number of dimensions turns the rating matrix into a tensor, less dense than the matrix, which might negatively impact the recommendations [AT11]. Therefore, it would be useful to know which variables are more important and use them to model contextual information. Latent context variables present a possibility for tackling these issues.

Contextual variables can be classified as latent or non-latent. Non-latent contextual variables have well-defined meanings. The time and location at which someone listened to a certain song

are easily interpretable examples of non-latent context. This type of context can be acquired explicitly, through user input for an example, or implicitly, through sensor data and other indirect information. On the other hand, latent contextual variables are extracted from non-latent contextual variables. They are learned from hidden patterns in non-latent data, and their meaning is not straightforward. Latent context variables' can potentially model context more accurately and with less variables, since they might reflect relations in the data. Besides, they also have advantages in terms of maintaining privacy as defended by [UBSR16], as their explicit meaning is unknown. Latent context-aware RS are a subset of CARS that use latent context instead of non-latent context [Ung15].

1.1 Motivation

This particular line of investigation is still relatively recent and consequently there is still room for contributions. This happens since latent variables' potential is not fully understood yet, and therefore more studying is required to understand their value.

This dissertation proposes a new method for a latent context-aware RS. This area is still in an early stage of development. While in related work methods such as neural networks are used to extract context, here a simple MF algorithm will be used. Even though this method is widely used for dimensionality reduction, its purpose in this project is innovative. Therefore, we will explore this idea, proposing a new CARS algorithm based on it.

The latent data resultant from MF will afterwards be integrated within a RS based on this method of extraction, in order to confirm whether latent context presents a viable option. Empirical validation will be made using standard, benchmark contextual data sets from different domains.

1.2 Goals and Research Questions

The main purpose of this research is to find an alternative, potentially more informative representation to use instead of the current non-latent variety of contextual data within a CARS. Therefore we will aim at developing a method to extract latent context variables from non-latent context variables. This contextual information will afterwards be integrated within a RS, in order to confirm whether latent context presents a viable option. This system will be empirically evaluated on different data sets, each pertaining to a different domain. In order to assess latent context performance, it will be compared against CF and CARS State of the Art algorithms and baselines. A high level scheme of the goals can be observed in Figure 1.1.

The research questions identified were the following:

1. **Is the extraction of latent context from non-latent context variables through MF viable?** With this question the focus is on ascertaining whether the extraction of latent context is possible through standard, off-the-shelf methods. MF is a common recommendation algorithm, but using it to extract latent context is a novel application.

Introduction

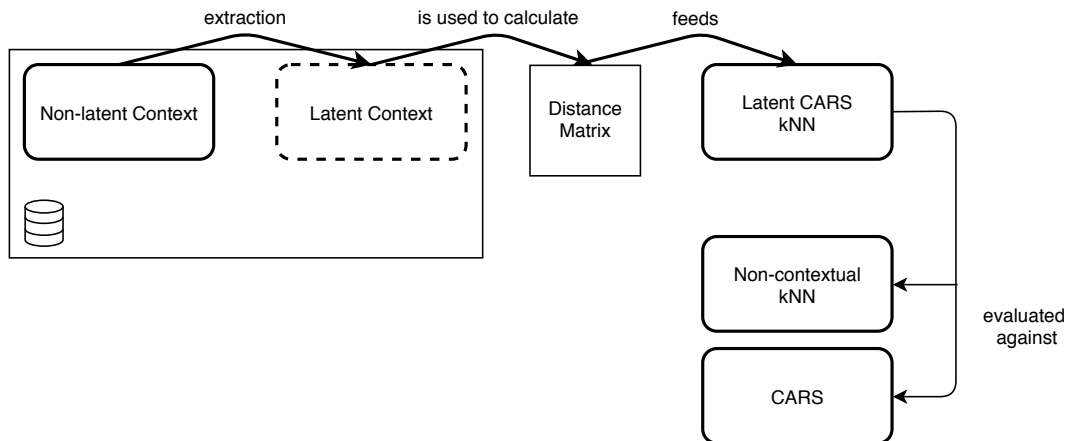


Figure 1.1: Illustration of the objectives for the project

2. **Can the quality of recommendations be improved by using latent context information obtained with an adaptation of MF?** The value of using latent variables for contextual recommendations will be put to test by evaluating the models created alongside other models from previous work [UBSR16]. The results will be evaluated using metrics such as Area Under the Curve (AUC) and Normative Discounted Cumulative Gain (NDCG), among others.

This project will then contribute to the exploration of latent contextual variables' value for improving the quality of recommendations. The contributions of this project are the following:

- **Adapting current methods to obtain latent context from non-latent contextual data.** The related work on this topic [UBSR16] described in section 2.4.5 uses different methods, so it will be interesting to see if alternative, simpler techniques are also viable. More specifically, this study will use MF, which is a widely used method for matrix decomposition [KBV09], but has yet to be applied to the extraction of latent variables.
- **Empirically compare the proposed method with a CARS approach and non-contextual approaches.** By the end of this study, some insights into the specific benefits and drawbacks of this type of variables are expected.
- **Propose a new CARS method to handle latent contextual data.** In order to integrate the previously extracted latent context information, a new CARS method will be developed. This method is based on precalculating distances between users in the latent space, which is less sparse, and then generating the recommendation on the original space. However, this is a general method, applicable with other methods for the extraction of latent context.

1.3 Dissertation Structure

This dissertation will be organised according to the following structure:

Introduction

- In chapter 2, the state of the art regarding RS and more specifically, Latent Context-Aware RS is presented, alongside other relevant information that frames the work to be developed.
- In chapter 3, the problem is presented, alongside the proposed solution. The approach and methodology taken will also be discussed and explained.
- In chapter 4, the results of the empirical study will be analysed and discussed. Some considerations about the data and setup will also be referred.
- Finally, in 5 the conclusions of this work will be drawn, with a brief summary of previous chapters. Also, some directions of future work will be suggested.

Chapter 2

Recommender Systems

RS are an ubiquitous constant in the average person's interactions with technology. Their main purpose is to help users navigate the huge amount of options available on entertainment services or other catalogues, pointing them towards items that should align more with their interests. RS are used in diverse areas to help with decision making, encompassing fields such as entertainment (including streaming services for music, television shows or movies), learning, books or even health and medicine. These types of systems are useful both for the end user and the platform where they are implemented. Assuming the user has limited time to spend, he will probably enjoy his time more and/or buy more products if he runs into personally interesting items. Essentially, RS can function as a personalised, automated sales assistant or guide.

2.1 Algorithms

RS work by collecting the preferences of the entire user base regarding the available items, whether through explicit methods like ratings, or through implicit methods, such as the time spent on a page, songs added to a playlist or movies watched. The information gathered is then used to calculate and predict a certain user's interest or rating for items. The way these predictions are made is what defines the different types of RS [AT05].

The main goal of RS is establishing a function R that estimates ratings. A formalisation for a non-contextual RS typically involves its set of users, $User$, and its set of items, $Item$. An example of a possible formalisation can be seen in Eq. 2.1.

$$R : User \times Item \rightarrow Rating \quad (2.1)$$

According to the paradigm they use to make recommendations, RS can be categorised as content-based filtering, collaborative filtering and hybrid approaches.

2.1.1 Content-based Filtering

To use content-based filtering, a user profile is created, describing the user’s likes, dislikes and past interests. Items are categorised and ascribed keywords, and are recommended if their keywords match the user’s interests. Product and user characteristics are often taken into account. Since these methods have their roots on information retrieval techniques, their main focus is finding what previously consumed items have in common. In the case of music, for example, the artist and genre would be considered. The goal is to recommend only similar items to the ones the user consumed in the past [Ado05].

The similarity can be calculated by traditional heuristics, such as TF-IDF, cosine similarity or by model-based techniques, like Bayesian classifiers, decision trees or artificial neural networks. Other methods include rule induction and Rocchio’s algorithm [BOHG13].

Content-based filtering sometimes implies collecting external information to complete the profiles, which can be troublesome [KBV09]. Since content-based filtering is mostly derived from information retrieval methods, it can struggle with automating the extraction of relevant information/tags for different types of items, including multimedia and text. Since this impacts the quality of recommendations, it also means that in practice, most of the content-based systems do not rely solely on this type of filtering [BOHG13]. Those approaches are classified differently, as hybrid systems. These will be discussed below in Section 2.1.3

2.1.2 Collaborative Filtering

Eq. 2.1 can also be seen represented in matrix format in Fig. 2.1. Having the current set of Users and Items, the objective is to estimate user ratings for items they did not rate yet. Using as guideline the notation in [EK19], users are denoted as $u \in User$, while items are denoted as $i \in Item$. Consequently, ui represents user u being recommended item i . When multiple users or items are required, subscripts such as u_1, u_2, \dots, u_n can be employed. If user u rated item i , this can be denoted by $r_{ui} \in R$, while the absence of rating from user u to item i can be represented by $r_{ui} \notin R$. Additionally, a rating predicted by the RS for user u and item i can be expressed as $r_{p_{ui}}$. For an example, in the case of Fig. 2.1, the system would generate $r_{p_{u_1 i_2}}$, $r_{p_{u_2 i_1}}$ and $r_{p_{u_3 i_3}}$.

	i_1	i_2	i_3
u_1	2	?	2
u_2	?	5	2
u_3	3	4	?

Figure 2.1: Matrix representation of a traditional Recommender System

Recommender Systems

Till We Have Faces: A Myth Retold Paperback – 14 Feb 2017
by C S Lewis (Author)
★★★★☆ 25 customer reviews

> See all 40 formats and editions

Audiobook £0.00 <small>Free with your Audible trial</small>	Library Binding from £20.56 <small>3 Used from £20.56</small>	Paperback £11.87 <small>4 Used from £5.96 8 New from £7.06</small>	Audio CD £29.95 <small>1 Used from £22.45 4 New from £24.94</small>
--	--	--	--

Want it delivered by **Monday, 22 July**? Order within **14 hrs 34 mins** and choose **Expedited Delivery** at checkout. [Details](#)

Note: This item is eligible for **click and collect**. [Details](#)

Customers who viewed this item also viewed Page 1 of 10

- The Complete C. S. Lewis Signature Classics: Boxed Set**
C. S. Lewis
★★★★☆ 50
Paperback
£35.99 ✓prime
- The Great Divorce (Cs Lewis Signature Classic)**
C. S. Lewis
★★★★☆ 121
Paperback
£6.62 ✓prime
- The Abolition of Man**
C. S. Lewis
★★★★☆ 27
Paperback
12 offers from £5.08
- Problem of Pain (C. Lewis Signature Classic) (C. S. Lewis Signature Classic)**
C. S. Lewis
★★★★☆ 60
Paperback
£5.61 ✓prime
- The Space Trilogy**
C. S. Lewis
★★★★☆ 43
Hardcover
£19.49 ✓prime
- Four Loves (C. Lewis Signature Classic) (C. S. Lewis Signature Classic)**
C. S. Lewis
★★★★☆ 59
Paperback
£7.99 ✓prime

Figure 2.2: Amazon collaborative filtering example

Collaborative filtering approaches are based on the principle that people who enjoyed similar things in the past will continue to do so in the future, much like how one has a higher chance of enjoying a movie recommended by a friend with similar taste. As such, users with similar past ratings are located in order to generate recommendations to each other. A well-known example of this type of RS is found on Amazon, through the section "Users who viewed item A also viewed item B", as seen in Fig. 2.2. In Eq. 2.1 it is possible to see a formalisation of this type of methods.

Collaborative filtering approaches can be classified as model-based, memory-based (also known as heuristic-based) or hybrid. While memory-based methods usually rely on existing user and/or item information to calculate similarity metrics, model-based methods use the available information to generate a model that will make its own recommendations. Memory-based approaches consist of neighbourhood algorithms, like k Nearest Neighbours (kNN) [PKCK12], which has user to user and item to item variations. In the user-based kNN the similarity is calculated between users according to their past ratings, while the item-based considers two different items instead. An illustrative scheme of how this algorithm works is observable in Fig. 2.3. The object to classify, which can be an user or an item, is represented by a question mark. The similarity is calculated between that object and the other objects in the database. Afterwards, the object is classified according to its k most similar neighbours. kNN relies heavily on similarity measures. Popular similarity measures include angular measures, such as cosine similarity and Pearson correlation [AJOP11]. Although simple and capable of providing good results, the kNN algorithm

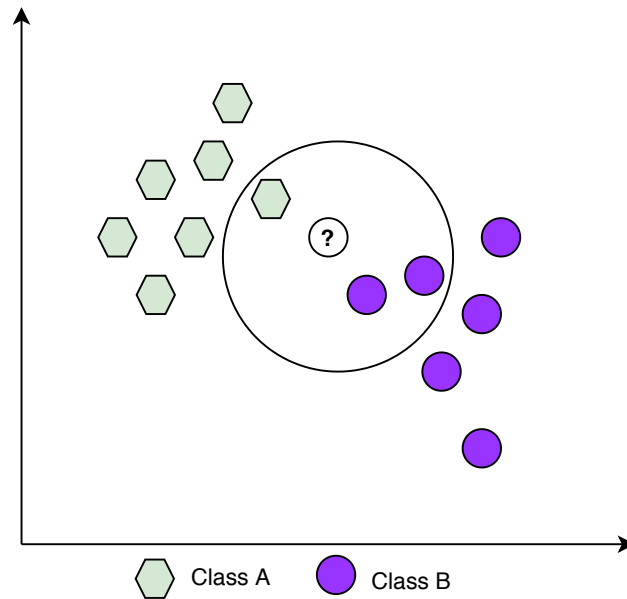


Figure 2.3: Schematic representation of the kNN algorithm

is vulnerable to sparsity and does not scale well [BOHG13]. Some model-based methods include probabilistic approaches, neural networks, MF, Restricted Boltzmann Machines clustering, among others. For an example, MF consists on characterising user and items through vectors of latent factors [KBV09]. These factors are inferred from item rating patterns, and in the case of music, as an example, can represent aspects such as genre, danceability or even other undecipherable traits.

Additionally, [Ado05] consider demographic filtering an extension of collaborative filtering that takes into account user information such as age, gender or nationality. This information can also be used to calculate users' similarity.

2.1.3 Hybrid Approaches

Hybrid methods combine both content-based filtering and collaborative filtering in order to get the best outcome possible, more specifically to avoid certain drawbacks of one of the approaches, like the cold start problem [Bur02]. Additionally, the author of [Bur02] also characterises hybrid systems according to the method they use to combine the different approaches. Seven different combinations are described:

- **Weighted** The different approaches produce different recommendations, but these are all aggregated into one single recommendation.
- **Switched** The system switches between the different approaches in accordance to some pre-established criterion.
- **Mixed** Different approaches produce different recommendations, which are then displayed alongside each other.

- **Feature combination** Different recommenders produce different features, which will feed a single recommendation algorithm.
- **Cascade** In this ordered process, a recommender first produces coarse recommendations which are then refined by another recommender.
- **Feature augmentation** One recommender produces outputs that are used as input features for the next recommender.
- **Meta-level** The model generated by one technique is used as input for another.

[BOHG13] present several ways of combining content-based filtering and collaborative filtering approaches. They separate four different approaches: A) content-based and collaborative based approaches are calculated separately and combined afterwards; B) content-based techniques are integrated into the collaborative filtering approach; C) both methods are integrated in the model; and finally D), where collaborative filtering techniques are incorporated in the content-based filtering method.

As alluded to in Section 2.1.1, one of the problems with content-based filtering is being limited to certain domains or having challenges with automated extraction of information or features about items. However, [TC00] created a hybrid model which switches between content-based or collaborative approaches in accordance with the quality of the results, which can adapt to the type of domain with more ease.

Other solutions involve including social or personal preference data. For instance, [YGK⁺06] use a Bayesian network with rating, content and latent user preferences data. Meanwhile, [LXG⁺14] use MF alongside item features to attribute more weight to “expert” opinions. These opinions are used to help users with a short past history of items.

A good overview of this type of approaches can be found on [BOHG13] and [VSDCT18].

2.2 Common Issues

The different types of RS mentioned in the previous sections all present their own set of challenges or issues. This section will go over each of the most significant ones individually.

Cold Start

The cold start problem, also referred to as new user or new item problem in [Ado05], is related to the amount of data necessary to jump-start the system. Some of the content-based filtering approaches require the user to rate some items before they can match his preferences to the rest of the items. For example, an entertainment streaming service might ask a set number of favourite movies or shows, upon the creation of an account. By doing this, the system immediately has some basic information about user interests that can be used to make recommendations.

Collaborative filtering systems are usually more affected by this problem, since they require the user to rate some items in order to find the users most similar to him and be able to make

recommendations. Additionally, a new item that was just added to the system will need to be rated by several users in order to be suggested to other users. A solution to this issue would be adding those item features to the recommendation process, including content-based techniques through a hybrid approach. This would combine the strengths of content-based filtering and collaborative filtering. Yet another option is using implicit feedback such as time spent on a certain item page or items downloaded to gather information about what the user prefers [MDW⁺12]. Another possibility is incorporating social network information in the recommendation process. For example, if a user has just joined and therefore has not rated any items yet, it is possible to suggest him some items that his friends or neighbours liked. Some examples of these approaches are summarised in [VSDCT18].

Overspecialisation and Diversity

Ideal recommendations should be in line with the user preferences, but should also be diverse enough not to saturate him. An example of this would be only recommending previously heard artists to a music streaming platform user. This can be a problem in a pure implementation of content-based filtering, since recommendations will consist of previously consumed content or items related to that content. Other items are usually not recommended, and the system sticks to what was certified relevant in the past [BOHG13, AT05].

Serendipity

In order to keep the system interesting to use, it is necessary to keep the recommendations fresh. This implies suggesting new items with a varying degree of randomness, allowing the user to choose items outside his safe zone. As an example, it would not be surprising to suggest a famous 80s band to a user who mostly listens to music of that period, and therefore that recommendation would not be particularly good. It would be interesting, however, to point him to a current indie band who is very inspired by the 80s. This seemingly random find would be more valuable to someone who already has a good grasp of the 80s artist scene and wants to expand his horizons [AT05, VSDCT18].

Sparsity

Rating sparsity is another of the problems that affect RS. Essentially, sparsity means the amount of ratings is not homogeneous across the entire catalogue of items. Additionally, it can mean there are few ratings comparatively with the the amount needed to make recommendations [Ado05]. In practice, even very active users have not rated the entirety of films or books available on a certain website. This would mean that using collaborative filtering, lesser-known films would never be recommended since only few users have rated them, even if those ratings were consistently high.

Another problem of sparsity in collaborative filtering is the inability to provide good recommendations to users who have a more niche set of interests, since these will naturally have less peers with similar interests [AT05].

Scalability

RS can be integrated in platforms with a large amount of items and/or users. These parameters can range from thousands to even millions in certain cases, such as in music recommendation [AT05]. This implies RS should be able to process these levels of information in an efficient manner. To attain this, implementations have been developed using parallel computation. One popular library that is capable of such parallelism in Machine Learning is TensorFlow¹. More specifically, TensorFlow allows for parallelism both for multiple cores and threads. Other options for scalable computation involve using the power of cloud computing [YL10].

[CKH⁺16] implemented a recommendation model using deep neural networks in TensorFlow. Other approaches include [PLS16], who developed an hybrid, parallel method for collaborative filtering on Apache Spark, a big data processing framework, through dimensionality reduction and clustering. Another option is the use of cloud computing platforms like Hadoop to tackle this issue, as done by [ZS10]. This last approach used Map-Reduce for parallelisation, a programming model that allows automatic execution on a large cluster of computers, allowing better scalability [DG08].

Privacy

As of late, privacy has risen to the top of the list of concerns of many consumers regarding information systems and social networks. It is therefore important for ethical reasons to make sure RS use information with users' consent and that data is subject to treatment when needed. This will increase users' trust in the system [VSDCT18].

2.3 Evaluation

The authors of [GS09] distinguish three types of recommendation systems, corresponding to different tasks. The first task consists of recommending a list of equally enjoyable or interesting items to the user, which require no ordering. The second consists on optimising the utility, which can be revenue or sales, for example. Finally, the third group consists on predicting user ratings for certain items. This project falls in the first and last categories. These categories should be taken into account when selecting the evaluation metrics. The metrics used to evaluate RS and the quality of their recommendations can be classified according to what they measure or are an indicator for accuracy or usefulness and usability [MDW⁺12]. Additionally, metrics should be selected according to the type of system.

Usefulness can be measured through surveys, focus group testing, trial experiments, A/B testing and other ways to collect user feedback. Usability metrics are useful for figuring out the perceived quality of the recommendations through the lens of the end user. This also usually involves collecting information about other usability characteristics of the system, such as responsiveness, transparency and trustworthiness.

¹<https://www.tensorflow.org/>

To evaluate the numeric quality of the predicted ratings, accuracy metrics are required. Since the ultimate goal of recommendation systems is to predict numeric values (ratings of items for different users), akin to a regression problem, statistical accuracy metrics such as Mean Average Error (MAE) and RMSE can be used to evaluate the difference between the predicted value and the real, expected value [Ado05]. Assuming there are n observations in the set and that e_n represents the error for each observation and prediction, MAE and RMSE can be calculated through Eq. 2.2 and Eq. 2.3 respectively [CD14]. The error $e_{r_{ui}}$ for a given rating r_{ui} is calculated through the difference between the predicted rating, $r_{p_{ui}}$ and the observed rating, r_{ui} , which can be seen in Eq. 2.4.

$$MAE = \frac{1}{n} \sum_{i=1}^n |e_i| \quad (2.2)$$

$$RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^n e_i^2} \quad (2.3)$$

$$e_{r_{ui}} = r_{p_{ui}} - r_{ui} \quad (2.4)$$

Even though both MAE and RMSE measure the average magnitude of the error in a set, they have some key differences. For instance, MAE uses the absolute value of the error, which means it does not consider if it is positive or negative. Additionally, all the errors have the same weight. Meanwhile, RMSE attributes more weight to the most significant errors, since the error is squared. This means that RMSE is more sensitive to outliers. Besides, these measures are adequate to express different error distributions. Since MAE fits uniformly distributed errors, RMSE is better suited when the error has a normal distribution [CD14].

Accuracy metrics are relatively simple to calculate and handy to establish comparisons, but there has been discussion regards their pertinency. This stems from the fact that in real-life scenarios, it is not necessary to exactly predict the rating a user would give to a given item. Instead, it is more compelling to be able to distinguish between items the user is interested in *versus* items the user has no interest in [MRK06]. Classification accuracy metrics can be used to measure whether the system decided correctly if the item was relevant or not, on a binary scale [HKTR04]. These metrics include precision, recall and F-measures. Precision and recall measure the rate of recommended items that are relevant and the rate of relevant items that were recommended, respectively [GS09]. However, these two metrics can be misleading when used individually, since they have a trade-off relation. A model with very high precision mostly makes recommendations that are very relevant to the user. However, increasing precision can lead to lower recall, which means many relevant recommendations are left out. Conversely, a model with high recall recommends most of the relevant items. Still, increasing recall can lower precision, meaning many irrelevant recommendations are also made. F-measures were created out of this problem, combining both metrics into one [AJOP11]. Eq. 2.7 shows how to calculate F_1 , or F-score, consisting on the harmonic average of precision (Eq. 2.5) and recall (Eq. 2.6). In order to calculate these measures, some concepts must be defined: relevant items that were correctly classified as such are

Recommender Systems

represented by TR ; irrelevant items that were correctly classified as irrelevant are represented by TI ; relevant items that were misclassified as irrelevant are FI and lastly, irrelevant items that were classified as relevant are FR .

$$Precision = \frac{TR}{TR + FR} \quad (2.5)$$

$$Recall = \frac{TR}{TR + FI} \quad (2.6)$$

$$F_1 = 2 \times \frac{Precision \times Recall}{Precision + Recall} \quad (2.7)$$

Metrics such as Receiver Operating Characteristics (ROC) and Swet's A measure present an alternative. The main goal of ROC is measuring how well the system can distinguish between relevant and irrelevant recommendations. The area under the ROC curve (AUC), also known as Swet's A measure, can provide the probability of the system being able to pick correctly between two randomly selected items, one from the set of relevant items, and one from the set of irrelevant items [HM82]. For an example, an AUC value of 0.5 would represent a random recommendation. Still, with this measure swaps at the same distance have similar impact regardless of their position in the ranking. Additionally, these measures require the nature of the recommendations to be binary: recommendations can either be relevant or irrelevant, interesting or uninteresting. This is not adequate when preference has a wider range of possible values.

Therefore, in RS it is also interesting to measure the capacity of the system ordering a list of recommendations by their relevancy, in a granular, non-binary manner. It is useful to show the most relevant recommendations first so the user can quickly find interesting items, without losing interest. Rank accuracy metrics, with roots in Information Retrieval, include precision at K ($P@K$), Normalised Cumulative Discounted Gain (NDCG), average rank and hit at K (Hit@K). These metrics are based on the assumption that the errors that matter are the ones in the limited list (top-K items) that is shown to the user. $P@K$ measures the number of relevant items in the top-K items list, where K is chosen by the researcher. This metric has little value by itself, since it does not account for the precision of the complete data. Even so, it can be used to establish some relative comparisons [Ste13]. Meanwhile, Hit@K is calculated as the number of hits in the top-K recommendations. A hit is counted when an item in the top-K recommendations is also present in the ratings of the user [YSGL12]. Aggregating this metric over all users results in recall at K ($R@K$). Another interesting measure is average rank of the correct recommendation in the list. If the item corresponding to the hit is close to the top, it can mean the user did not have to search a lot to find the match. On the other hand, NDGC evaluates the top-K list items taking into account their position. By associating a certain discount according to the item position on the list, it is possible to account for user interest in items that show up first. The formula to calculate the Discounted Cumulative Gain (DCG) can be seen in Eq. 2.8, considering N users and an ordered list of K items. g_{ui_k} represents the gain of a user u when recommended item i with position k . The logarithm base, y , is usually chosen between 2 and 10 [SG11]. NDGC is the normalised version

of DGC.

$$DCG = \frac{1}{N} \sum_{u=1}^N \sum_{k=1}^K \frac{g_{uik}}{\max(1, \log_y k)} \quad (2.8)$$

The metrics presented so far evaluate the utility of a ranking. Nevertheless, the ranking can be evaluated based on whether the ordering is correct or not. To do this, the generated ranking can be correlated with a reference ranking. A reference ranking, such as a ranking provided by the user is required. Metrics for this type of correlation are Spearman's rho, Kendall's tau coefficient and Normalised Distance-based Performance Measure (NDPM). The main difference between Spearman's rho/Kendall's tau and NDPM regards the way they deal with ties [SG11]. Spearman's rho and Kendall's tau are better suited to cases where the reference ranking is complete, including all user preferences. Thus, a tie between items in the user ranking means indifference, which means they should be ranked equally. NDPM, on the other hand, is adequate to systems where items will appear ordered and ties are not allowed. Therefore, the evaluation has to allow for tied items to be ranked differently. This measure cannot evaluate the value predicted, only the list order [HKTR04]. The formula to calculate NDPM can be seen in Eq. 2.9. C^- represents the items that the system ranked incorrectly, while C^+ represents the ones correctly ranked. C^u represents the number of items of the reference rank that did not tie and C^{u0} is the number of pairs that tied in the system ranking but not in the reference ranking.

$$NDPM = \frac{C^- + 0,5C^{u0}}{C^u} \quad [SG11] \quad (2.9)$$

[HKTR04] All of these rank correlation metrics have the disadvantage of penalising shifts at the bottom and top of the list equally.

Finally, coverage can be measured to know the percentage of items or users of the system that are effectively used to provide recommendations [IFO15]. To measure the coverage of items, Gini Index and Shannon Entropy are used. User coverage can be evaluated through offline testing the amount of profile information required to make recommendations, according to the type of RS [SG11].

2.4 Contextual Recommender Systems

People's consuming habits are influenced by the context they are inserted in [Ado05]. Using as an example the case of film recommendations, it is understandable how a person might want to watch a different type of film during a weekend night with family comparatively to a weekday afternoon with friends. Therefore, it became important to add contextual dimensions to traditional RS, which then became known as CARS [Ado05, BOHG13].

Therefore, the formalisation from Eq. 2.1 can be adapted to include a *Context* dimension, as seen in Eq. 2.10.

$$R : User \times Item \times Context \rightarrow Rating \quad [AT11] \quad (2.10)$$

2.4.1 Definition of Context

Context can be defined as the set of conditions that describe the situation of an entity [ADB⁺99]. Essentially, context describes a situation and the environment a device or user is in. Context includes time, human factors, such as the user, social environment and the task, but also physical factors, such as conditions, infrastructure and location [SBG99].

In terms of acquisition, context can be classified as implicit or explicit. Explicit context is acquired through direct user input, such as inserting a user’s current location or social company. Meanwhile, implicit context is gathered automatically, by using the device’s GPS location, for example. Implicit context can also be inferred through the analysis of user interactions or other sensor data [MDW⁺12], using processes such as Activity Recognition. An example of this would be inferring whether a person is running or driving through their mobile sensor data [DPDM14]. In both types of approaches the meaning of the variables is easy to understand.

2.4.2 Non-latent Context versus Latent Context

As previously mentioned, non-latent context is all the context that is captured or inferred through either implicit or explicit means. However, including these variables also increases dimensionality in the recommendation matrix, which in turn further complicates the sparsity problem [Ung15].

For this reason, attempts at maintaining contextual information while tackling dimensionality and privacy have leaned towards the usage of latent context variables. Latent context variables are inferred from non-latent contextual data. However, unlike traditional, non-latent variables, latent variables’ meaning is unknown. As the name indicates, latent context consists of hidden contextual variables which can be extracted from non-latent context through different techniques [UBSR16]. This hidden context has the potential to reflect only the most significant contextual variables, making it possibly more accurate and concise. A diagram that summarises this classification can be seen in Fig. 2.4.

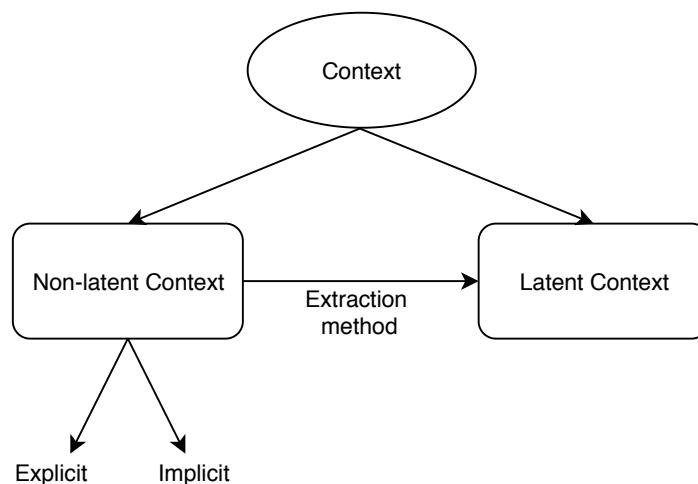


Figure 2.4: Categorisation of context according to acquisition and latency

Recommender Systems

CARS techniques	CARS algorithms	Used model (in case of contextual modelling techniques)
Contextual pre-filtering	(Herlocker and Konstan, 2001)	Not Applicable
	Reduction based approach (Adomavicius and Tuzhilin, 2001) (Adomavicius et al., 2005)	
	Item splitting technique (Baltrunas and Ricci, 2009) (Baltrunas and Ricci, 2014)	
	User splitting technique (Said et al., 2011)	
	User Item splitting technique (Zheng et al., 2013)	
	Distributional-Semantics Pre-filtering (Codina et al., 2016)	
Contextual post-filtering	Weight and Filter post-filtering methods (Paniello et al., 2009)	Not Applicable
Contextual modelling	Context aware SVM (Oku et al., 2006)	Support Vector Machines
	Multi-verse Recommendation (Karatzoglou et al., 2010)	Tensor Factorization
	Context Aware Matrix Factorization (Baltrunas et al., 2011)	Matrix Factorization
	Context Aware Factorization Machines (Rendle et al., 2011)	Factorization Machines
	TFMAP (Shi et al., 2012)	Tensor Factorization
	iTALS (Hidasi and Tikk, 2012)	Tensor Factorization
	Gaussian Process Factorization Machines (Nguyen et al., 2014)	Gaussian Process
	Contextual SLIM (Zheng et al., 2014) (Zheng, 2014) (Zheng et al., 2015)	Sparse Linear Method
	Contextual Operating Tensor for CARS (Liu et al., 2015)	Matrix Factorization

Table 2.1: Classification of context-aware recommendation algorithms by [LBK17]

[Ung15] identifies three steps to his process of extracting latent context:

1. **Gathering raw data from sensors and other sources** Data from mobile sources is gathered, such as accelerometer, GPS and applications currently in use.
2. **Using feature engineering to extract features from raw data** Several statistics are calculated from the previous collected raw data, with the goal of representing it in a simple and informative manner.
3. **Training unsupervised model to extract the latent contextual variables** With the goal of extracting latent contextual variables, an autoencoder neural network is trained to find relations and patterns in the initial data.

Other methods for the extraction of latent variables involve topic modelling techniques, SVD, MF techniques or PCA [UBSR16]. Topic modelling techniques include latent Dirichlet allocation (LDA) [LX13], and MF techniques include Biased MF [LW15].

2.4.3 Algorithms

Decomposition methods are widely used in CARS. These include traditional dimensionality reduction techniques, such as MF techniques, Singular Value Decomposition (SVD), or Principal Component Analysis (PCA) [AT05, AJOP11]. Other approaches like Support Vector Machines (SVM), Factorisation Machines (FM) and Tensor Factorisation (TF) are also used. Of these, MF techniques are popular for not being demanding in computational terms, for being scalable and offering good results [KBV09].

The authors of [LBK17] summarised and classified the most frequently used CARS algorithms in Table 2.1. The most relevant ones will be explained and discussed here.

Three main ways of including contextual information in the recommendation have been identified in [AT05], each with its own set of advantages and challenges. These are also used by the authors of [LBK17] to classify contextual algorithms:

- **Contextual pre-filtering** Using this approach, context is first considered in order to eliminate ratings that are not relevant for the specific situation. The recommendation algorithm is only applied afterwards, using only ratings with relation to the user's context.
- **Contextual post-filtering** Using this approach, recommendations are generated first on the data without influence of contextual variables. Afterwards, the recommendations are adapted (filtered) for each user's context.
- **Contextual modelling** This approach consists of including the contextual information directly in the recommendation algorithm. Usually this implies an increase in dimensionality, thanks to the addition of the contextual information to the model.

Regarding contextual pre-filtering for example, the item-splitting method works by splitting each item into varied, different virtual items depending on the possible contexts. User-splitting, on the other hand, separates user profiles into different profiles for different contexts, which are then used for the recommendations [AT11].

In contextual post-filtering, there are two approaches, Weight and Filter. The Weight method attributes a weight to a recommendation according to the probability of it being relevant in the corresponding context, and uses these weighted values to reorder the recommendations. Meanwhile, the Filter method removes items with little probability of being relevant to a certain context [AT11]. Essentially these approaches consist in filtering the generated recommendations and eliminating items with a probability below a certain minimum value. Alternatively, recommendations can be ranked according to their weight and likeliness of being relevant.

Finally, in contextual modelling, multidimensional algorithms are used [MDW⁺12]. These approaches can also be classified as heuristic or model-based. Heuristic methods incorporate context in the calculations of the recommendations, and include neighbour-based approaches which were adapted for the extra dimensions. This usually means adopting n-dimensional similarity metrics that take into account more than just users or items. Meanwhile, model-based approaches include for example SVM, a supervised learning ML technique that learns how to classify an item as interesting or uninteresting. Another possibility is Context-Aware MF (CAMF), which introduces more parameters to 2D MF to model the context. CAMF presents good scalability and accuracy [LBK17]. However, it is used often alongside SVD to aid in the matrices decomposition. The SVD implementations are often adaptations such as SVD++ to better deal with sparsity. Lastly, another option is using TF, which calculates the recommendations as the product of the latent factors of the separate items, users and context matrices [VSDCT18]. However, as more contextual variables are introduced, the number of model parameters also increases. Besides, with CAMF using a linear, reduced set of parameters it is possible to obtain equivalent results [UBSR16].

Both pre and post-filtering approaches have the drawback of requiring supervision alongside each step of the process and of requiring a sufficient amount of ratings in all the possible contexts

in order to overcome sparsity. Another problem with these methods is that in order to decide which method better suits the case, either pre-filtering or post-filtering, it is necessary to test and compare several implementations of each [PTG⁺09]. Contextual modelling has the advantage of integrating context information directly into the model. At the same time this can also create more sparsity thanks to the number of variables included, which can prove demanding [UBSR16].

However, in all presented approaches, the inclusion of contextual variables leads to an inevitable increase in dimensionality and, consequently, sparsity [UBSR16]. Privacy is also a pressing concern, seeing as information about an individual's context can be highly personal and its collection and usage should be made transparent [VSDCT18].

2.4.4 Literature Review of Latent Approaches

So far, the most widely used algorithms were discussed according to their technique for including contextual information. However, this summary only considered the aforementioned technique and, in contextual modelling approaches, the used model.

With this in mind, a survey of specific contextual approaches was conducted. Articles selected were analysed in terms of the type of data used, the field of application, the type of context (latent or non-latent), the techniques and type of model employed and evaluation measures. Results are summarised in Table 2.4.4, from the most recent to the least recent. The intention was to have a comprehensive overview of the evolution of the methods used across a recent timeline.

Recommender Systems

Appr.	Data type	Field	Context	Representation	Technique	Algorithm
[PZ18]	Playlist name, track analysis, user listening history	Music	Latent	Matrix	Contextual modelling	SVM, FM, MF using MCMC
[ARD ⁺ 16]	Social tags	Music	Non-latent	Tensor, matrix	Contextual modelling	Normalized frequency matrix, cosine similarity
[UBSR16]	Mobile sensor data	POI	Latent	Matrix	Contextual modelling	Autoencoder, PCA, MF
[LW15]	Demographic, mood, time	Movies, books, music	Latent	Matrix	Contextual modelling	Biased MF, SGD
[SLH13]	Mood tags, plot keywords	Movies	Latent	Matrix	Contextual modelling	JMF, latent similarity
[LHZ13]	Social data, time	Movies	Latent	Matrix	Contextual modelling	MF
[HMB12]	Sequence of songs, social tags	Music	Latent	Matrix	Contextual modelling	LDA, GSP
[SBCH12]	Location, form of transportation, mood	POI	Non-latent	Decision tree	Contextual pre-filtering	Hidden Markov Model, Bag of words
[KABO10]	Companion, time, place, hunger level, demographics	Movies, food	Non-latent	Tensor	Contextual modelling	Tensor factorisation
[Ado05]	Time, place, companion	Movies	Non-latent	Tensor	Pre-filtering	Cosine similarity, sum of products

It is possible to see that the fields of application are varied, from music to POI and movies. In fact, 4 out of 10 articles used data related to music, while 5 used movie related data. Some used data from more than one field of application.

The type of representation usually alternates between matrices and tensors. Nonetheless, one of the articles analysed used a decision tree [SBCH12]. In fact, matrix and tensor decomposition techniques are the most widely used to tackle the issues such as scalability and sparsity [Sym16]. By representing user, item and context information in matrix form, it is possible to apply matrix factorisation techniques. This process will reveal latent factors that associate user and item information, for an example. However, in order to represent ternary relations directly, structures

such as tensors can be used [Sym16]. Using tensors usually requires different algorithms, since algorithms used for matrices can no longer be applied to this type of representation. Even though tensors allow for more complex representations, they also require more computational resources as the rank increases [FO17].

It seems as though pre-filtering and post-filtering approaches have been less used with time, giving way to contextual modelling. Even though contextual modelling involves some increase in dimensionality, its use seems to be increasing since it is advantageous to include context information directly in the process. Unlike pre and post-filtering approaches, which only use contextual information to filter items rated for a given situation, contextual modelling integrates the context in the recommendation system. This means it is possible for the model to learn to recommend taking into account both the item and context. The algorithms used are also varied, with MF being used in different variations.

Most approaches are model-based, with only two of nine being heuristic-based instead. Additionally, the cases in which an heuristic-based approach was used were less recent. Model-based approaches also make more sense seeing as contextual modelling is prevalent.

In terms of evaluation metrics, prediction metrics such as MAE and RMSE are used, as are coverage metrics such as Precision and Recall. Additionally, ranking metrics such as P@K and Hit@K are also used.

2.4.5 Related Work

There has been some previous work in this research area, more specifically an approach in [UBSR16]. This article's main goal was to present a solution for the inclusion of latent context in a recommendation algorithm, improving accuracy and addressing privacy and usability concerns. The aforementioned article showcases a new approach to integrating context in recommendations, using contextual modelling. Contextual information is represented by data collected from mobile sensors, from which only a small group of the most appropriate features is selected through feature engineering.

The latent context is extracted from hidden patterns contained in the raw data, through unsupervised Deep Learning (DL) techniques, more specifically an autoencoder, and through statistical procedures such as PCA. Afterwards, the authors describe an implementation of a CARS using the extracted latent context, and another hybrid one that uses latent context in conjunction with non-latent context. An overview of their method can be seen on Fig. 2.5.

These models were evaluated against a non-contextual algorithm using MF and a simple, non-latent contextual model from [BLR11] as baselines. The evaluation was carried out by analysing the use of an Android application that provided recommendations about points of interest. User feedback was captured alongside their contextual information through the sensors. Metrics used were RMSE, NDCG, Hit@K, and average rank of the selected recommendation.

This research suggests that latent contextual variables are a viable alternative to non-latent context. However, other methods for latent context extraction have yet to be explored, and the value of these variables is still not fully understood.

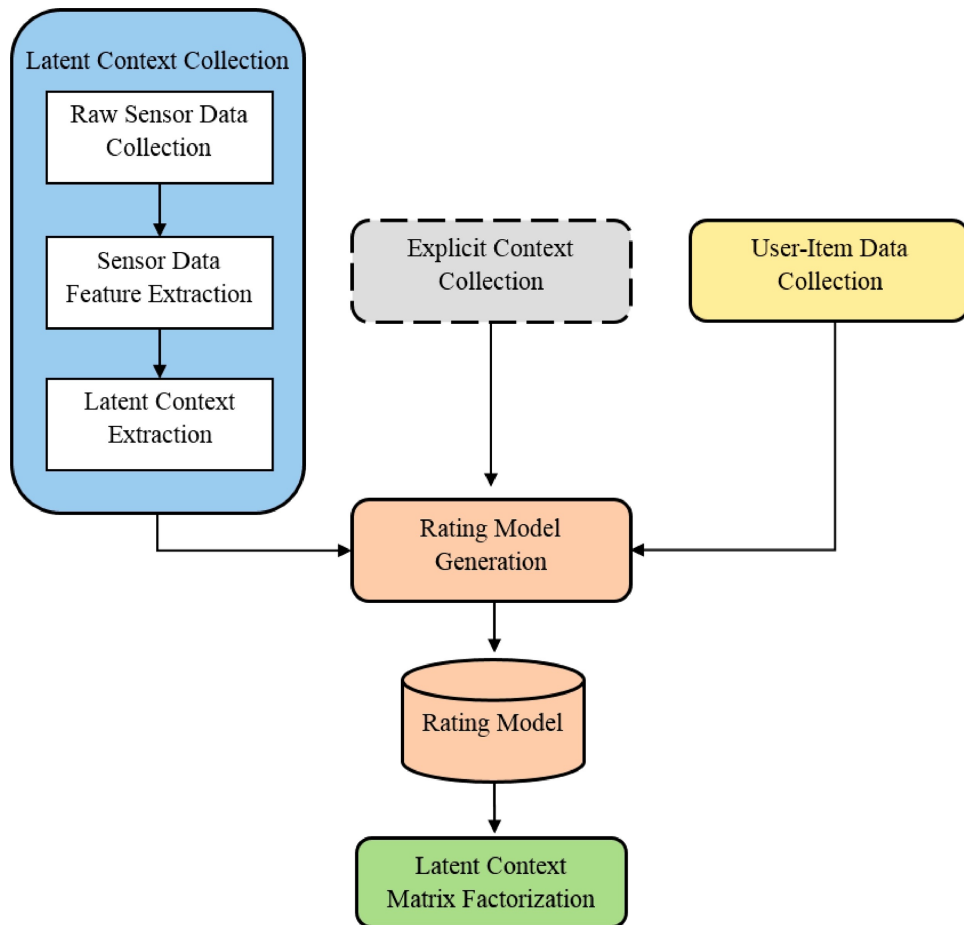


Figure 2.5: Method used in related work [UBSR16]

2.5 Summary

From the study and review of the current developments in the area of RS and, more specifically, CARS, it seems there is some room for further study and improvement. Some issues inherent to RS, such as dimensionality, sparsity and privacy are still pressing and concerning even shifting to contextual RS, and therefore require new solutions. Latent contextual variables are one of the proposed solutions for these issues, but their role and value is still not completely established or clear.

Moreover, there has been some previous work related to the one proposed here. However, even though the general goal of that approach is similar, the methods to attain it proposed here are different. It is our goal to find alternatives which allow to tackle the issues of time and space complexity and which do not necessarily rely on Machine Learning (ML) techniques. Besides, the proposed approach will be evaluated using different data sets from various domains.

Recommender Systems

Chapter 3

A Latent Context-aware Recommender System Approach

This chapter details the method developed for latent context-aware recommendations. Afterwards, the empirical setup will also be explained.

3.1 Approach Overview

A simple overview of the method used in our approach consists of the following steps: the data is projected into a latent context space; the nearest neighbours of the test cases are then found; and the representation of those cases in the original space is then used to make recommendations. We opted for a contextual modelling approach, since this has the advantage of including context directly in the generation the recommendations. Moreover, in [PTG14] different approaches such as content-based, collaborative filtering and hybrid are evaluated and compared regarding accuracy and diversity. The results revealed that contextual modelling obtained consistently good results, even if not always the absolute best solution. In order to have a model-based method, translating between latent and non-latent dimensions is required. This is difficult since it would be necessary to invert the matrix of latent dimensions obtained through MF. However, the inversion operation requires a square matrix, which is limiting. In order to address this concern, we opted for a memory-based approach instead. As explained in Section 2.1.2, memory-based methods generalise to new users, items and contexts by using a similarity measure and comparing unknown occurrences to the known examples. This means the system does not build a model for the data, but instead finds the most similar labelled user or items and uses their information to make a prediction [RESK15]. Besides, collaborative filtering approaches are still very popular and widely used [PKCK12].

Furthermore, instead of implementing a new method for extraction of latent context, we opted for applying Matrix Factorisation. This decision was made since the main goal of this project was to use off-the-shelf methods to learn latent context.

3.1.1 Context-aware Recommender Systems

As previously defined in Eq. 2.10, in CARS each User, Item and Context combination are associated with a different Rating. This means that the most direct way to represent this type of data is with a three-dimensional tensor, as seen in Fig. 3.1.

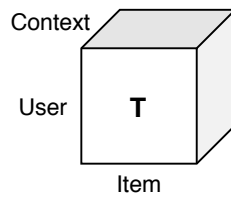


Figure 3.1: Tensor representation of contextual data

However, CARS data can be approximated by a matrix, preserving the tensor information. But in order to approximate this three-dimensional association in 2D format, it is required that all the possible user-item-context combinations are represented.

For an example, we have defined context in section 2.4.1 as the variables that characterise the environment of a user. Therefore, it is important to maintain the intrinsic relation between each user-item pair and its context. This means any given user-item pair should be able to have more than one possible context associated with him, as seen in Eq. 3.1. For an example, user John can consume movie A in different times (during the week or during the weekend), or with different company (friends, family or alone). Therefore, the possible combinations are $(John, A, week)$, $(John, A, weekend)$, $(John, A, friends)$, $(John, A, family)$, $(John, A, alone)$.

$$Context = f(User, Item) \quad (3.1)$$

To safeguard these conditions, the matrix approximation in Fig. 3.2 was used. In this representation, context variables still depend on the user-item combination. More specifically, the rows of the matrix embody all the possible contextual variables. In turn, the columns identify all the possible user and item combinations. This can be seen formalised in Eqs. 3.2,3.2,3.2.

$$\begin{aligned} M_{m \times n}, \\ m = user_item, \\ n = context_conditions \end{aligned}$$

This representation was chosen for its convenience and simplicity for the rest of the process.

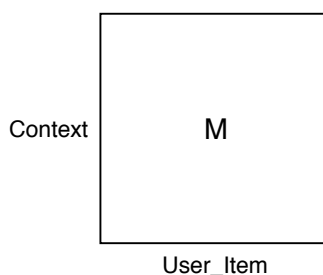


Figure 3.2: Matrix approximation of contextual data

3.1.2 Latent Context Extraction

In addition to the notation established in Chapter 2, some elements more specific to this problem will be introduced. Since the chosen extraction algorithm is MF, some concepts about how it works will be presented in this subsection.

Eq. 3.2 refers to the general process commonly known as MF. This algorithm decomposes a matrix M into two different matrices, P and Q .

$$M = P \times Q \quad (3.2)$$

In case the matrix includes only User, Item and Ratings the decomposition is translated by Eq. 3.3. The original matrix M is decomposed into two matrices that represent user factors (UF) and item factors (IF), respectively [KBV09]. The way this algorithm does this decomposition is essentially by projecting the original matrix M to a latent factor space. In this latent space, the algorithm characterises users and items on factors learned from user feedback [KB15]. User feedback can be explicit, as in the case of ratings, or implicit, such as browsing history or mouse movements [KBV09]. Each item will then have a corresponding matrix IF that quantifies the factors that match its characteristics. Meanwhile, each user has its respective matrix UF that estimates the appeal of items according to their factors for that user [KB15], as seen in Eq. 3.3.

$$M = UF \times IF \quad (3.3)$$

Consequently, the rating r of item i by user u can be given by the scalar product of UF_u and IF_i .

Nonetheless, in this approach we are not using this algorithm for rating prediction, but instead for the mapping of the original matrix to a latent space dimension. Adapting the approach to our contextual matrix, the process can be translated by Eq. 3.4.

$$M = CF \times UIF \quad (3.4)$$

In this case, CF and UIF consist of two matrices, corresponding to context factors and user-item factors, respectively. Matrix UIF will be a matrix with no sparsity, since every user-item combination will have an associated interest in each context, according to its latent factors. For

an example, given three contexts A , B and C , the combination *userJohn – productWizardOfOz* will have three different ratings, one for each of the contexts. These measures of interest will then be used as ratings. Since these ratings result from latent context factors, they can be considered *latent contextual ratings*. It is also worth noting that the number of lines in the *UIF* matrix can be parameterised, by setting the number of factors. This means this process allows customisation of the number of contextual variables.

These concepts and decomposition are the key to the extraction of latent context.

3.2 Empirical Evaluation

The method is divided into two components: extraction of latent context and generation of recommendations. The main components and flow can be seen in Fig. 3.3. In the aforementioned figure, the two main processes are represented with an oval shape. The two components will be explained in the following sections.

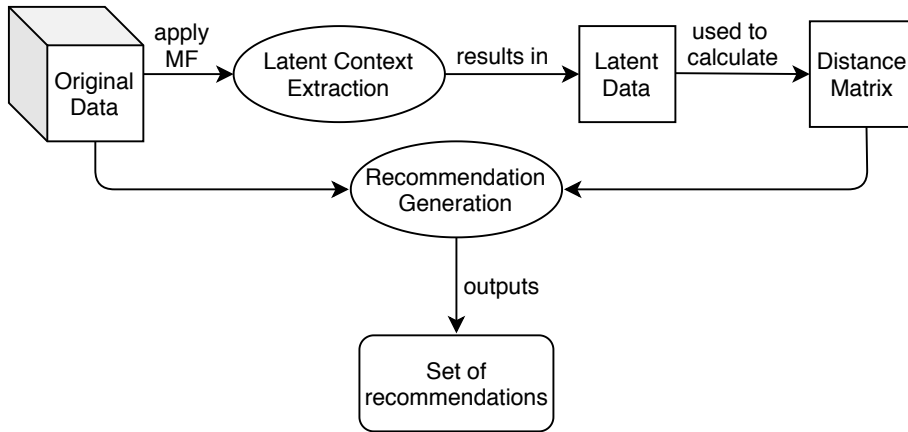


Figure 3.3: High level scheme of the developed method

3.2.1 Latent Context Extraction

The first part of the approach consists in the extraction of the latent contextual variables by Matrix Factorisation, more specifically SVD++ in our implementation. However, our method is independent of the type of MF used. In Fig. 3.4 ([3]) it is possible to visualise this step of the process of extraction through decomposition techniques. First, the data is split into train and test sets, respectively T_{train} and T_{test} . Some of the ratings in the test set are hidden and afterwards the data is aggregated into one set again. Next, the full set with some hidden ratings, T_{hid} , containing User-Item-Context data can be approximated by the User_Item-Context matrix M for these operations. This matrix is then decomposed in two other matrices, in the operation equivalent to Eq. 3.4, through a MF method. Since we can define the parameters of the process, in this approach we will ensure the number of latent contexts, C' , will always be inferior to the number of original contexts, C . The reason for this decision is the simplification of the problem and reduction of dimensions.

The result of this operation is matrix $\langle Q \rangle$, filled with latent contextual ratings which are used afterwards to calculate the similarities between users in the next step.

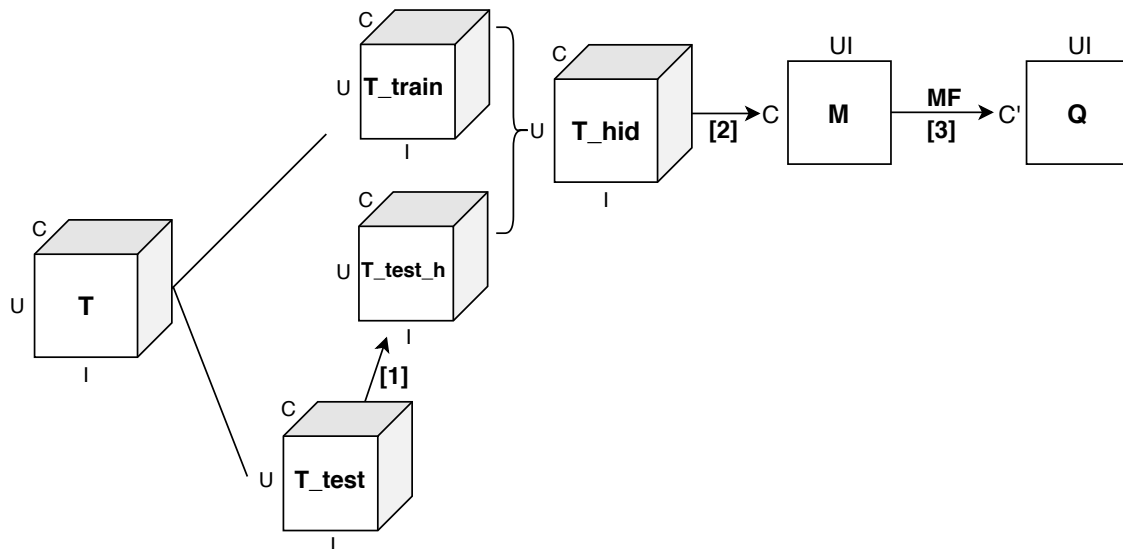


Figure 3.4: Process for latent context extraction

3.2.2 Latent Context-aware Recommender System

After the latent context extraction, the similarities between the users are calculated using a similarity measure. In our implementation, we used Pearson. This similarity measure assumes a linear relation between variables, while also adjusting for variations in how users rate along the scale [IFO15]. All the users are projected to a latent context space, where the similarities between them are calculated. Afterwards, for a given active user, a recommendation is computed using the data in the original space.

The recommendations are generated using a traditional collaborative filtering algorithm, kNN. However, our approach differs from common kNN methods regarding the similarity matrix. The similarity matrix used in Fig. 3.5 is precalculated using the latent ratings information. Even though the distances are calculated in the latent space dimension, the recommendations are actually generated in the original dimension. Thus, this approach eliminates the need to convert between latent and non-latent dimensions. Essentially, we obtain a matrix with latent ratings through the MF extraction process. The similarities between users are calculated using this matrix, which means they take into account the latent ratings and therefore latent context. Afterwards, this similarity matrix is used in a kNN algorithm that generates the recommendations.

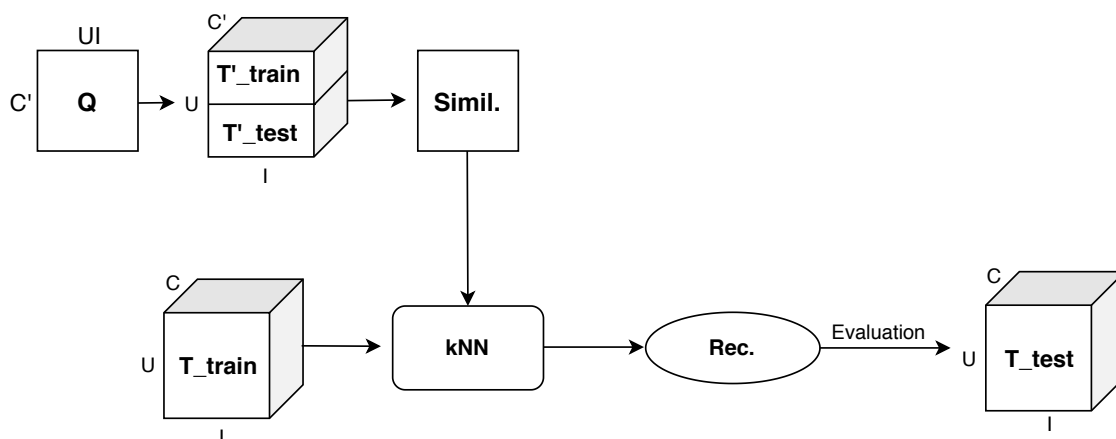


Figure 3.5: Process for generation of recommendations and empirical evaluation

3.3 Experimental Setup

3.3.1 Data

The data used consists of three data sets, all from different domains. All these data sets include at least two contextual variables, and the rating scale is from 1 to 5. Their domains and various statistics can be consulted in Table 4.1. These data sets were available in the CARSKit github repository in English.¹

The data sets are all relatively small, with the largest one having 97 users, 79 items and 11255 ratings. In contrast, the smallest one has 50 users, 40 items and 2828 ratings. The smallest data set is also the most dense, with a data density of 10.09%. The lowest data density value is 2.40%, which can be considered very high in large scale applications, and is found in the data set with the highest number of context variables and conditions. Even though the data sets are small in size, they are the best publicly available solutions to study the CARS problem in the context of latent contextual dimension extraction.

It is also noteworthy that the number of items is mostly on pair with the number of items, except for the *InCarMusic* data set. In the case of this data set, the number of items are approximately triple the number of users. Attending to these characteristics, we opted for an user-based kNN approach [DKR⁺11]. In this type of approach, a prediction for a given user is generated based on the ratings of similar users.

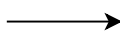
Since these data sets are small-sized, the analysis and conclusions drawn from these results are illustrative and would therefore benefit from further validation.

3.3.2 Data Preparation

All the aforementioned data sets were subject to similar preparation. All user ids, item ids and ratings were converted to numeric types. Furthermore, all invalid rows (non-numeric) were dropped.

¹https://github.com/irecsys/CARSKit/tree/master/context-aware_data_sets

User	Item	Rating	Context1	Context2
u1	i1	5	A	B
u2	i2	4	B	A
u3	i3	3	A	A
u1	i4	2	B	B



User	Item	Rating	Context
u1	i1	5	AB
u2	i2	4	BA
u3	i3	3	AA
u1	i4	2	BB

Figure 3.6: Schematic representation of aggregation of two contextual variables

The detailed rules for dropping rows and other specifics can be consulted in App. A.

Regarding contextual variables, each one of these was converted into a categorical type. Afterwards, all these categories are aggregated into a single contextual variable. This variable has a unique value for each combination of contexts. Fig. 4.1 schematises this aggregation with a small example. In this case, two contextual variables can each take two different values, A or B. Consequently, the aggregated context variable can take four values, representing all the possible combinations between A and B. This is useful so that the method can be applied to any number of context variables and context conditions. However, this step was not necessary in data set *TijuanaRestr* [RGGaV14], since the two contextual variables were already aggregated.

As detailed in Fig. 3.4 and Fig. 3.5, the method developed is made up of two processes: the extraction of latent context and the generation of recommendations. However, the pipeline for the method includes different tools for different processes. More specifically, the preprocessing and extraction of latent context is done in Python [PVG⁺11, Oli06, McK10] with surprise [Hug17].

3.3.3 Evaluation

3.3.3.1 Evaluation Protocol

As previously mentioned, our method calculates the distance matrix in the latent space dimension, while it generates the recommendations in the original non-latent dimension. Thus, it is necessary to adapt the evaluation method to conform to these different spaces. Due to this, in order to evaluate our approach, a percentage of the ratings is hidden before applying MF, as seen in Fig. 3.4([1]). The recommendations are evaluated afterwards in the original non-latent space, as is visible in Fig. 3.5.

Furthermore, in RS a common research protocol consists in using a predetermined number of known items or, alternatively, hidden items in the test set [SG11]. A scheme for this approach can be seen in Fig. 4.2. In case the predetermined number corresponds to the number of known ratings, this protocol is known as "given n", while in the case it is the number of hidden ratings it is known as "all but n". The recommendations generated by the model (*Predicted*) will be evaluated against the previously hidden ratings from the test set. In order to hide items for the evaluation, the data is split into training T'_{train} and test sets T'_{test} , some of the ratings hidden and only

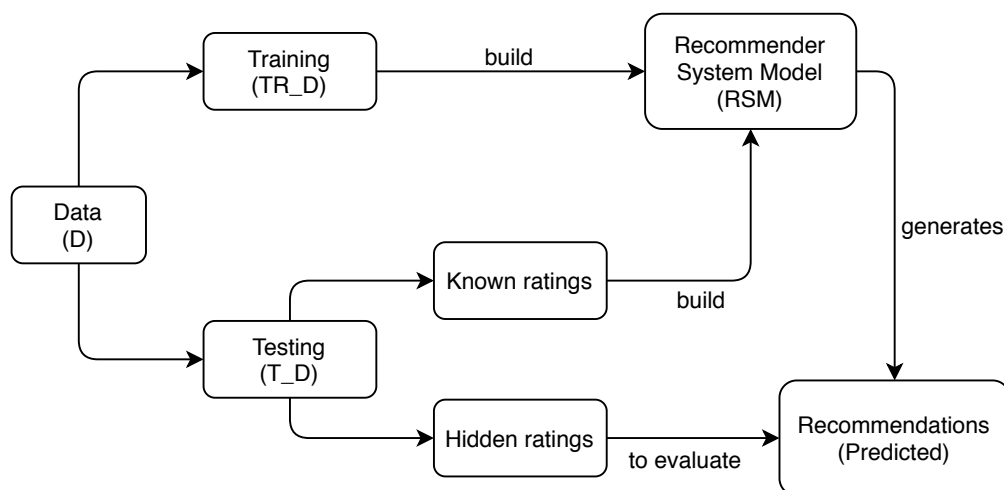


Figure 3.7: Evaluation protocol for Recommender Systems

then are the combined sets projected. The input for this calculation will be T'_{train} and T'_{test} , as seen in Fig. 3.4([5]).

3.3.3.2 Additional Procedures

To evaluate whether latent context provides better recommendations, we will compare our method with different, preexistent methods:

- **State of the art CARS** A contextual baseline is important to ascertain the differences between our latent contextual approach and a non-latent contextual approach. As such, a state of the art algorithm for context-aware recommendations is considered, namely CAMF.
- **Non-latent version of our approach** Since our latent CARS implementation has multiple components, we compare the proposed method with a version in which we turn off the use of latent context, while every other aspect remains the same. As such, we also consider a plain version of the kNN algorithm we use in our approach. However, for this baseline we will not provide the precalculated distance matrix. This means the distances will be calculated in the same space as the recommendations, using non-latent ratings. It is also worth noting that this algorithm does not take context into account whatsoever in this form.

Additionally, traditional baseline algorithms will be applied to serve as reference for the difficulty of the problem, namely MF implementations such as BiasedMF and SVD++. These algorithms do not include contextual information in the recommendation process. Even so, they facilitate drawing some conclusions regarding general quality and whether the contextual dimension is relevant to the recommendation problem.

The evaluation protocol will then consist of cross-validation, by splitting the data set D into two parts, one for training (TR_D) and another as test set (T_D). The model will be trained using the training set, and afterwards some of the metrics referred in section 2.3 will be calculated in the test set.

3.3.3.3 Metrics

Even though there are many different types of metrics available, it is necessary to choose the most appropriate ones taking into account the type of data, their meaning and the task at hand, as explained in section 2.3. For quite some time, the metrics used for the evaluation of RS have been a topic of discussion in the community. The authors of [MRK06] claim that even though rating accuracy metrics such as RMSE or MAE have been helpful to compare algorithms, better alternatives should be explored. To this end, they defend the evaluation of recommendation quality in list form instead of individually. In fact, this is how the recommendations are commonly displayed to the end user on several applications. In [LMY⁺12] it is mentioned how RMSE and MAE are not adequate to evaluating the utility of a list of recommendations.

Alternatives more suited to this task include metrics such as AUC@K, MAP@K, P@K and R@K. These metrics help measure how well the system distinguishes between relevant and irrelevant items without focusing as much on the accuracy of the predicted ratings [LMY⁺12, SG11]. Plus, since users pay more attention to the first items in the recommendation list, it is also important to evaluate the items position in the list. Thus NDGC@K will be calculated, a metric originated in Information Retrieval. This metric is based on ranking the relevance of items the further up they are on a list [SG11]. This will ensure that there is a gradation of relevance instead of just binary values for "recommend" or "do not recommend".

3.3.3.4 Considerations

The first paragraph of this section is concerned with evaluation; the other two are concerned with parameters of the methods. As will be explained in the next section, it is not possible to assure the exact same environment for baseline recommendations and the ones generated by our approach. This considered, the setup was as homogenous as possible. The validation consisted of k-fold cross-validation, with 5 folds. To obtain more reliable results, all experiments were repeated 10 times and the results averaged.

In the case of kNN-based approaches, the experiments were repeated with $k=1$, $k=3$, $k=5$ and $k=10$, to find which value obtained the best results. If k is too small, the algorithm may be more susceptible to noise, and if it is too large the distinctions between labels will be blurrier, hence the importance of finding a reasonable value. Additionally, the number should be prime to minimise ties in the voting. The experiments were done with varying factors, in order to observe how changing the number of factors might affect the results, and also to obtain the best results possible with each method. In the case of model-based approaches, the experiments were done with 100 iterations and repeated with number of factors $\text{numFact} = 1$, $\text{numFact} = 3$, $\text{numFact} = 5$ and $\text{numFact} = 10$. In the specific case of our approach, we also ran several experiments with a varying number of factors for the extraction of latent context, to see if changing this number improves the results. This will also provide some insights as to whether the number of latent context variables has influence in the information they contain. At least three different number of

factors were tried for each data set. The minimum of these numbers was 1, while the maximum was half of the number of context conditions.

3.4 Implementation

After extracting the latent ratings that result of the first part of our method, the similarities between users are calculated and used as input to generate recommendations. For this part of the process, `librec`, a Java-based tool [GZSYS15, GZYS15, SGZ15] is used. `Librec` uses the latent ratings to calculate the distance matrix. Next, `librec` uses these distances to calculate the recommendation. Additionally, `librec` is also used to run baseline experiments with the same algorithm as the one in our approach (kNN) but without the distance matrix provenient from the latent ratings.

However, `librec` does not support contextual recommendations. Therefore, in order to run baseline experiments with contextual algorithms, another toolkit named `CARSKit` [ZMB15] was used. `CARSKit` is based on `librec`, which allows the comparability of results while providing a complete library of context-aware algorithms.

3.5 Discussion

We theorise that this projection of the contexts to a latent dimension will reduce sparsity. Additionally, the distances between users will also be reduced, in comparison with the non-latent approach. As explained in 4.1.3.1, the data is split, some of the ratings in the test set are hidden and only then are the sets projected into the latent dimension. This solves the question of computational cost, since in this case the projection does not need to be done multiple times for different test sets. The most pressing issue in this case is cold start, discussed in 2.2, since in the case there are not enough users in the system it might be complicated to find similar users. Additionally, in case the existing users have not rated enough items, it is complicated to measure how similar they are to other users.

Chapter 4

Empirical Study

This chapter presents the results of the empirical study conducted. Afterwards, the research questions will be studied, in face of the empirical results obtained.

4.1 Experimental Setup

4.1.1 Data

The data used consists of three data sets, all from different domains. All these data sets include at least two contextual variables, and the rating scale is from 1 to 5. Their domains and various statistics can be consulted in Table 4.1. These data sets were available in the CARSKit github repository in English.¹

The data sets are all relatively small, with the largest one having 97 users, 79 items and 11255 ratings. In contrast, the smallest one has 50 users, 40 items and 2828 ratings. The smallest data set is also the most dense, with a data density of 10.09%. The lowest data density value is 2.40%, which can be considered very high in large scale applications, and is found in the data set with the highest number of context variables and conditions. Even though the data sets are small in size, they are the best publicly available solutions to study the CARS problem in the context of latent contextual dimension extraction.

It is also noteworthy that the number of items is mostly on pair with the number of users, except for the *InCarMusic* data set. In the case of this data set, the number of items are approximately triple the number of users. Attending to these characteristics, we opted for an user-based kNN approach [DKR⁺11]. In this type of approach, a prediction for a given user is generated based on the ratings of similar users.

Since these data sets are small-sized, the analysis and conclusions drawn from these results are illustrative and would therefore benefit from further validation.

¹https://github.com/irecsys/CARSKit/tree/master/context-aware_data_sets

	DePaulMovie [ZMB15]	InCarMusic [BKL⁺11]	TijuanaRestr [RGGaV14]
Domain	Film	Music	POI/Restaurant
Users	97	42	50
Items	79	139	40
Ratings	11255	8846	2828
Context variables	3	8	2
Context conditions	13	27	7
Data density	5.05%	2.40%	10.09%
Average rating	3.3297	2.4462	3.9030
Median rating	3	1	4
Mode rating	5	1	5

Table 4.1: Collection of statistics for the used data sets

4.1.2 Data Preparation

All the aforementioned data sets were subject to similar preparation. All user ids, item ids and ratings were converted to numeric types. Furthermore, all invalid rows (non-numeric) were dropped. The detailed rules for dropping rows and other specifics can be consulted in App. A.

Regarding contextual variables, each one of these was converted into a categorical type. Afterwards, all these categories are aggregated into a single contextual variable. This variable has a unique value for each combination of contexts. Fig. 4.1 schematises this aggregation with a small example. In this case, two contextual variables can each take two different values, A or B. Consequently, the aggregated context variable can take four values, representing all the possible combinations between A and B. This is useful so that the method can be applied to any number of context variables and context conditions. However, this step was not necessary in data set *TijuanaRestr* [RGGaV14], since the two contextual variables were already aggregated.

As detailed in Fig. 3.4 and Fig. 3.5, the method developed is made up of two processes: the extraction of latent context and the generation of recommendations. However, the pipeline for the method includes different tools for different processes. More specifically, the preprocessing and extraction of latent context is done in Python [PVG⁺11, Oli06, McK10] with surprise [Hug17].

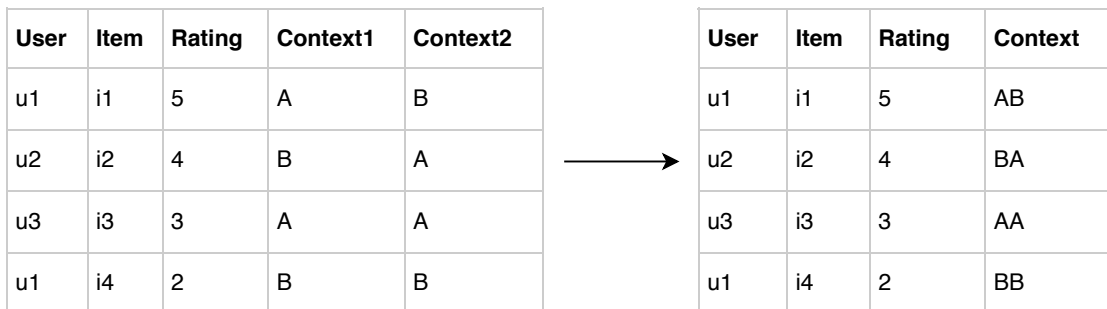


Figure 4.1: Schematic representation of aggregation of two contextual variables

4.1.3 Evaluation

4.1.3.1 Evaluation Protocol

As previously mentioned, our method calculates the distance matrix in the latent space dimension, while it generates the recommendations in the original non-latent dimension. Thus, it is necessary to adapt the evaluation method to conform to these different spaces. Due to this, in order to evaluate our approach, a percentage of the ratings is hidden before applying MF, as seen in Fig. 3.4([1]). The recommendations are evaluated afterwards in the original non-latent space, as is visible in Fig. 3.5.

Furthermore, in RS a common research protocol consists in using a predetermined number of known items or, alternatively, hidden items in the test set [SG11]. A scheme for this approach can be seen in Fig. 4.2. In case the predetermined number corresponds to the number of known ratings, this protocol is known as "given n", while in the case it is the number of hidden ratings it is known as "all but n". The recommendations generated by the model (*Predicted*) will be evaluated against the previously hidden ratings from the test set.

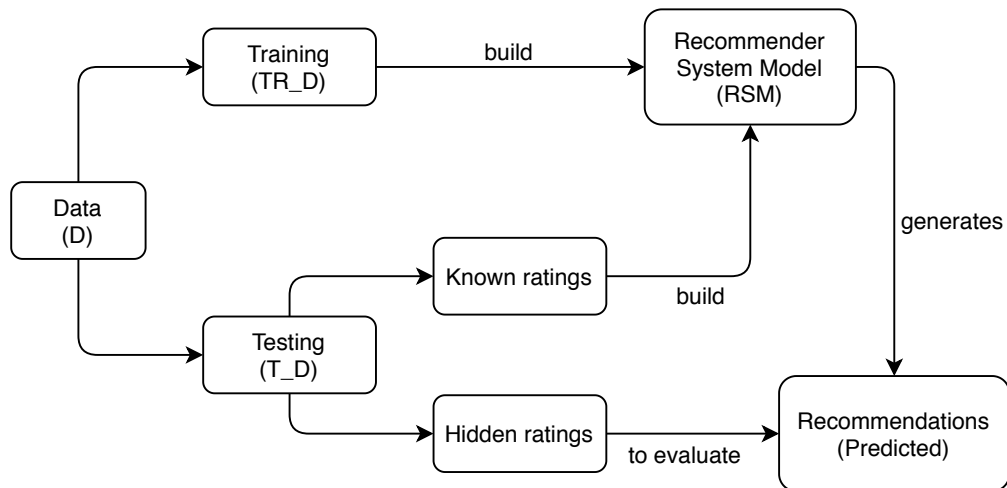


Figure 4.2: Evaluation protocol for Recommender Systems

4.1.3.2 Additional Procedures

To evaluate whether latent context provides better recommendations, we will compare our method with different, preexistent methods:

- **State of the art CARS** A contextual baseline is important to ascertain the differences between our latent contextual approach and a non-latent contextual approach. As such, a state of the art algorithm for context-aware recommendations is considered, namely CAMF.
- **Non-latent version of our approach** Since our latent CARS implementation has multiple components, we compare the proposed method with a version in which we turn off the use of latent context, while every other aspect remains the same. As such, we also consider a plain

version of the kNN algorithm we use in our approach. However, for this baseline we will not provide the precalculated distance matrix. This means the distances will be calculated in the same space as the recommendations, using non-latent ratings. It is also worth noting that this algorithm does not take context into account whatsoever in this form.

Additionally, traditional baseline algorithms will be applied to serve as reference for the difficulty of the problem, namely MF implementations such as BiasedMF and SVD++. These algorithms do not include contextual information in the recommendation process. Even so, they facilitate drawing some conclusions regarding general quality and whether the contextual dimension is relevant to the recommendation problem.

The evaluation protocol will then consist of cross-validation, by splitting the data set D into two parts, one for training (TR_D) and another as test set (T_D). The model will be trained using the training set, and afterwards some of the metrics referred in section 2.3 will be calculated in the test set.

4.1.3.3 Metrics

Even though there are many different types of metrics available, it is necessary to choose the most appropriate ones taking into account the type of data, their meaning and the task at hand, as explained in section 2.3. For quite some time, the metrics used for the evaluation of RS have been a topic of discussion in the community. The authors of [MRK06] claim that even though rating accuracy metrics such as RMSE or MAE have been helpful to compare algorithms, better alternatives should be explored. To this end, they defend the evaluation of recommendation quality in list form instead of individually. In fact, this is how the recommendations are commonly displayed to the end user on several applications. In [LMY⁺12] it is mentioned how RMSE and MAE are not adequate to evaluating the utility of a list of recommendations.

Alternatives more suited to this task include metrics such as AUC@K, MAP@K, P@K and R@K. These metrics help measure how well the system distinguishes between relevant and irrelevant items without focusing as much on the accuracy of the predicted ratings [LMY⁺12, SG11]. Plus, since users pay more attention to the first items in the recommendation list, it is also important to evaluate the items position in the list. Thus NDGC@K will be calculated, a metric originated in Information Retrieval. This metric is based on ranking the relevance of items the further up they are on a list [SG11]. This will ensure that there is a gradation of relevance instead of just binary values for "recommend" or "do not recommend".

4.1.3.4 Considerations

The first paragraph of this section is concerned with evaluation; the other two are concerned with parameters of the methods. As will be explained in the next section, it is not possible to assure the exact same environment for baseline recommendations and the ones generated by our approach. This considered, the setup was as homogenous as possible. The validation consisted of k-fold

cross-validation, with 5 folds. To obtain more reliable results, all experiments were repeated 10 times and the results averaged.

In the case of kNN-based approaches, the experiments were repeated with $k=1$, $k=3$, $k=5$ and $k=10$, to find which value obtained the best results. If k is too small, the algorithm may be more susceptible to noise, and if it is too large the distinctions between labels will be blurrier, hence the importance of finding a reasonable value. Additionally, the number should be prime to minimise ties in the voting. The experiments were done with varying factors, in order to observe how changing the number of factors might affect the results, and also to obtain the best results possible with each method. In the case of model-based approaches, the experiments were done with 100 iterations and repeated with number of factors $\text{numFact} = 1$, $\text{numFact} = 3$, $\text{numFact} = 5$ and $\text{numFact} = 10$. In the specific case of our approach, we also ran several experiments with a varying number of factors for the extraction of latent context, to see if changing this number improves the results. This will also provide some insights as to whether the number of latent context variables has influence in the information they contain. At least three different number of factors were tried for each data set. The minimum of these numbers was 1, while the maximum was half of the number of context conditions.

4.2 Implementation

After extracting the latent ratings that result of the first part of our method, the similarities between users are calculated and used as input to generate recommendations. For this part of the process, `librec`, a Java-based tool [GZSYS15, GZYS15, SGZ15] is used. Librec uses the latent ratings to calculate the distance matrix. Next, librec uses these distances to calculate the recommendation. Additionally, librec is also used to run baseline experiments with the same algorithm as the one in our approach (kNN) but without the distance matrix provenient from the latent ratings.

However, librec does not support contextual recommendations. Therefore, in order to run baseline experiments with contextual algorithms, another toolkit named CARSKit [ZMB15] was used. CARSKit is based on librec, which allows the comparability of results while providing a complete library of context-aware algorithms.

4.3 Analysis of the Results

As established in section 1.2, one of the main research goals for this project was to determine whether recommendations could be improved using latent context. More precisely, our hypothesis was that using latent context would yield better recommendations. This is due to the fact that even in the case of some reduction of information, we expect the reduction in sparsity to compensate for it.

To better examine this assumption, we tested it empirically in comparison with the groups of approaches in Table 4.2, as detailed in section 4.1.3.1. These identifications will be used throughout the document.

Empirical Study

Description	Identification
Our novel latent context-aware RS	L_kNN
State of the art CARS	CAMF_CU
Standard, non-contextual version of our approach	S_kNN
Popular traditional non-contextual algorithms	BiasedMF SVD++

Table 4.2: Set of approaches tested

The results obtained from our experiences indicate that in most of the cases tested, our approach obtains better results than the other methods. Furthermore, in the cases where it does not obtain the best results globally, these results are not considerably worse than the best method. This confirms our initial hypothesis. Additionally, it reveals that latent context is an avenue of research with potential to be explored.

It is worth noting that the approaches in all the following plots are the ones with the number of factors that yielded better results. This stands for both the state of the art methods and ours. As such, in the *TijuanaRestr* data set our approach uses 4 factors for the latent context extraction, and the best results obtained with state of the art CARS were the ones with 10 factors. In the case of *DePaulMovie* data set the best results appeared when 1 factor was used, both for the extraction of latent context and also for the state of the art models. For the *InCarMusic* data set the state of the art algorithms showed their best results with 10 factors. In the same data set, our approach performed the best with 1 factor for latent context extraction.

As is visible through the figures, in two out of the three data sets used, our approach had overall the best results. To be more precise, the values obtained by our approach were generally higher compared to other approaches, except in data set *InCarMusic*.

The full results are available in the form of tables in the App. B. The results for each data set will be analysed individually in the following subsections.

4.3.1 DePaulMovie

The frequencies for each rating in this data set can be seen in Table 4.4. It is observable that most of the ratings are positive, considering 3 as the threshold. More precisely, ratings equal to 1 or 2 make up 28.88% of the total number of ratings.

Data set	Non-latent context conditions C	Optimal latent context conditions C'
DePaulMovie	13	1
InCarMusic	27	1
TijuanaRestr	7	4

Table 4.3: Non-latent context conditions and latent context conditions extracted per data set

Empirical Study

Rating	Frequency
1	829
2	625
3	1005
4	1209
5	1367

Table 4.4: Frequency distribution of data set *DePaulMovie*

In this data set, P@5 is the only metric whose value is slightly higher on S_kNN . Our approach obtained a P@5 value of 0.112, while S_kNN has 0.113. In Fig. 4.3 it is possible to see the comparison of the values of precision across the different approaches. However, the highest value globally for MAP@5 and P@5 still come from our method, using 7 factors for latent context extraction. Even so, we opted to consider the approach with 1 factor of extraction in the graphs, since this is the factor that performed better in general.

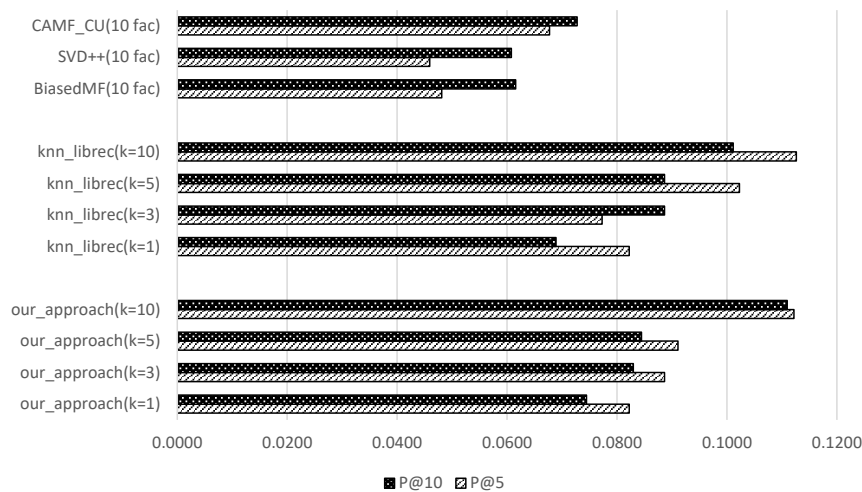


Figure 4.3: Comparison of P@5 and P@10 in the DePaulMovie data set

Fig. 4.4 shows a comparison of the different methods in terms of their AUC@5 and AUC@10. As mentioned in section 2.3, this metric represents the area under the ROC curve, considering a top set of 5 and 10 elements, respectively. This metric measures the probability of a random relevant item being ranked higher than a random irrelevant item. In the plot it is observable that our approach obtains better AUC@5 results as the number of neighbours increases, with a maximum of 0.701. This does not happen with the similar approach S_kNN , which does not include contextual information. However, the S_kNN approach obtains similar results to our approach when the number of neighbours is reduced. Regarding AUC@10, the highest value is obtained with our approach, using the maximum number of neighbours, obtaining 0.760.

By observing Fig. 4.5 we can compare the values of MAP@5 and MAP@10 across the different approaches. This measure represents the mean of the Average Precision for all the users, and is well suited to ranking recommendations. This is due to giving more weight to items recommended

Empirical Study

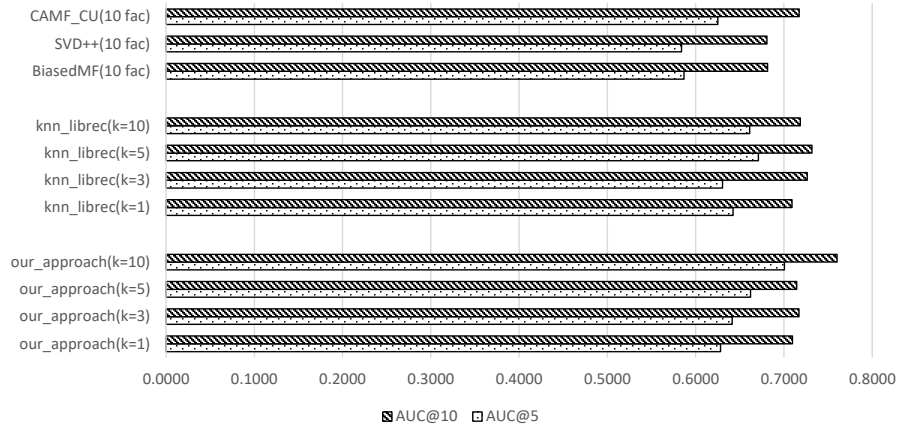


Figure 4.4: Comparison of AUC@5 and AUC@10 in the *DePaulMovie* data set

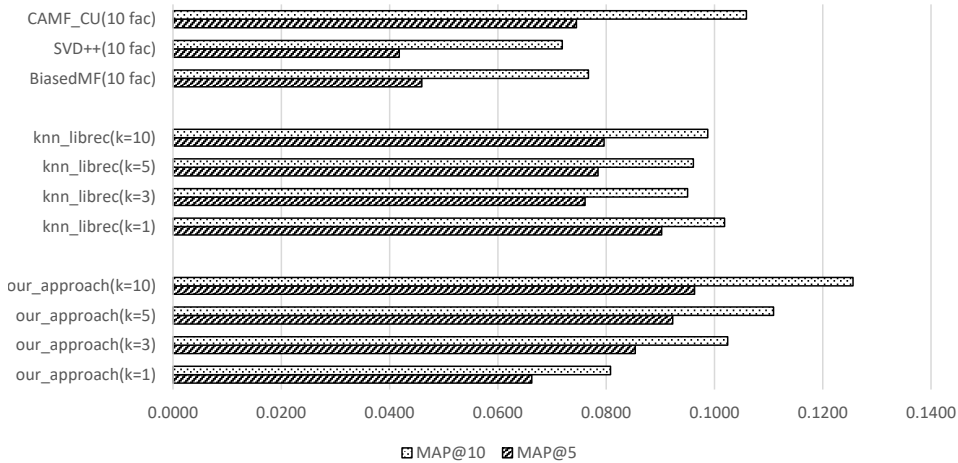


Figure 4.5: Comparison of MAP@5 and MAP@10 in the *DePaulMovie* data set

Empirical Study

at the top of the list than to the ones at the bottom. In this metric we can see a reasonable difference between the contextual and non-contextual approaches. *CAMF_CU* and *L_kNN* present the best values, with 0.106 and 0.126, respectively. The *S_kNN* method presents their best results with only one neighbour, while *L_kNN* seems to improve with higher k values.

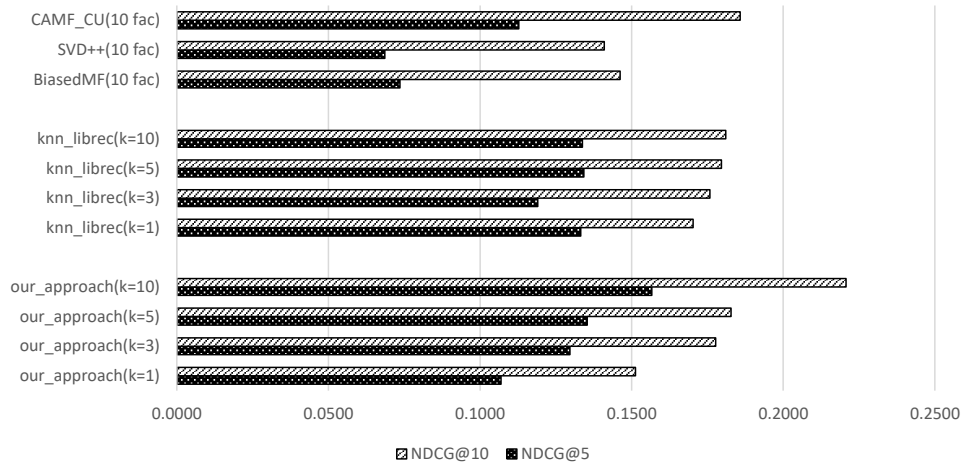


Figure 4.6: Comparison of NDCG@5 and NDCG@10 in the *DePaulMovie* data set

In Fig. 4.6 is a graph with the comparison of NDCG@5 and NDCG@10 obtained with the different methods. NDCG assumes that top recommendations are more important than bottom ones, and measures the utility of a given recommendation based on its position in the ranked list. All the methods had relatively low values in comparison with the values obtained in [UBSR16]. However, once again context-aware approaches generally outperformed the others. *L_kNN* seems to improve with higher k numbers, while the performance of *S_kNN* does not seem to be as affected by the value of k. *L_kNN* has the highest value again, followed by *S_kNN* in the case of NDCG@5 and *CAMF_CU* in the case of NDCG@10.

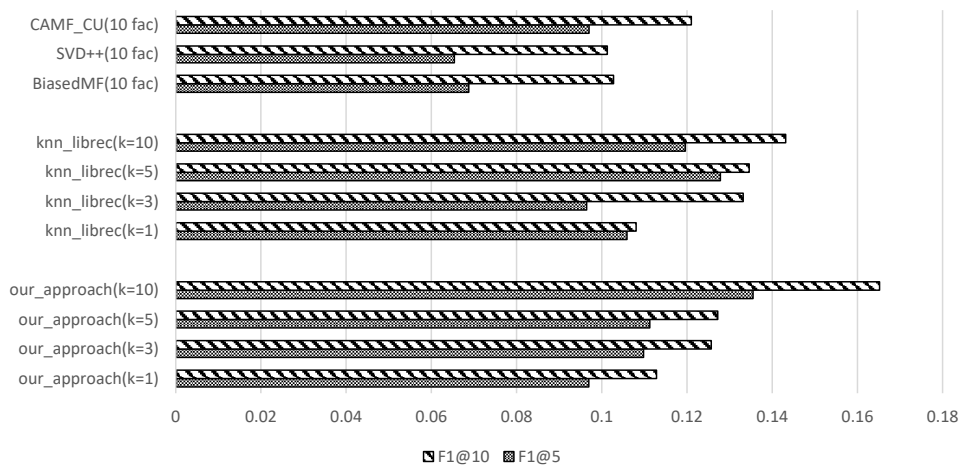


Figure 4.7: Comparison of F1@5 and F1@10 in the *DePaulMovie* data set

Empirical Study

Rating	Frequency
1	1452
2	705
3	652
4	513
5	494

Table 4.5: Frequency distribution of data set *InCarMusic*

Finally, in Fig. 4.7 it is possible to compare $F1@5$ and $F1@10$ in the different methods. As mentioned in section 2.3, $F1$ is the harmonic average of Precision and Recall. This means it measures both the precision and the robustness of the algorithm, assuming there is always a trade-off between the two. Essentially, in this data set L_kNN has the best results, followed by S_kNN . Once again, the traditional approaches $SVD++$ and $BiasedMF$ are outperformed by all other methods.

4.3.2 InCarMusic

The frequency distribution for this data set can be seen in Table 4.5. The threshold is also considered to be 3. It is noteworthy that the mode of the distribution is 1, which means the most common rating is a negative one. Plus, negative ratings make up more than half of the data set, with 56.53% of the total number of ratings.

In this data set our approach has the best results in two metrics, $P@5$ and $P@10$. In the other metrics $SVD++$ has higher values. However, it is worth noting that in this data set our approach still performs better than S_kNN . Additionally, our approach does not yield considerably worse results for most of the metrics, with the worst difference being relative to $R@10$.

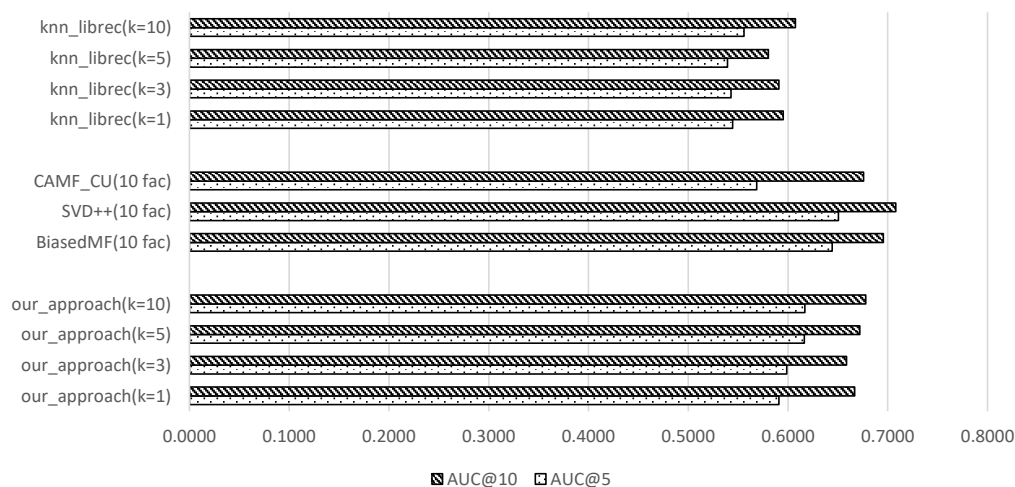


Figure 4.8: Comparison of AUC@5 and AUC@10 in the *InCarMusic* data set

Empirical Study

In Fig. 4.8, a comparison of AUC@5 and AUC@10 across the varying approaches can be seen, which shows similarity between the values obtained with our approach and with $SVD++$.

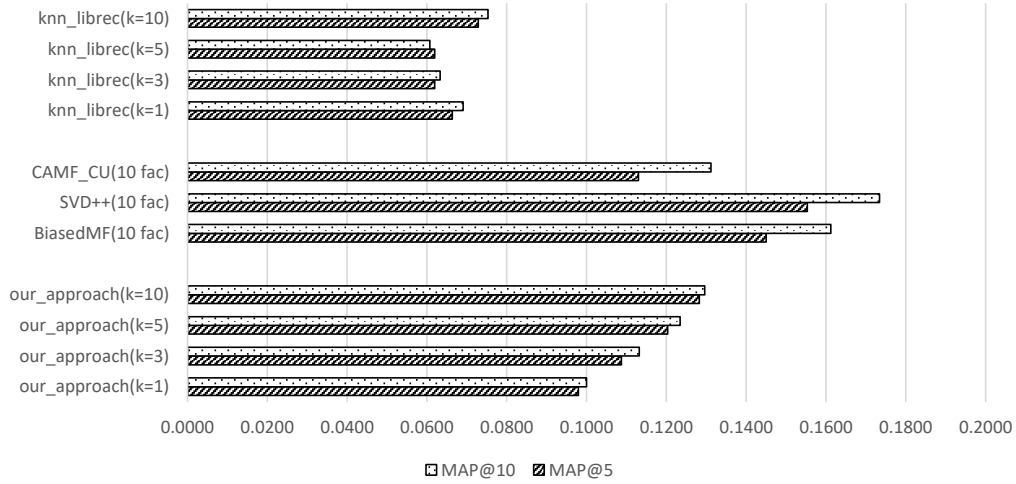


Figure 4.9: Comparison of MAP@5 and MAP@10 in the *InCarMusic* data set

The comparison of MAP@5 and MAP@10 for the different methods can be seen in Fig. 4.9. Once more S_kNN staggers behind the other approaches by a considerable margin. L_kNN seems to get better results as the k increases.

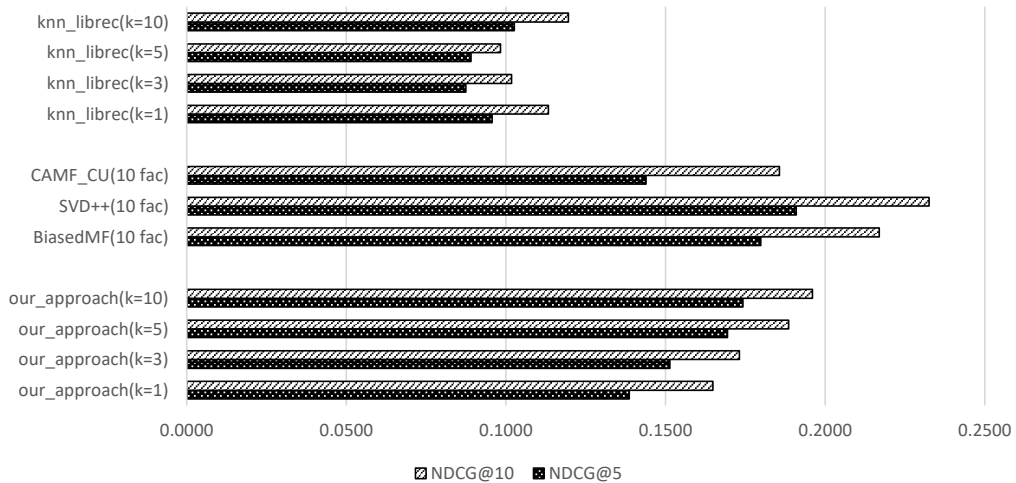


Figure 4.10: Comparison of NDCG@5 and NDCG@10 in the *InCarMusic* data set

A similar reading is done for Fig. 4.10. The results are also low in general, with the best result being obtained with $SVD++$. However, the second best method was $Biased_MF$, followed by L_kNN .

In Fig. 4.11 the comparison of F1@5 and F1@10 can be observed for the different methods. Even though the values are low in general, it is interesting to see that in terms of these metrics L_kNN is the winner with the higher absolute value.

Empirical Study

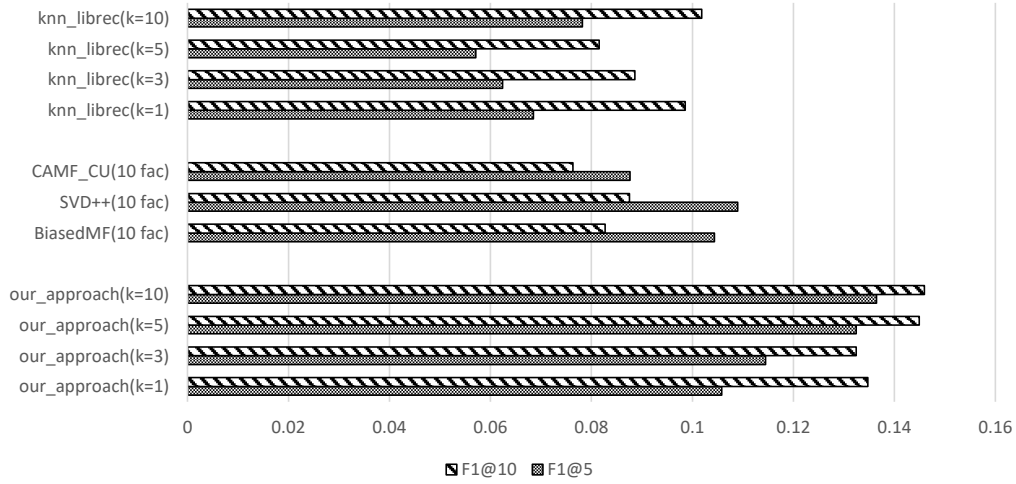


Figure 4.11: Comparison of F1@5 and F1@10 in the *InCarMusic* data set

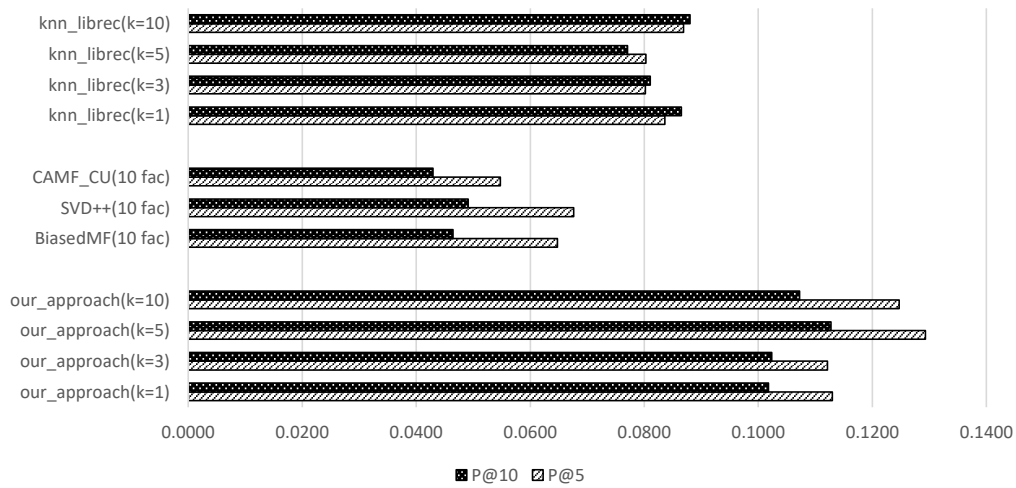


Figure 4.12: Comparison of P@5 and P@10 in the *InCarMusic* data set

Empirical Study

Rating	Frequency
1	154
2	122
3	188
4	192
5	757

Table 4.6: Frequency distribution of data set *TijuanaRestaurant*

Furthermore, in Fig. 4.12, we have a comparison of P@10 and R@10, which are the metrics with the most pronounced differences between L_kNN and $SVD++$. In the case of R@10, $SVD++$ has more 0.17 points than our approach, while in terms of P@10 our approach is 0.6 points higher than $SVD++$. $SVD++$ seems to have very low precision, which means a lot of irrelevant recommendations are made alongside the relevant ones. However, it seems to have high recall, which means it recommends most of the relevant items. L_kNN seems more balanced in the sense that it has higher precision and lower recall.

An hypothesis is that the unusual characteristics of this data set, such as low median rating, prevalence of negative ratings and higher item-to-user ratio might explain the performance of our approach. However, this would require further validation and testing.

4.3.3 TijuanaRestaurant

The frequency distribution for this data set can be seen in Table 4.6. Since the rating scale is identical to the previous two, the threshold is also considered to be 3. This data set is the smallest one, and also the densest. Plus, most ratings are positive.

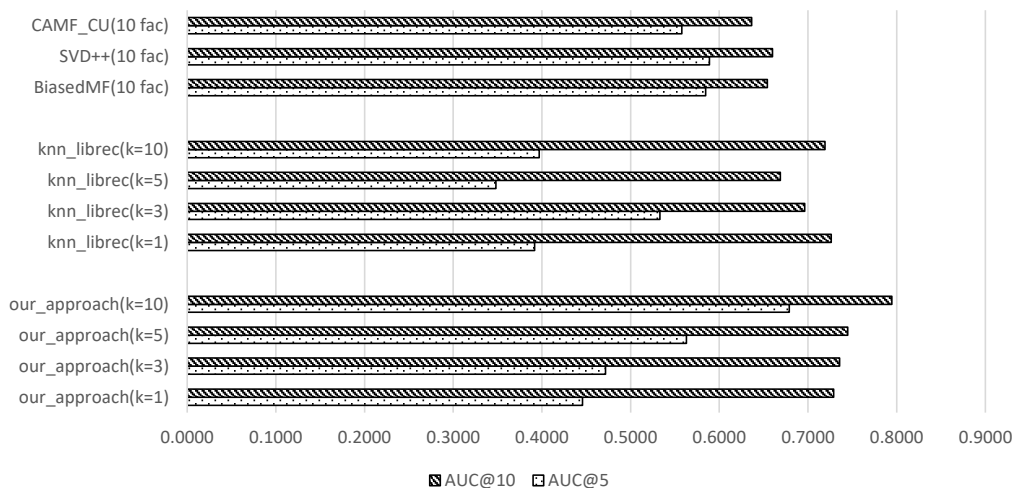


Figure 4.13: Comparison of AUC@5 and AUC@10 in the *TijuanaRestaurant* data set

The comparison for AUC@5 and AUC@10 for this data set can be seen in Fig. 4.13. It is observable that the best results are obtained by L_kNN . $SVD++$ is the second best in terms of

Empirical Study

AUC@5, but S_kNN is better in terms of AUC@10. Additionally, the value of AUC@10 in our method is the highest globally in this data set. One hypothesis to explain this could be the higher density of data.

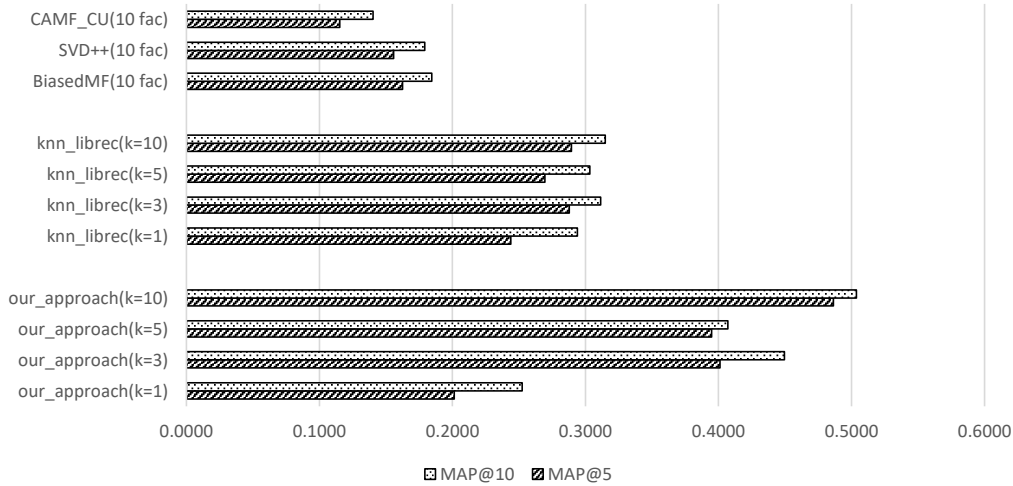


Figure 4.14: Comparison of MAP@5 and MAP@10 in the *TijuanaRestaurant* data set

In Fig. 4.14 it is possible to observe the comparison of MAP@5 and MAP@10 for this data set. Our method vastly outperforms all the others, with the second best being S_kNN . However, the values are not particularly high. The state of the art contextual approach $CAMF_CU$ showed the worst results in these metrics.

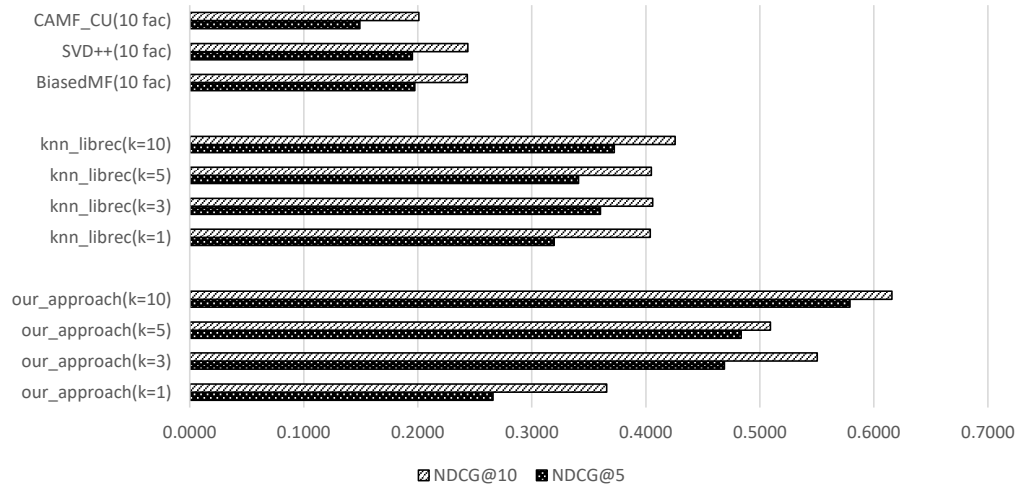


Figure 4.15: Comparison of NDCG@5 and NDCG@10 in the *TijuanaRestaurant* data set

In Fig. 4.15 the comparison of NDCG@5 and NDCG@10 for this data set is visible. It is worth pointing out that this data set shows the highest values out of all the data sets in terms of this metric. This reveals that the ranking of recommendations is done with the most relevant items at the top of the list, since changing the order of the recommendations would alter/decrease the

Empirical Study

value of NDCG. *CAMF_CU* displayed the worst results here, while *S_kNN* was the second best. In both *L_kNN* and *S_kNN* increasing k seems to improve the results.

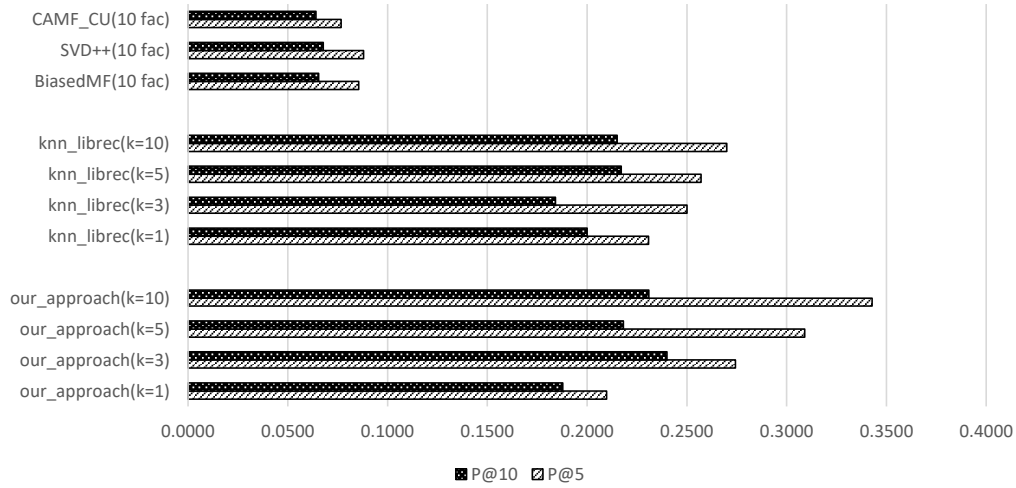


Figure 4.16: Comparison of P@5 and P@10 in the *TijuanaRestaurant* data set

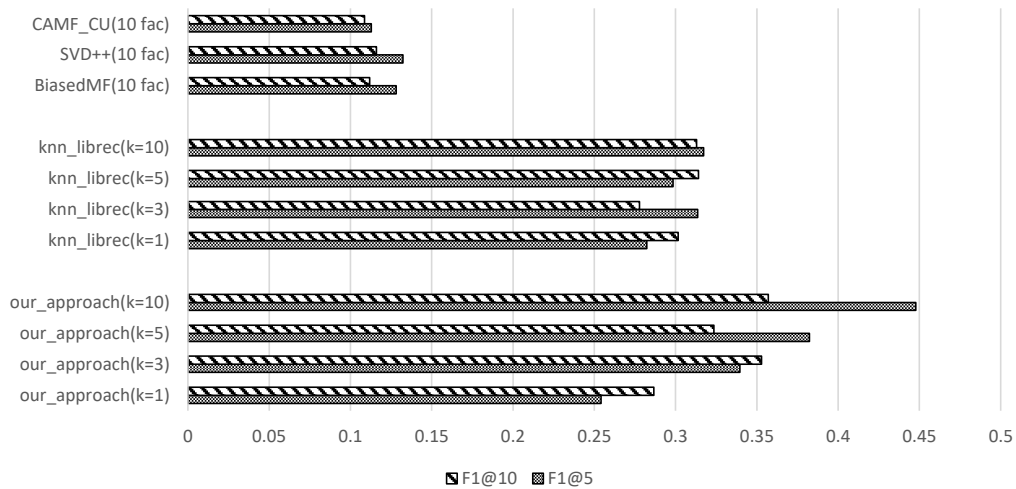


Figure 4.17: Comparison of F1@5 and F1@10 in the *TijuanaRestaurant* data set

In Fig. 4.16, it is possible to see the comparison for the precision metric in this data set. Once again, our approach outperforms all the other methods, obtaining the highest values.

The comparison of F1@5 and F1@10 can be consulted in Fig. 4.17. *L_kNN* displays the best results, followed by *S_kNN*. In last place is the *CAMF_CU* approach. Especially relatively to F1@5, we can see an improvement in the results with the increase in k in our method. It is also in this data set that we see the highest values out of all the data sets. More research would be required in order to exhaust all the possibilities, but this might be caused by the characteristics aforementioned.

4.4 Discussion

After some analysis, the results obtained can be summarised in the following points:

1. In two out of the three data sets tested, our approach obtains better results than the other methods;
2. The data set where our approach does not obtain the best results (*InCarMusic*) has different characteristics, like a low median rating and also a larger percentage of negative ratings;
3. The ideal number of factors for the latent context extraction is highly variable; in the data set *TijuanaRestr*, the best results were obtained with 4 latent context conditions vs the original 7, while in the *InCarMusic* and *DePaulMovie* data sets the best results were obtained with 1 context condition vs the original 27 and 13, respectively
4. In all the data sets tested, our approach always obtained the best performance in terms of precision compared with the remaining approaches. This happened even in the data set where our method was not the best performer

After scrutiny of the aforementioned points, some observations can be made. First of all, dense matrices seem to be related to better performances, as expected. Additionally, it seems as when the number of initial context conditions is large, our method does better with a smaller number of extracted latent conditions,. An explanation for point 2 can be that it is harder to predict which items will be liked if the data includes more information about the users' dislikes than likes. In the case of this data set, more than half of the ratings being negative means there are also fewer positive ratings the system can learn to generalise to new instances. When latent context is extracted from this data, there is a possibility that sparsity is reduced but also exacerbating the percentage of negative ratings at the same time. Point 3 might indicate that the more contextual information available, namely, contextual conditions, the easier it is to find relevant relationships between the conditions. This might also mean that systems with more contextual conditions would benefit more from a latent approach. Nevertheless, more research would be necessary to derive more conclusions.

4.5 Research Questions

The results can also be discussed in terms of the Research Questions first presented in section 1.2.

1. **Is the extraction of latent context from non-latent context variables through MF viable?** After our experiments, it is possible to conclude that MF is a viable method to extract latent context. We managed to extract latent contextual variables from non-latent context, while parameterising the number of latent contextual variables to be extracted.

- 2. Can the quality of recommendations be improved by using latent context information obtained with an adaptation of MF?** The results from our experiments suggest that the recommendations can be improved using latent context. In fact, in two out of the three data sets tested, the performance improved in regard to various metrics using latent context instead of non-latent context.

Empirical Study

Chapter 5

Conclusions and Future Work

Nowadays, RS have become significantly relevant to tackle the current information overload problem. It has been established that context can alter user preferences, and thus including it in RS yields better recommendations. These contextual variables can either be non-latent or latent. Non-latent contextual variables' meaning is very intelligible, but adding too many of these variables can increase sparsity and consequently worsen performance. Because of this, latent variables are seen as an alternative. These variables are obtained from the previously mentioned non-latent contextual variables, but their meaning is unknown since they are extracted from hidden patterns. For reasons such as privacy, usability and sparsity reduction, latent contextual variables have become more and more appealing as of late. However, their value and contribution power is not confirmed yet. The motivation for this study was to develop a method to extract and use latent contextual variables and empirically evaluate it. A possible advantage of these variables is a more precise representation of context with less dimensionality. This would diminish the need for feature engineering and lower computational cost, without compromising recommendation quality.

The main objective of this project was fulfilled, since we experimented with a methodology for the extraction of latent context from non-latent contextual variables and for generating recommendations. As such, we assembled a latent context-aware RS.

The developed method was evaluated against a state of the art CARS approach and also the kNN algorithm without the latent similarity matrix. Additionally, some popular non-latent algorithms were used as baseline. The results obtained were very favourable and demonstrated clear potential to improve recommendations. In two out of three data sets, the developed approach presented more positive results than the other methods in metrics such as AUC, MAP and NDCG.

However, only three data sets were used, and these are also relatively small and therefore more research is still required to consolidate the results. In addition, there are still interesting questions that could be potential future areas of work. For an example, it would be compelling to investigate the relation between latent and non latent contextual variables. This could be done in terms of

Conclusions and Future Work

similarity/distance between both representations, through metrics such as Jensen-Shannon divergence. This metric is a symmetric adaptation of the Kullback-Leibler divergence, which allows multivariable comparisons and can also be non-parametric. This is useful since we do not know if the distribution of the data follows a known distribution. Doing this would give some insight about the level of information in each type of context. Plus, it might also provide additional information that could help improve the extraction process. For example, knowing which variables are highly correlated might prove useful for the extraction of latent variables on future iterations. Visualization algorithms could also be used to reveal possible patterns in the representations. Another interesting avenue would be trying more methods of extraction of latent contextual variables, such as autoencoders, and comparing their performances.

References

- [ADB⁺99] Gregory D Abowd, Anind K Dey, Peter J Brown, Nigel Davies, Mark Smith, and Pete Steggles. Towards a better understanding of context and context-awareness. In *International symposium on handheld and ubiquitous computing*, pages 304–307. Springer, 1999.
- [Ado05] Incorporating Contextual Information in Recommender Systems Using a Multidimensional Approach. *ACM Transactions on Information Systems*, 23(1):103–145, 2005.
- [AJOP11] Xavier Amatriain, Alejandro Jaimes, Nuria Oliver, and Josep M Pujol. Data mining methods for recommender systems. In *Recommender systems handbook*, pages 39–71. Springer, 2011.
- [ARD⁺16] Mohammed F. Alhamid, Majdi Rawashdeh, Haiwei Dong, M. Anwar Hossain, and Abdulmotaleb El Saddik. Exploring Latent Preferences for Context-Aware Personalized Recommendation Systems. *IEEE Transactions on Human-Machine Systems*, 2016.
- [AT05] Gediminas Adomavicius and Alexander Tuzhilin. Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions. *IEEE Transactions on Knowledge and Data Engineering*, 17(6):734–749, 2005.
- [AT11] Gediminas Adomavicius and Alexander Tuzhilin. *Context-Aware Recommender Systems*. Springer, 2011.
- [BKL⁺11] Linas Baltrunas, Marius Kaminskas, Bernd Ludwig, Omar Moling, Francesco Ricci, Aykan Aydin, Karl-Heinz Lüke, and Roland Schwaiger. Incarmusic: Context-aware music recommendations in a car. In *E-Commerce and Web Technologies*, pages 89–100. Springer, 2011.
- [BLR11] Linas Baltrunas, Bernd Ludwig, and Francesco Ricci. Matrix factorization techniques for context aware recommendation. *Proceedings of the Fifth ACM Conference on Recommender Systems - RecSys '11*, page 301, 2011.
- [BOHG13] J. Bobadilla, F. Ortega, A. Hernando, and A. Gutiérrez. Recommender systems survey. *Knowledge-Based Systems*, 46:109–132, 2013.
- [Bur02] Robin Burke. Hybrid recommender systems: Survey and experiments. *User modeling and user-adapted interaction*, 12(4):331–370, 2002.
- [CD14] Tianfeng Chai and Roland R Draxler. Root mean square error (rmse) or mean absolute error (mae)?—arguments against avoiding rmse in the literature. *Geoscientific model development*, 7(3):1247–1250, 2014.

REFERENCES

- [CKH⁺16] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishu Aradhya, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, et al. Wide & deep learning for recommender systems. In *Proceedings of the 1st workshop on deep learning for recommender systems*, pages 7–10. ACM, 2016.
- [DG08] Jeffrey Dean and Sanjay Ghemawat. Mapreduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113, 2008.
- [DKR⁺11] Christian Desrosiers, George Karypis, F Ricci, L Rokach, B Shapira, and PB Kantor. Recommender systems handbook. In *A comprehensive survey of neighborhood-based recommendation methods*, pages 107–144. Springer, 2011.
- [DPDM14] Toon De Pessemier, Simon Doooms, and Luc Martens. Context-aware recommendations through context and activity recognition in a mobile environment. *Multimedia Tools and Applications*, 72(3):2925–2948, 2014.
- [EK19] Michael D. Ekstrand and Joseph A. Konstan. Recommender Systems Notation: Proposed Common Notation for Teaching and Research. 2019.
- [FO17] Evgeny Frolov and Ivan Oseledets. Tensor methods and recommender systems. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 7(3):e1201, 2017.
- [GS09] Asela Gunawardana and Guy Shani. A survey of accuracy evaluation metrics of recommendation tasks. *Journal of Machine Learning Research*, 10(Dec):2935–2962, 2009.
- [GZSYS15] Guibing Guo, Jie Zhang, Zhu Sun, and Neil Yorke-Smith. Librec: A java library for recommender systems. In *UMAP Workshops*, volume 4, 2015.
- [GZYS15] Guibing Guo, Jie Zhang, and Neil Yorke-Smith. Trustsvd: Collaborative filtering with both the explicit and implicit influence of user trust and of item ratings. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.
- [HKTR04] Jonathan L Herlocker, Joseph A Konstan, Loren G Terveen, and John T Riedl. Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems (TOIS)*, 22(1):5–53, 2004.
- [HM82] James A Hanley and Barbara J McNeil. The meaning and use of the area under a receiver operating characteristic (roc) curve. *Radiology*, 143(1):29–36, 1982.
- [HMB12] Negar Hariri, Bamshad Mobasher, and Robin Burke. Context-aware music recommendation based on latent topic sequential patterns. page 131, 2012.
- [Hug17] Nicolas Hug. Surprise, a Python library for recommender systems. <http://surpriselib.com>, 2017.
- [IFO15] FO Isinkaye, YO Folajimi, and BA Ojokoh. Recommendation systems: Principles, methods and evaluation. *Egyptian Informatics Journal*, 16(3):261–273, 2015.
- [KABO10] Alexandros Karatzoglou, Xavier Amatriain, Linas Baltrunas, and Nuria Oliver. Multiverse recommendation: n-dimensional tensor factorization for context-aware collaborative filtering. In *Proceedings of the fourth ACM conference on Recommender systems*, pages 79–86. ACM, 2010.

REFERENCES

- [KB15] Yehuda Koren and Robert Bell. Advances in collaborative filtering. In *Recommender systems handbook*, pages 77–118. Springer, 2015.
- [KBV09] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.
- [LBK17] Fatima Zahra Lahlou, Houda Benbrahim, and Ismail Kassou. Context aware recommender system algorithms: State of the art and focus on factorization based methods. *Electronic Journal of Information Technology*, 2017.
- [LHZ13] Nathan N. Liu, Luheng He, and Min Zhao. Social temporal collaborative ranking for context aware movie recommendation. *ACM Transactions on Intelligent Systems and Technology*, 4(1):1–26, 2013.
- [LMY⁺12] Linyuan Lü, Matúš Medo, Chi Ho Yeung, Yi-Cheng Zhang, Zi-Ke Zhang, and Tao Zhou. Recommender systems. *Physics reports*, 519(1):1–49, 2012.
- [LW15] Xin Liu and Wei Wu. Learning context-aware latent representations for context-aware collaborative filtering. In *Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval*, pages 887–890. ACM, 2015.
- [LX13] Bin Liu and Hui Xiong. Point-of-interest recommendation in location based social networks with topic and location awareness. In *Proceedings of the 2013 SIAM International Conference on Data Mining*, pages 396–404. SIAM, 2013.
- [LXG⁺14] Chen Lin, Runquan Xie, Xinjun Guan, Lei Li, and Tao Li. Personalized news recommendation via implicit social experts. *Information Sciences*, 254:1–18, 2014.
- [McK10] Wes McKinney. Data structures for statistical computing in python. In Stéfan van der Walt and Jarrod Millman, editors, *Proceedings of the 9th Python in Science Conference*, pages 51 – 56, 2010.
- [MDW⁺12] Nikos Manouselis, Hendrik Drachler, Martin Wolpers, Erik Duval, Katrien Verbert, Xavier Ochoa, and Ivana Bosnic. Context-Aware Recommender Systems for Learning: A Survey and Future Challenges. *IEEE Transactions on Learning Technologies*, 5(4):318–335, 2012.
- [MRK06] Sean M McNee, John Riedl, and Joseph A Konstan. Being accurate is not enough: how accuracy metrics have hurt recommender systems. In *CHI’06 extended abstracts on Human factors in computing systems*, pages 1097–1101, 2006.
- [Oli06] Travis E Oliphant. *A guide to NumPy*, volume 1. Trelgol Publishing USA, 2006.
- [PKCK12] Deuk Hee Park, Hyea Kyeong Kim, Il Young Choi, and Jae Kyeong Kim. A literature review and classification of recommender systems research. *Expert Systems with Applications*, 39(11):10059–10072, sep 2012.
- [PLS16] Sasmita Panigrahi, Rakesh Ku Lenka, and Ananya Stitipragyan. A hybrid distributed collaborative filtering recommender engine using apache spark. *Procedia Computer Science*, 83:1000–1006, 2016.

REFERENCES

- [PTG⁺09] Umberto Panniello, Alexander Tuzhilin, Michele Gorgoglione, Cosimo Palmisano, and Anto Pedone. Experimental comparison of pre- vs. post-filtering approaches in context-aware recommender systems. 2009.
- [PTG14] Umberto Panniello, Alexander Tuzhilin, and Michele Gorgoglione. Comparing context-aware recommender systems in terms of accuracy and diversity. *User Modeling and User-Adapted Interaction*, 24(1-2):35–65, 2014.
- [PVG⁺11] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.
- [PZ18] Martin Pichl and Eva Zangerle. Latent feature combination for multi-context music recommendation. *Proceedings - International Workshop on Content-Based Multimedia Indexing*, 2018-Sept:1–6, 2018.
- [RESK15] Neil Rubens, Mehdi Elahi, Masashi Sugiyama, and Dain Kaplan. Active learning in recommender systems. In *Recommender systems handbook*, pages 809–846. Springer, 2015.
- [RGGaV14] Xochilt Ramirez-Garcia and Mario Garc a Valdez. Post-filtering for a restaurant context-aware recommender system. In *Recent Advances on Hybrid Approaches for Designing Intelligent Systems*, volume 547 of *Studies in Computational Intelligence*, pages 695–707. Springer International Publishing, 2014.
- [SBCH12] Norma Saiph Savage, Maciej Baranski, Norma Elva Chavez, and Tobias H ollerer. I’m feeling loco: A location based context aware recommendation system. In *Advances in Location-Based Services*, pages 37–54. Springer, 2012.
- [SBG99] Albrecht Schmidt, Michael Beigl, and Hans-W Gellersen. There is more to context than location. *Computers & Graphics*, 23(6):893–901, 1999.
- [SG11] Guy Shani and Asela Gunawardana. Evaluating recommendation systems. In *Recommender systems handbook*, pages 257–297. Springer, 2011.
- [SGZ15] Zhu Sun, Guibing Guo, and Jie Zhang. Exploiting implicit item relationships for recommender systems. In *International Conference on User Modeling, Adaptation, and Personalization*, pages 252–264. Springer, 2015.
- [SLH13] Yue Shi, Martha Larson, and Alan Hanjalic. Mining contextual movie similarity with matrix factorization for context-aware recommendation. *ACM Transactions on Intelligent Systems and Technology*, 4(1):1–19, 2013.
- [Ste13] Harald Steck. Evaluation of recommendations: rating-prediction and ranking. In *Proceedings of the 7th ACM conference on Recommender systems*, pages 213–220. ACM, 2013.
- [Sym16] Panagiotis Symeonidis. Matrix and tensor decomposition in recommender systems. In *Proceedings of the 10th ACM Conference on Recommender Systems*, pages 429–430. ACM, 2016.

REFERENCES

- [TC00] Thomas Tran and Robin Cohen. Hybrid recommender systems for electronic commerce. In *Proc. Knowledge-Based Electronic Markets, Papers from the AAAI Workshop, Technical Report WS-00-04*, AAAI Press, page 12, 2000.
- [UBSR16] Moshe Unger, Ariel Bar, Bracha Shapira, and Lior Rokach. Towards latent context-aware recommendation systems. *Knowledge-Based Systems*, 104:165–178, jul 2016.
- [Ung15] M. Unger. Latent context-aware recommender systems. *RecSys 2015 - Proceedings of the 9th ACM Conference on Recommender Systems*, pages 383–386, 2015.
- [VSDCT18] Norha M. Villegas, Cristian Sánchez, Javier Díaz-Cely, and Gabriel Tamura. Characterizing context-aware recommender systems: A systematic literature review. *Knowledge-Based Systems*, 140:173–200, 2018.
- [YGK⁺06] Kazuyoshi Yoshii, Masataka Goto, Kazunori Komatani, Tetsuya Ogata, and Hiroshi G Okuno. Hybrid collaborative and content-based music recommendation using probabilistic model with latent user preferences. In *ISMIR*, volume 6, page 7th, 2006.
- [YL10] Zhang Yujie and Wang Licai. Some challenges for context-aware recommender systems. In *2010 5th International Conference on Computer Science & Education*, pages 362–365. IEEE, 2010.
- [YSGL12] Xiwang Yang, Harald Steck, Yang Guo, and Yong Liu. On top-k recommendation using social networks. In *Proceedings of the sixth ACM conference on Recommender systems*, pages 67–74. ACM, 2012.
- [ZMB15] Yong Zheng, Bamshad Mobasher, and Robin Burke. Carskit: A java-based context-aware recommendation engine. In *Proceedings of the 15th IEEE International Conference on Data Mining Workshops*. IEEE, 2015.
- [ZS10] Zhi-Dan Zhao and Ming-Sheng Shang. User-based collaborative-filtering recommendation algorithms on hadoop. In *2010 Third International Conference on Knowledge Discovery and Data Mining*, pages 478–481. IEEE, 2010.

REFERENCES

Appendix A

Preparations

In this appendix are listed some preparation details. Some rows were dropped from the original data in order to clean and prepare it for the subsequent operations. The rows which did not fulfil the following rules were dropped:

- Any alphabetic characters are removed from user ids
- Any alphabetic characters are removed from item ids
- User ids are converted to numeric types
- Item ids are converted to numeric types
- Ratings are converted to numeric types

Context variables were aggregated into one single context variable with all the possible conditions. Additionally, after these steps the data was indexed in terms of all the possible user-item combinations. This was done by using the cartesian product of the unique user ids and item ids as index. Finally, new user-item-context combinations, with no previous rating information were filled with zeros.

Preparations

Appendix B

Results

In this appendix the full results for each data set are presented, with the highest global value for each column highlighted.

In Figs. [B.1](#) and [B.2](#), it is possible to see the full results for the *DePaulMovie* data set. In Figs. [B.3](#) and [B.4](#) are the results for the *InCarMusic* data set, while in Figs. [B.5](#) and [B.6](#) it is possible to observe the results for the *TijuanaRestaurant* data set. In these figures, all the metrics gathered are visible, as well as the various runs of the algorithms with different parameters.

Results

UIC matrix statistics			
0	6220	Median of all rating values:	3
1	829	Standard deviation all ratings:	1.906723
2	625	Mode of all rating values:	5
3	1005	Context conditions:	13
4	1209	Data density:	5.05%
5	1367	Avg val of all ratings:	3.329688
		Nr users:	97
		Nr items:	79
		Nr ratings:	11255

ALGORITHM\METRIC	AUC@5	AUC@10	MAP@5	MAP@10	NDCG@5	NDCG@10	P@5	P@10	R@5	R@10
knn_librec(k=1)	0.6424	0.7090	0.0903	0.1019	0.1333	0.1703	0.0822	0.1487	0.0689	0.2507
knn_librec(k=3)	0.6305	0.7263	0.0761	0.0950	0.1190	0.1758	0.0773	0.1283	0.0886	0.2677
knn_librec(k=5)	0.6710	0.7317	0.0785	0.0961	0.1343	0.1796	0.1023	0.1705	0.0886	0.2798
knn_librec(k=10)	0.6610	0.7185	0.0796	0.0987	0.1337	0.1810	0.1126	0.1276	0.1011	0.2449
BiasedMF(1 fac)	0.6268	0.7040	0.0707	0.0966	0.1092	0.1698	0.0683	0.1613	0.0662	0.3177
SVD++(1 fac)	0.6253	0.7015	0.0725	0.0979	0.1103	0.1699	0.0668	0.1601	0.0655	0.3128
CAMF_CU(1 fac)	0.6092	0.6830	0.0563	0.0725	0.0903	0.1461	0.0581	0.1373	0.0590	0.2799
BiasedMF(3 fac)	0.6029	0.6836	0.0548	0.0801	0.0868	0.1472	0.0554	0.1315	0.0598	0.2879
SVD++(3 fac)	0.6013	0.6840	0.0532	0.0784	0.0772	0.1459	0.0539	0.1299	0.0592	0.2891
CAMF_CU(3 fac)	0.5957	0.6718	0.0494	0.0727	0.0790	0.1355	0.0512	0.1178	0.0556	0.2653
BiasedMF(5 fac)	0.5879	0.6725	0.0451	0.0703	0.0726	0.1345	0.0478	0.1134	0.0568	0.2753
SVD++(5 fac)	0.5903	0.6722	0.0446	0.0697	0.0729	0.1337	0.0478	0.1141	0.0562	0.2729
CAMF_CU(5 fac)	0.5863	0.6695	0.0433	0.0677	0.0702	0.1305	0.0464	0.1061	0.0551	0.2636
BiasedMF(10 fac)	0.5868	0.6815	0.0460	0.0767	0.0736	0.1462	0.0481	0.1208	0.0615	0.3117
SVD++(10 fac)	0.5840	0.6806	0.0418	0.0719	0.0686	0.1410	0.0459	0.1135	0.0608	0.3040
CAMF_CU(10 fac)	0.6251	0.7173	0.0745	0.1059	0.1128	0.1858	0.0677	0.1709	0.0727	0.3597

shrinkage=10
niter: 100,lr:0.02,maxlr: -1.0

Figure B.1: First worksheet of the results from the *DePaulMovie* data set

DATASET		Movie_DePaulMovie								
SETUP		kcv 5 folds trainset ratio = 0.8 MF 1 fact threshold=3								
ALGORITHM\METRIC	AUC@5	AUC@10	MAP@5	MAP@10	NDCG@5	NDCG@10	P@5	P@10	R@5	R@10
our_approach(k=1)	0.6282	0.7097	0.0663	0.0808	0.1070	0.1513	0.0822	0.1181	0.0744	0.2329
our_approach(k=3)	0.6414	0.7169	0.0854	0.1025	0.1297	0.1777	0.0886	0.1443	0.0830	0.2593
our_approach(k=5)	0.6623	0.7143	0.0923	0.1109	0.1353	0.1828	0.0911	0.1428	0.0844	0.2578
our_approach(k=10)	0.7005	0.7601	0.0963	0.1256	0.1567	0.2207	0.1122	0.1710	0.1110	0.3233

DATASET		Movie_DePaulMovie								
SETUP		kcv 5 folds trainset ratio = 0.8 MF 3 fact threshold=3								
ALGORITHM\METRIC	AUC@5	AUC@10	MAP@5	MAP@10	NDCG@5	NDCG@10	P@5	P@10	R@5	R@10
our_approach(k=1)	0.6056	0.6910	0.0444	0.0629	0.0790	0.1312	0.0652	0.1027	0.0697	0.2388
our_approach(k=3)	0.6521	0.7215	0.0799	0.0961	0.1282	0.1736	0.0932	0.1377	0.0846	0.2517
our_approach(k=5)	0.6471	0.7035	0.0828	0.1006	0.1318	0.1756	0.0955	0.1415	0.0875	0.2471
our_approach(k=10)	0.6629	0.7200	0.0849	0.0986	0.1363	0.1765	0.0901	0.1508	0.0791	0.2523

DATASET		Movie_DePaulMovie								
SETUP		kcv 5 folds trainset ratio = 0.8 MF 7 fact threshold=3								
ALGORITHM\METRIC	AUC@5	AUC@10	MAP@5	MAP@10	NDCG@5	NDCG@10	P@5	P@10	R@5	R@10
our_approach(k=1)	0.6086	0.6781	0.0585	0.0684	0.0927	0.1311	0.0736	0.0976	0.0690	0.2038
our_approach(k=3)	0.6573	0.7293	0.0877	0.1043	0.1353	0.1849	0.0966	0.1378	0.0908	0.2723
our_approach(k=5)	0.6675	0.7342	0.0861	0.1045	0.1392	0.1881	0.1101	0.1494	0.1022	0.2722
our_approach(k=10)	0.6699	0.7462	0.0969	0.1219	0.1535	0.2126	0.1200	0.1551	0.1067	0.3021

Figure B.2: Second worksheet of the results from the *DePaulMovie* data set

Results

UIC matrix statistics										
0	5030	Median of all rating values:	1							
1	1452	Standard deviation all ratings:	1.473793							
2	705	Mode of all rating values:	1							
3	652	Context conditions:	27	Nr users:	42					
4	513	Data density:	2.40%	Nr items:	139					
5	494	Avg val of all ratings:	2.446235	Nr ratings:	8846					

ALGORITHM\METRIC	AUC@5	AUC@10	MAP@5	MAP@10	NDCG@5	NDCG@10	P@5	P@10	R@5	R@10	
knn_librec(k=1)	0.5444	0.5953	0.0664	0.0691	0.0956	0.1133	0.0836	0.0580	0.0865	0.1146	shrinkage=10
knn_librec(k=3)	0.5429	0.5908	0.0619	0.0633	0.0874	0.1018	0.0802	0.0511	0.0810	0.0978	
knn_librec(k=5)	0.5393	0.5802	0.0620	0.0608	0.0891	0.0983	0.0803	0.0443	0.0771	0.0865	
knn_librec(k=10)	0.5558	0.6075	0.0729	0.0753	0.1026	0.1195	0.0869	0.0711	0.0880	0.1209	
BiasedMF(1 fac)	0.5475	0.5792	0.0356	0.0445	0.0493	0.0706	0.0225	0.0809	0.0198	0.1433	niter: 100,lr:rate: 0.02,maxlr: -1.0
SVD++(1 fac)	0.5471	0.5783	0.0364	0.0452	0.0498	0.0706	0.0223	0.0796	0.0196	0.1404	
CAMF_CU(1 fac)	0.5442	0.5792	0.0306	0.0409	0.0442	0.0687	0.0207	0.0768	0.0197	0.1497	
BiasedMF(3 fac)	0.5458	0.5808	0.0387	0.0490	0.0514	0.0755	0.0218	0.0829	0.0204	0.1531	
SVD++(3 fac)	0.5414	0.5802	0.0349	0.0459	0.0463	0.0726	0.0202	0.0741	0.0203	0.1519	
CAMF_CU(3 fac)	0.5380	0.5740	0.0299	0.0401	0.0409	0.0653	0.0184	0.0673	0.0189	0.1397	
BiasedMF(5 fac)	0.5486	0.5918	0.0420	0.0545	0.0555	0.0852	0.0234	0.0907	0.0232	0.1787	
SVD++(5 fac)	0.5434	0.5918	0.0381	0.0519	0.0499	0.0831	0.0212	0.0796	0.0233	0.1791	
CAMF_CU(5 fac)	0.5576	0.6056	0.0447	0.0589	0.0616	0.0951	0.0276	0.1033	0.0264	0.2030	
BiasedMF(10 fac)	0.6442	0.6955	0.1450	0.1612	0.1798	0.2169	0.0648	0.2686	0.0464	0.3772	
SVD++(10 fac)	0.6504	0.7080	0.1553	0.1734	0.1909	0.2325	0.0676	0.2802	0.0491	0.4024	
CAMF_CU(10 fac)	0.5687	0.6758	0.1130	0.1312	0.1439	0.1857	0.0547	0.2198	0.0429	0.3439	

Figure B.3: First worksheet of the results from the *InCarMusic* data set

DATASET		InCarMusic									
SETUP		kcv 5 folds trainset ratio = 0.8 MF 1 fact threshold=3									
ALGORITHM\METRIC	AUC@5	AUC@10	MAP@5	MAP@10	NDCG@5	NDCG@10	P@5	P@10	R@5	R@10	
our_approach(k=1)	0.5908	0.6668	0.0980	0.1000	0.1386	0.1648	0.1130	0.0995	0.1018	0.1991	shrinkage=10
our_approach(k=3)	0.5987	0.6586	0.1087	0.1132	0.1513	0.1731	0.1121	0.1170	0.1024	0.1875	
our_approach(k=5)	0.6163	0.6718	0.1203	0.1234	0.1694	0.1886	0.1293	0.1357	0.1127	0.2028	
our_approach(k=10)	0.6168	0.6780	0.1282	0.1296	0.1742	0.1960	0.1247	0.1506	0.1072	0.2284	

DATASET		InCarMusic									
SETUP		kcv 5 folds trainset ratio = 0.8 MF 3 fact threshold=3									
ALGORITHM\METRIC	AUC@5	AUC@10	MAP@5	MAP@10	NDCG@5	NDCG@10	P@5	P@10	R@5	R@10	
our_approach(k=1)	0.5444	0.5941	0.0812	0.0838	0.1079	0.1253	0.0976	0.0555	0.0983	0.1160	shrinkage=10
our_approach(k=3)	0.5446	0.5888	0.0769	0.0723	0.1019	0.1117	0.0916	0.0518	0.0862	0.1014	
our_approach(k=5)	0.5534	0.6025	0.0893	0.0828	0.1181	0.1272	0.0987	0.0622	0.0885	0.1225	
our_approach(k=10)	0.5696	0.6241	0.0835	0.0859	0.1169	0.1335	0.0999	0.0753	0.0966	0.1275	

DATASET		InCarMusic									
SETUP		kcv 5 folds trainset ratio = 0.8 MF 7 fact threshold=3									
ALGORITHM\METRIC	AUC@5	AUC@10	MAP@5	MAP@10	NDCG@5	NDCG@10	P@5	P@10	R@5	R@10	
our_approach(k=1)	0.5307	0.5860	0.0574	0.0606	0.0837	0.1005	0.0763	0.0376	0.0812	0.0928	shrinkage=10
our_approach(k=3)	0.5178	0.5592	0.0644	0.0568	0.0834	0.0880	0.0719	0.0351	0.0683	0.0708	
our_approach(k=5)	0.5228	0.5636	0.0617	0.0548	0.0829	0.0870	0.0738	0.0360	0.0700	0.0670	
our_approach(k=10)	0.5346	0.5728	0.0582	0.0539	0.0801	0.0875	0.0739	0.0362	0.0693	0.0742	

DATASET		InCarMusic									
SETUP		kcv 5 folds trainset ratio = 0.8 MF 14 fact threshold=3									
ALGORITHM\METRIC	AUC@5	AUC@10	MAP@5	MAP@10	NDCG@5	NDCG@10	P@5	P@10	R@5	R@10	
our_approach(k=1)	0.5372	0.6077	0.0913	0.0887	0.1154	0.1327	0.0969	0.0539	0.0959	0.1301	shrinkage=10
our_approach(k=3)	0.5770	0.6329	0.0950	0.0928	0.1303	0.1447	0.1147	0.0752	0.1040	0.1406	
our_approach(k=5)	0.5568	0.6173	0.0875	0.0888	0.1223	0.1373	0.1008	0.0756	0.0866	0.1308	
our_approach(k=10)	0.5568	0.6173	0.0875	0.0888	0.1223	0.1373	0.1008	0.0756	0.0866	0.1308	

Figure B.4: Second worksheet of the results from the *InCarMusic* data set

Results

UIC matrix statistics			
0	1415	Median of all rating values:	4
1	154	Standard deviation all ratings:	2.191017
2	122	Mode of all rating values:	5
3	188	Context conditions:	7
4	192	Data density:	10.09%
5	757	Avg val of all ratings:	3.903043
		Nr users:	50
		Nr items:	40
		Nr ratings:	2828

ALGORITHM\METRIC	AUC@5	AUC@10	MAP@5	MAP@10	NDCG@5	NDCG@10	P@5	R@5	P@10	R@10	
knn_librec(k=1)	0.3919	0.7261	0.2438	0.2939	0.3197	0.4040	0.2308	0.3637	0.2000	0.6137	
knn_librec(k=3)	0.5331	0.6963	0.2877	0.3114	0.3603	0.4059	0.2500	0.4206	0.1841	0.5652	
knn_librec(k=5)	0.3480	0.6688	0.2696	0.3032	0.3409	0.4048	0.2571	0.3554	0.2171	0.5670	
knn_librec(k=10)	0.3970	0.7192	0.2895	0.3148	0.3724	0.4257	0.2700	0.3845	0.2150	0.5745	
shrinkage=10											
BiasedMF(1 fac)	0.5344	0.6182	0.0849	0.1112	0.1148	0.1695	0.0671	0.1732	0.0603	0.3229	
SVD++(1 fac)	0.5322	0.6154	0.0848	0.1101	0.1134	0.1671	0.0655	0.1674	0.0592	0.3164	
CAMF_CU(1 fac)	0.5476	0.6306	0.0971	0.1229	0.1292	0.1842	0.0710	0.1904	0.0628	0.3424	
niter: 100,lr:rate: 0.02,maxlr: -1.0											
BiasedMF(3 fac)	0.5477	0.6389	0.0973	0.1276	0.1325	0.1972	0.0728	0.2016	0.0669	0.3823	
SVD++(3 fac)	0.5447	0.6378	0.0992	0.1308	0.1331	0.1994	0.0723	0.1975	0.0670	0.3817	
CAMF_CU(3 fac)	0.5366	0.6215	0.0861	0.1126	0.1163	0.1726	0.0659	0.1767	0.0605	0.3311	
BiasedMF(5 fac)	0.5643	0.6455	0.1136	0.1417	0.1513	0.2106	0.0779	0.2234	0.0668	0.3872	
SVD++(5 fac)	0.5571	0.6397	0.1100	0.1376	0.1467	0.2047	0.0768	0.2183	0.0660	0.3781	
CAMF_CU(5 fac)	0.5456	0.6268	0.0962	0.1227	0.1277	0.1828	0.0709	0.1854	0.0622	0.3372	
BiasedMF(10 fac)	0.5847	0.6541	0.1625	0.1845	0.1973	0.2434	0.0855	0.2556	0.0655	0.3828	
SVD++(10 fac)	0.5887	0.6598	0.1558	0.1791	0.1952	0.2439	0.0880	0.2672	0.0678	0.4017	
CAMF_CU(10 fac)	0.5578	0.6366	0.1152	0.1403	0.1491	0.2010	0.0767	0.2133	0.0641	0.3561	

Figure B.5: First worksheet of the results from the *RestrTijuana* data set

DATASET		RestrTijuana									
SETUP		kcv 5 folds trainset ratio = 0.8 MF 1 fact threshold=3									
ALGORITHM\METRIC	AUC@5	AUC@10	MAP@5	MAP@10	NDCG@5	NDCG@10	P@5	R@5	P@10	R@10	
our_approach(k=1)	0.4181	0.7102	0.2957	0.3403	0.3697	0.4434	0.2650	0.3938	0.2275	0.6224	
our_approach(k=3)	0.5396	0.6755	0.3319	0.3560	0.4148	0.4510	0.2895	0.4229	0.2000	0.5469	
our_approach(k=5)	0.4920	0.7271	0.3973	0.4225	0.4897	0.5373	0.3167	0.5178	0.2389	0.6821	
our_approach(k=10)	0.4566	0.7076	0.3946	0.4284	0.4754	0.5298	0.3135	0.4581	0.2405	0.6464	
shrinkage=10											
DATASET		RestrTijuana									
SETUP		kcv 5 folds trainset ratio = 0.8 MF 2 fact threshold=3									
ALGORITHM\METRIC	AUC@5	AUC@10	MAP@5	MAP@10	NDCG@5	NDCG@10	P@5	R@5	P@10	R@10	
our_approach(k=1)	0.3831	0.7108	0.2388	0.2793	0.2925	0.3727	0.2378	0.2876	0.2270	0.5502	
our_approach(k=3)	0.4360	0.7049	0.3524	0.4061	0.4032	0.5147	0.2629	0.3281	0.2514	0.6662	
our_approach(k=5)	0.6493	0.7227	0.3719	0.4040	0.4611	0.5126	0.2974	0.5000	0.2154	0.6473	
our_approach(k=10)	0.5832	0.7406	0.3506	0.3786	0.4145	0.4738	0.2850	0.4313	0.2175	0.6373	
DATASET		RestrTijuana									
SETUP		kcv 5 folds trainset ratio = 0.8 MF 4 fact threshold=3									
ALGORITHM\METRIC	AUC@5	AUC@10	MAP@5	MAP@10	NDCG@5	NDCG@10	P@5	R@5	P@10	R@10	
our_approach(k=1)	0.4458	0.7288	0.2014	0.2524	0.2661	0.3656	0.2098	0.3225	0.1878	0.6056	
our_approach(k=3)	0.4718	0.7356	0.4012	0.4495	0.4690	0.5503	0.2743	0.4456	0.2400	0.6664	
our_approach(k=5)	0.5630	0.7448	0.3948	0.4069	0.4836	0.5094	0.3091	0.5014	0.2182	0.6265	
our_approach(k=10)	0.6788	0.7946	0.4864	0.5036	0.5791	0.6159	0.3429	0.6455	0.2310	0.7868	

Figure B.6: Second worksheet of the results from the *RestrTijuana* data set