GDAŃSK UNIVERSITY
OF TECHNOLOGY

Faculty of Electronics,
Telecommunications and Informatics

Aleksandra Karpus

Context-Aware User Modelling and Generation of Recommendations in Recommender Systems

Doctoral Dissertation

Supervisor:
prof. dr hab. inż. Krzysztof Goczyła
Faculty of Electronics, Telecommunications and Informatics
Gdańsk University of Technology

Gdańsk, 2017

Contents

Acknowledgments

Science, my lad, is made up of mistakes, but they are mistakes which it is useful to make, because they lead little by little to the truth.

Jules Verne, *A Journey to the Center of the Earth*

In a special way I would like to thank my supervisor Professor Krzysztof Goczyła for his advising, meaningful insights and in particular for the time that he always had for me. Without him, this dissertation would not have been created.

I would also like to thank Professor Tommaso Di Noia from SisInf Lab for hosting me at Polytechnic University of Bari and for the help with developing Contextual Conditional Preferences. I am very grateful to other people from SisInf Lab, especially Paolo Tomeo, Jessica Rosati and Silvia Giannini. Many thanks to Paolo Tomeo for his help with extending datasets that we use for experimentations and with clustering of actors and directors in LDOS-CoMoDa dataset. Words of gratitude for Jessica Rosati for meaningful talks about CP-nets and for Silvia Gannini for her useful insights how to improve the questionnaire. Thank you all also for other things that happened during my stay in Bari.

I sincerely thank Iacopo Vagliano from Politecnico di Torino for creating the RSCtx ontology and for results of our joint work.

I would like to thank Doctor Tomasz Boiński for creating and sharing the LaTeX template for doctoral dissertation. I also thank to all faculty members, students and others who helped me by participating in an on-line survey.

I wish to express my sincere gratitude to my Parents who taught me that learning is an important part of living and who supported me during all of those years of study.

Finally, I would like to acknowledge with gratitude support, patience and love of my husband Oliwer. This dissertation would not have come to existence without him.

Glossary

BPR	Bayesian Personalized Ranking
CARS	Context-Aware Recommender Systems
CASA	Context-Aware Splitting Approaches
CB	Content-based (Recommender Systems)
CCP	Contextual Conditional Preference
CF	Collaborative Filtering
COUP	Contextual Ontological User Profile
CSLIM	Contextual SLIM
DL	Description Logics
FISM	Factored Item Similarity Model
GCCP	General CCP
ICCP	Individual CCP
ILD	Intra-List Diversity
kNN	k Nearest Neighbors
LDA	Latent Dirichlet Allocation
MAE	Mean Absolute Error
MAP	Mean Average Precision
MF	Matrix Factorization
MRR	Mean Reciprocal Rank
nDCG	Normalized Cumulative Discounted Gain
precision@k	Precision computed on a list of top k recommendations
recall@k	Recall computed on a list of top k recommendations
re-rankCCP	Context-aware post-filtering method that uses CCPs
RMSE	Root of Mean Square Error
RS	Recommender Systems
SIM	Structured-Interpretation Model
SLIM	Sparse Linear Methods
SVD++	MF algorithm based on Singular Value Decomposition method
Time SVD++	Time-aware SVD++
UserKNN	User-based k Nearest Neighbors
WRMF	Weighted Regularized Matrix Factorization

Chapter 1

Introduction

This chapter provides background information that helps to understand motivations for the topic of this dissertation. It introduces theses and goals of this work and describes the structure of the dissertation.

1.1 Motivations

Nowadays we can have access to unlimited information from many fields thanks to the emergence of the Internet. The problem starts when we do not know what we are looking for. One person is not able to process such big amount of data. Even search engines are not enough to solve this problem which is widely known as *information overload* [101]. Recommender systems are intended to be a solution to this problem. Their role is to give the most appropriate recommendations from the set of all possibilities to users in different situations [91].

In the last decade researchers came to a conclusion that known recommendation techniques are not sufficient to predict user decisions. It has been noticed that user preferences strongly depend on the context in which a user currently is [3]. The notion of context will be described in details in further part of the dissertation. Here, we only use the following example to better understand the intuitions. When we want to recommend some books to a Java developer who is a father, we do not want to recommend him a book with fairy tales when he is at work, but we want to do that when he is at home. Instead, when he is at work we want to recommend him a book about a new framework in Java. This raises new challenges for researchers such as how to obtain relevant contextual information, how to model user preferences in a context and use them to make predictions, just to name a few.

During last decade many context-aware approaches were proposed. However, they usually consider the situation where a lot of data is available. On the other hand, recommender systems research still strive for solving *the cold-start problem*, where we do not have enough information about users and their ratings. Different situations of this kind described in the literature are called the cold-start problem. Two of them are well-known and have also other names: the *new item problem* and the *new user problem*. Both occur when a recommender system is well-established and a lot of ratings are available. When we introduce a new item into such a system, many recommendation algorithms will not recommend it to a user because of lack of its history, i.e. former users ratings. The same happens when a new user registers into a recommender system. She will not receive interesting recommendations just because

the system does not know her preferences yet [6, 52].

One of possible solutions to solve the cold start problem in recommender systems is to use *cross-domain recommendation* [33]. A cross-domain algorithm can deal with items from different domains. Cremonesi et al. [24] distinguish three types of cross-domain recommendations: single-domain, cross-domain and multi-domain. For simplicity, we use only two domains in following explanations. Single-domain cross-domain recommendation uses information (e.g. user ratings) from both domains to make predictions in only one of them. Cross-domain recommendation uses information from one domain to make predictions in another domain. The third kind, multi-domain recommendation, recommends items from both domains using all possible information (from both domains). The biggest advantage of multi-domain recommendation is that it offers added value to recommendations, i.e. diversity, novelty and serendipity [19]. These concepts will be discussed in a further part of the dissertation.

An important aspect of human to human recommendations is giving an explanation. For example, when Bob recommends a book to Alice, he can say "You should read this book, it's really great!" The same happens in shops. When we buy a new camera, we usually ask a seller for recommendations. However, a recommendation like "This camera is better than those" is unsatisfactory. We expect some explanation, e.g. "This camera is the cheapest one that has these [known from conversation] parameters." Thus, recommender systems should also provide explanations for recommendations that they give. Explanations help a user to comprehend how recommendations were generated. This increases a user trust in a recommender system and helps her to better understand her personal needs, which leads to maximizing satisfaction [53].

The aspects outlined in this section will be thoroughly discussed in this dissertation.

1.2 Goals and Theses

Motivations presented in the previous section lead us to formulating the main thesis of this dissertation:

> *Context-aware user models and recommendation approaches proposed in this dissertation allow to create context-aware recommender systems that can be successfully applied in real-life scenarios giving satisfactory results considering their quantitative and qualitative properties.*

The main thesis is supported by the following two auxiliary theses:

1. *User modelling in the form of contextual ontology and usage of an ontology of context enable us to generate multi-domain recommendations and allow dynamic generalization of contextual parameters values.*
2. *User modelling in the form of contextual conditional preferences allows to generate explanations and contextual recommendations in new user cold-start situations as well as in typical scenarios.*

To prove above theses, we undertook fulfillment of the following goals:

- A comprehensive review of existing recommendation algorithms and evaluation measures,

- Development of a method to generate multi-domain recommendations that applies contextual ontology as a user model,
- Creating a new user model that allows to easily build explanations and generate recommendations also in the new user situations and development of recommendation algorithms based on this model,
- Performing offline experiments to prove the usability of proposed methods,
- Performing an evaluation of proposed methods by an on-line survey with real users.

1.3 Structure of Dissertation

The further part of this dissertation is organized as follows.

Chapter 2 provides intuitions and formal definitions of a context in general. It describes methods of dealing with contextual information. Moreover, it explains how context should be understood in context-aware applications and how we can decide whether a factor is a contextual parameter.

Chapter 3 presents different definitions of ontology and classification criteria. It also introduces basic information about description logics and ontologies that are based on it. Furthermore, this chapter describes the Structured Interpretation Model which is one of the ways to modularize ontologies. It also contains short review of existing ontologies of context.

In Chapter 4 we present a classification of recommendation approaches and we review existing recommendation approaches. We also explain context-awareness of recommendation methods and describe ways how user preferences can be modelled. This chapter contains detailed information on how recommender systems are evaluated and a review of evaluation measures.

Chapter 5 provides description of a novel idea of how contextual ontology built according to SIM modularization approach can be used to model context-dependent user preferences. It also introduces a new method, the Ontology-based Contextual Pre-filtering technique for recommender systems that can be applied to generate multi-domain recommendations as well as recommendations in one domain.

In Chapter 6 we define Contextual Conditional Preferences, a new formalism for modelling context-aware user interests. We also describe our algorithm of learning Contextual Conditional Preferences from a ratings matrix and two new algorithms that utilize the formalism: the Rating Prediction with CCP (to predict a user rating) and the re-rankCCP (to generate a list of top k recommendations).

Chapter 7 contains detailed information about performed experiments as well as justification of the main and auxiliary theses. In this chapter we show that re-rankCCP algorithm can be used in the new user cold-start problem as well as in the typical scenario. We also validate the proposed methods by performing an on-line survey with real users.

Chapter 8 summarizes the dissertation, emphasizing the original work presented in it. It also provides possible directions of future works in the topic of context-aware recommender systems.

In order to help with studying the dissertation, at the beginning of the text a list of abbreviations and symbols is provided.

Chapter 2

Notion of Context

This dissertation is focused on context-awareness in recommender systems. Thus, we have to explain what we understand by the notion of context. This chapter provides some intuitions and formal definitions of a context.

In everyday life people use context in a very natural way, so they do not realize that. For example, when we talk with other people we often use words such as *here* and *there* to describe places or *I*, *you* and *she* to describe people. However, we still understand each other and know what *here* means because we know the context of a sentence. We know where we are and who we are talking with. Following this intuition, we can provide first definition of a context. According to the Oxford dictionary[1], *context* is defined as "the circumstances that form the setting for an event, statement, or idea, and in terms of which it can be fully understood".

In [10, 11, 34] the metaphor of a box was used to describe the notion of context, as shown in Fig. 2.1. A collection of sentences is contained inside the box, while a collection of parameters $P_1, ..., P_n, ...$ is placed outside the box. Each parameter P_i is associated with a value V_i. The content of what is inside the box depends on the values of parameters associated with the box. For example, if the speaker is Alice, i.e. the value of the parameter *speaker* is set to *Alice*, then any occurrence of *I* will refer to Alice.

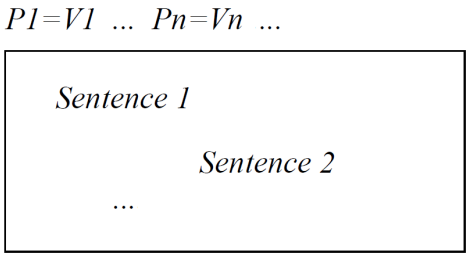


Figure 2.1: Context as a box (from [10]).

There is no agreement on the number of parameters, what features of context they should capture or whether they are fixed for all contexts. Some theories claim that all contexts depend on the finite set of parameters [60, 77]. Others argue that the number of parameters is

[1] https://en.oxforddictionaries.com/definition/context

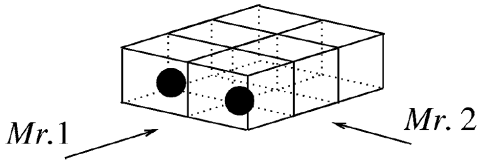


Figure 2.2: The magic box (from [94]).

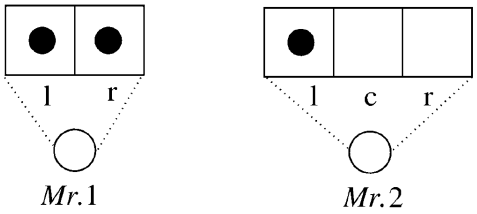


Figure 2.3: Partial views on the magic box (from [94]).

very large, virtually infinite, and is different for different contexts [41, 80]. Some sentences like "Alice lives next door" or "He is going to move out next week" suggest that these parameters should be spatio-temporal. But actually there are many more contextual features to take into consideration.

In [10] the magic box example was presented. It illustrates an importance of considering context dependance. In [94] the same example was slightly modified. Here, we present the simplified version of the example, which is shown in Fig. 2.2. The box consists of six sectors, each sector possibly containing a ball. Suppose there are two observers, Mr. 1 and Mr. 2, who look at the magic box from different viewpoints. The box is "magic", because Mr. 1 and Mr. 2 cannot guess the depth of the box. Fig. 2.3 shows what Mr. 1 and Mr. 2 see.

Partial views of Mr. 1 and Mr. 2 can be treated as two different contexts. Both observers have their local representations of the box. This representation depends on the perspective view of the observer. Mr. 1 sees two sections: left and right, with a ball in both of them. Mr. 2 sees three sections: left, center and right, where a ball is placed only in the left section. These contexts are dependent of each other. If Mr. 1 sees at least one ball, then Mr. 2 also has to see it in at least one section. If Mr. 2 does not see any ball in the box, then Mr. 1 also has to see all the sections empty.

The main aim of using contexts in logic is to reduce the amount of knowledge needed to solve a certain problem, i.e. to perform *localised reasoning*. Localised reasoning follows the intuition that some reasoning processes are local for a single context (box) disregarding what is outside the box. Thus, given the fixed collection of contextual parameters and their values, we can solve some problem taking into consideration only information placed inside the context. However, localised reasoning is not simply equivalent to reasoning with a partition of knowledge, because some facts can be inferred locally only because other facts can be inferred in other contexts. Benerecetti et al. [10, 11] proposed two methods of dealing with a knowledge from outside of the box: *push and pop* and *shifting*.

The first method assumes that there exists the state where a collection of contextual parameters is empty, i.e. uncontextual state. We can call this state *ideal*, because it never appears in practice. The process of *contextualization* consists of *push* operations which move

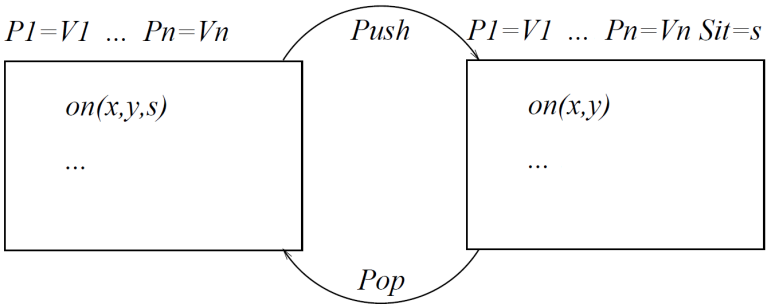


Figure 2.4: Push and pop (from [11]).

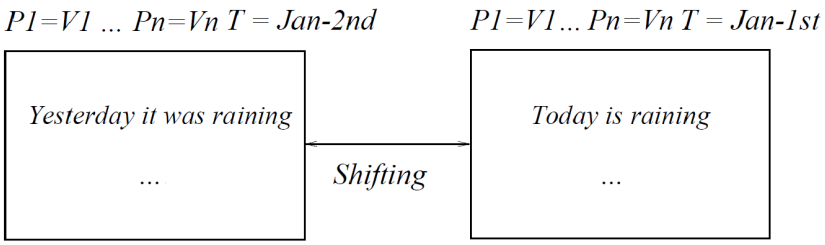


Figure 2.5: Shifting (from [11]).

information that was explicitly encoded in sentences inside the box to the contextual parameters outside of the box. It means that *push* operation produces a flow of information from the inside to the outside of the box. We can reverse this process by using *pop* operation which removes a contextual parameter from outside of the box and explicitly represents it in the sentence inside the box. An example of such two-directional process is presented in Fig. 2.4. On the left side we have a box with a fixed collection of contextual parameters. Inside the box there is a sentence that represents information that in some situation s an object x is placed on an object y. Because s can be treated as a contextual information, we can remove it from the sentence inside the box and add a new contextual parameter Sit with value s to the collection outside the box (the *push* operation). On the right we have resulting box with a simplified sentence that an object x is placed on an object y. As it was mentioned before, we can reverse this process by removing parameter $Sit = s$ from the collection of contextual parameters and encode it in a sentence inside the box (the *pop* operation), which is represented by the bottom arrow in Fig. 2.4.

The second method, *shifting*, is based on the assumption that values of some contextual parameter can be related to each other. Such parameters can be, e.g. *time* and *location*. If we assign specific days, e.g. January 1st and January 2nd, as values for the *time* parameter, then we see clear relationship between them. Thus, we can assume that the same relationship exists for the boxes containing these parameters and value pairs in their collections of contextual parameters. Indeed, if on January 1st it is raining in some place in the world (given by other, not considered now, contextual parameter), then on January 2nd we can say that "Yesterday it was raining" (in the same place in the world). This example is illustrated in Fig. 2.5. It should be noticed that *shifting* is not limited to contextual parameters such as *time*. It will work also with changing of viewpoints, like in the magic box example, and others parameters that can have values related to each other.

This dissertation is focused on context-aware user modeling and generation of recommendations. Thus, our main interest is the context from computers/applications perspective. Shilit et al. [93] defined the context by examples. The list of contextual parameters includes: location, people and resources around, lighting, noise level, network connectivity, communication costs and communication bandwidth. This definition is not good enough, because different contextual features can be applicable in different kinds of systems or domains. It is also difficult to apply while building a new system, because we cannot be sure if something is a contextual feature if it was not listed. The best and most general definition of context from the application viewpoint is the one given by Dey [26]:

Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves.

This definition makes it easier to decide whether something is a contextual feature or not. It means that parameters like time, companion, mood or weather could be considered as contextual variables for different domains and applications. For example, when recommending a movie to John it would make sense to take into account if he is planning to watch it with girlfriend, family or alone and when he wants to watch it, while the weather could not influence his decision. But when we think about travel recommendations, the weather could be crucial.

The importance of using only relevant contextual information in a certain application was emphasized by Odic et al. [83]. They notice that contextual information is dynamic and sometimes the same parameter could be considered as a context and sometimes not. For example, all user attributes, such as sex or education, could be used only in collaborative approaches. In other cases, these information is fixed for each user and cannot be treated as context. The important factor that should be checked while considering relevance of a contextual feature is variability. Odic et al. distinguish three measures that could be used for checking variability of a potentially contextual variable, i.e. entropy, variance and unalikeability. The first two are well-known measures. Worth attention is the third one, the unalikeability proposed by Kader and Perry [59, 87] and given by the formula (2.1). It represents how often observations differ one from another.

$$\eta(v) = \frac{\sum_{i \neq j} c(x_i, x_j)}{n^2 - n} \quad , \tag{2.1}$$

where v is a contextual variable, x_i and x_j are the observations of the variable v, n is the number of all observations of v, and $c(x_i, x_j) = 0$ if $x_i = x_j$ and $c(x_i, x_j) = 1$ if $x_i \neq x_j$.

To conclude this chapter, we provide a definition of context-aware system proposed by Day and Abowd [27]: "A system is context-aware if it uses context to provide relevant information and/or services to the user, where relevancy depends on the user's task."

Chapter 3

Introduction to Ontologies

In this chapter the notion of ontology is discussed. Section 3.1 provides some ontology definitions and classification criteria. It also introduces basic information about description logics and ontologies that are based on it. Section 3.2 describes the Structured Interpretation Model which is one of the ways to modularize ontologies. Section 3.3 provides an overview of existing ontologies of context.

3.1 Notion of Ontology in Informatics

The world *ontology* is known from philosophy. According to Merriam-Webster dictionary[1] it is "a branch of metaphysics concerned with the nature and relations of being". Similarly to philosophical *ontology*, *ontologies* in information sciences are focused on describing the world, the basic categories and relationships of beings, and defining entities and types of entities. There are many complementary definitions of what an ontology is. According to Gruber [40] "an ontology is an explicit specification of a conceptualization". This definition is very broad and gives possibilities for different interpretations.

Borst [15] modified a bit the definition given by Gruber and described the notion of an ontology as "a formal specification of a shared conceptualization". Studer et al. [102] merged and explained these two definition as follows:

> "An ontology is a formal, explicit specification of a shared conceptualization. Conceptualization refers to an abstract model of some phenomenon in the world by having identified the relevant concepts of that phenomenon. Explicit means that the type of concepts used, and the constraints on their use are explicitly defined. Formal refers to the fact that the ontology should be machine-readable. Shared reflects the notion that an ontology captures consensual knowledge, that is, it is not private of some individual, but accepted by a group."

Sometimes taxonomies are considered full ontologies, because they provide a consensual conceptualization of a given domain, e.g. the Yahoo! Directory, a taxonomy for searching the Web [76]. The ontology community calls them *lightweight ontologies* and distinguish them

[1] `https://www.merriam-webster.com/dictionary/ontology`

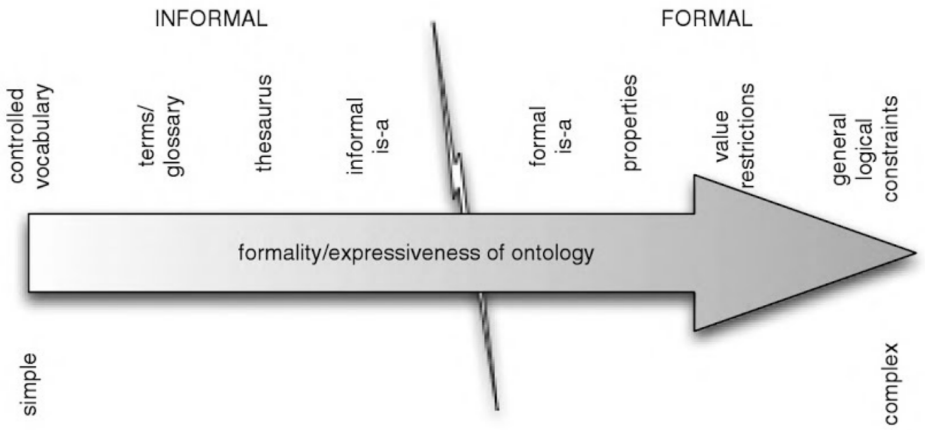


Figure 3.1: Classification of ontologies by the level of language formalization (from [108]).

from *heavyweight ontologies* which provide axioms and constraints in addition to: concepts, concept taxonomies, relationships between concepts, and properties that describe concepts (which are contained also in lightweight ontologies). Axioms and constraints clarify the intended meaning of the terms collected in the ontology [39].

We can classify ontologies by the level of language formalization. If an ontology is expressed in a natural language, it is *highly informal*. Since it is not machine-readable, it is not an ontology according to the definition by Studer et al. [102]. If an ontology is expressed in structured and restricted natural language, it is *semi-informal*. If it is expressed in some formally defined language like OWL [45, 84, 97], it is *semi-formal*. Ontologies are "rigorously formal if they provide meticulously defined terms with formal semantics, theorems and proofs of properties such as soundness and completeness" [39]. The spectrum of possible ontologies is presented in Fig. 3.1.

Another way of ontologies classification considers the role in the information systems that they play [35]. The most general ones are called *upper ontologies* and describe general terms that can be reused and shared between cooperating applications in the Internet. We distinguish here *foundational ontologies* which define abstract terms from philosophy like *entity*. The most popular are *domain ontologies* which model concepts and relations specific to certain domain like healthcare or art. They could be connected by a relation of specification to upper ontologies. *Application ontologies* are the most specific ones. They are built to work with just one system and typically they are not reused.

Ontologies based on Description Logic (DL) are the most interesting for the purposes of this dissertation. Description logics are decidable fragments of first-order logic. Ontologies based on DL consist of two parts, *TBox* and *ABox* [7]. The TBox contains terminological knowledge, i.e. definitions of concepts and roles, while the ABox contains assertional knowledge, i.e. definitions of individuals.

A DL-based ontology contains three kinds of components:

- concepts,
- roles,
- individuals.

Concepts represent classes of objects which share the same properties (but with possible

Constructor	Description
$\top$	top (universal) concept
$\bot$	bottom concept
$\neg C$	negation of concept
$C \sqcap D$	conjunction (intersection) of concepts
$C \sqcup D$	disjunction (union) of concepts
$\forall R.C$	value restriction
$\exists R.C$	existential quantification

Table 3.1: The ALC language constructors.

different values), e.g. *Human*. Roles describe relationships between concepts like *has_child* which could connect concept *Human* with itself in some demographic ontology. Roles are also used to represent properties of concepts, e.g. *name* or *age* for concept *Human*. Individuals represent instances of concepts and values of their properties (expressed in the form of roles). For example, JOE DOE could be an individual for the concept *Human* with value *joe* for the property *name*. An assignment of individual to a concept is called *unary assertion* and is placed in the ABox.

We distinguish two types of concepts, i.e. *atomic* and *complex* concepts. An atomic concept A could be any name written with capital letter, e.g. *Human*, *Woman*, *Man* or *Parent*. Complex concepts (denoted by C and D) are built from atomic concepts and roles (R) with a set of constructors like conjunction, disjunction, negation, value restriction, existential quantification, existential restriction, qualified number restriction, etc. Depending on the selection of the different constructors we can design and use different DL languages [7, 39]. Concepts descriptions are built with a relation of subsumption, i.e. $A \sqsubseteq C$ which means that all A are C. When subsumption applies in both directions, i.e. $C \sqsubseteq D$ and $D \sqsubseteq C$ are true, then we can replace it with a relation of equivalence, i.e. $C \equiv D$.

ALC is the most simple DL langage that is sufficient for building practical ontologies. The ALC language is consists of constructors which are collected in Tab. 3.1. An example of TBox defined in ALC is shown below.

$$\text{Human} \equiv \text{Woman} \sqcup \text{Man} \tag{3.1}$$

$$\text{Woman} \sqcap \text{Man} \equiv \bot \tag{3.2}$$

$$\text{Parent} \equiv \text{Human} \sqcap \exists \text{hasChild.Human} \tag{3.3}$$

$$\text{Mother} \equiv \text{Woman} \sqcap \text{Parent} \tag{3.4}$$

$$\text{Father} \equiv \text{Man} \sqcap \text{Parent} \tag{3.5}$$

$$\text{FatherOfDaughters} \equiv \text{Father} \sqcap \forall \text{hasChild.Woman} \tag{3.6}$$

$$\text{Married} \equiv \text{Human} \sqcap \exists \text{hasSpouse.Human} \tag{3.7}$$

$$\text{Single} \sqsubseteq \text{Human} \sqcap \neg \text{Married} \tag{3.8}$$

We can read above axioms as: humans are men and women (3.1); there are no individuals that are man and woman at the same time (3.2); a parent is a human with at least one child (3.3); a mother is a woman and a parent (3.4); a father is a man and a parent (3.5); a father

whose all children are women, is father of daughters (3.6); a human who has a spouse is married (3.7); a single is a human who is not married (3.8). A corresponding ABox can look like follows.

$$\text{Human}(\textsc{carol}) \tag{3.9}$$

$$\text{Man}(\textsc{joeDoe}) \tag{3.10}$$

$$\text{Woman}(\textsc{alice}) \tag{3.11}$$

$$\text{Woman}(\textsc{mary}) \tag{3.12}$$

$$\text{hasChild}(\textsc{joeDoe},\textsc{alice}) \tag{3.13}$$

$$\text{hasChild}(\textsc{joeDoe},\textsc{mary}) \tag{3.14}$$

$$\text{hasSpouse}(\textsc{joeDoe},\textsc{carol}) \tag{3.15}$$

$$\neg\text{Married}(\textsc{alice}) \tag{3.16}$$

We can read above assertions as: CAROL is a human (3.9); JOEDOE is a man (3.10); ALICE and MARY are women (3.11, 3.12); ALICE and MARY are children of JOEDOE (3.13, 3.14); CAROL is a spouse of JOEDOE (3.15); ALICE is not married (3.16). Based on above TBox and ABox we can derive additional assertions which are shown below.

$$\text{Father}(\textsc{joeDoe}) \tag{3.17}$$

$$\text{FatherOfDaughters}(\textsc{joeDoe}) \tag{3.18}$$

$$\text{Married}(\textsc{carol}) \tag{3.19}$$

$$\text{Married}(\textsc{joeDoe}) \tag{3.20}$$

There exist many design methods for building ontologies of good quality. More details can be found in [36, 61].

3.2 Contextual Ontology

A contextual ontology is an ontology built according to modularization approach called *Structured-Interpretation Model* (SIM) developed at Knowledge Management Group at Gdańsk University of Technology [37, 38, 106]. This approach enables us to represent different, sometimes contradictory, points of view in one ontology and to reason with it.

Basic elements of a contextual ontology are *context types* and *context instances* which correspond to parts of TBox and ABox respectively. As a *context type* we understand a part of TBox defined by values of a set of contextual parameters. Similarly as for classical ontologies, *context instances* do not introduce new terminology. These basic elements are connected with each other by three kinds of relations. Context types are arranged in an inheritance hierarchy by the relation of *inheritance*. More specialized terminologies may "see" more general ones, but more general terminologies are unaware of the existence of more specialized ones. Context instances are connected by relation of *aggregation*. Aggregating context instance merges information from aggregated context instances. There exists a connection between context types and instances and it is described by the relation of instantiation which assign each

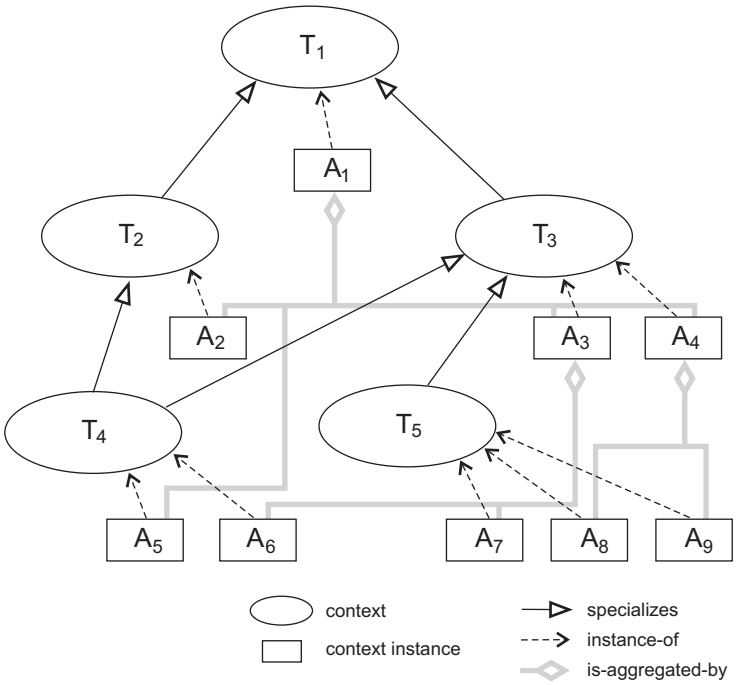


Figure 3.2: Structured-Interpretation Model (from [106]).

context instance to its type. This structure makes terminological and assertional parts of ontology independent from each other.

An example ontology built according to SIM method is presented in Fig. 3.2. This ontology consists of five context types and nine context instances. Some of the types have more than one context instance. For example, context instances A_7, A_8, A_9 instantiate context type T_5.

There is always only one context instance that aggregate all other context instances and only one context type from which other types inherit. These are maximal elements of aggregation and inheritance relations respectively. Thus, all the context instances must be consistent with each other on the maximal level of generalization.

A simple example taken from [106] is used to explain how the contextual ontology works. Let assume that an ontology consists of two context types (TBoxes) and three context instances (ABoxes). Context type T_1 provides concept *Can_resuscitate* from which concept *Doctor* inherits. *Doctor* is a concept provided in the terminology of context type T_2. Context instances A_2 and A_3 describe a situation of an individual called *john_doe* from different points of view: he is a doctor in Poland but legally he is not a doctor in United Kingdom. Assertions $Doctor(john\_doe)$ and $\neg Doctor(john\_doe)$ are contradictory, nevertheless the ontology is consistent. It is because the concept *Doctor* is defined below the context instance A_1 which aggregates context instances A_2 and A_3. The concept *Can_resuscitate* is defined on the level of the context instance A_1. Therefore the conclusion reflecting the fact that John Doe can resuscitate can flow between the two contexts (in Poland and in the United Kingdom). This example is shown in Fig. 3.3.

Another example is shown in Fig. 3.4. An ontology consists of three context types and three context instances. Context type T_1 describes general notions of *Man* and *Woman*.

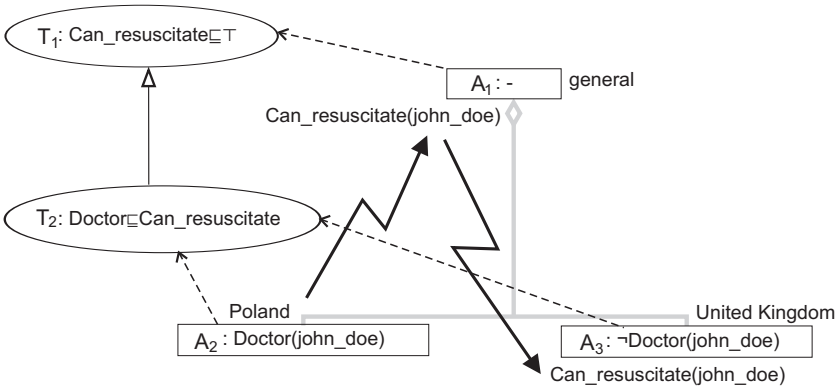


Figure 3.3: An example of SIM ontology (based on [106]).

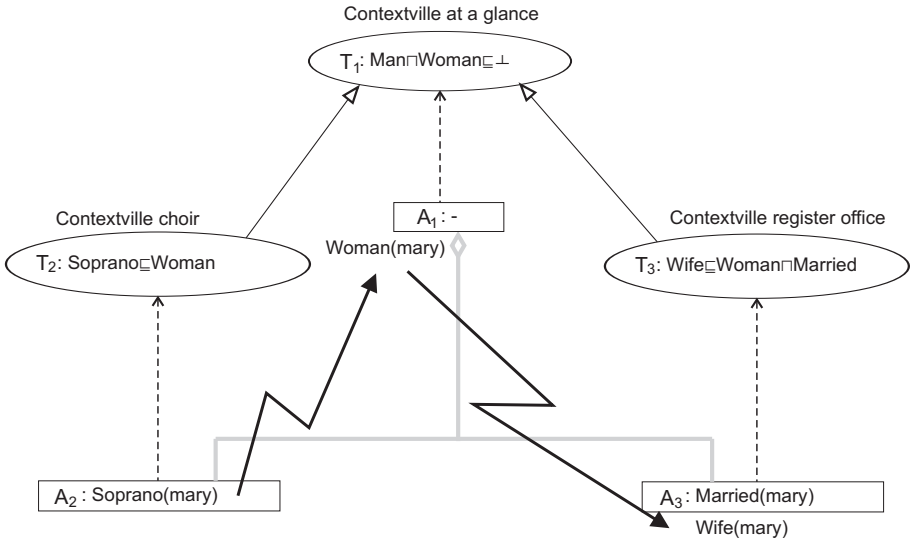


Figure 3.4: An example of SIM ontology (from [38]).

Context type T_2 specializes T_1 towards description of voices in a choir while context type T_3 specializes T_1 towards description of social relations. Context instances A_2 and A_3 are aggregated by context instance A_1 which ABox is empty. Because concept *Woman* is defined at the level of context type T_1, information that Mary is a woman flows through context instance A_1 to the context instance A_3. Hence, the conclusion that Mary is also a wife can be obtained.

3.3 Ontologies of Context

In one of the proposed in this thesis recommendation method we use an ontology of context. Hence, in this section we shortly review some of existing and most popular ontology-based models of context information.

Many approaches for modelling contextual information have been already proposed. These models are sometimes less, sometimes more formal (while utilize UML, Object-Role Modelling or ontologies [12, 14, 75]). Bettini et al. gave three reasons why modelling contexts by on-

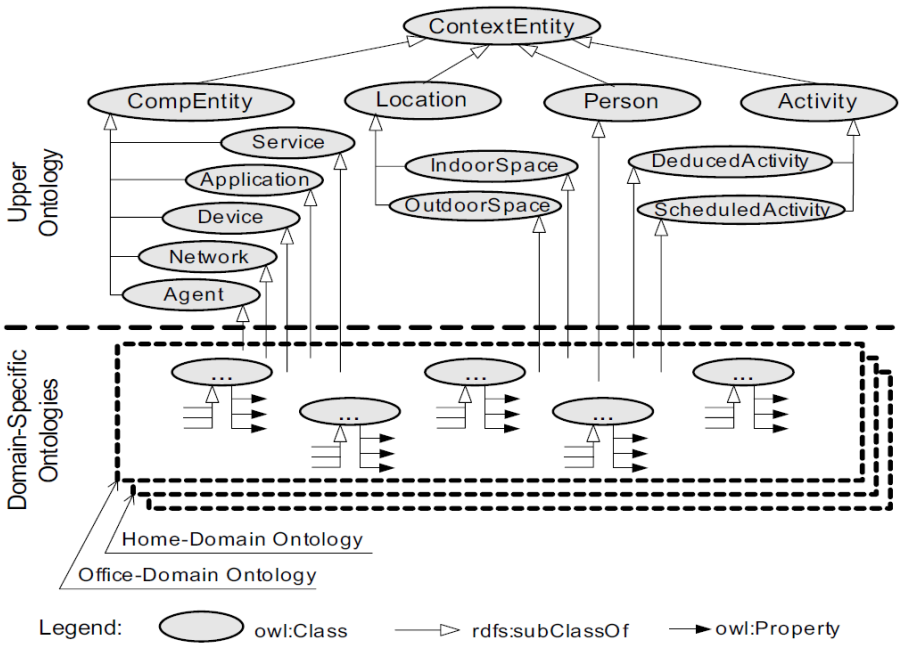


Figure 3.5: Partial Definition of CONON upper ontology (from [107]).

tologies is a good solution:

1. the expressiveness of the language - it is hard to represent complex context data by simple languages;
2. possibility to share and/or integrate context among different sources which is given by providing a formal semantics to context data;
3. the available reasoning tools - they can be used to check for consistency of the set of relationships describing a context scenario and to reveal the presence of a more abstract context characterization [12].

Whang et al. [107] proposed an extensible CONtext ONtology (CONON) for modeling context in pervasive computing environments. The context model is divided into upper ontology and domain-specific ontologies which is shown in Fig. 3.5. The upper ontology captures general features of basic contextual entities. CONON defines 14 core classes to model Person, Location, Activity and Computational Entities.

The CoDAMoS project [88] proposes an extensive ontology-based model for creating context-aware computing infrastructures. The ontology is classified into four basic concepts used to model Users, Environment, Platforms and Services. It is focused on the modeling of profiles for human users and applications, and might be limited with respect to future context-awareness tasks in service-service interaction models. It is difficult to express contexts at different granularity levels with this model.

The goal of the SOUPA project was to define ontologies for supporting pervasive computing applications [22]. SOUPA is written in a very modular way by combining subontologies for time, location, policies and persons (FOAF - Friend Of A Friend). It consists of two distinctive but related sets of ontologies: SOUPA Core and SOUPA Extension which is shown in Fig. 3.6.

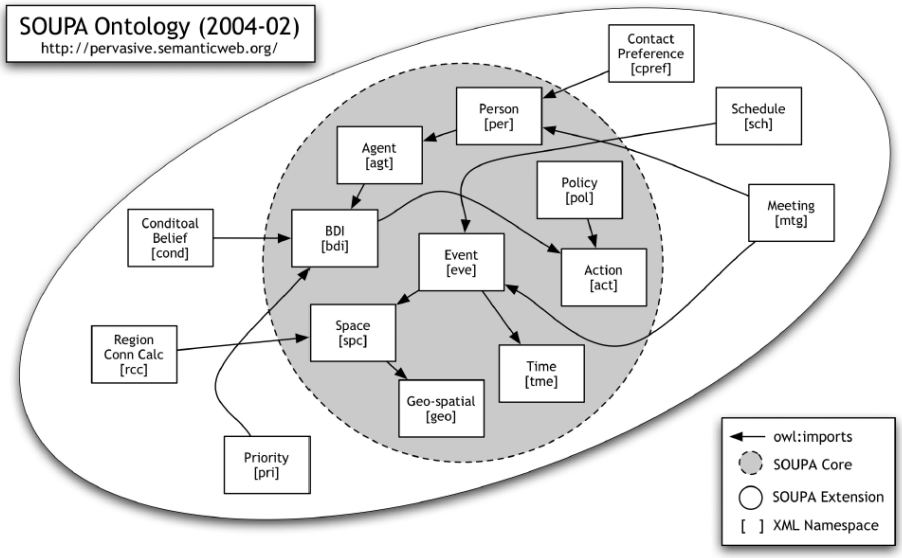


Figure 3.6: The SOUPA ontology (from [22]).

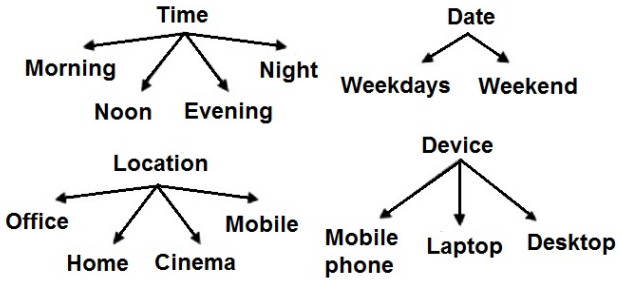


Figure 3.7: Examples of context taxonomies (from [44]).

Interesting approach was proposed by Hawalah and Fasli [44]. Authors suggest that each context dimension should be described by its own taxonomy. Time, date, location and device are considered as default context parameters. Their taxonomies are shown in Fig. 3.7. It is possible to add other domain specific context variables as long as they have a clear hierarchical representations.

PRISSMA[2] [23], a vocabulary based on Dey's definition of context [26], relies on the W3C Model-Based User Interface Incubator Group proposal[3], which describes mobile context as an encompassing term, defined as the sum of three different dimensions: user model and preferences, device features, and the environment in which the action is performed. A graph-based representation of PRISSMA is provided in Fig. 3.8.

Recommender System Context (RSCtx) ontology [68] was designed according to METHONTOLOGY [32], a well known ontology design method. It follows an assumption that, in a given application, there is a predefined set of contextual dimensions, each with a defined

[2] http://ns.inria.fr/prissma
[3] http://www.w3.org/2005/Incubator/model-based-ui/XGR-mbui/

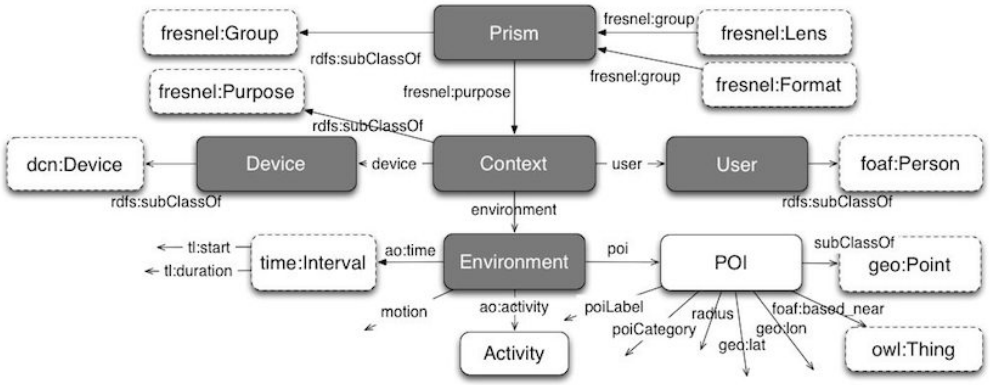


Figure 3.8: The PRISSMA vocabulary (from [23]).

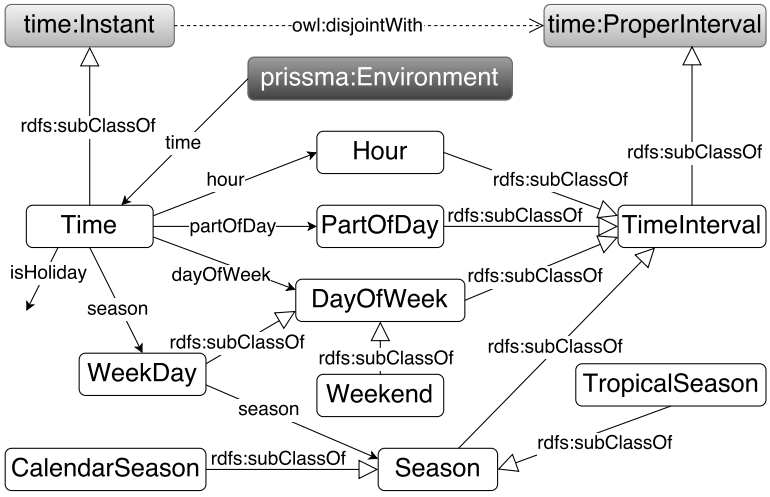


Figure 3.9: Concepts and relations of RSCtx representing the time dimension (from [68]).

set of attributes. The contextual information that are relevant to provide recommendations, were modelled. RSCtx ontology is not focused on any particular domain, on the contrary it is aimed at reusing it in different applications. As in PRISSMA, the point of view used to describe the context itself is the application point of view, thus the user itself is considered as part of the context. The main goal in this contextual model is a possibility to express different levels of granularity for different context parameters. Fig. 3.9 illustrates how time is represented and the relations with PRISSMA and the Time [46] ontology.

Chapter 4

Basics of Recommender Systems

In this chapter we describe what Recommender Systems are. We also present a classification of recommendation approaches in Section 4.1. In Section 4.2 we explain context-awareness of recommendation methods. Section 4.3 provides a systematic review of existing recommendation algorithms that are used in further chapters. In Section 4.4 we described ways in which user preferences can be modeled. Section 4.5 concerns evaluation approaches and measures.

4.1 Classification of Recommender Systems

Recommendation Systems (RSs) are software tools and techniques which aim at suggesting new items that may possibly be interesting to a user. An item could be a book, a movie, a concert, a job or even a friend in some social recommenders (e.g. Facebook). In everyday life we interact with RSs when we search for information using Google or when we buy something through Internet.

Adomavicius and Tuzhilin [4] described recommendation as a two-dimensional function

$$R : Users \times Items \mapsto Ratings \tag{4.1}$$

that maps the user and the item dimensions onto a rating score. For each user, function R computes rating just for items that were not yet seen. Therefore, we could obtain rating for whole $Users \times Items$ space and use the highest ones for a specific user to make recommendations for her.

We distinguish two types of recommendation task: *rating prediction* and *generating ranking*. In the first one, the system provides a set of predicted ratings for a set of input users and items - one rating per each user and item pair. In the second one, also known as *recommending good items* or *generating a list of top k recommendations*, the system finds k best items from whole set of possible items that should be of interests to a user.

Besides recommendation tasks, we classify recommenders according to how they work. We distinguish four kinds of RSs [52]:

1. Content-based RSs,
2. Collaborative RSs,
3. Knowledge-based RSs,
4. Hybrid approaches.

Title	Director	Genre	Actors
Donnie Darko	Richard Kelly	drama, supernatural	Jake Gyllenhaal, Jena Malone, Drew Barrymore
Girl Interrupted	James Mangold	drama	Winona Ryder, Angelina Jolie
Inception	Christopher Nolan	heist, thriller, science fiction	Leonardo DiCaprio, Ken Watanabe, Marion Cotillard
Hunger Games	Gary Ross	science fiction, adventure	Jennifer Lawrence, Josh Hutcherson, Liam Hemsworth
Sleepless In Seattle	Nora Ephron	drama, comedy, romantic	Tom Hanks, Meg Ryan

Table 4.1: Exemplary catalog of movies.

Person	Director	Genre	Actors
Alice	Christopher Nolan, Nora Ephron	drama, science fiction, adventure	Leonardo DiCaprio, Tom Hanks, Jennifer Lawrence

Table 4.2: Exemplary preference profile.

4.1.1 Content-based Recommender Systems

Content-based (CB) RSs rely on the assumption that a user likes things which are similar to each other. Thus, she should be satisfied with recommendations of items that are similar to those consumed[1] by her in the past. It is very important how items and user preferences are represented and compared in this kind of recommenders. The simplest way to represent items is to provide a list of *features* (also called *attributes*) for each item. For a movie recommender we could consider the genre, the director, the film studio or the language as useful item features. An example catalog of movies is presented in Tab. 4.1. Usually, user preferences are expressed in exactly the same dimensions. An example preference profile of Alice is depicted in Tab. 4.2.

In order to recommend some movie to Alice, we should find an item that is most similar to her preferences. To compute the similarity between two movies, we need to decide on a similarity measure and item features that will be used. In considered example we have categorical variables only. For most of them more than one value can be assigned at the same time. For this kind of situations, the best similarity measure is Jaccard similarity [50, 51] given by the formula (4.2).

$$jacc(A, B) = \frac{|A \cap B|}{|A \cup B|} \ , \tag{4.2}$$

[1]This is a commonly used in the literature word which denotes that a user saw or bought, or listened to, or knew an item (different verbs can be used for different kinds of items). The verb *consume* is just domain-independent.

User	Movie1	Movie2	Movie3	Movie4	Movie5
Alice	3	1	4	4	?
Bob	4	2	5	4	5
Carol	1	5	5	4	3
Dave	5	3	4	3	4
Eve	3	4	2	1	2

Table 4.3: Exemplary matrix of user-item ratings.

where $\cap$ indicates the intersection and $\cup$ the union of sets A and B.

Let us assume that we want to determine the similarity of movies by taking into account only the genre. In that case, the best movie for Alice is *Hunger Games* with Jaccard similarity equal to 2/3, while other movies obtain score equal or less than 1/3 (*Girl Interrupted* - 1/3, *Donnie Darko* and *Sleepless In Seattle* - 1/4, *Inception* - 1/5). If we wish to use all the attributes, we can for instance compute an average of similarities for each item feature. In this case, the best movie for Alice is *Sleepless In Seattle* with score 1/3, followed by *Inception* (9/30) and *Hunger Games* (13/45).

Content-based methods have their limitations. One of the most important is that they lead to overspecialization by looking only for similar items. Thus, they do not recommend to a user new things that a user does not know yet and that she may like. Another big problem occurs for new users who have just registered in a CB RS. They have not rated many items yet or did not provide information about their interests so it is hard to provide any recommendations (it is impossible to find any similar thing to something unknown). This phenomenon is called the new user cold-start problem.

4.1.2 Collaborative Recommender Systems

Collaborative Filtering (CF) methods use an intuition that users with similar interests like similar things. Hence, if two users have very similar history of interacting with a system (purchases, searches, etc.), they probably will like the same things in the future. So, if users A and B liked similar items in the past, it makes sense to recommend to user A items from user B history if user A has not seen or consumed them before. This kind of RSs is widely used in industries, e.g. in e-commerce shops.

CF methods take a matrix of given user-item ratings as the only input. An example of such matrix in the movie domain is presented in Tab. 4.3. We use here 1-5 scale where 1 means *strongly dislike* and 5 *like a lot*. Let us assume that we want to predict Alice interests in Movie5. Each user's preferences are represented by one row in this matrix and consist of user's ratings for each movie that she watched. First, we need to find users with similar to Alice rating history. We call them k nearest neighbors (kNN), where k is the constraint for a number of similar users. The most common similarity measure is Pearson correlation coefficient [86, 100] which is presented in the formula (4.3). This measure considers the fact that users interpret rating scale in different ways. Some of them rate just items that they really like (only ratings 4 and 5) and some of them use the whole rating scale.

$$sim(x,y) = \frac{\sum_{i \in I}(r_{x,i} - \bar{r_x})(r_{y,i} - \bar{r_y})}{\sqrt{\sum_{i \in I}(r_{x,i} - \bar{r_x})^2}\sqrt{\sum_{i \in I}(r_{y,i} - \bar{r_y})^2}} \ , \tag{4.3}$$

where i indicates an item from the set of items I, $r_{x,i}$ is the rating that a user x gave to an item i and the symbol $\bar{r_x}$ corresponds to the average rating of user x.

Another very popular similarity measure is the cosine similarity [96] given by the formula (4.4).

$$cos(x,y) = \frac{x \bullet y}{\|x\| \, \|y\|} \ , \tag{4.4}$$

where $\bullet$ indicates vector dot product and $\|x\|$ is the norm of a user vector x.

In the rest of our example we use Pearson correlation coefficient to find nearest neighbors for Alice. We obtain following results for each user in the rating matrix:

$$sim(\text{Alice}, \text{Bob}) \approx 0.91 \ , \tag{4.5}$$
$$sim(\text{Alice}, \text{Carol}) \approx -0.12 \ , \tag{4.6}$$
$$sim(\text{Alice}, \text{Dave}) \approx 0.24 \ , \tag{4.7}$$
$$sim(\text{Alice}, \text{Eve}) \approx -0.91 \ . \tag{4.8}$$

The most similar user to Alice is Bob, and the least similar is Eve. In this case it make sense to use $k = 2$ nearest neighbors, i.e. Bob and Dave, to predict Alice rating for *Movie5*. For further computation of predicted ratings we use the formula (4.9).

$$pred(x,i) = \bar{r_x} + \frac{\sum_{y \in U} sim(x,y)(r_{y,i} - \bar{r_y})}{\sum_{y \in U} sim(x,y)} \ , \tag{4.9}$$

where U is a set of k nearest neighbors, $sim(x,y)$ is Pearson correlation coefficient and other symbols are as in the equation (4.3). We obtain:

$$pred(\text{Alice}, \text{Movie5}) \approx 3.83 \ . \tag{4.10}$$

Predicted rating is quite high, so in the case of recommending good items, this movie can be included in the list. The described approach is called user-based k nearest neighbors (UserKNN) and was the first known collaborative method [70].

Similarly to CB RSs, also CF methods have their limitations. One of the biggest problem is data sparsity. In the example we missed only one rating, but in reality missing ratings are in majority. Thus, finding similar users in such a sparse matrix is really difficult. Another problem is so called a new item cold-start problem. This is other type of the mentioned before cold-start problem. It occurs when a new item is introduced into a system, and none of the users have rated it yet. In that case, it will never be recommended to any of the users [54].

4.1.3 Knowledge-based Recommender Systems

Besides well-known and widely-used collaborative and content-based recommendation techniques there exist also knowledge-based ones which depend on detailed knowledge about items to be recommended. An example of an item catalog in the digital camera domain is

ID	Price	Mpix	Opt-zoom	LCD-size	Movies	Sound	Waterproof
p_1	148	8.0	4x	2.5	no	no	yes
p_2	182	8.0	5x	2.7	yes	yes	no
p_3	189	8.0	10x	2.5	yes	yes	no
p_4	196	10.0	12x	2.7	yes	no	yes
p_5	151	7.1	3x	3.0	yes	yes	no
p_6	199	9.0	3x	3.0	yes	yes	no
p_7	259	10.0	3x	3.0	yes	yes	no
p_8	278	9.1	10x	3.0	yes	yes	yes

Table 4.4: Exemplary product assortment: digital cameras (from [31]).

presented in Tab. 4.4. Knowledge-based RSs are helpful in situations when other recommendation methods do not perform well, e.g. recommendations of cars, houses or computers. These are the things that we do not buy often and when we do, we know exactly what we want or expect. Additionally, it does not make sense to recommend someone a computer basing on a rating that he gave to some machine two years ago [55].

We distinguish three types of knowledge-based systems, i.e. constraint-based RSs, rule-based (RBR) and case-based (CBR) reasoning. In any case, the recommendation problem consists of selecting items from an item catalog that match the user's needs, preferences, or hard requirements.

Constraint-based RSs try to find the best solution based on user requirements. They solve so called constraint satisfaction problem [30]. In the digital camera example, a user requirement could be "the price should be lower that 200" or "the camera should be suited for sports photography."

In RBR systems the knowledge about items and users' interests is represented in the form of "IF condition THEN action" rules and new problems are answered by reasoning with them. In the recommendation task, when some condition holds the matching rule is fired [49].

CBR systems store knowledge in the casebase in the form of cases. During recommendation task, the cases are compared to user requirements according to some similarity measure. The items suggested by the most (least) similar cases are then tested for success by active user. The process has many iterations and all of them are kept in the casebase as new cases [98].

4.1.4 Hybrid Recommender Systems

The last category of RSs are hybrid approaches. As the name suggests, they combine several algorithm implementations or recommendation components from other categories to minimize the effect of specific problems for each category. There are many ways on how to do that. To combine different kinds of recommenders we need data that are required for them. These requirements are collected in Tab. 4.5. Columns represent types of input data. User profile stands for all information about a user like her age, exact preferences or items rated in the past. Clusters of similar users and information how to compare users are examples of community data. Product features are all available item attributes. Knowledge models represent all additional information that helps to map user constraints with product features.

Paradigm	User profile	Community data	Product features	Knowledge models
Collaborative	yes	yes	no	no
Content-based	yes	no	yes	no
Knowledge-based	yes	no	yes	yes

Table 4.5: Input data requirements of recommendation algorithms (from [56]).

Recommenders can be combined in different orders sequentially or can be run in parallel and then results may be merged. However, these approaches are not important for this dissertation. More information can be found in [56].

4.2 Context-aware Recommender Systems

Context-aware recommender systems (CARS) are a particular category of recommender systems which exploit contextual information to provide more adequate recommendations. For example, a restaurant recommendation for a Saturday evening with your friends should be different from one suggested for a workday lunch with co-workers.

When we incorporate additional contextual parameters we achieve a multidimensional system, because each context variable (like time, weather or mood) is treated as another dimension. The new rating function R would be described as follows [91]:

$$R : Users \times Items \times Contexts \mapsto Ratings \ , \tag{4.11}$$

where $Contexts$ denotes the contextual information specific for an application domain.

Adomavicius and Tuzhilin [5] distinguish tree main types of context-aware recommender systems, i.e. contextual pre-filtering, contextual post-filtering and contextual modeling. The paradigms differ in the way they incorporate context in the recommendation process, which is shown in Fig. 4.1.

In contextual pre-filtering, we first do selection of preferences from $Users \times Items \times Contexts \times Ratings$ space, taking into account only relevant context. Thus, we filter an initial set of ratings and return the contextualized data. After this preparation any known two-dimensional recommender algorithm could be used to predict further users preferences. Subsequently, we choose just those ratings that concern a specific user and return recommendation based on the highest predicted ratings. This last step is the same as in traditional two-dimensional RSs. The whole process is shown in Fig. 4.1a.

In contrast, contextual post-filtering applies context after traditional recommendation process. It means that from a predicted set of recommendations we select just those that match the context under current consideration. Fig. 4.1b presents the details. Contextual pre- and post- filtering have a big advantage that they can be used with every known two-dimensional recommendation algorithm.

Contextual modeling differs radically from previously described paradigms. In this kind of recommenders we do not filter anything, but we incorporate a context in a prediction model. Usually, well-known machine learning algorithms, like Random Forest or Support Vector Machine, are applied on multidimensional $Users \times Items \times Contexts \times Ratings$ space. The

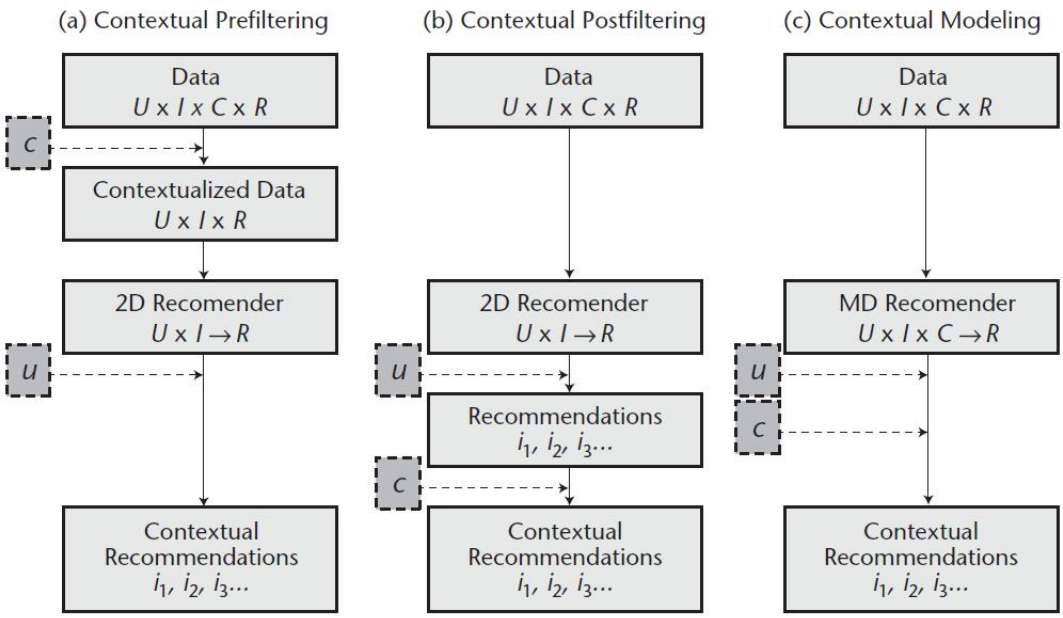


Figure 4.1: Paradigms for incorporating context in recommender systems (from [5]).

recommendations are achieved directly from the model, taking into account current user-context situation, which is presented in Fig. 4.1c.

4.3 Review of Recommendation Approaches

In this section we describe all algorithms that are used for comparison and evaluation performed in further chapters. They are representatives of different kinds of RSs describes in Sections 4.1 and 4.2.

Besides sophisticated algorithms, there exist also some naïve baselines like Random Guess or User Average. Both baselines are used for rating prediction task. Random Guess uses a random number generator to "guess" a user rating for some item. Because there is no logic here, the method usually gives a huge prediction error. User Average considers the fact that users differ from each other according to the way in which they use rating scale. Prediction is made with average value of all ratings that a user already gave to consumed items.

4.3.1 *k* Nearest Neighbors

User-based k nearest neighbors algorithm has been already described in Section 4.1.2. Here, we focus only on item-based k nearest neighbors method [25]. Let us consider again an example from Tab. 4.3. This time we will not look for users that are similar to Alice, but items that are similar to *Movie5*, for which we want to predict Alice rating. We observe that *Movie3* and *Movie1* are somehow similar to *Movie5*. The idea is to simply check the ratings that Alice gave to these similar item. Alice rating for *Movie1* and *Movie3* are 3 and 4, respectively. Thus, the recommender computes a weighted average of these values and returns something between 3 and 4. It should be noticed that similar result was obtained with user-based kNN algorithm.

4.3.2 SVD and SVD++

Matrix factorization (MF) methods are a special case of collaborative filtering approaches which helps to uncover latent features that explain observed ratings. MF models map users and items to a joint latent factor space of a dimensionality f. In a movie domain, these factors could measure some obvious features like genre and soundtrack music, or completely uninterpretable features.

SVD is an example of MF method which is based on Singular Value Decomposition method. The model associates each user u with a user-factors vector $\mathbf{p}_u \in \mathbb{R}^f$, and each item i with an item-factors vector $\mathbf{q}_i \in \mathbb{R}^f$. The dot product $\mathbf{q}_i^T \mathbf{p}_u$ captures the overall interest of the user u in characteristics of the item i. A rating is computed according to equation (4.12).

$$\hat{r}_{ui} = \mu + b_i + b_u + \mathbf{q}_i^T \mathbf{p}_u \ , \tag{4.12}$$

where μ is global average of all ratings, b_i and b_u are item and user biases respectively, i.e. they determine how much item and user ratings differ from the average [73].

SVD++ is very similar to SVD. However, it incorporates additional information about items that user u rated, independently of the rating score. We denote this set of items by symbol $R(u)$. A second item-factors vector $\mathbf{y}_i \in \mathbb{R}^f$ is added to characterize users based on the set of items that they rated. A rating is computed according to equation (4.13) [71].

$$\hat{r}_{ui} = \mu + b_i + b_u + \mathbf{q}_i^T \left(\mathbf{p}_u + |R(u)|^{\frac{1}{2}} \sum_{j \in R(u)} y_j \right) \ . \tag{4.13}$$

4.3.3 Time SVD++

The main reason why Time SVD++ was created is the fact that user preferences evolve over time. Thus, it make sense to incorporate time factor into the computation. Three effects that change over time were identified, i.e. item bias $b_i(t)$, user bias $b_u(t)$ and user preferences $\mathbf{p}_u(t)$. The first can be justified by the fact that an item's popularity vary over time. Let us consider the movie domain again. When a movie is new and just had a premier in cinemas, it could be very popular because people are talking about it. The same happens with user bias, i.e. a user could rank a movie differently (e.g. 3 stars instead of 4 that he gave earlier) after watching other (maybe better?) movie.

User preferences are the third thing that changes over time. A user who liked drama movies two years ago can now be a fan of animated movies, because she became a parent. This is natural part of life that with changing circumstances we are changing and our preferences are also changing.

In Time SVD++, a current rating is computed according to equation (4.14).

$$\hat{r}_{ui} = \mu + b_i(t) + b_u(t) + \mathbf{q}_i^T \left(\mathbf{p}_u(t) + |R(u)|^{\frac{1}{2}} \sum_{j \in R(u)} y_j \right) \ , \tag{4.14}$$

where t denotes a time factor and other symbols mean the same as in equation (4.13) [72].

4.3.4 BPR: Bayesian Personalized Ranking

Bayesian Personalized Ranking (BPR) is a collaborative method designed for a task of recommending a list of top k items [90]. As it was mentioned in previous sections, collaborative methods use a ratings matrix $\mathbf{S} \subseteq U \times I$, where U is a set of all users and I is a set of all items, for making recommendations. This matrix consists of known user ratings and missing values (denoted by ?). Usually, these missing values are replaced with zeros, which could be interpreted as negative feedback. The main idea in BPR method is that missing values can also describe items that are unknown to a user (and may be she would like to get to know them). Thus, the task of RS is now to provide the user with a personalized total ranking $>_u \subset I^2$ of all items, where $>_u$ has to meet the properties of a total order:

$$\forall i, j \in I : i \neq j \Rightarrow i >_u j \vee j >_u i \ , \tag{4.15}$$

$$\forall i, j \in I : i >_u j \wedge j >_u i \Rightarrow i = j \ , \tag{4.16}$$

$$\forall i, j, k \in I : i >_u j \wedge j >_u k \Rightarrow i >_u k \ . \tag{4.17}$$

For each user, parts of $>_u$ are reconstructed from the matrix $\mathbf{S}$. This reconstruction follows an assumption that the user prefers items which she already ranked. For two items that both have been seen by a user, any preference can be inferred. The same is true for two items that a user has not seen yet. It is shown if Fig. 4.2. On the left side of the figure, the traditional matrix of users ratings is shown. The symbol "+" denotes that a user interact with an item, while "?" stands for missing value. On the right side of Fig. 4.2, item to item preference matrices obtained by BPR method are presented. For each user we obtain one matrix where columns and rows represent same items (columns are denoted by i and rows by j, but index is the same for the same item). It does not make sense to check preference

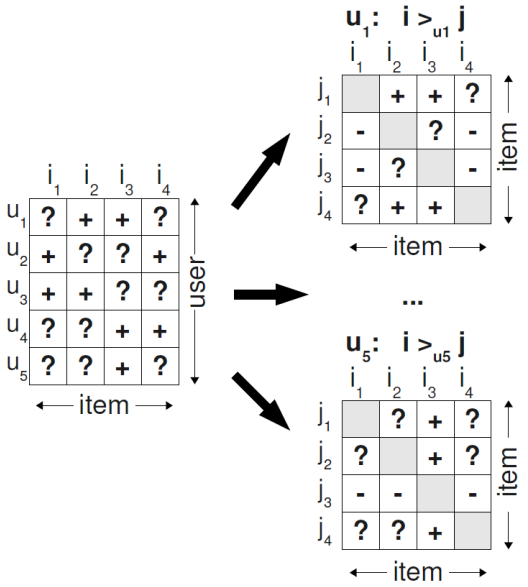


Figure 4.2: Partial reconstruction of a personalized ranking $>_u$ from a ratings matrix $\mathbf{S}$ in BPR method (from [90]).

between same items, so a diagonal of a matrix is grayed out. Here, the symbol "+" denotes that a user prefers an item i over an item j, "-" means that she prefers j over i and "?" is unknown value, i.e. values for both items in the source matrix were the same.

In further recommendation task, triples (u, i, j) are used, where user u is assumed to prefer i over j. In [90], the recommendation task has been reduced to an optimization problem and solved using a stochastic gradient descent algorithm that chooses triples randomly with a uniform distribution.

4.3.5 WRMF: Weighted Regularized Matrix Factorization

Weighted Regularized Matrix Factorization (WRMF) is a collaborative method designed for generating a list of top-k recommendations [47]. Similarly to BPR, this method also assumes that a user prefers items that she consumed to other ones, i.e. those that she did not consume. However, here a user's row from the ratings matrix is translated into a set of binary variables p_{ui}, as shown in formula (4.18).

$$p_{ui} = \begin{cases} 1, & r_{ui} > 0 \\ 0, & r_{ui} = 0 \end{cases}, \tag{4.18}$$

where u is a user, i is an item and r_{ui} is an entry from the ratings matrix that corresponds to a user u and an item i. These variables indicate user's preferences in items. If a user u consumed an item i ($r_{ui} > 0$), then we assume that u likes i ($p_{ui} = 1$). On the other hand, if u never consumed i, we assume no preference ($p_{ui} = 0$). Because preferences are assumed, a confidence level was introduced into a model. It is represented by a set of variables c_{ui} which measure confidence in observing p_{ui}:

$$c_{ui} = 1 + \alpha r_{ui} \ , \tag{4.19}$$

where α is a constant dependent on the dataset.

The goal is to find a vector $\mathbf{x}_u \in \mathbb{R}^f$ for each user u, and a vector $\mathbf{y}_i \in \mathbb{R}^f$ for each item i that will factor user preferences. In other words, preferences are assumed to be the inner products:

$$p_{ui} = \mathbf{x}_u^T \mathbf{y}_i \ . \tag{4.20}$$

These factors are computed by solving the following optimization problem:

$$\underset{x_*, y_*}{\text{minimize}} \quad \sum_{u,i} c_{ui}(p_{ui} - \mathbf{x}_u^T \mathbf{y}_i)^2 + \lambda \left(\sum_u \|\mathbf{x}_u\|^2 + \sum_i \|\mathbf{y}_i\|^2 \right) \ , \tag{4.21}$$

where λ is a regularization parameter.

4.3.6 Latent Dirichlet Allocation

Latent Dirichlet Allocation (LDA) is a generative probabilistic model designed for a collection of discrete data like a text corpora [13]. The basic idea is that documents are related to latent topics, which are characterized by a distribution over words.

We will use a terminology from the work of Blei et al. [13] and in the end of this section we will translate it into the recommendation problem.

The basic unit of discrete data is a *word*, which is contained in some vocabulary V. A sequence of N words is a *document* denoted by $\mathbf{w} = (w_1, w_2, ..., w_N)$, where w_n is the n-th word in the sequence. A collection of M documents is a *corpus* denoted by $D = \{\mathbf{w}_1, \mathbf{w}_2, ..., \mathbf{w}_M\}$. As assumed in LDA, each document $\mathbf{w}$ in a corpus D is generated by the following process. First, random variable θ is sampled from a Dirichlet$(\alpha, \alpha_2, ..., \alpha_k)$ distribution. θ lies in the $(k-1)$-dimensional simplex (a k-vector θ lies in the $(k-1)$-simplex if $\theta_i \geq 0$, $\sum_{i=1}^k \theta_i = 1$). Then, for each word, a topic $z_n \in 1, 2, ..., k$ is sampled from a Multinomial(θ) distribution. Finally, each word w_n is sampled from $p(w_n|z_n, \beta)$, a multinomial probability conditioned on the topic z_n. The distribution of a document is given by equation (4.22).

$$p(\mathbf{w}|\alpha, \beta) = \int p(\theta|\alpha) \left(\prod_{n=1}^{N} \sum_{z_n} p(z_n|\theta) p(w_n|z_n, \beta) \right) d\theta \ , \tag{4.22}$$

where $p(\theta|\alpha)$ is Dirichlet distribution, $p(z_n|\theta)$ is a multinomial distribution parametrized by θ, and $p(w_n|z_n, \beta)$ is a multinomial distribution over the words. This model is parametrized by the k-dimensional Dirichlet parameter $\alpha = (\alpha_1, \alpha_2, ..., \alpha_k)$ and a $k \times |V|$ matrix β. Parameters α and β have to maximize the log-likelihood of the data:

$$l(\alpha, \beta) = \sum_{d=1}^{M} \log\, p(\mathbf{w}_d|\alpha, \beta) \ . \tag{4.23}$$

The graphical model representation of LDA is presented in Fig. 4.3. The outer box represents documents, while the inner box represents the iterative selection of topics and words within a document.

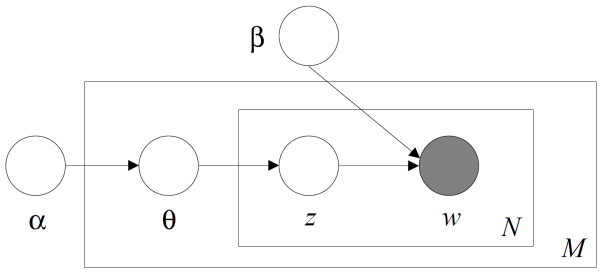


Figure 4.3: Graphical model representation of LDA (from [13]).

To apply this method to RSs, we have to treat users as documents and items as words. Thus, a set of all users corresponds to a corpus. In this way, we can obtain a distribution of user preferences over items and use it for further predictions.

4.3.7 SLIM: Sparse Linear Methods

Sparse Linear Method (SLIM) was designed for generating a list of top-k recommendations. Let us assume that we have an $m \times n$ matrix $\mathbf{R}$ that consists of ratings r_{ij} that the i-th user gave to the j-th item. The recommendation score on an unrated j-th item of i-th user is calculated as a sparse aggregation of items that have been rated by i-th user. It is shown in equation (4.24).

$$\hat{r}_{ij} = \mathbf{r}_i^T \mathbf{w}_j \ , \tag{4.24}$$

where $r_{ij} = 0$ and $\mathbf{w}_j$ is a sparse n-size column vector of aggregation coefficients.

To compute the values for $\mathbf{w}_j$, the following optimization problem has to be solved:

$$\begin{aligned} \underset{\mathbf{w}_j}{\text{minimize}} \quad & \frac{1}{2} \left\| \mathbf{r}_j - \mathbf{R}\mathbf{w}_j \right\|_2^2 + \frac{\beta}{2} \left\| \mathbf{w}_j \right\|_2^2 + \lambda \left\| \mathbf{w}_j \right\|_1 \\ \text{subject to} \quad & \mathbf{w}_j \geq 0 \\ & w_{jj} = 0 \ . \end{aligned} \tag{4.25}$$

In equation (4.25), $\mathbf{r}_j$ is the j-th column of $\mathbf{R}$, the constants β and λ are regularization parameters, $\left\| \mathbf{w}_j \right\|_2 = \sqrt{\sum_{i=1}^{n} |w_{ij}|^2}$ is l_2-norm of vectors, and $\left\| \mathbf{w}_j \right\|_1 = \sum_{i=1}^{n} |w_{ij}|$ is l_1-norm of vector $\mathbf{w}_j$. Because this problem is represented on columns instead of the whole matrix $\mathbf{W}$, it can be easily parallelized [82].

4.3.8 Contextual SLIM

Contextual SLIM (CSLIM) is a context-aware extension of the SLIM algorithm. A contextual situation is denoted by a binary vector. Let us assume that all contextual conditions can be represented by {`time`=*weekday*, `time`=*weekend*, `location`=*home*, `location`=*work*}. The vector $\mathbf{c} = \langle 0, 1, 1, 0 \rangle$ indicates that the current contextual situation is {`time`=*weekend*, `location`=*home*}. CSLIM follows the idea of an aggregation of users' ratings on other items, and add contextual factors into this aggregation. In the case when no other items were ranked in a certain context, the rating is estimating based on user's non-contextual ratings on this item [111].

4.3.9 FISM: Factored Item Similarity Model

Factored Item Similarity Model (FISM) is a collaborative method designed for generating a list of top-k recommendations [58]. The recommendation score $\hat{r}_{ui}$ for a user u on an unrated item i is calculated as an aggregation of the items that have been rated by u with the corresponding product of $\mathbf{p}_j$ latent vectors from $\mathbf{P}$ and the $\mathbf{q}_i$ latent vector from $\mathbf{Q}$, as shown if formula:

$$\hat{r}_{ui} = b_i + b_u + (n_u)^{-\alpha} + \sum_{j \in R_u^+} \mathbf{p}_j \mathbf{q}_i^T \ , \tag{4.26}$$

where R_u^+ is the set of items rated by user u, $\mathbf{p}_j$ and $\mathbf{q}_i$ are the learned item latent factors, n_u is the number of items rated by u, and α is a user specified parameter between 0 and 1. In FISM, matrices $\mathbf{P}$ and $\mathbf{Q}$ are learned by minimizing the following regularized optimization problem:

$$\underset{\mathbf{P},\mathbf{Q}}{\text{minimize}} \quad \frac{1}{2} \sum_{u \in U} \sum_{j \in R_u^+, k \in R_u^-} \|(r_{uj} - r_{uk})(\hat{r}_{uj} - \hat{r}_{uk})\|_F^2 + \frac{\beta}{2}(\|\mathbf{P}\|_F^2 + \|\mathbf{Q}\|_F^2) + \frac{\gamma}{2} \|\mathbf{b_i}\|_2^2 \ , \tag{4.27}$$

where U is a set of all users, R_u^+ is the set of items seen by user u, R_u^- is the set of items unseen by user u, a vector $\mathbf{b}_i$ corresponds to the vector of item biases, and β and γ are the regularization weights for latent factor matrices and item bias vector, respectively. Note that there are no user bias b_u terms, since the terms cancel out when taking the difference of the ratings.

4.3.10 Context-Aware Splitting Approaches

One of the possibilities for contextual pre-filtering are Context-Aware Splitting Approaches (CASA). We could distinguish three kinds of them, i.e. item splitting, user splitting and UI splitting that combines the first two [110].

An item splitting method was proposed by Baltrunas and Ricci [9]. The underlying idea of this method is that one item may vary from the user point of view in different contexts, therefore it could be useful to split it into two items. To better understand, consider the point of interests (POI) recommendation example shown in Tab. 4.6. There are one user and one item. We have two ratings in the training data and one unknown that we want to predict (it is denoted by "?"). There are three contextual parameters: `time`, `weather` and `companion`. Item splitting tries to find a condition (contextual variable and its value) on which to split each item. For contextual parameter weather we have three values: *sunny*, *cloudy* and *rainy*. Consequently, we have three possibilities for splitting with this parameter, i.e. "sunny and not sunny", "cloudy and not cloudy" and "rainy and not rainy". Let us assume that the best criterion to split item p1 in Tab. 4.6 is "`companion` = children and not children". Hence we represent it as two items: p11 (POI visited with children) and p12 (POI visited with other companions than children). The resulting rating matrix is presented in Tab. 4.7. After such data preparation it is possible to use any two-dimensional recommendation algorithm.

Analogously, we could split user into two users based on the contextual condition. This approach is also called *micro profiles* [8]. The UI splitting uses both kinds of splits, for items and for users. It should be noticed that the best contextual factor for splitting users and items could be, and usually is, different, i.e. we do not use the same contextual condition to split users and items.

User	Item	Rating	Time	Weather	Companion
u1	p1	2	Weekend	Cloudy	Children
u1	p1	4	Weekend	Sunny	Girlfriend
u1	p1	?	Weekday	Rainy	Friend

Table 4.6: POI ratings in contexts (from [110]).

User	Item	Rating
u1	p11	2
u1	p12	4
u1	p12	?

Table 4.7: Rating matrix transformed by *item splitting* (from [110]).

4.4 User Preferences Models

In the previous section we have presented recommendation approaches that work with user representations in the form of a vector or a matrix. The ratings matrix is the most common in RSs and gives a uniform user representation for different kinds of recommendation algorithms. These two aspects can be seen as most important advantages of a matrix user model. However, it has also some disadvantages. First of all, the ratings matrix is usually very sparse and very big. Thus, it is hard to move it from one RS to another. As was mentioned in previous section, it is typically uncontextual and adding each contextual parameter increases dimensionality of a matrix. Last disadvantage of a ratings matrix is its separation of content information about items.

In this section we discuss other user preferences models that are important for this dissertation.

4.4.1 CP-nets

To describe CP-nets, first we need to introduce *ceteris paribus* preferential statements [16]. Let us start with an example given by Hansson [43]:

> *When discussing with my wife what table to buy for our living room, I said: "A round table is better than a square one." By this I did not mean that irrespectively of their other properties, any round table is better than any square-shaped table. Rather, I meant that any round table is better (for our living room) than any square table that does not differ significantly in its other characteristics, such as height, sort of wood, finishing, price, etc. This is preference* ***ceteris paribus*** *or "everything else being equal". Most of the preferences that we express or act upon seem to be of this type.*

This example gives very good intuitive understanding of what a *ceteris paribus* preferential statement is. The formal definition of CP-net needs to be preceded by explanation of additional notions.

We assume that the world can be in one of a number of *states* S and at each state s there are number of *actions* A_s that can be performed. Each action, when performed at a state,

has a specific *outcome*. A set of all outcomes is denoted by O. A *preference ranking* is a total preorder $\succeq$ over the set of outcomes. $o_1 \succeq o_2$ means that outcome o_1 is equally or more preferred to a user than o_2.

Let us consider a set of *variables* (*features*) $\mathbf{V} = X_1, ..., X_n$ over which a user has preferences. Each variable X_i is associated with a domain $Dom(X_i) = x_1^i, ..., x_{n_i}^i$ of *values* it can take. An *assignment* $\mathbf{x}$ of values to a set $\mathbf{X} \subseteq \mathbf{V}$ of variables is a function that maps each variable in $\mathbf{X}$ to an element of its domain. We denote the set of all assignments to $\mathbf{X} \subseteq \mathbf{V}$ by $Asst(\mathbf{X})$. If $\mathbf{x}$ and $\mathbf{y}$ are assignments to disjoint sets $\mathbf{X}$ and $\mathbf{Y}$ ($\mathbf{X} \cap \mathbf{Y} = \emptyset$) respectively, we denote the combination of $\mathbf{x}$ and $\mathbf{y}$ by $\mathbf{xy}$.

Definition 4.4.1. A set of variables $\mathbf{X}$ is *preferentially independent* of its complement $\mathbf{Y} = \mathbf{V} - \mathbf{X}$ iff, for all $\mathbf{x}_1, \mathbf{x}_2 \in Asst(\mathbf{X})$ and $\mathbf{y}_1, \mathbf{y}_2 \in Asst(\mathbf{Y})$, we have

$$\mathbf{x}_1\mathbf{y}_1 \succeq \mathbf{x}_2\mathbf{y}_1 \text{ iff } \mathbf{x}_1\mathbf{y}_2 \succeq \mathbf{x}_2\mathbf{y}_2 \ . \tag{4.28}$$

If the relation (4.28) holds, we say that $\mathbf{x}_1$ is preferred to $\mathbf{x}_2$ *ceteris paribus*.

Definition 4.4.2. Let $\mathbf{X}$, $\mathbf{Y}$ and $\mathbf{Z}$ be nonempty sets that partition $\mathbf{V}$. $\mathbf{X}$ is *conditionally preferentially independent* of $\mathbf{Y}$ given an assignment $\mathbf{z}$ to $\mathbf{Z}$ iff, for all $\mathbf{x}_1, \mathbf{x}_2 \in Asst(\mathbf{X})$ and $\mathbf{y}_1, \mathbf{y}_2 \in Asst(\mathbf{Y})$, we have

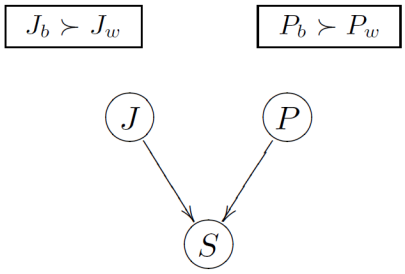
$$\mathbf{x}_1\mathbf{y}_1\mathbf{z} \succeq \mathbf{x}_2\mathbf{y}_1\mathbf{z} \text{ iff } \mathbf{x}_1\mathbf{y}_2\mathbf{z} \succeq \mathbf{x}_2\mathbf{y}_2\mathbf{z} \ . \tag{4.29}$$

Each user should be able to identify a set of *parent* variables $Pa(X_i)$ that can affect her preference over various values of X_i. That is, given a particular value assignment to $Pa(X_i)$, a user is able to determine a preference order for the values of X_i, assuming that all other things are equal. It should be noticed, that X_i is conditionally preferentially independent of $\mathbf{V} - (Pa(X_i) \cup \{X_i\})$.

Definition 4.4.3. A *CP-net* over variables $\mathbf{V} = X_1, ..., X_n$ is a directed graph G over $X_1, ..., X_n$ whose nodes are annotated with conditional preference tables $CPT(X_i)$ for each $X_i \in \mathbf{V}$. Each conditional preference table $CPT(X_i)$ associates a total order $\succ_u^i$ with each assignment $\mathbf{u}$ of X_i's parents $Pa(X_i) = \mathbf{U}$.

Let us consider an example about choosing an outfit for some formal evening. We assume that John has only black and white jackets and pants and only white and red shirts in his wardrobe. He prefers to wear black clothes, so he always chooses black pants and jacket (independently of each other), if they are clean. If not, he has to wear white clothes. However, John does not like to look like for a funeral or be whole in white, so he chooses a red shirt if his jacket and pants are in the same color. He also does not like to look too flashy, so if his pants and jacket have different colors, he prefers to wear a white shirt. We see that his choices for a shirt are strongly dependent of a color of a jacket and pants that he chose. Thus, for a shirt variable S, we can define its parents $Pa(S) = \{J, P\}$, where J and P denote a jacket and pants variables, respectively. This CP-net with corresponding conditional preference tables is presented in Fig. 4.4. Please note that presented John's preferences did not consider different ties or shoes. Contrary, they are assumed to be the same in each case.

Main advantages of this model are its compactness and portability between different systems (RSs or even others). CP-nets are also very clear and intuitive for people, so a user is



$J_b \wedge P_b$	$S_r \succ S_w$
$J_w \wedge P_b$	$S_w \succ S_r$
$J_b \wedge P_w$	$S_w \succ S_r$
$J_w \wedge P_w$	$S_r \succ S_w$

Figure 4.4: The CP-Net for choosing an outfit for a formal evening (from [16]).

able to express her preferences easily. If we use CP-nets for recommendations, it is easy to explain to a user how decisions have been made. However, CP-nets have also their disadvantages. It is very hard problem (probably unfeasible) to find a series of dependent variables, which is crucial to build the structure of a CP-net. The *ceteris paribus* assumption in combination with sparsity of a ratings matrix gives almost no chances to find any user preference. The last disadvantage is that CP-net allows to store only preferences of a single user.

4.4.2 Ontologies

An interesting direction in the field of recommender systems is to apply ontologies to capture user preferences. The simplest way to do that is mapping both users and items to some taxonomy or domain ontology. Items can be recommended to a user by using some similarity measure. This solution is very similar to CB RSs and was reported by Maidel et al. [78], Rack et al. [89] and Middleton et al. [81]. The last RS uses an ontological user profile to recommend research papers. Both research papers and user profiles are represented through a taxonomy of topics and the recommendations are generated considering topics of interest for the user and papers classified to those topics.

Slightly different approach to user modeling with ontologies was used in News@hand [18]. The news items are automatically and periodically retrieved via RSS feeds and annotated with semantic concepts from system domain ontologies. During an interaction with a user a set of weighted concepts from the domain ontologies is collected. A user context is represented by this set. The importance of concepts fades away with the time by a decay factor. This helps to keep the user context up to date. Existing relations between concepts in the ontologies are used to find semantic paths linking preferences to a context.

A multi-dimensional ontology model was used in a context-aware system which recommends Web services [92]. The multi-dimensional ontology model consists of three independent ontologies: a user context ontology, a Web service ontology and an application domain ontology, which are combined into one ontology by some relations between concepts from the three ontologies. Data from a WSDL file for a Web service are automatically added into the

Web service ontology during the registration process. The user must specify the name, birth date, sex and occupation to build her profile. The user context ontology consists of those parameters and a list of interests. Every item in the list has a level of interest property, which is used to assign a weight to the item during the recommendation process.

An interesting approach was proposed by Hawalah and Fasli [44]. Proposed context taxonomies have been already described in Section 3.3. Besides context taxonomies, this approach uses a reference ontology for building contextual personalized ontological profiles. The key feature of this profile is the possibility of assigning user interests in groups, if these interests are directly associated with each other by a direct relation, are sharing the same super-class or the same property.

The main advantages of representing user preferences with ontologies are a unified terminology and better matching between users and items than in other models. However, there are also disadvantages of ontological user models. First of all, a context is not directly connected to a user model, which can be hard to manage. Due to domain ontologies/taxonomies, the model is strongly domain-dependent. It means that for each domain the same effort is needed to build a model. The last disadvantage is that typically we have to create a new recommendation algorithms to cope with the ontological user representation.

4.5 Methods of Evaluating Recommender Systems

This section is focused on evaluation of RS and measures used for this purpose. The section is divided into two parts, one for each type of recommendation tasks which were explained in Section 4.1. Because of a specificity of these tasks and differences in an output of algorithms, they require different evaluation measures, described later in this section.

Independently from recommendation tasks, there exist two ways for evaluating RSs, i.e. offline and online experiments. In offline experiments historical ratings are used. Typically, data are split into two sets, i.e. training set and test set. A model is trained on the first set and then evaluation measures are computed on the second one. Sometimes, additional set is used for verification to avoid over-fitting to the data.

Online experiments require people involvement. Users verify the system while using it. It is possible to analyze usefulness of the system by recording and analyzing user actions with received recommendations. However, usually users are asked to answer some questions about their experience with the system.

Since online studies strongly depend on the specific kind of RS and the domain of recommended items, this section is focused only on offline experiments and measures related to them.

4.5.1 Predicting Ratings

In the predicting ratings task, the system provides a set of predicted ratings for a set of input items - one rating per each item. This task is evaluated on the accuracy of these predictions. Typically, this accuracy is measured by means of some statistical error. The most popular errors used for that purpose are the Mean Absolute Error (MAE) and the Root of the Mean Square Error (RMSE). If $\hat{r}_{u,i}$ is the predicted rating for user u over item i from the test set T and $r_{u,i}$ is the actual rating, we define MAE and RMSE as in formulas (4.30) and (4.31).

Item	Rating
i1	2
i2	4
i3	5
i4	3

Table 4.8: An example of a test set.

Item	RS1	RS2
i1	4	2
i2	2	4
i3	3	1
i4	3	3

Table 4.9: Ratings predicted by two RSs: RS1 and RS2 for the test set from Tab. 4.8.

The values of actual ratings $r_{u,i}$ are usually known because they are included in a test set in offline experiments.

$$\text{MAE} = \frac{1}{|T|} \sum_{(u,i)\in T} |\hat{r}_{u,i} - r_{u,i}| \ , \tag{4.30}$$

$$\text{RMSE} = \sqrt{\frac{1}{|T|} \sum_{(u,i)\in T} (\hat{r}_{u,i} - r_{u,i})^2} \ . \tag{4.31}$$

Let us consider an example. Assume that we have four items in the test set (given in Tab. 4.8 and two RSs. Values predicted by both recommenders are presented in Tab. 4.9. When we apply values for the first RS from Tables 4.8 and 4.9 to formulas (4.30) and (4.31) we obtain following results:

$$\text{MAE} = \frac{1}{4}(|4-2| + |2-4| + |3-5| + |3-3|) = 1.5 \ , \tag{4.32}$$

$$\text{RMSE} = \sqrt{\frac{1}{4}(|4-2|^2 + |2-4|^2 + |3-5|^2 + |3-3|^2)} \approx 1.7 \ . \tag{4.33}$$

And for the second RS:

$$\text{MAE} = \frac{1}{4}(|2-2| + |4-4| + |1-5| + |3-3|) = 1 \ , \tag{4.34}$$

$$\text{RMSE} = \sqrt{\frac{1}{4}(|2-2|^2 + |4-4|^2 + |1-5|^2 + |3-3|^2)} = 2 \ . \tag{4.35}$$

From this example we see that RMSE disproportionately penalizes large errors in comparison with MAE, because it would prefer the first system from our example, while MAE would prefer the second one [95].

4.5.2 Recommending Good Items

In the recommending good items task, the system does not try to predict any specific ratings. Instead, RS proposes a list of k best items that a user might be interested in. Since this task is more complicated, we can not focus only on accuracy measures. Thus, many different measures were proposed. In this section the most important ones are described.

In this task, an accuracy is understood by means of classical information retrieval measures: *precision* and *recall*. *Precision* reflects how many of relevant items appeared in the recommendation list, while *recall* shows what percentage of relevant items was recommended. The corresponding formulas are as follows:

$$\text{precision} = \frac{|\{\text{relevant items}\} \cap \{\text{recommended items}\}|}{|\{\text{recommended items}\}|} \ , \tag{4.36}$$

$$\text{recall} = \frac{|\{\text{relevant items}\} \cap \{\text{recommended items}\}|}{|\{\text{relevant items}\}|} \ . \tag{4.37}$$

Recommended items are those which appear in the recommendation list. Items that are interesting to a user are marked as *relevant items*. Items rated positively by a user in the past (from the test set) are usually used as *relevant items* in offline experiments.

Typically we compute *precision* and *recall* at some cut-off k of the list, e.g. top 10 recommendations, when k is equal to 10. Then, we call them *precision@k* and *recall@k* respectively.

F-measure (or *F1-score*) of a recommendation list is a harmonic mean of its precision and recall and is given by the formula:

$$\text{F} = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \ . \tag{4.38}$$

The value of this measure is high only when values of both, precision and recall are high. The *F-measure* is equal to 0 when no relevant items have been recommended, and it is equal to 1 if all recommended items are relevant and all relevant items have been recommended.

Besides *precision* measure, we also use *average precision* (AP) given by the formula (4.39).

$$\text{AP} = \frac{\sum_{k=1}^{n} (\text{precision@}k \cdot \text{rel}(k))}{|\{\text{relevant items}\}|} \ , \tag{4.39}$$

where n is the size of a recommendation list and $\text{rel}(k)$ is an indicator function equaling 1 if the item at a position k in the list is relevant, zero otherwise [104].

In order to not report many numbers, typically only *a mean of average precision* (MAP) is reported and compared. Assuming that we have N recommendation lists, MAP is computed as follows:

$$\text{MAP} = \frac{\sum_{l=1}^{N} \text{AP}(l)}{N} \ . \tag{4.40}$$

When MAP value is equal to 0.5, it means that on the average every second item in the list provided by RS is considered relevant by a user. Obviously, $MAP = 1$ means that all items in the list are interesting to a user and $MAP = 0$ means that none of items is interesting to a user.

To evaluate quality of an RS, it is also important at what positions the relevant items are placed. To this purpose we can use *mean reciprocal rank* (MRR) which is the average of the reciprocal ranks of results for N recommendation lists (4.41).

$$\text{MRR} = \frac{1}{N}\sum_{l=1}^{N}\frac{1}{\text{rank}_l} \ , \tag{4.41}$$

where rank_l refers to the rank position of the first relevant item for the l-th recommendation list. The higher the position of the first relevant item is, the higher the value of *MRR* is.

Another measure that takes into account a position of an item in the recommendation list is *Discounted Cumulative Gain* (DCG). Each user u has some level of interest in an item i placed at the position k in a recommendation list. We call this level of interest a *gain* and denote by g_{ui_k}. Assuming that we have N users and n items in each list, *DCG* is defined as:

$$\text{DCG} = \frac{1}{N}\sum_{u=1}^{N}\sum_{k=1}^{n}\frac{g_{ui_k}}{max(1, log_b k)} \ , \tag{4.42}$$

where b is a free parameter varying from 2 to 10. *Normalized Cumulative Discounted Gain* (nDCG) is a normalized version of *DCG*, since results sets could vary a lot between recommendation systems and make it hard to compare. The *nDCG* is given by:

$$\text{nDCG} = \frac{\text{DCG}}{\text{DCG}^*} \ , \tag{4.43}$$

where DCG^* is the ideal DCG, where items in the recommendation list are sorted according to relevance to a user [57].

Besides the accuracy and the rank of an item in a recommendation list, other factors are important to assure a user satisfaction. One of such factors is *novelty* [20] which expresses how much items from the list are unknown to a user. The measure is given by a formula:

$$novelty = \frac{1}{k}\sum_{i \in R_{u,k}} log_2(pop(i)) \ , \tag{4.44}$$

where u denotes a user, k is the size of a recommendations list $R_{u,k}$, i denotes an item and $pop(i)$ is its popularity, i.e. a number of users who ranked item i normalized by the number of users. It does not matter if the rating given by a user is positive or negative.

It is also very important that items in the list differ from each other, since a user will not find a recommendation system useful if it always recommends the same or similar things. Smyth and McClave [99] proposed a *diversity* measure, i.e. Intra-List Diversity (ILD) that computes the average distance between each couple of items in the list R:

$$ILD(R) = \frac{1}{|R|\,(|R|-1)}\sum_{i,j \in R, i \neq j}(1 - sim(i,j)) \ , \tag{4.45}$$

where i, j are items. The *sim* function is configurable and application dependent.

One of the most interesting measures is *serendipity* which catches the essence of RSs. According to the Oxford dictionary[2], *serendipity* is "the occurrence and development of events

[2] `https://en.oxforddictionaries.com/definition/serendipity`

by chance in a happy or beneficial way". However, the common definition of serendipity in recommender systems does not exist yet, since it is challenging to say which items are serendipitous and why [48].

Ziegler et al. described serendipitous items as those with a low popularity [113]. Results obtained by Maksai et al. [79] confirmed this intuition. They have proved that the most popular items have serendipity equal to zero. Further, we will refer to Ziegler measure as *expectedness* and use the following formula to describe it:

$$expectedness = \frac{1}{k}\sum_{i=1}^{k} pop(i) \ , \tag{4.46}$$

where k is the size of the recommendations list, i denotes an item and $pop(i)$ is the popularity of an item i like in formula (4.44).

Adamopoulos and Tuzhilin require that items have to be novel and unexpected to the user, but they add a third feature: a positive emotional response. "Serendipity, the most closely related concept to unexpectedness, involves a positive emotional response of the user about a previously unknown (novel) item and measures how surprising these recommendations are" [2].

Simpler definition was proposed by Zhang et al. "Serendipity represents the *unusualness* or *surprise* of recommendations" [109]. They called their measure *unserendipity* and defined it by the formula:

$$unserendipity = \frac{1}{|H_u|}\sum_{h\in H_u}\frac{1}{k}\sum_{i\in R_{u,k}} sim(i,h) \ , \tag{4.47}$$

where u denotes a user, h is an item from a user history H_u (the user's past ratings), k is the size of the user u recommendation list $R_{u,k}$ and i denotes an item from the recommendations list $R_{u,k}$. The sim function could be any similarity function, e.g. a cosine similarity.

Expectedness is a simple measure which sums up the popularity of all items in the recommendations list. The unserendipity measure is more complicated and checks how much items from a recommendations list are similar to those from a user history. Both measures are in opposite to the definition of serendipity. Thus, the lower values of those measures are, the better serendipity of a recommendations list is.

Chapter 5

Usage of Contextual Ontology in Recommender Systems

In this chapter we present how the SIM ontology can be applied in RSs. Section 5.1 provides description how a contextual ontology built according to SIM modularization approach can be used to model context-dependent user preferences. Section 5.2 introduces the ontology-based contextual pre-filtering technique for RSs. While combined with non-context-aware recommendation algorithms, it produces context-aware results. We show that this method can be used for both kinds of recommendation tasks: rating prediction and generation of top k recommendations. We also show that the ontology-based contextual pre-filtering technique is applicable for multi-domain recommendations.

5.1 Contextual Ontological User Profile

A contextual ontology built according to modularization approach, which is described in Section 3.2, has interesting properties that can be used to create a contextual user model [66, 67]. It enables us to represent different points of view in different situations easily. And, what is most important, it enables a reasoning process and a flow of information between interconnected contexts[1]. Thus, a user profile can be represented as a hierarchy of contexts which contain information about user preferences in different domains and in different contextual situations [68].

Contextual Ontological User Profile (COUP) is a proposed in this dissertation user model that is based on the SIM approach. It supports a storage of preferences from multiple domains, which is done by adding context types related to different domains. Because this part strongly depends on the domain of RS that will use the model, the process of choosing and modeling domains has to be done by system designers.

Context types and context instances related to contextual parameters are added to COUP in a dynamic way. As a consequence, we can use as many variables as needed. Another useful property is that many user profiles can be stored in one SIM ontology. An example of

[1] Recall that the context in a SIM ontology is a pair of context type and context instance (see Section 3.2). For distinction, in this chapter we will use a phrase *contextual situation* to refer to a context as a specific user situation.

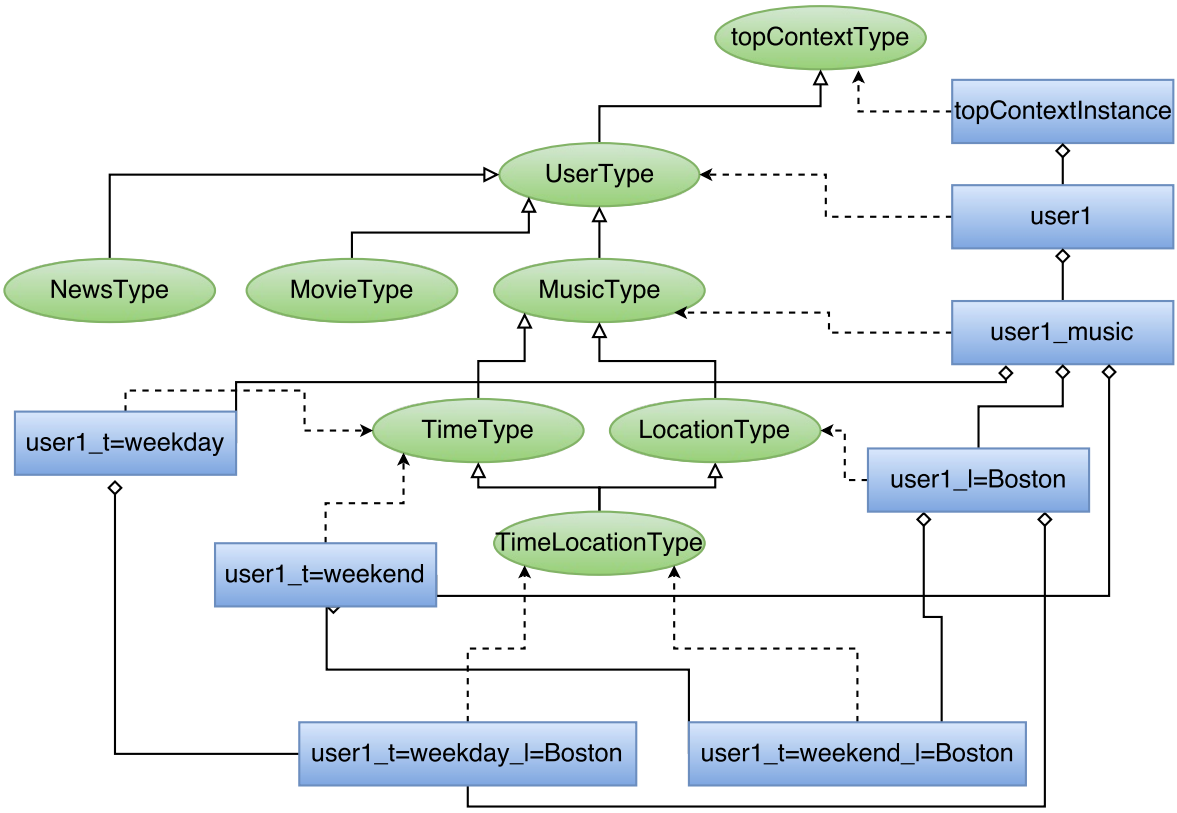


Figure 5.1: An example of the Contextual Ontological User Profile.

a contextual profile for one user is shown in Fig. 5.1.

Three modules in the example in Fig. 5.1 are fixed: `UserType`, `topContextType` and `topContextInstance`. All the others are configurable or can be added dynamically. Modules `topContextType` and `topContextInstance` are obligatory in the SIM model. `UserType` is artificial and is present in the SIM ontology because it enables to add many user profiles to the ontology. In `topContextType` we defined the concept `Rating` and its corresponding roles, e.g. `isRatedWith` and `hasValue`. In the next level of the hierarchy, there are context types that describe domains of interests related to the RS which will use the profile. At this level existing domain ontologies can be reused.

At the next levels, all context types and instances are added to the contextual user profile during the creation of profile (e.g. from a ratings matrix) or later, when a new contextual situation occurs. User preferences are stored in these lower context instances.

Because the SIM method does not enable to easily store and obtain the exact values of contextual parameters, we use the following naming convention for the lower context instances. Since we can store profiles of many users in one COUP, we put a user identifier at the beginning of the name. Further parts of the name consist of the contextual parameter symbol (here we use `t` for `time` and `l` for `location`) and its value separated by "=" (see Fig. 5.1). All of the parts of the name are separated by "_" to easily parse a name of a contextual instance.

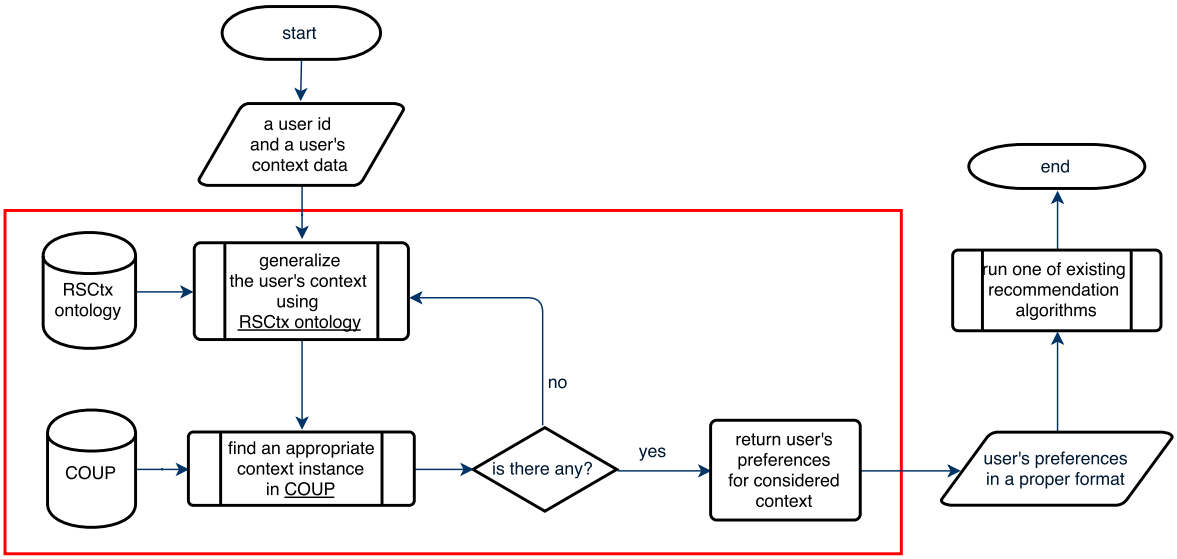


Figure 5.2: The schema of the Ontology-based Contextual Pre-filtering Approach.

5.2 Generation of Recommendations with Pre-filtering Technique

We use COUP and RSCtx ontology, which was presented in Section 3.3, for pre-filtering in the recommendation process. The aim was to provide a universal context-aware improvement for existing algorithms.

The approach consists of three main functional components: (I) context detection and generalization, (II) user profile and pre-filtering, and (III) recommendation [69]. In the first component, we use the RSCtx ontology to identify the user contextual situation from raw data and generalize it in the desired granularity level. The second component is responsible for building a user profile, finding a context instance that fits the considered user contextual situation, and returning only relevant preferences. By relevant preferences we mean all users ratings that were rated in the considered contextual situation, i.e. preferences from all context instances for the same values of contextual parameters from different user profiles.

The last component, recommendation, uses a state-of-the-art non-context-aware algorithm, e.g. Item kNN, for providing recommendations.

The general recommendation process is presented in Fig. 5.2 and proceeds as follows. Given a user and his current contextual situation, a proper generalization of values for his contextual parameters is generated by using the RSCtx ontology. Then, an appropriate context instance from COUP is identified by using the generalized context information. If a context instance is not found in the user profile, the generalization step is repeated to search for a module that corresponds to the new values of contextual parameters. If it is found, relevant preferences (for considered user and all other users who have the context instance with the same value for the same contextual parameters) are prepared to be used with a recommendation algorithm.

The flow of the ontology-based contextual pre-filtering method is presented in Algorithm 1.

Let us consider some examples to explain better how the ontology-based contextual pre-

Algorithm 1 Ontology-based Contextual Pre-filtering Technique

Require: u - a user, ctx - contextual situation of a user
Ensure: $ratings$ - relevant preferences for considered contextual situation

```
ratings ← newList();
while ratings.isEmpty() do
    context ← generalize(ctx);
    instanceName ← findContextInstance(u, context);
    if instanceName! = null then
        ratings ← getRelevantData(instanceName);
    else
        ctx ← context;
    end if
end while
return ratings;
```

Table 5.1: Example for rating prediction with COUP.

User	Item (Movie)	Rating	Companion	Day
Alice	**Donnie Darko**	1	friend	Sunday
Alice	**Girl Interrupted**	2	friend	Friday
Alice	How To Hook Up Your Home Theater	4	family	Sunday
Alice	**Inception**	5	friend	Friday
Alice	The Imaginarium of Doctor Parnassus	?	friend	Saturday
Alice	Shrek	5	family	Saturday
Alice	Spiderman	1	family	Sunday
Alice	**The Counselor**	4	friend	Friday
Alice	The Lion King	4	family	Sunday

filtering method is used in different recommendation tasks. We start with a simple example for rating prediction. The rating history of Alice is presented in Tab. 5.1. We want to predict Alice rating for the movie *The Imaginarium of Doctor Parnassus* assuming that she will watch it on Saturday with a friend. The initial values of contextual parameters are not raw data, so we can skip the first generalization step of pre-filtering method. We have to find context instance corresponding to the specific contextual situation (`companion` = *friend*, `time` = *Saturday*). In our naming convention, it would be `Alice_t=Saturday_c=friend`. However, this is the first occurrence of such contextual situation for Alice according to Tab. 5.1. Thus, we have to do the generalization step. It is hard to generalize contextual parameter `companion`, but `time` has a natural hierarchy. The upper level for `time` granularity will be a division into *weekday* and *weekend*[2]. The context instance corresponding to this contextual situation exists, so we can return the items and their corresponding rating values (the corresponding movies are marked in bold in Tab. 5.1). The last step is to compute rating for the considered movie based on the returned ratings. For simplicity, we use the user average predictor. Thus, the predicted Alice rating for the movie *The Imaginarium of Doctor Parnassus* is $(1 + 2 + 5 + 4)/4 = 3$.

The second example concerns the process of ontology-based contextual pre-filtering for

[2]As a weekend in this example we understand Friday, Saturday and Sunday.

multi-domain ranking task. However, we will not show the final ranking list, but we will stop after returning a set of ratings on which any non-context-aware ranking algorithm can be applied.

Historical users ratings from two different domains: movies and restaurants, are presented in Tables 5.2 and 5.3 respectively. Let us assume that we want to generate a list of top 5 recommendations for Alice in the same contextual situation as in previous example, i.e. on Saturday with a friend. Again, Alice does not have the needed context instance in her COUP. Thus, we have to do the same generalization as in previous example. The value of the contextual parameter `time` is equal to *weekend*. In Alice COUP there exists such a context instance. The same happens for Bob and Carol. The returned set of ratings is presented in Tab. 5.4.

Please note how easy it was to obtain ratings from different domains. The reason for that is the way in which modules in COUP are connected with each other. Because context types that represent contextual parameters can inherit from context types that describe domains, context instances related to certain contextual situations will contain preferences from different domains. Of course, it is not obligatory to have this inheritance relation. Everything depends on desired application of COUP and decisions of RS designers.

It should be noticed that if we consider any other user than Alice, i.e. Bob or Carol, we would not have to perform any generalization of contextual parameters. Both users, Bob and Carol, have their COUP with context instances corresponding to the contextual situation *on Saturday with a friend*. It is one of the biggest advantages of the approach described in this chapter. For each user we allow (and can obtain) different level of granularity for values of contextual parameters. It enables us to better model and deal with context-aware user preferences.

Table 5.2: Sample user preferences in the movie domain of Alice, Bob and Carol.

User	Item (Movie)	Rating	Companion	Day
Alice	Donnie Darko	1	friend	Sunday
Alice	Girl Interrupted	2	friend	Friday
Alice	How To Hook Up Your Home Theater	4	family	Sunday
Alice	Inception	5	friend	Friday
Alice	The Imaginarium of Doctor Parnassus	5	friend	Friday
Alice	Shrek	5	family	Saturday
Alice	Spiderman	1	family	Sunday
Alice	The Counselor	4	friend	Friday
Alice	The Lion King	4	family	Sunday
Bob	An Unexpected Journey	5	friend	Saturday
Bob	City Of Angels	2	girlfriend	Saturday
Bob	Armageddon	2	friend	Friday
Bob	Inception	1	friend	Tuesday
Bob	Green Mile	5	friend	Saturday
Bob	Hunger Games	2	friend	Saturday
Bob	Tourist	4	girlfriend	Friday
Bob	Sleepless In Seattle	4	girlfriend	Friday
Bob	The Desolation Of Smaug	5	friend	Tuesday
Carol	At Worlds End	5	friend	Friday
Carol	Dead Mans Chest	5	friend	Friday
Carol	Gangs Of New York	2	friend	Saturday
Carol	The Imaginarium of Doctor Parnassus	5	friend	Saturday
Carol	Return Of The King	5	alone	Saturday
Carol	The Curse Of The Black Pearl	5	friend	Friday
Carol	The Fellowship Of The Ring	5	alone	Saturday
Carol	Two Towers	5	alone	Tuesday
Carol	Cast Away	2	alone	Saturday

Table 5.3: Sample user preferences in the restaurant domain of Alice, Bob and Carol.

User	Item (Restaurant)	Rating	Companion	Day
Alice	Fish Bar	4	friend	Friday
Alice	McDonalds	2	friend	Sunday
Alice	Italian Restaurant	5	family	Sunday
Alice	Noodle Bar	3	family	Saturday
Alice	Pizza Hut	5	friend	Friday
Bob	McDonalds	5	friend	Saturday
Bob	Jamie Oliver's Diner	5	girlfriend	Saturday
Bob	Russian Cuisine	2	friend	Friday
Bob	Pizza Hut	1	girlfriend	Tuesday
Bob	Pizza Hut	5	friend	Saturday
Carol	Noodle Bar	5	friend	Friday
Carol	Russian Cuisine	2	friend	Friday
Carol	Italian Restaurant	1	alone	Saturday
Carol	Fish Bar	4	friend	Saturday
Carol	McDonalds	5	alone	Saturday

Table 5.4: Preferences of Alice, Bob and Carol after ontology-based contextual pre-filtering for Alice on Saturady with a friend (both domains).

User	Item	Rating
Alice	Donnie Darko	1
Alice	Girl Interrupted	2
Alice	Inception	5
Alice	The Imaginarium of Doctor Parnassus	5
Alice	The Counselor	4
Alice	Fish Bar	4
Alice	McDonalds	2
Alice	Pizza Hut	5
Bob	An Unexpected Journey	5
Bob	Armageddon	2
Bob	Green Mile	5
Bob	Hunger Games	2
Bob	McDonalds	5
Bob	Russian Cuisine	2
Bob	Pizza Hut	5
Carol	At Worlds End	5
Carol	Dead Mans Chest	5
Carol	Gangs Of New York	2
Carol	The Imaginarium of Doctor Parnassus	5
Carol	The Curse Of The Black Pearl	5
Carol	Noodle Bar	5
Carol	Russian Cuisine	2
Carol	Fish Bar	4

Chapter 6

Contextual Conditional Preferences and their Application in Recommender Systems

In this chapter we introduce contextual conditional preferences and describe how to apply them in RSs. Section 6.1 provides the definition of the contextual conditional preferences and description of a new user model based on them. Section 6.2 describes the Prism algorithm that is used to extract contextual conditional preferences from explicit user ratings, which is explained in Section 6.3. Section 6.4 presents how contextual conditional preferences can be used in RSs to predict a user rating and to generate a list of top k recommendations.

6.1 Contextual Conditional Preferences

Contextual Conditional Preferences (CCPs) were inspired by CP-nets described in Section 4.4.1. However, CCPs are not a special case of CP-nets. The main difference is that the conditional relationship between any two variables like it is in CP-nets is not allow. Here, in the conditional part we allow only contextual parameters, on which values of other variables depend. CCPs were introduced to provide compact and context-aware representation of user interests for RSs [65]. Thus, we have to deal with the data sparsity problem, as described in Chapter 4. Therefore, we cannot keep the *ceteris paribus* assumption.

Definition 6.1.1. Contextual Conditional Preference (CCP) is an expression of the form:

$$(\gamma_1 = c_1) \wedge \ldots \wedge (\gamma_n = c_n) \mid (\alpha_1 = a_1) \succ (\alpha_1 = a'_1) \wedge \ldots \wedge (\alpha_m = a_m) \succ (\alpha_m = a'_m) \quad (6.1)$$

with γ_i being contextual variables, α_i item attributes, and $c_1, ..., c_n$, $a_1, a'_1, ..., a_m, a'_m$ being concrete values of these parameters. Symbol $\succ$ denotes a preference relation, e.g. $x \succ y$ means that someone prefers x over y.

The above CCP is read as *given the context* $(\gamma_1 = c_1) \wedge \ldots \wedge (\gamma_n = c_n)$ *I prefer* a_1 *over* a'_1 *for* α_1 *and* ... *and* a_m *over* a'_m *for* α_m. An example of the CCP is shown below.

$$\begin{gathered}\text{weather} = \textit{sunny} \wedge \text{companion} = \textit{with children} \\ \mid \ \text{category} \in \{\textit{walk or trail}, \textit{park}\} \succ \text{category} \in \{\textit{museum}\}\end{gathered}$$

It means that for a given context, i.e. sunny weather and the company of children, a user prefers places with categories "walk or trail" and "park" to those with category "museum".

We distinguish two kinds of CCPs: individual and general. Individual CCPs (ICCPs) express preferences of a single user, as typically different preferences models do. It is quite different with general CCPs (GCCPs), because they catch general trends of interests in the whole considered population, i.e. all users of a certain RS. GCCPs follow an intuition that in some situations different people like the same things, e.g. on Friday evening with a partner people usually prefer to watch romantic movies than dramas.

A user model consists of a set of CCPs. This set could be explicitly given by the user during a registration into RS or during further interaction with RS. Furthermore, we can extract CCPs from existing user ratings, which is explained in more details in Section 6.3. But before we introduce the precise algorithm, we have to recall the Prism algorithm.

6.2 Prism Algorithm

`Prism` is the algorithm used for induction of decision rules which are used for classification tasks [21]. Steps of the algorithm are given below.

Algorithm 2 Prism algorithm

If the training set contains instances of more than one class, then for each class δ_n, in turn:

1. Calculate the probability of occurrence $p(\delta_n|(\alpha, x))$ of the class δ_n for each attribute-value pair (α, x).
2. Select the (α, x) for which $p(\delta_n|(\alpha, x))$ is maximum and create a subset of the training set comprising all the instances which contain the selected (α, x).
3. Repeat steps 1 and 2 for this subset until it contains only instances of class δ_n. The induced rule is a conjunction of all the attribute-value pairs used in creating the homogeneous subset.
4. Remove all instances covered by this rule from the training set.
5. Repeat steps 1-4 until all instances of class δ_n have been removed.

When the rules for one class have been induced, the training set is restored to its initial state and the algorithm is applied again to induce a set of rules covering the next class. As the classes are considered separately, their order of presentation is irrelevant. If all instances are of the same class then that class is returned as the rule, and the algorithm terminates.

To better understand Prism algorithm, let us consider the following example. We have a set of predefined weather conditions and the corresponding decision whether to go to a trip or not. The training set in presented in Tab. 6.1.

In this example we have only two classes: *Yes* and *No*. The order in which we consider classes is insignificant for the final result. We start with *Yes* class, since we have more instances for it. Thus, we look for a rule of the form "If ? then decision = *Yes*". In our case, probabilities from Step 1 of Algorithm 2 are simply frequencies of specific attribute-value

Outlook	Temperature	Windy	Decision
Overcast	Cool	No	Yes
Overcast	Hot	No	Yes
Overcast	Chill	Yes	No
Rainy	Hot	No	No
Rainy	Chill	No	No
Rainy	Cool	Yes	No
Sunny	Cool	No	Yes
Sunny	Chill	No	Yes
Sunny	Hot	Yes	Yes

Table 6.1: The trip decision example.

pairs, which are shown below.

$$\begin{aligned}
\text{Outlook} &= \mathit{Overcast} && 2/3\\
\text{Outlook} &= \mathit{Rainy} && 0/3\\
\text{Outlook} &= \mathit{Sunny} && 3/3\\
\text{Temperature} &= \mathit{Chill} && 1/3\\
\text{Temperature} &= \mathit{Hot} && 2/3\\
\text{Temperature} &= \mathit{Cool} && 2/3\\
\text{Windy} &= \mathit{Yes} && 1/3\\
\text{Windy} &= \mathit{No} && 4/6
\end{aligned}$$

According to steps 2 and 3 we have to choose (Outlook, *Sunny*) pair and create the following rule: "If Outlook = *Sunny* then decision = *Yes*". Then, we remove all instances containing (Outlook, *Sunny*) pair (Step 4) and repeat everything for *Yes* class again. The frequencies of remaining attribute-value pairs are shown below.

$$\begin{aligned}
\text{Outlook} &= \mathit{Overcast} && 2/3\\
\text{Outlook} &= \mathit{Rainy} && 0/3\\
\text{Temperature} &= \mathit{Chill} && 0/2\\
\text{Temperature} &= \mathit{Hot} && 1/2\\
\text{Temperature} &= \mathit{Cool} && 1/2\\
\text{Windy} &= \mathit{Yes} && 0/2\\
\text{Windy} &= \mathit{No} && 2/4
\end{aligned}$$

In this case, one attribute-value pair is not enough. The best is (Outlook, *Overcast*), but we still have two classes in the subset of the training set that consist of those instances that contain this pair (this subset is shown in Tab. 6.2). The second best pair is (Windy, *No*), which has frequency equal to 2/2 in the subset from Tab. 6.2. Thus, resulting second rule will be: "If Outlook = *Overcast* and Windy = *No* then decision = *Yes*". Computed rules are sufficient to cover each instance of the class *Yes*. We should do the same steps to create rules for the *No* class. We obtain the following rules: "If Outlook = *Rainy* then decision = *No*" and "If Temperature = *Chill* and Windy = *Yes* then decision = *No*".

Outlook	Temperature	Windy	Decision
Overcast	Cool	No	Yes
Overcast	Hot	No	Yes
Overcast	Chill	Yes	No

Table 6.2: The subset of training set for trip decision example.

6.3 Extraction of Contextual Conditional Preferences

As it was mentioned earlier in this chapter, CCPs can be learned from explicit user ratings. We assume that beside a rating value we have contextual parameters with their values as well as content information about items available in a dataset. In order to elicit preference relations we split the dataset into two parts based on the value of the ratings. Depending on a rating scale for a dataset we have to use a different thresholds to divide ratings into positive and negative ones. Then, both subsets are divided into smaller sets containing all of the contextual information and one of the item features. With such prepared data we computed context-aware individual preferences for each user by running the `Prism` algorithm from the WEKA library[1] (version 3.6.11) to generate rules of the form shown in the formula (6.2).

$$(\gamma_1 = c_1) \wedge \ldots \wedge (\gamma_n = c_n) \mid (\alpha_1 = a_1) \succ (\alpha_1 = a_1') \ , \tag{6.2}$$

where all symbols have the same meaning as in the formula (6.1).

We tested also other algorithms for generation of decision rules, like `RIPPER`, `M5` or `PART`. However, only `Prism` algorithm produced many nicely readable rules. Other algorithms gave insufficient number of rules or the rules were hard to understand.

The next step is to compact preferences with the same "conditional part" into one preference of the form shown below.

$$\begin{aligned}(\gamma_1 = c_1) \wedge \ldots \wedge (\gamma_n = c_n) \mid \left(\alpha_1 \in \left\{a_1^1, \ldots, a_1^k\right\}\right) \succ \left(\alpha_1 \in \left\{a_1^{1\prime}, \ldots, a_1^{l\prime}\right\}\right) \wedge \\ \ldots \wedge \left(\alpha_n \in \left\{a_n^1, \ldots, a_n^m\right\}\right) \succ \left(\alpha_n \in \left\{a_n^{1\prime}, \ldots, a_n^{p\prime}\right\}\right) \ ,\end{aligned} \tag{6.3}$$

where k, l, m and p are natural numbers denoting number of possible values for parameters α_n. An example rule could look like the following one.

$$\begin{aligned}&season = 3 \wedge weather = 1 \wedge time = 2 \wedge mood = 1 \\ \mid\ &genre \in \{18\} \succ genre \in \{8, 12, 7\} \wedge\ director \in \{5, 8\} \succ director \in \{3\}\,.\end{aligned}$$

It means that for a given context (e.g. season is 3 - Autumn) a user prefers a genre with id 18 to those with 8, 12 or 7 and directors from clusters 5 and 8 to those from cluster 3, etc.

If the value of some content parameter is the same on both sides of a preference relation for some certain user's context, then this value is marked as meaningless and is not taken into consideration in this context for the user. The described algorithm is presented in Algorithm 3. It can be used to extract both kinds of CCPs: ICCPs and GCCPs. The main difference in the computation of general and individual CCPs is that in the first case all the ratings from the dataset were treated like they were made by one person. As a consequence,

[1] `http://www.cs.waikato.ac.nz/ml/weka/`

Algorithm 3 Extraction of Contextual Conditional Preferences

Require: RI - a dataset with user ratings, contextual parameters and content information, CF - item features, t - threshold to divide ratings
Ensure: CCP - a set of CCPs

```
PRI, NRI ← new List();
for all ri in RI do
    if rating(ri) > t then
        for all cf in CF do
            PRI.add(context(ri), attribute(ri, cf));
        end for
    else
        for all cf in CF do
            NRI.add(context(ri), attribute(ri, cf));
        end for
    end if
end for
for all cf in CF do
    PRules ← Prism(PRI, cf);
    NRules ← Prism(NRI, cf);
end for
CCP ← compact(PRules, NRules);
CCP ← removeMeaningless(CCP);
return CCP;
```

we removed many contradictory values during the merging phase. To better understand the issue, let us consider an example in the movie domain from Tab. 6.3, which is a bit moddified example from Tab. 5.2. Besides information about rating for an item, we have also two contextual factors, i.e. *companion* and *day*, and one movie feature, i.e. *genre* in sample user profiles. For all three users we could compute ICCPs. The rating scale is 1-5, so the threshold to divide ratings into positive and negative ones is set to 3. For multivalued attributes, we have to split each instance into number of instances equal to the number of values that this attribute has. The resulting positive and negative subsets of Alice ratings are represented in Tables 6.4 and 6.5 respectively. On such prepared training set we apply Prism algorithm. Some rules obtained for the subset of positive ratings are given below.

$$\text{If day} = Sunday \text{ and companion} = family \text{ then animated}$$
$$\text{If day} = Sunday \text{ and companion} = family \text{ then adventure}$$
$$\text{If day} = Saturday \text{ and companion} = friend \text{ then fantasy}$$

Some rules obtained for the subset of negative ratings are given below.

$$\text{If day} = Saturday \text{ and companion} = friend \text{ then supernatural}$$
$$\text{If day} = Saturday \text{ and companion} = friend \text{ then drama}$$
$$\text{If day} = Sunday \text{ and companion} = family \text{ then superhero}$$

Table 6.3: Sample user profiles of Alice, Bob and Carol.

User	Item (Movie)	Rating	Companion	Day	Genre
Alice	Donnie Darko	1	friend	Saturday	drama, supernatural
Alice	Girl Interrupted	2	friend	Friday	drama
Alice	How To Hook Up Your Home Theater	4	family	Sunday	animated
Alice	Inception	5	friend	Friday	heist, thriller, science fiction
Alice	The Imaginarium of Doctor Parnassus	5	friend	Saturday	fantasy
Alice	Shrek	5	family	Saturday	animated, fantasy
Alice	Spiderman	1	family	Sunday	superhero
Alice	The Counselor	4	friend	Friday	thriller
Alice	The Lion King	4	family	Sunday	animated, adventure
Bob	An Unexpected Journey	5	alone	Saturday	fantasy, epic, adventure
Bob	City Of Angels	2	girlfriend	Saturday	fantasy, romantic, drama
Bob	Armageddon	2	alone	Friday	thriller, disaster, science fiction
Bob	Inception	1	alone	Tuesday	heist, thriller, science fiction
Bob	Green Mile	5	alone	Saturday	drama, fantasy
Bob	Hunger Games	2	alone	Saturday	science fiction, adventure
Bob	Tourist	4	girlfriend	Friday	thriller, comedy, romantic
Bob	Sleepless In Seattle	4	girlfriend	Friday	drama, comedy, romantic
Bob	The Desolation Of Smaug	5	alone	Tuesday	adventure, epic, fantasy
Carol	At Worlds End	5	friend	Friday	fantasy, swashbuckler
Carol	Dead Mans Chest	5	friend	Friday	fantasy, swashbuckler
Carol	Gangs Of New York	2	friend	Saturday	historical, drama, epic
Carol	The Imaginarium of Doctor Parnassus	5	friend	Saturday	fantasy
Carol	Return Of The King	5	alone	Saturday	epic, fantasy
Carol	The Curse Of The Black Pearl	5	friend	Friday	swashbuckler, fantasy
Carol	The Fellowship Of The Ring	5	alone	Saturday	epic, fantasy
Carol	Two Towers Film	5	alone	Tuesday	epic, fantasy
Carol	Cast Away	2	alone	Saturday	drama, adventure

Table 6.4: Subset of positive Alice ratings.

Companion	Day	Genre
family	Sunday	animated
friend	Friday	heist
friend	Friday	thriller
friend	Friday	science fiction
friend	Saturday	fantasy
family	Saturday	animated
family	Saturday	fantasy
friend	Friday	thriller
family	Sunday	animated
family	Sunday	adventure

Table 6.5: Subset of negative Alice ratings.

Companion	Day	Genre
friend	Saturday	drama
friend	Saturday	supernatural
friend	Friday	drama
family	Sunday	superhero

After merging these rules according to the contextual parameters, we obtained following individual preferences for Alice:

$$\begin{aligned} &\text{day} = Sunday \wedge \text{companion} = family \\ &\mid \text{genre} \in \{animated, adventure\} \succ \text{genre} \in \{superhero\} \ , \end{aligned} \tag{6.4}$$

$$\begin{aligned} &\text{day} = Saturday \wedge \text{companion} = friend \\ &\mid \text{genre} \in \{fantasy\} \succ \text{genre} \in \{drama, supernatural\} \ , \end{aligned} \tag{6.5}$$

$$\begin{aligned} &\text{companion} = friend \\ &\mid \text{genre} \in \{thriller\} \succ \text{genre} \in \{drama\} \ . \end{aligned} \tag{6.6}$$

We see that Alice's movie preferences vary depending on the company and the day. The same applies to Bob and Carol. Exemplary general preferences (GCCPs) computed for the sample profiles are shown below.

$$\begin{aligned} &\text{companion} = alone \\ &\mid \text{genre} \in \{fantasy\} \succ \text{genre} \in \{sciencefiction\} \ , \end{aligned} \tag{6.7}$$

$$\begin{aligned} &\text{companion} = friend \\ &\mid \text{genre} \in \{fantasy\} \succ \text{genre} \in \{drama\} \ , \end{aligned} \tag{6.8}$$

$$\begin{aligned} &\text{day} = Saturday \\ &\mid \text{genre} \in \{fantasy\} \succ \text{genre} \in \{drama\} \ . \end{aligned} \tag{6.9}$$

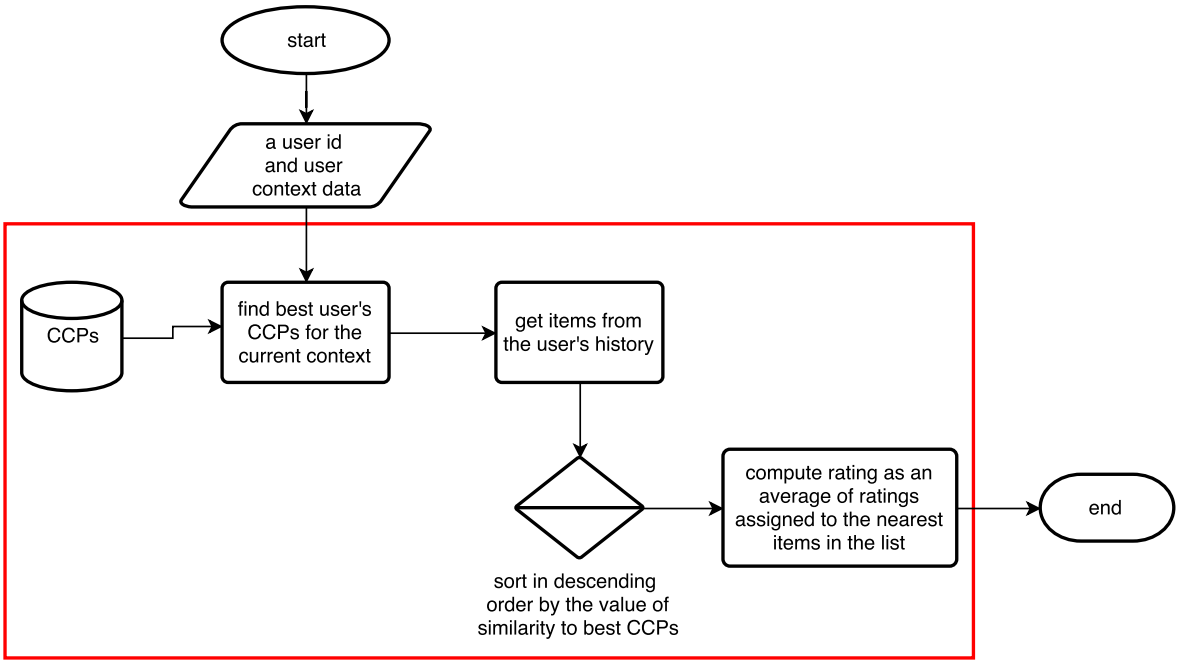


Figure 6.1: Rating prediction with CCPs.

6.4 Generation of Recommendations with Contextual Conditional Preferences

This section presents how CCPs can be applied in RSs. In the first part of the section an algorithm for rating prediction is described. The second part is concerned on generating a list of top k recommendations with usage of CCPs.

6.4.1 Rating Prediction Task

The algorithm for rating prediction with CCPs was introduced by Karpus et al. [64]. The overall process in depicted in Fig. 6.1 and in Algorithm 4.

Algorithm 4 Rating prediction with CCPs

Require: CCP - a set of CCPs, ti - the test instance, sim - minimal similarity
Ensure: $rating$
$best \leftarrow findMostSimilarRules(CCP, ti, sim)$;
$items \leftarrow getTrainingData(context(ti), sim)$;
$items \leftarrow reorder(items, best, sim)$;
$rating \leftarrow computeAverageFromNeighbors(items)$;
return $rating$;

Having a specific user and his context, and wanting to predict his rating for some item, first we need to find the best CCPs (the user's or general ones) that will be used during a prediction process. In this case, the best preferences are those which are most similar to the considered context. In order to count a contextual similarity between a CCP p and a current

user context $ctx(u)$ we used the following measure [103]:

$$sim\,(p, ctx(u)) = \sum_{(\gamma_i, c_i) \in p} overlap(ctx(u), (\gamma_i, c_i)) \ . \tag{6.10}$$

We also used the overlap function defined as:

$$overlap(ctx(u), (\gamma_i, c_i)) = \begin{cases} 1 & (\gamma_i, c_i) \in ctx(u); \\ 0.5 & c_i = -1; \\ 0 & otherwise. \end{cases} \tag{6.11}$$

The overlap function returns 1 when we are sure that the pair (γ_i, c_i) is contained both in the contextual part of p and in the current user context $ctx(u)$. When it is uncertain, i.e. when the value c_i for the dimension γ_i is equal -1 (the unknown value), it returns 0.5. Otherwise 0 is returned. It should be noticed, that the current user context $ctx(u)$ is also a set of pairs (γ_i', c_i'), i.e. the name of the contextual variable and its value.

For an item whose rating we want to predict, we construct a list containing this item and items seen by the user in the context similar to current context in at least some percentage (this value is configurable and depends on the data set).

Identified in the previous step best preferences are used to order the constructed list. For each pair of items, we choose the one that has the most similar values for the attributes $attr(i)$ (a set of attribute name and value pairs (α_i, a_i)) with the CCP p. For this purpose we used another similarity measure and overlap function defined as:

$$sim_{cont}\,(p, attr(i)) = \sum_{(\alpha_i, a_i) \in p} overlap(attr(i), (\alpha_i, a_i)) \ , \tag{6.12}$$

$$overlap(attr(i), (\alpha_i, a_i)) = \begin{cases} 1 & (\alpha_i, a_i) \in attr(i); \\ 0 & a_i = -1; \\ -1 & otherwise. \end{cases} \tag{6.13}$$

The overlap function used here is quite different from the one used above. In the case of item features it is more crucial to have strict matching. This is the reason why we do not reward the unknown value and why we give penalty for unmatched parameter values.

It should be noticed that we need to compare the similarity of the item attributes with both sides of the preference relation in the current preference statement.

The process of reordering is repeated as long as nothing can be changed. Depending on the final place of the considered item in the list, we compute its rating. If the context is new, i.e. if there is no other item in the list, we rate the current item with some baseline algorithm (it is a configurable option). If the item is first or last on the list, we assign to it a rating of the nearest neighbor. Otherwise, we compute the rating as an average of two or four, depending on the size of a list, nearest neighbors' ratings, i.e. the one/two above considered item and one/two below it. We assume that we do not have much data in one context, so we cannot take more than four neighbors.

Let us consider an example from Tab. 6.3. We want to predict Alice rating for the movie *Cast Away* assuming that she will watch it on Friday with a friend. The first step is to find the best CCPs to use. In this case we have two CCPs matching: one ICCP and one GCCP, both for contextual parameter `companion` = *friend*. The next step is to find all previous

Algorithm 5 re-rankCCP algorithm

Require: alg - a name of a baseline algorithm,
k - a number of recommendations in the final list,
u - a user,
ctx - a user context,
$ccps$ - a list of all CCPs for user u
Ensure: $topK$ - an ordered list of top k recommendations
$list \leftarrow$ generateTop100Recommendations(alg, u);
2: $best \leftarrow$ findBestCCPs$(ccps, u, ctx)$;
$map \leftarrow$ empty HashMap;
4: **for all** $item$ in $list$ **do**
$sum \leftarrow 0$;
6: **for all** ccp in $best$ **do**
$sat \leftarrow$ satisfiability$(item, ccp)$;
8: $sum \leftarrow sum + sat$;
end for
10: $avg \leftarrow sum/$sizeOf$(best)$;
$map[item] \leftarrow avg$;
12: **end for**
$rec \leftarrow$ order(map);
14: $topK \leftarrow$ cutOff(rec, k);

Alice ratings from considered contextual situation. We found three such movies, i.e. *Girl Interrupted*, *Inception* and *The Counselor*. We extend this list with considered movie. Now, we have to use measure (6.12) to count the similarity of movies to CCPs and to reorder movies in the list. The resulting list is as follows:

The Counselor, Inception, Girl Interrupted, Cast Away.

The considered movie is the last one in the list, so we assign to it the rating value of the nearest neighbor. In this case, we predict that Alice will give rating equal 2 to the movie *Cast Away* if she will watch it on Friday with a friend.

6.4.2 Ranking Task

An algorithm for generating a list of top k recommendations with CCPs, that is called re-rankCCP, is presented in Algorithm 5. It was introduced by Karpus et al. in [63]. We describe it and refer to its specific lines below.

We assume that ICCPs and GCCPs are generated for all non-new users, since new users do not have any rating history.

For a certain user and his current context, first we generate a primary list of top m recommendations with some existing non-context-aware algorithm, e.g. UserKNN (line 1). The value of m has to be significantly greater than k. Then we have to find the best CCPs that will be further used in the reshuffling process (line 2).

In this case, the best preferences are those which are most similar to the considered context. In order to count a contextual similarity between a CCP p and a current user

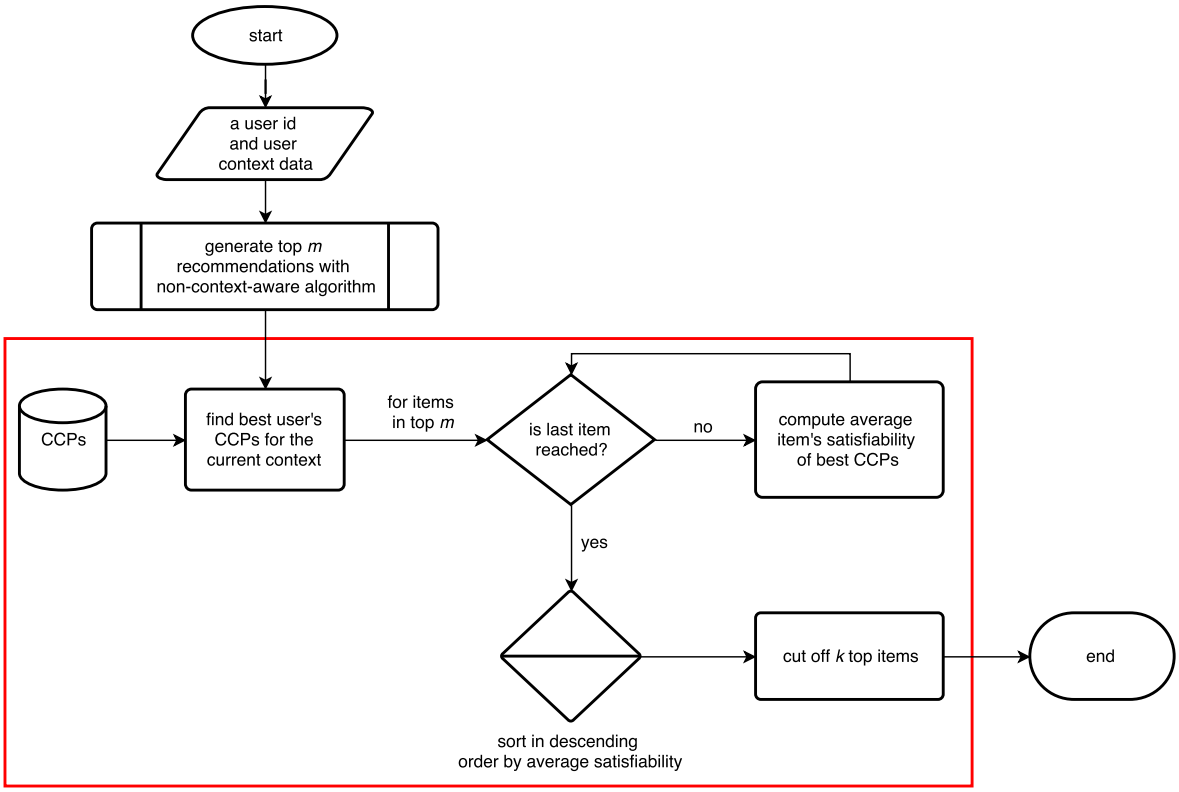


Figure 6.2: Post-filtering with re-rankCCP algorithm.

context $ctx(u)$ we used measures given by formulas (6.10) and (6.11).

For each item in the primary recommendations list and each best CCP we have to compute how much an item i satisfies a CCP p (line 7). For this purpose, we proposed *satisfiability* measure:

$$sat(i,p) = \frac{\sum_{\alpha \in a(p)} \left(sim(v_\alpha^m(p), v_\alpha(i)) - sim(v_\alpha^l(p), v_\alpha(i)) \right)}{|a(p)|} , \tag{6.14}$$

where sim denotes Jaccard similarity, α is the name of an item feature, $a(p)$ is the set of item attributes considered in the CCP p, $v_\alpha(i)$ is the set of values of an attribute α for an item i. Similarly $v_\alpha^m(p)$ and $v_\alpha^l(p)$ denote the sets of values of an attribute α for a CCP p on both sides of the preference relation - m stands for *more preferred* and l for *less preferred*.

The *satisfiability* measure represents the difference between item similarities to the both sides of the CCP preference relation, i.e. the similarity to most preferred part minus the similarity of the less preferred part. In this way we reward items that fit best to user preferences and penalize items that have features that user does not like. The size of a set of item attributes serves as a normalization factor. Thus, regardless of the number of item features, the value of *satisfiability* is always between 0 and 1.

The next step is to order the primary recommendations list according to the value of average *satisfiability* of the best CCPs (line 13). The last part is to cut off unneeded items from resulting recommendations list to receive top 5, top 10 or other top k ranking (line 14). This process is illustrated in Fig. 6.2.

Let us consider again an example from Tab. 6.3. We assume that some traditional rec-

ommendation algorithm returned a following top 10 list for Alice:

Gangs Of New York, The Curse Of The Black Pearl, Cast Away, An Unexpected Journey, City Of Angels, Armageddon, Green Mile, Hunger Games, Tourist, Sleepless in Seattle.

We consider a situation when Alice wants to watch a movie with a friend. With our reshuffling method, using two rules: ICCP for Alice profile and GCCP for this contexts, i.e. (6.6) and (6.8) respectively, we obtained the final top 5 recommendations list:

An Unexpected Journey, Armageddon, Tourist, The Curse Of The Black Pearl, City Of Angels.

Fantasy and *thriller* movies are higher in the final list, while *drama* movies have been mostly cut off the list as expected from the user preferences. At this point, we will not evaluate results of this example. A comprehensive evaluation of the algorithm is presented in the next chapter.

One of the biggest advantage of the proposed method is the ability to explain to the user how recommendations were made. For each user-context situation, we use a specific set of CCPs. Moreover, we know exactly which of them are the user's individual CCPs and which are general ones. Thus, as an explanation, we can show to the user all the CCPs that were used in recommendation process in an understandable way. Considering again the above example, if we display to Alice the CCPs in addition to the top 5 recommendation list, it would give her better understanding of how the RS works. Moreover, she could find out something new about her personal preferences, which seems even more important.

Chapter 7

Analysis of Usability of Proposed Approaches

This chapter contains detailed information about performed experiments as well as justification of theses defined in Chapter 1. Section 7.1 describes context-aware datasets that are used for experiments. In Section 7.2 explanation on how to choose relevant contextual parameters is presented. Descriptions of used libraries and the way in which splits for test and training sets was performed are placed in Section 7.3. The most important part of this chapter is Section 7.4 which consists of justification of the main and auxiliary theses.

7.1 Datasets

Models and recommendation approaches proposed in this dissertation are context-aware. Thus, they should be evaluated on context-aware datasets. The most popular in recommendation field datasets like `MovieLens`[1], do not contain contextual information. They usually contain only `timestamp` variable, however its values represent time of rating an item, not the time of consuming it, and differ only by seconds from each other (for the same user). Thus, this variable is useless as a contextual parameter.

We found five datasets which contain contextual information and can be used for the evaluation of the proposed methods. These datasets are: `LDOS-CoMoDa`[2] dataset, `Unibz-STS`[3] dataset, `Restaurant & consumer`[4] dataset, `ConcertTweets`[5] dataset and `MovieTweetings`[6] dataset. More details about these datasets are provided in the next sections.

7.1.1 LDOS-CoMoDa

The `LDOS-CoMoDa` dataset [74] was collected by a web application that enables contextual rating of a movie just after watching it. The dataset contains 30 variables among which 12

[1] `https://grouplens.org/datasets/movielens/`
[2] `http://212.235.187.145/spletnastran/raziskave/um/comoda/comoda.php`.
[3] `https://github.com/irecsys/CARSKit/tree/master/context-aware_data_sets`
[4] `https://www.researchgate.net/publication/254258246_Restaurant_consumer_data_Data_Set`
[5] `https://github.com/padamop/ConcertTweets`
[6] `https://github.com/sidooms/MovieTweetings`

Table 7.1: Contextual parameters from `LDOS-CoMoDa` dataset.

Parameter name	Possible values
time	Morning, Afternoon, Evening, Night
daytype	Working day, Weekend, Holiday
season	Spring, Summer, Autumn, Winter
location	Home, Public place, Friend's house
weather	Sunny/clear, Rainy, Stormy, Snowy, Cloudy
social	Alone, My partner, Friends, Colleagues, Parents, Public, My family
end_emo	Sad, Happy, Scared, Surprised, Angry, Disgusted, Neutral
dominant_emo	Sad, Happy, Scared, Surprised, Angry, Disgusted, Neutral
mood	Positive, Neutral, Negative
physical	Healthy, Ill
decision	User decided which movie to watch, User was given a movie
interaction	First interaction with a movie, n-th interaction with a movie

Table 7.2: Basic statistics of four datasets: `LDOS-CoMoDa` (CoMoDa), `Unibz-STS` (STS), `Restaurant & consumer` (RC) and `MovieTweetings` (MT).

	CoMoDa	STS	RC	MT
Number of users	121	325	138	39025
Number of items	1232	249	130	22395
Number of ratings	2296	2534	1161	425729
Max number of ratings per user	275	175	18	1102
Min number of ratings per user	1	1	3	1
Avg number of ratings per user	18.98	7.80	8.41	10.91
Max number of ratings per item	26	282	36	2919
Min number of ratings per item	1	1	3	1
Avg number of ratings per item	1.86	10.18	8.93	19.01
Rating scale	1-5	1-5	0-2	0-10
Number of context parameters	12	14	13/0[7]	1
Number of content information	7	1	23	3

are contextual parameters. Other variables are basic information about user (user id, `age`, `sex`, `city` and `country`), a rating in a 5-star scale (higher values denote higher preference) and content information about multiple item dimensions (item id, `director`, `country`, `language`, `year`, 3 main `genres`, 3 main `actors` and `budget`). Since contextual variables are of special interest to us, we describe them in Tab. 7.1. All of the attributes are categorical and their values are represented as numbers. Unknown values are denoted by "-1". Basic statistics of the dataset are presented in Tab. 7.2.

In order to find replicable preferences in such a limited dataset, we had to cluster actors and directors. The process was executed by mapping each actor and director to its corresponding Wikipedia page and eventually by considering their common Wikipedia categories[8]. The number of clusters are 13 for directors and 15 for actors. The choice of those numbers

[7]There are 13 contextual parameters in the dataset. However, their values are fixed for each user.
[8]`https://en.wikipedia.org/wiki/Help:Category`

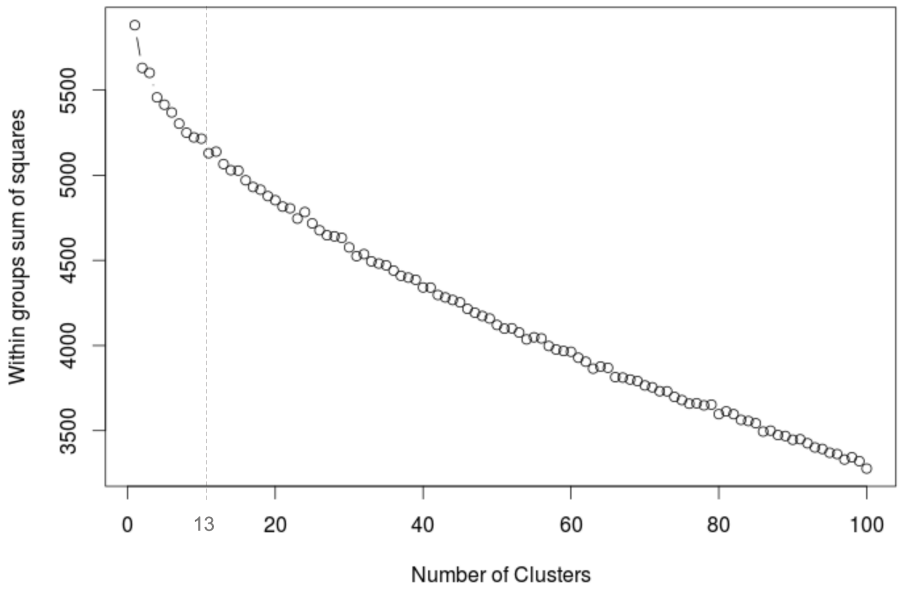


Figure 7.1: Chart for choosing the number of clusters for `director` variable.

was based on the calculation of the within groups sum of squares (`withinSS` measure from the `R Stats Package`, version 2.15.3), picking the number corresponding to an evident break in the distribution of the withinSS measure against the number of clusters. The resulting chart for clustering of directors is shown in Fig. 7.1.

7.1.2 Unibz-STS

The `Unibz-STS` dataset [17, 29] was collected by a mobile application that recommends places of interests (POIs) in South Tyrol in Italy. The recommender is called South Tyrol Suggests (STS). The dataset contains ratings in a 5-star scale, an information about users and their personality (birth date, gender, openness to experience, conscientiousness, extraversion, agreeableness and emotional stability), a context of visiting a POI, which is presented in Tab. 7.3, and a POI's category. All of the attributes are categorical and their values are represented as numbers. Unknown values are denoted by *NULL*, which we replaced with "-1". Basic statistics of the dataset are presented in Tab. 7.2.

7.1.3 Restaurant & consumer

The `Restaurant & consumer` dataset [105] was collected by a prototype application which recommends restaurants to users. The dataset consists of nine files containing three types of information: restaurant data (latitude, longitude, address, city, state, country, fax, ZIP, alcohol, smoking, dress, accessibility, price, franchise, ambiance, space, services, parking, cuisine, phone, accepts, days, hours), user information (variables without limits on possible values: latitude, longitude, birth year, weight and height; cuisine with 103 possible values;

Table 7.3: Contextual parameters from `Unibz-STS` dataset.

Parameter name	Possible values
distance	far away, near by
time available	half day, one day, more than one day
temperature	burning, hot, warm, cool, cold, freezing
crowdedness	crowded, not crowded, empty
knowledge of surroundings	new to area, returning visitor, citizen of the area
season	spring, summer, autumn, winter
budget	budget traveler, price for quality, high spender
daytime	morning, noon, afternoon, evening, night
weather	clear sky, sunny, cloudy, rainy, thunderstorm, snowing
companion	alone, with friends, with family, with partner, with children
mood	happy, sad, active, lazy
weekday	weekday, weekend
travel goal	visiting friends, business, religion, health care, social event, education, landscape, fun, sport
means of transport	no transportation means, a bicycle, a car, public transport

others variables are presented in Tab. 7.4) and a rating that a user gave to a restaurant, its food and service. In this dataset ratings are expressed in a 0-2 scale, where 0 indicates that the user does not like the restaurant, and 2 denotes the highest preference. Unknown values are denoted by "?", which we replaced with "-1". Contextual parameters such as information about a user's mood or companion are not available in this dataset. Basic statistics are presented in Tab. 7.2.

Table 7.4: Contextual parameters from `Restaurant & consumer` dataset.

Parameter name	Possible values
smoker	false, true
drink level	abstemious, social drinker, casual drinker
dress preference	informal, formal, no preference, elegant
ambiance	family, friends, solitary
transport	on foot, public, car owner
marital status	single, married, widow
children	independent, kids, dependent
interest	variety, technology, none, retro, eco-friendly
personality	thrifty-protector, hunter-ostentatious, hard-worker, conformist
religion	none, Catholic, Christian, Mormon, Jewish
activity	student, professional, unemployed, working-class
color	black, red, blue, green, purple, orange, yellow, white
budget	medium, low, high

7.1.4 ConcertTweets

The `ConcertTweets` dataset [1] was collected from publicly available and well-structured tweets shared on Twitter[9] social media. The dataset contains user feedback that refers to musical shows and concerts of various artists and bands. Since the dataset was generated automatically, it combines implicit and explicit user feedback. Thus, there are two rating scales: one numerical scale with ratings in the range [0.5, 5.0], which corresponds to explicit user ratings, and one descriptive scale for implicit feedback with possible values equal to *yes*, *maybe* and *no*, although *no* never occurs. We decided to split the dataset into two separate sets according to the scale type and we mapped the descriptive values *yes*, *maybe* and *no* with numerical values 2, 1 and 0, respectively. Besides the rating data, the dataset contains basic information about each musical event: event date, city, state (or country), latitude, longitude, venue, and event URL. We decided to extend this dataset by adding parameters associated with musical artist, i.e. genres that they play, a year of starting their career and other musical artists that are somehow related. It was done by executing appropriate queries on DBpedia[10]. An examplary SPARQL query is presented in Listing 7.1.

Listing 7.1: SPARQL query to DBpedia for ConcertTweets enrichment.

```
PREFIX dbo: <http://dbpedia.org/ontology/>

SELECT distinct ?band
WHERE { {{?band dbo:genre ?g}
UNION {?band dbo:associatedMusicalArtist ?a}
UNION {?band dbo:activeYearsStartYear ?y}} .
{?band a dbo:Agent} .
FILTER regex(str(?band),"cherry_vanilla","i")}
ORDER BY ?band
```

This query returns the URI (that contains the string "cherry_vanilla") of artist who belongs to DBpedia category `Agent` and has at least one of the properties: `genre`, `associatedMusicalArtist` or `activeYearsStartYear`, associated to her on DBpedia.

Tab. 7.5 presents some statistics about the enriched dataset by considering the whole dataset and each of the sets generated when splitting by scale type.

7.1.5 MovieTweetings

The `MovieTweetings` dataset [28], similarly to `ConcertTweets`, consists of ratings extracted from publicly available and well-structured tweets shared on Twitter social media. The dataset contains basic information about movies (title, year and genres), ratings in a 0-10 scale (higher rating value means higher preference) and an exact time when a user rated a movie. Other user information and contextual parameters are unavailable in this dataset. However, we checked the values of the `timestamp` parameter for randomly chosen users. They differ from each other of at least couple of hours and there are no more than two movies rated on the same day. Thus, we can assume that users rated a movie just after they

[9] `https://twitter.com/`
[10] `https://dbpedia.org/sparql`

Table 7.5: Statistics on the data contained in `ConcertTweets` dataset.

	Ratings		
	All	Descriptive	Numeric
Number of users	61803	56519	16479
Number of musical events	116320	110207	21366
Number of pairs: artist and musical events	137382	129989	23383
Number of ratings	250000	219967	30033
Maximum number of ratings per user	1423	1419	92
Minimum number of ratings per user	1	1	1
Average number of ratings per user	4.045	3.892	1.823
Maximum number of ratings per item	218	216	38
Minimum number of ratings per item	1	1	1
Average number of ratings per item	2.149	1.996	1.406
Rating scale	NA	0-2	0.5-5.0
Number of context parameters	2	2	2
Number of content information	2	2	2

watched it. Hence, we can use this variable as a contextual parameter which represents the time of watching a movie. Basic statistics of the dataset are presented in Tab. 7.2.

7.2 Selection of Contextual Parameters

Usage of contextual parameters in recommendation process is considered as a good solution to improve quality of recommendations. However, irrelevant contextual parameters can increase the noisiness of data, which leads to poor results. According to the analysis reported in [62], different domains typically use different contextual features. But, this does not seem a good solution to select relevant contextual parameters.

In Chapter 2 three measures: entropy, variance and unalikeability, were mentioned. According to Odic et al. [83] they are applicable for checking variability of a variable, which is an important factor for verification of contextual parameter relevance. We decided to use unalikeability which represents how often observations differ one from another. It was devised for categorical variables, so it is good for our purposes. However, we found it a bit difficult to interpret, since there is no known "perfect value" for that measure. Let us recall the formula (2.1):

$$\eta(v) = \frac{\sum_{i \neq j} c(x_i, x_j)}{n(n-1)} \ , \tag{7.1}$$

where v is a contextual variable, x_i and x_j are observations of v, n is the number of all observations of v, and $c(x_i, x_j) = 0$ if $x_i = x_j$ and $c(x_i, x_j) = 1$ if $x_i \neq x_j$.

Obviously, if every observation has the same value, then unalikeability is equal to zero. This situation means that the variable cannot be a relevant contextual parameter because it does not affect any user decision (the value is always the same, independently from a user's choice). Similarly, if each value of a variable differs from others, i.e. there are no two same values, then unalikeability is equal to 1. It is obvious that in this situation the variable also cannot be considered as a relevant contextual parameter. Therefore, what value of unalikeability guarantees the relevance of contextual feature?

According to Pal and Michel[85] the "perfect" value for unalikeability measure is around 0.5. We will show that this is true only for variables with two possible values, like for the variable `sex`.

We assume that we have a variable `x` with k possible values. The ideal situation is when we have n observations of each of k values. Thus, we have nk observations in total. Now, we need to compute the value of unalikeability for variable `x`. In the denominator we have $nk(nk-1)$. In the nominator we should have value equal to the number of pairs that have different values. If all the values were different, then we would have $nk(nk-1)$ pairs that give 1 as a value of the function c. However, in this situation we have $kn(n-1)$ pairs that give 0 as a value of the function c. Thus, the nominator is equal to $nk(nk-1) - kn(n-1)$. Further computations are as follows:

$$\eta(x) = \frac{nk(nk-1) - kn(n-1)}{nk(nk-1)} = 1 - \frac{nk(n-1)}{nk(nk-1)} = 1 - \frac{(n-1)}{(nk-1)} = 1 - \frac{n(1-\frac{1}{n})}{n(k-\frac{1}{n})} =$$
$$= 1 - \frac{(1-\frac{1}{n})}{(k-\frac{1}{n})} \ .$$

Hence,

$$\lim_{n \to \infty} \eta(x) = 1 - \frac{1}{k} \ . \tag{7.2}$$

Table 7.6: Unalikeability analysis for `LDOS-CoMoDa` dataset.

Contextual parameter	Unalikeability	k	Perfect value	Difference
time	0.66	4	0.75	**0.09**
daytype	0.58	3	0.67	**0.08**
season	0.72	4	0.75	**0.03**
location	0.26	3	0.67	0.41
weather	0.66	5	0.80	0.14
social	0.61	7	0.86	0.25
end_emo	0.68	7	0.86	0.18
dominant_emo	0.73	7	0.86	0.12
mood	0.55	3	0.67	0.12
physical	0.18	2	0.50	0.32
decision	0.39	2	0.50	0.11
interaction	0.31	2	0.50	0.19
age	0.37	5	0.80	0.43
sex	0.50	2	0.50	**0.00**
country	0.63	5	0.80	0.17
city	0.70	22	0.95	0.25

Therefore in large datasets, the "perfect" value for unalikeability measure can be estimated by $1 - \frac{1}{k}$, where k is the number of possible values of a contextual parameter `x`. If k is 2, then the "perfect" value equals 0.5.

In three of the five datasets we have many contextual parameters available. For sure not all of them are relevant for making recommendation. Thus, we have to analyze unalikeability values for these parameters and choose those for which the value is close to the "perfect" value.

We performed the unalikeability analysis on two levels. The first level of analysis is the dataset level, which checks the variability of contextual parameters in the whole dataset. The second one is the user level whose aim is to verify if certain contextual parameters really can influence user decisions.

Table 7.6 contains results of the unalikeability analysis on the dataset level for `LDOS-CoMoDa` dataset. Contextual parameters with best unalikeability value are those with the lowest difference between the "perfect" value and computed unalikeability. These values are marked in bold. For `LDOS-CoMoDa` dataset, `time`, `daytype`, `season` and `sex` are best contextual parameters.

We did not use the four last parameter from Tab. 7.6 for the unalikeability analysis on the user level because they are always the same for a user in `LDOS-CoMoDa` dataset. It is hard to present results in a tabular form, so we show them on box plots (see Fig. 7.2). The bold line inside each box denotes the median value of unalikeability for a specific parameter. If we compare it with corresponding values in Tab. 7.6, we observe some differences. First of all, the `season` parameter is not the best one anymore. `Daytype`, `weather`, `end_emo` and `dominant_emo` are better than this one. However, we will not consider `end_emo`, `dominant_emo` and `mood` as a sensible contextual factors, since they represent user emotions during and after watching a movie. Thus, they should be treated more like a measure of user

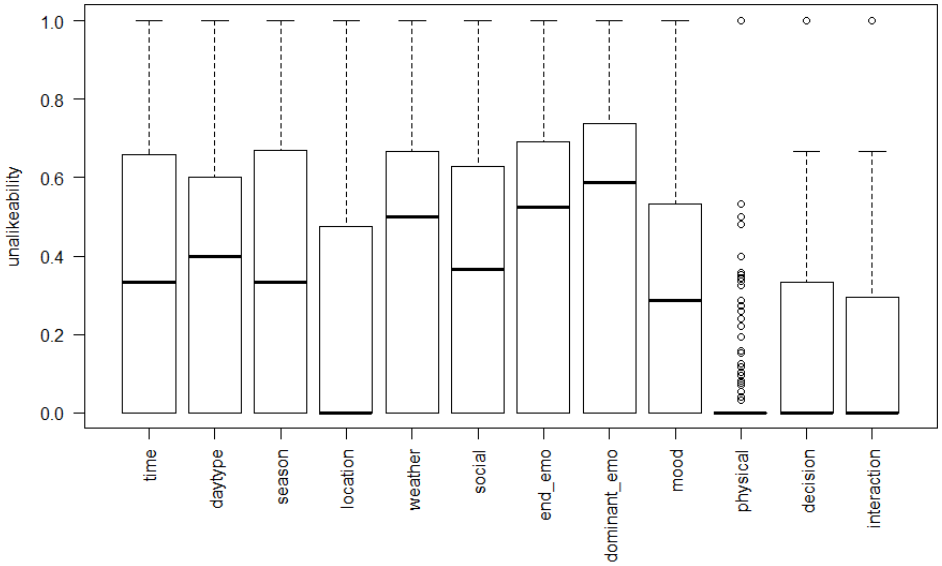


Figure 7.2: Boxplots for unalikeability analysis for `LDOS-CoMoDa` dataset (user level).

interest in a movie, not the contextual information.

The second difference is with the `decision` parameter. The difference between the "perfect" value and the computed value of the `decision` unalikeability is small (0.11) on the dataset level. However, the median value for this factor is almost 0 on the user level. Thus, this parameter cannot be considered as conveying any relevant contextual information.

To conclude, in further experiments on `LDOS-CoMoDa` dataset we used four contextual parameters: `time`, `daytype`, `season` and `weather`. In cases where we use GCCPs, we also used the `sex` parameter, since we found it relevant on the dataset level[11].

The same analysis was performed for `Unibz-STS` dataset. The results are presented in Tab. 7.7 and in Fig. 7.3. On the dataset level, the best contextual parameters are: `weekday`, `transport`, `travel goal`, `season`, `budget`, `knowledge of surrounding`, `distance`, `mood` and `time available`. However, the dataset is rather small and we do not want to use too many contextual factor.

Box plots in Fig. 7.3 are worth discussing. They suggest that contextual parameters usually have the same values, so others are treated as outliers on the box plot. If we take a deeper look into the dataset, we can observe that most of the values for contextual parameters are missing. Thus, these outliers values are most important. We can see the groups of points in the box plots. They usually oscillate around the "perfect" value of unalikeability for considered parameter, e.g. `season` or `distance`. Thus, we can assume that the same contextual parameters are relevant on the user level as those on the dataset level.

To conclude, in further experiments on `Unibz-STS` dataset we used five best contextual parameters: `weekday`, `transport`, `travel goal`, `season` and `budget`.

The results for the unalikeability analysis for `Restaurant & consumer` dataset are presented in Tab. 7.8 and in Fig. 7.4. The best contextual parameters on the dataset level are: `drink level`, `dress preference`, `ambiance`, `interest` and `color`. Similarly to the results

[11]Recall that GCCPs catch the general trend of users interests.

Table 7.7: Unalikeability analysis for `Unibz-STS` dataset.

Contextual parameter	Unalikeability	k	Perfect value	Difference
distance	0.43	2	0.50	**0.07**
time available	0.57	3	0.67	**0.09**
temperature	0.67	6	0.83	0.16
crowdedness	0.56	3	0.67	0.11
knowledge of surrounding	0.60	3	0.67	**0.06**
season	0.72	4	0.75	**0.03**
budget	0.62	3	0.67	**0.05**
daytime	0.66	5	0.80	0.14
weather	0.73	6	0.83	0.11
companion	0.56	5	0.80	0.24
mood	0.67	4	0.75	**0.08**
weekday	0.50	2	0.50	**0.00**
travel goal	0.87	9	0.89	**0.02**
transport	0.75	4	0.75	**0.00**

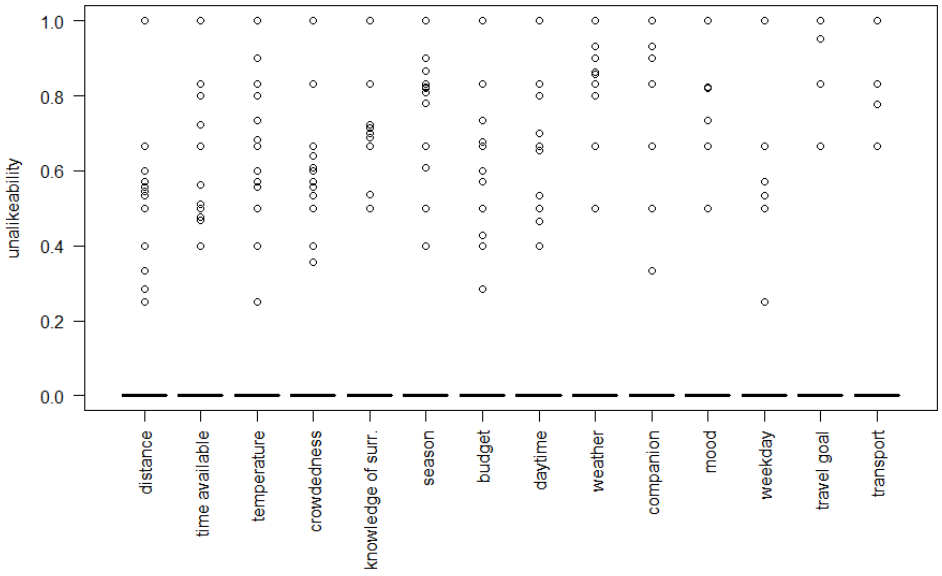


Figure 7.3: Boxplots for unalikeability analysis for `Unibz-STS` dataset (user level).

Table 7.8: Unalikeability analysis for `Restaurant & consumer` dataset.

Contextual parameter	Unalikeability	k	Perfect value	Difference
smoker	0.31	2	0.50	0.19
drink level	0.66	3	0.67	**0.00**
dress preference	0.68	4	0.75	**0.07**
ambiance	0.58	3	0.67	**0.09**
transport	0.55	3	0.67	0.11
marital status	0.14	3	0.67	0.52
children	0.18	3	0.67	0.49
interest	0.74	5	0.80	**0.06**
personality	0.61	4	0.75	0.14
religion	0.40	5	0.80	0.40
activity	0.22	4	0.75	0.53
color	0.81	8	0.88	*0.06*
budget	0.46	3	0.67	0.20

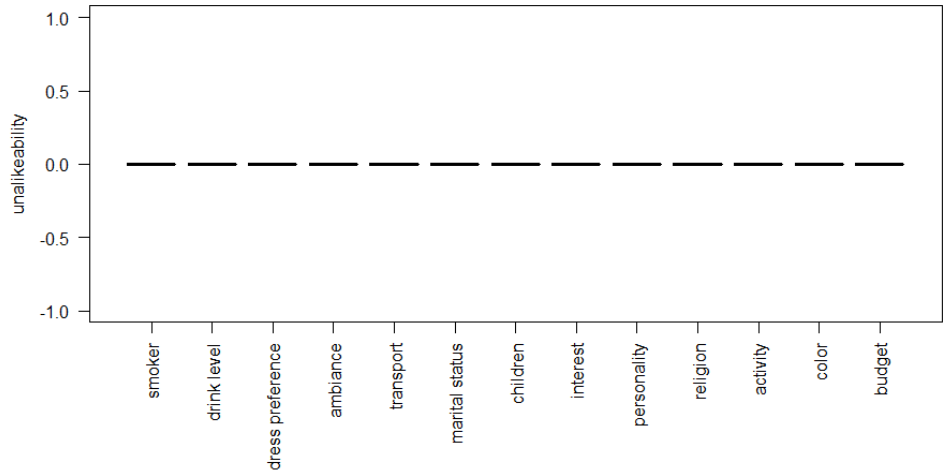


Figure 7.4: Boxplots for unalikeability analysis for `Restaurant & consumer` dataset (user level).

for `Unibz-STS` dataset, the box plots for unalikeability of contextual parameters on the user level needs special attention. They show that the value of a specific factor for a specific user is always fixed. Thus, such factors cannot affect user's individual decisions. However, they can be used to find general trends of users interests. To summarize, we will use five best contextual parameters mentioned above only for experiments with GCCPs. For other experiments we cannot use `Restaurant & consumer` dataset because it does not contain truly contextual factors.

7.3 Preparation of Experiments

This section provides detailed information about libraries that were used in experiments. It also describes how test and training sets were constructed from datasets presented in Section 7.1.

7.3.1 Libraries

For our experiments we used two libraries: LibRec[12] and CARSKit[13]. We describe them separately in the following sections.

LibRec Library

We use LibRec [42] library in the version 1.3, which was the latest version at the time of starting experimentation. The library contains implementations of many state-of-the-art recommendation algorithms, from naïve baselines like Random Guess, through well-established methods like User and Item kNN, to simple context-aware algorithms like Time SVD++. The class structure of the LibRec library is shown in Fig. 7.5.

The library implements many measures for evaluation of algorithms for both recommendation tasks. We used the following ones: MAE, RMSE, precision@5, precision@10, recall@5, recall@10, MAP, nDCG and MRR. Measures of expectedness, unserendipity, novelty and diversity we implemented by ourselves.

The library is easy to work with because it enables two ways of usage: as a library from Java code or directly from a compiled jar file. The only needed things are 3 files: one for configuration, and two for test and training datasets. An exemplary configuration file is shown in Listing 7.2.

Some experiments were performed directly by using jar file. However, we also used the LibRec as a library and created the needed files from the source code.

[12] `https://www.librec.net/`
[13] `https://github.com/irecsys/CARSKit`

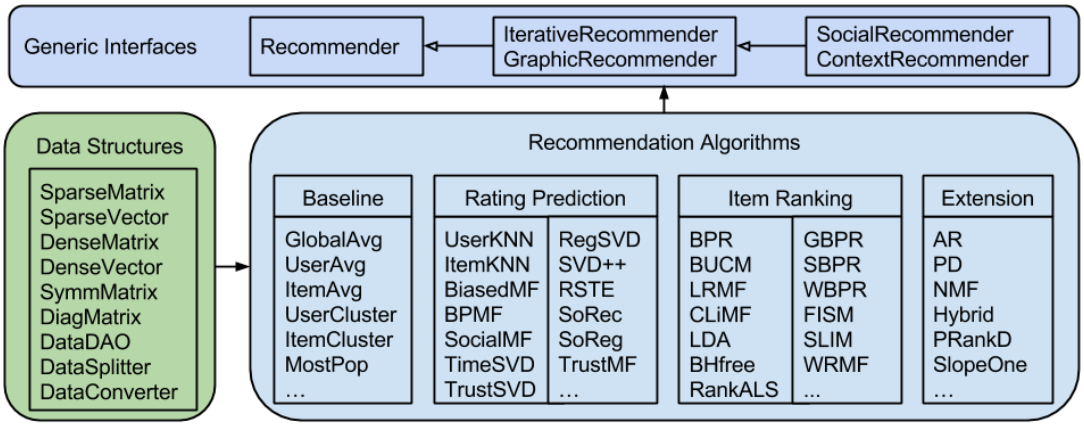


Figure 7.5: The Class Structure of the LibRec Library (from [42]).

Listing 7.2: An exemplary configuration file for LibRec library.

```
dataset.ratings.wins=.\\Datasets\\RC\\training.csv
dataset.ratings.lins=./Datasets/RC/training.csv

ratings.setup=-columns 0 1 2 -threshold -1

recommender=BPR
evaluation.setup=test-set -f .\\Datasets\\RC\\test.csv
item.ranking=on -topN 10 -ignore -1

num.factors=10
num.max.iter=30

learn.rate=0.01 -max -1 -bold-driver
reg.lambda=0.1 -u 0.1 -i 0.1 -b 0.1

output.setup=on -dir ./Results/
```

CARSKit Library

We used CARSKit [112] library in version 0.3.0 to generate context-aware recommendations for comparison with our methods. This library is built based on LibRec library in version 1.3, so we can be sure that results obtained with both of them are comparable with each other. It also follows the running convention with files. However, in CARSKit we need more attributes in training and test sets for contextual parameters.

The library contains implementations of all context-aware algorithms described in Chapter 4. It also reuses implementations of some non-context-aware algorithms from LibRec library. They can be used for running contextual pre- or post-filtering methods, e.g. CASA. The class structure of the CARSKit library is shown in Fig. 7.6.

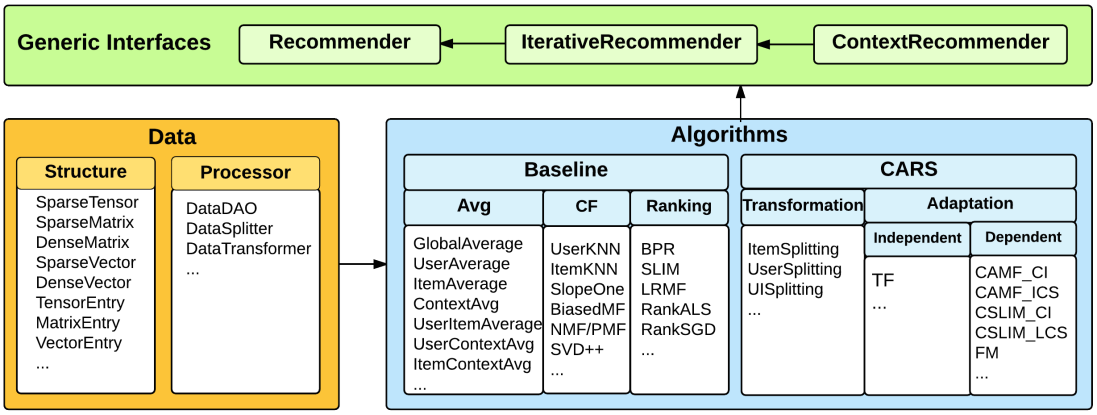


Figure 7.6: The Class Structure of the CARSKit Library (from [112]).

7.3.2 Preparation of Data Splits

In all experiments the hold-out validation was used. Thus, we needed to prepare pairs of training and test sets. For each dataset described in the previous sections we prepared 3 pairs. The first pair was designed for the rating prediction task and was obtained by randomly splitting each user ratings in a 4:1 ratio. For `MovieTweeting` and `ConcertTweets` (both rating scales) datasets we used the `timestamp` attribute and the split was performed based on rating time value.

The second pair of training and test sets was designed for testing the typical scenario of the recommending good items task. In this case we also used a 4:1 ratio. However, in a test set we put only positive ratings of users. The third pair of training and test sets was also designed for generating the list of top k recommendation, but for the new user cold-start situation. For this purpose we had to simulate users who do not have any rating history yet. Thus, we split each dataset by user, not by ratings. We used the same ratio, 4:1, for the split.

7.4 Evaluation of Proposed Methods

This section is a collection of descriptions of experiments that we performed to prove the main and auxiliary theses.

7.4.1 Rating Prediction

We performed experiments for the rating prediction task. As evaluation measures we used the mean absolute error (MAE) and the root of the mean square error (RMSE), which are commonly used for evaluating accuracy of a rating in the rating prediction task. We chose four baseline algorithms from LibRec library for comparison with our methods, i.e. Random Guess, User Average, Item kNN and SVD++. We applied also Time SVD++ algorithm on those datasets that contain information about exact time of consuming an item, i.e. `MovieTweeting` and `ConcertTweets`. It was impossible to use this algorithm on other datasets.

As it was mentioned before, `Restaurant & consumer` dataset does not contain truly

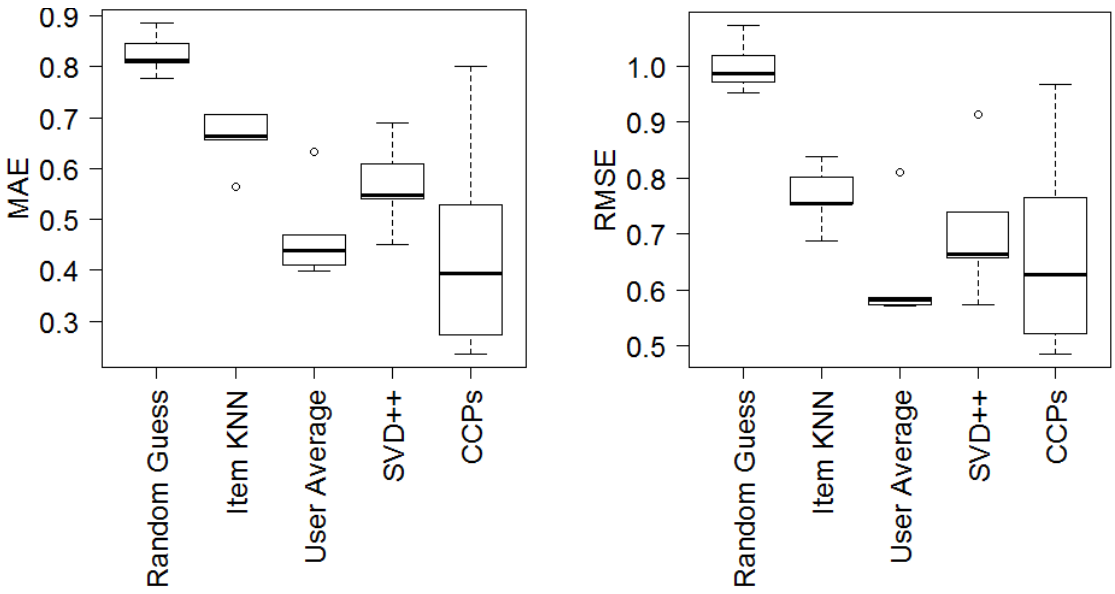


Figure 7.7: Values of MAE and RMSE for different methods applied on `Restaurant & consumer` dataset.

contextual information. Thus, we were unable to perform our ontology-based contextual pre-filtering method on this dataset. Moreover, for rating prediction with CCPs we could use only GCCPs. Results are presented in Fig. 7.7. Our method is denoted by "CCPs".

It should be noticed that User Average algorithm outperforms SVD++. This may be due to the way in which users rate restaurants: it may be possible that they do not use the whole rating scale but just a part of it, e.g. a user evaluates only those restaurants that she likes (her ratings are always greater that 1.0).

Our rating prediction approach with CCPs outperforms all of the baseline algorithms besides User Average. Our method obtained better MAE values but worse RMSE values than User Average. Considering the interpretation of these measures in Chapter 4, we can deduce that our algorithm predicts an accurate rating in more cases than User Average, but the error is bigger than for User Average when it predicts incorrect rating.

We do not know the distribution of results and we cannot guarantee that it is the Gaussian distribution so we cannot use statistical tests for normal distribution, like T-test. Thus, we decided to use the Wilcoxon test which confirmed the statistical significance of a prediction accuracy improvement on all algorithms for CCPs with the p-value smaller than 0.05.

We could use both our methods: ontology-based contextual pre-filtering (with COUP) and rating prediction with CCPs, for all other datasets. In the following figures we denote the first method by the "CPF" prefix.

Results obtained for `LDOS-CoMoDa` and `Unibz-STS` datasets are shown in Figures 7.8, 7.9, 7.10 and 7.11. Since these datasets are quite small, our contextual pre-filtering method is not so effective as for bigger datasets. It gives only slight improvement for `LDOS-CoMoDa` dataset with three of four algorithms. For SVD++ algorithm, contextual pre-filtering received bigger MAE and RMSE errors than the algorithm without pre-filtering. Results obtained for `Unibz-STS` dataset are worth discussing. They show that our ontology-based

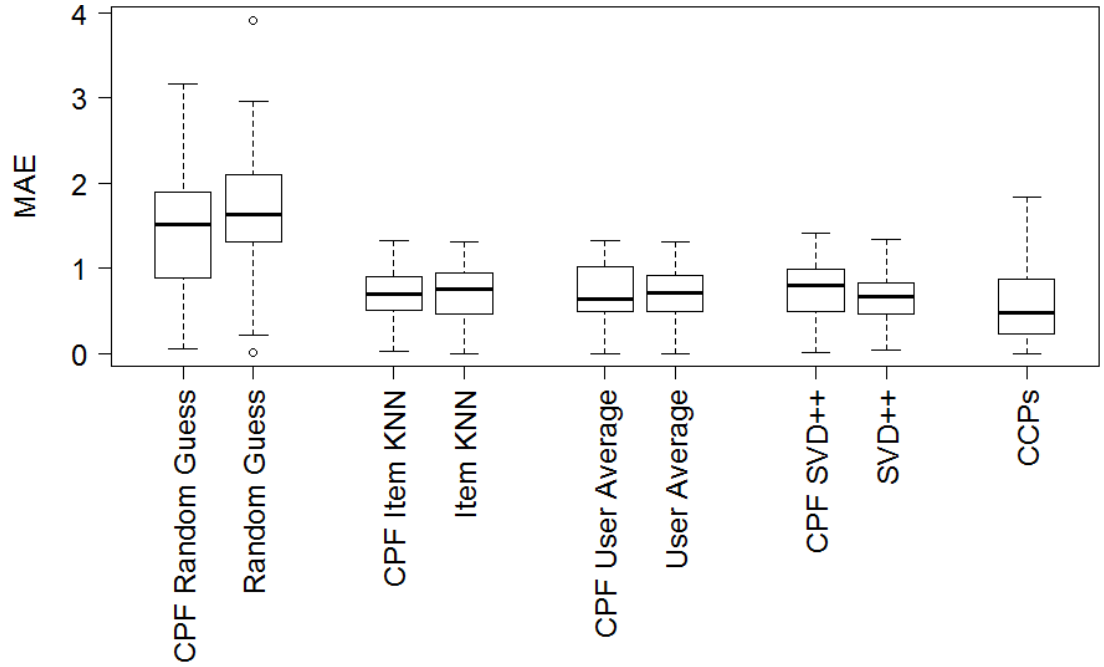


Figure 7.8: Values of MAE for different methods applied on LDOS-CoMoDa dataset.

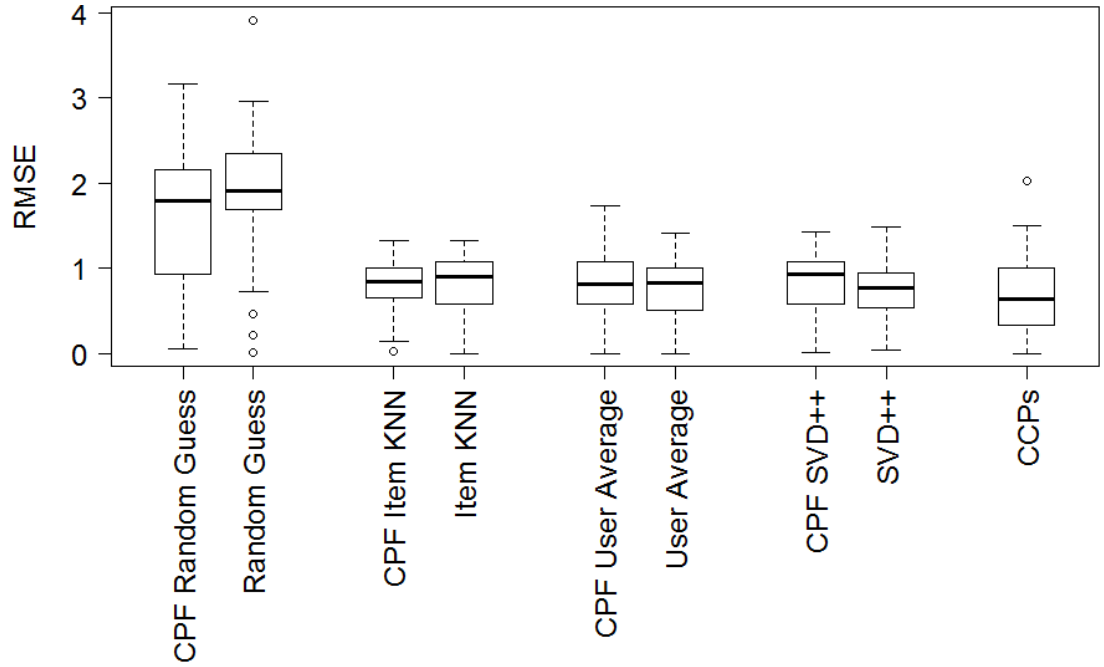


Figure 7.9: Values of RMSE for different methods applied on LDOS-CoMoDa dataset.

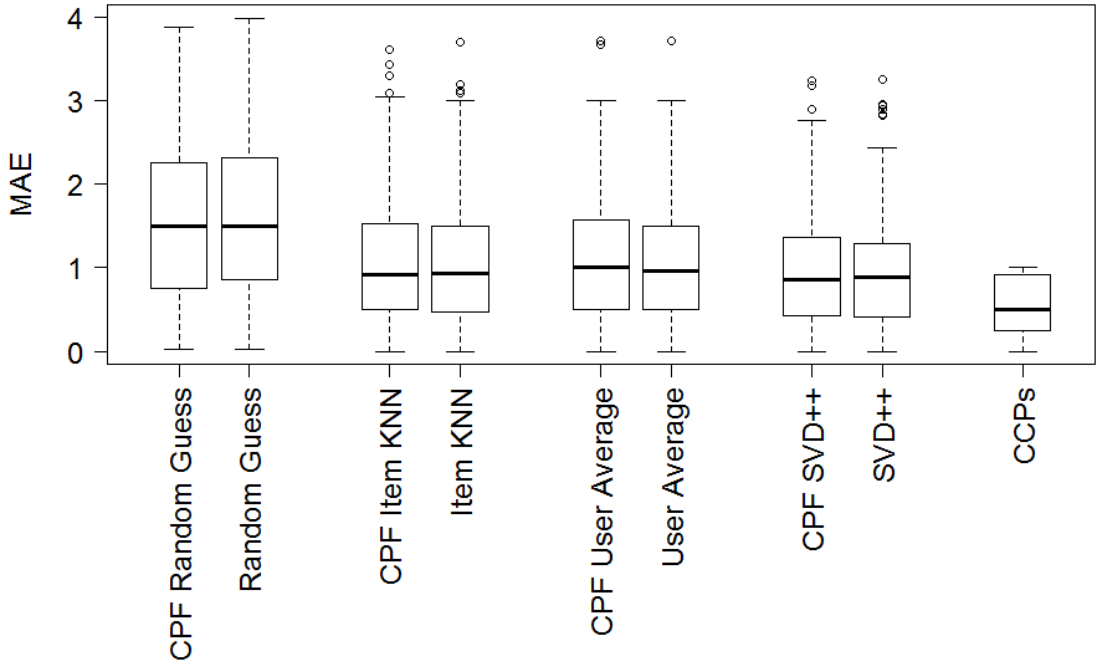


Figure 7.10: Values of MAE for different methods applied on `Unibz-STS` dataset.

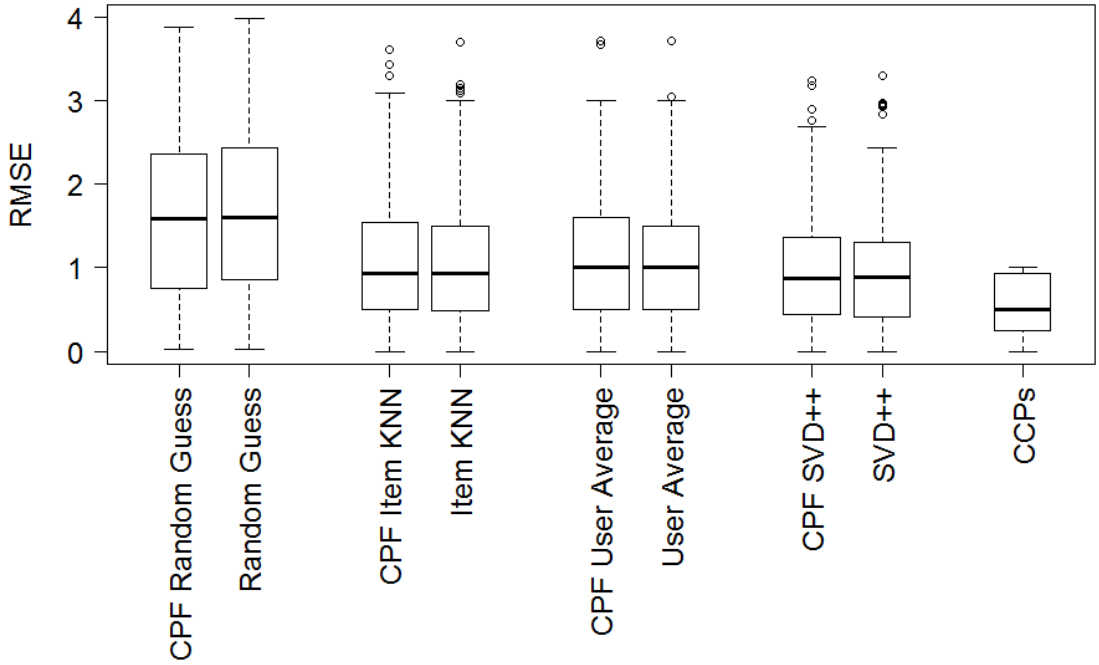


Figure 7.11: Values of RMSE for different methods applied on `Unibz-STS` dataset.

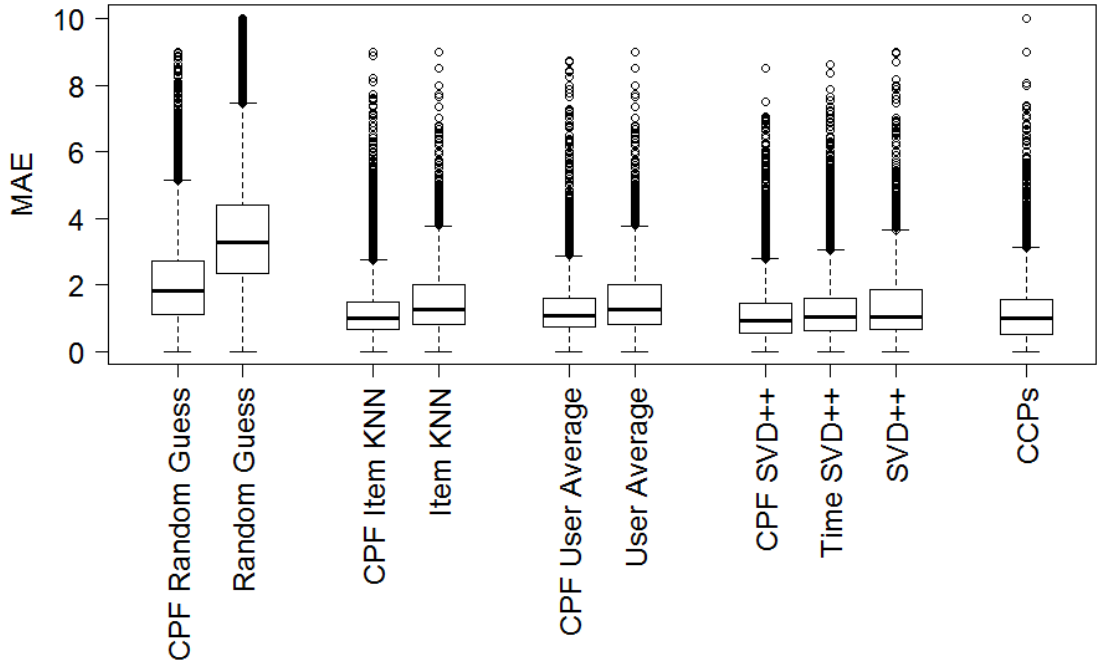


Figure 7.12: Values of MAE for different methods applied on `MovieTweetings` dataset.

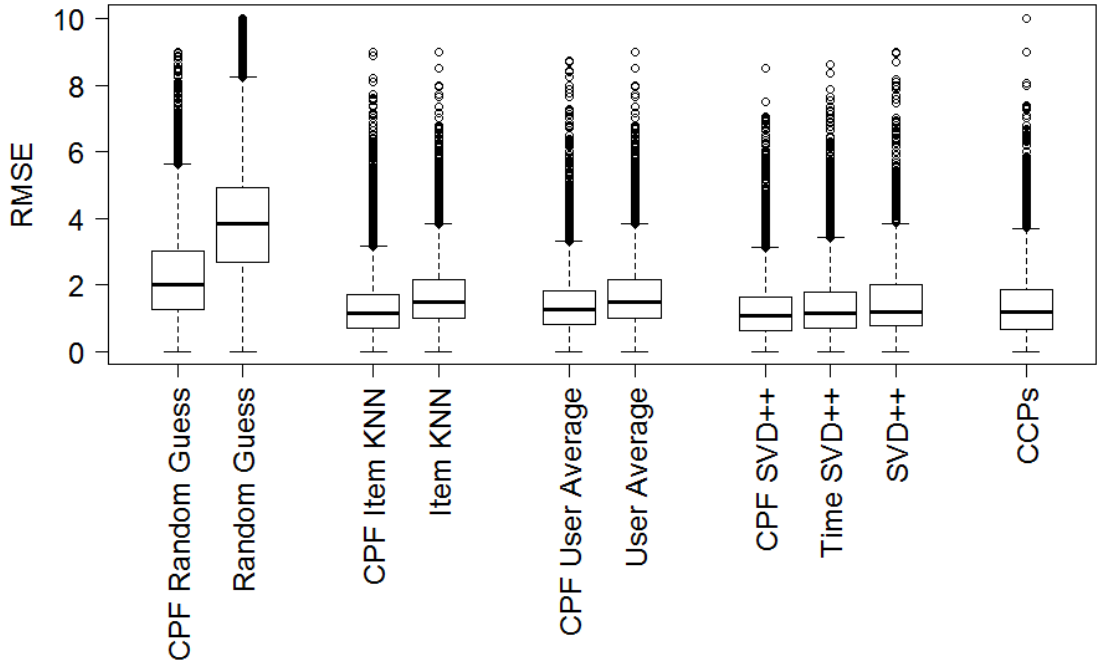


Figure 7.13: Values of RMSE for different methods applied on `MovieTweetings` dataset.

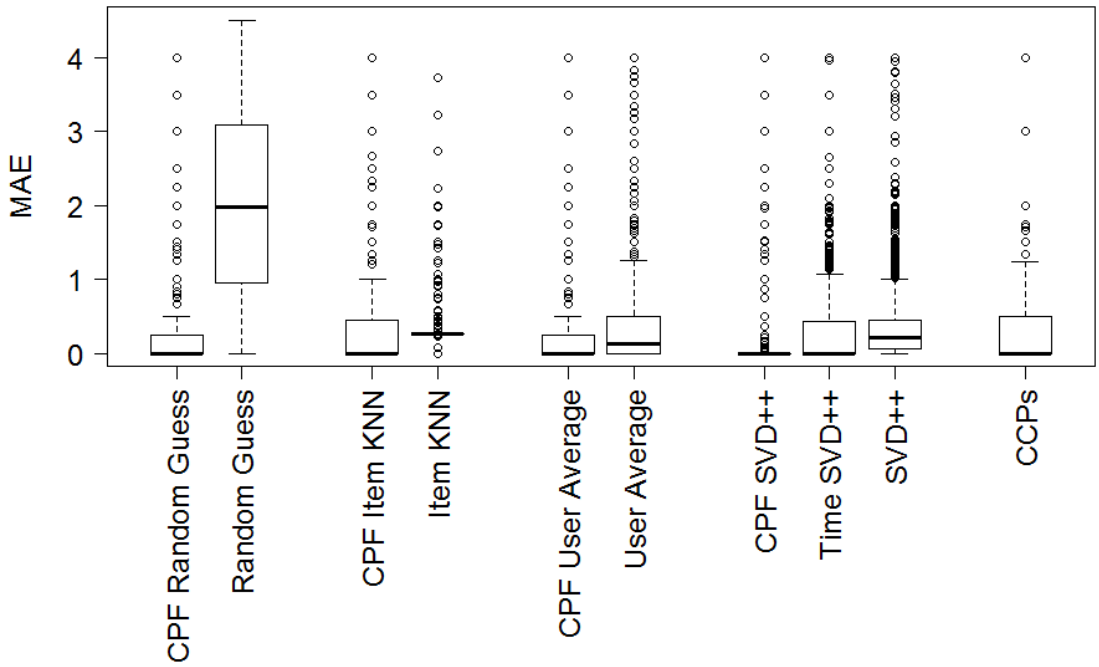


Figure 7.14: Values of MAE for different methods applied on `ConcertTweets` dataset with numerical rating scale.

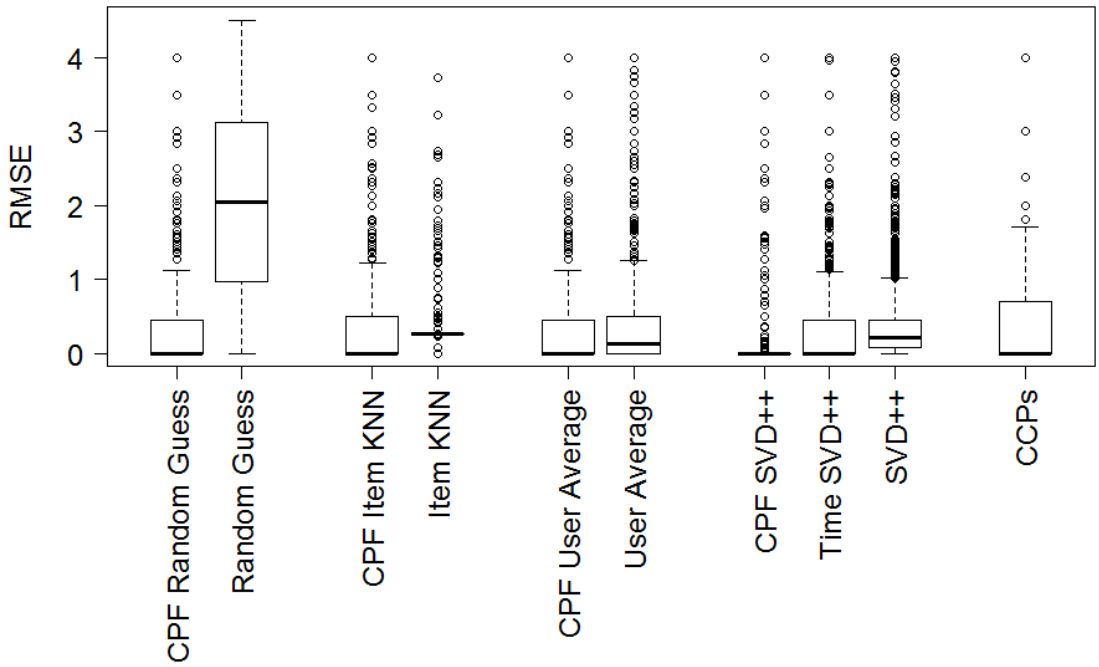


Figure 7.15: Values of RMSE for different methods applied on `ConcertTweets` dataset with numerical rating scale.

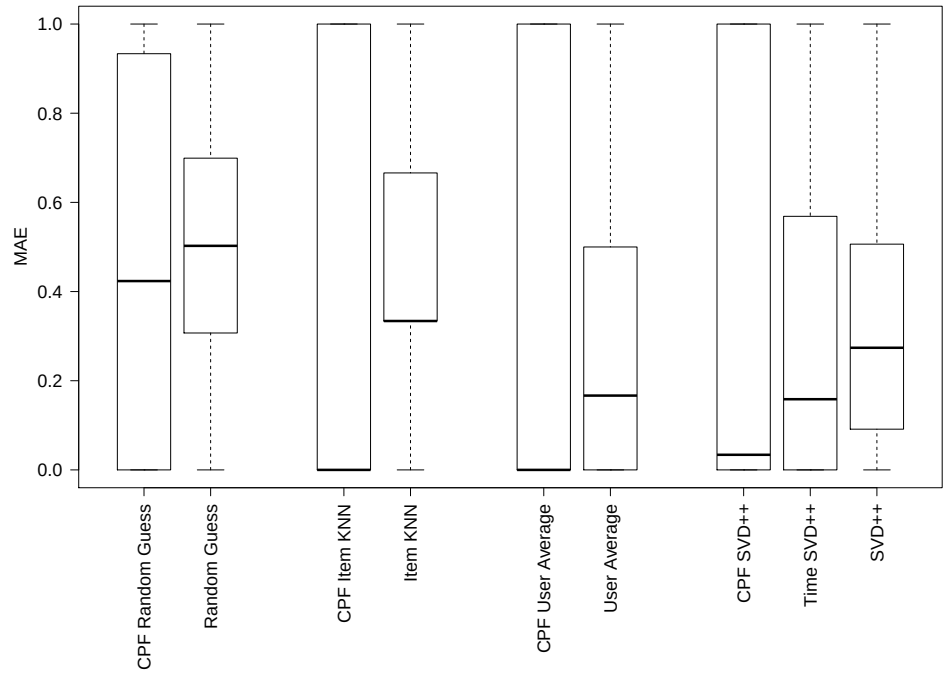


Figure 7.16: Values of MAE for different methods applied on `ConcertTweets` dataset with descriptive rating scale.

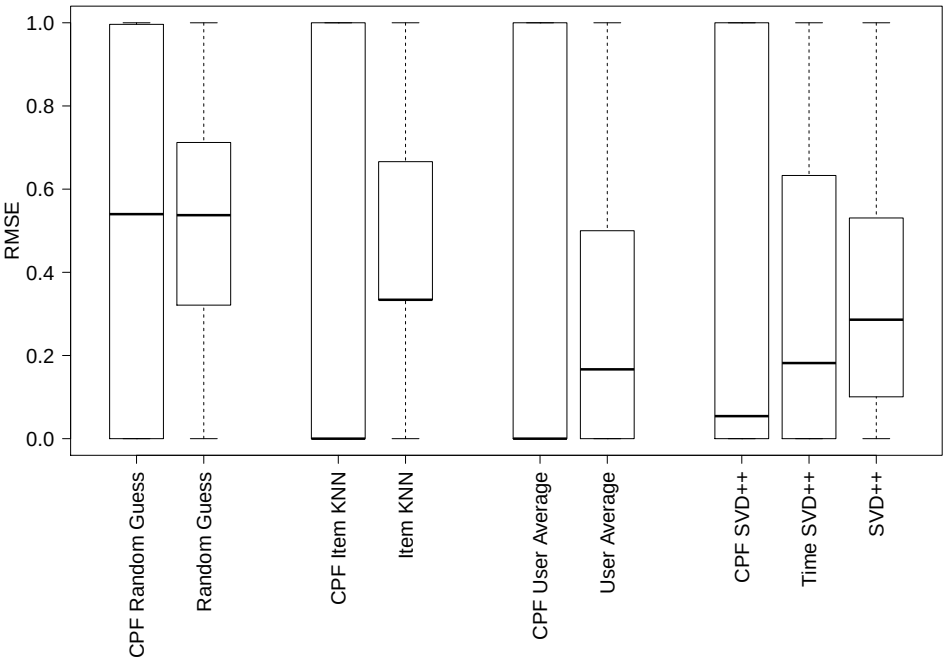


Figure 7.17: Values of RMSE for different methods applied on `ConcertTweets` dataset with descriptive rating scale.

contextual pre-filtering approach did not change values of MAE and RMSE in comparison with pure algorithms. Indeed, this dataset contains many unknown values for contextual parameters, which causes substantial generalization of context during pre-filtering. Thus, for many cases we reached the maximum generalization level that is equivalent to non-contextual data. The difference in errors values for this dataset is statistically insignificant according to the Wilcoxon test. It shows that user modelling in the form of contextual ontology and usage of an ontology of context allow dynamic generalization of contextual parameters values, which partially proves the first auxiliary thesis.

It should be noticed that our algorithm for rating prediction with CCPs outperforms all other algorithms taking into account values of MAE and RMSE on `LDOS-CoMoDa` and `Unibz-STS` datasets. The Wilcoxon test confirmed the statistical significance of results with p-value smaller than 0.05.

Results obtained for `MovieTweetings` and `ConcertTweets` (with numerical rating scale) datasets are shown in Figures 7.12, 7.13, 7.14 and 7.15. We observe that ontology-based contextual pre-filtering method outperforms all other algorithms on these datasets. Our algorithm for rating prediction with CCPs performs similarly to Time SVD++ on these datasets, which means that the algorithm with CCPs works best only on small datasets. Ontology-based contextual pre-filtering is better for bigger datasets.

CCPs could be obtained from numerical ratings as decribed in Chapter 6. Thus, we cannot use the algorithm for rating prediction with CCPs on the subset of `ConcertTweets` dataset with descriptive rating scale. Results obtained for baseline algorithms and ontology-based contextual pre-filtering are shown in Figures 7.16 and 7.17. We see that the median values for our approach are improved for all algorithms but the overall MAE and RMSE values for *descriptive scale* subset is worst for all of the cases. This suggests that in the case of binary scale (*yes*/*maybe*) contextual pre-filtering may increase the sparsity and noisiness of data. Thus, the recommendation algorithm may not always predict the rating. However, the difference between results for two subsets could be caused not by wrong pre-filtering method but by psychological differences between *a priori* and *a posteriori* evaluation by a user. It is more reliable when a user evaluates an item after she consumed it than when she declares what she would do or prefer. This could lead us to the conclusion that this approach could be successfully applied in RSs where numeric scale is used to rate items in *a posteriori* way.

7.4.2 Recommending Good Items

Recommending good items is the most important task in the RSs field from practical point of view. RSs that we see and use in everyday life run computations to generate a list of top k items. Thus, it is important to check the performance of new methods in this task.

We performed several series of experiments to confirm usability of the proposed methods for recommending good items. For evaluation we used nine measures: *precision*, *recall*, *MAP*, *nDCG*, *MRR*, *expectedness*, *unserendipity*, *novelty* and *diversity*, which were described in Chapter 4. Besides *MAP*, *nDCG* and *MRR*, all other measures were computed on the lists of top 5 and top 10 recommendations. However, since the results were similar for both lists we report here only results obtained for the list of top 10 recommendations.

We chose six baseline algorithms: BPR, FISM, LDA, SLIM, User kNN and WRMF from LibRec library and two context-aware methods: CSLIM and User-Item Splitting from

CARSKit library. We tested two scenarios: the typical scenario and the new user cold-start scenario. The first one we tested with both our approaches, while the second one only with the re-rankCCP algorithm.

In the following tables the prefix *ccp-* denotes that re-rankCCP was applied on the list obtained by the algorithm and the prefix *onto-* denotes the algorithm applied on the data returned by ontology-based contextual pre-filtering. The best results within one algorithm group are marked in bold. Globally best results are marked in italic. If results are best globally and within one algorithm group at the same time, they are marked in bold italic.

First we will focus only on the ontology-based contextual pre-filtering and then we will proceed with the re-rankCCP method.

We applied each chosen algorithm on each dataset twice: once as is, and then on data generated by our ontology-based contextual pre-filtering technique. We were unable to finish computations for two algorithms: SLIM and User kNN, on both subsets of `ConcertTweets` and on `MovieTweetings` dataset, because of their computational complexity and the size of datasets. For easy comparison, we did not report results for these algorithms on `LDOS-CoMoDa`. We did not apply contextual pre-filtering on `Restaurant & consumer` dataset because it does not contain truly contextual information. We also did not apply this method on `Unibz-STS` dataset because of the results that we obtained for rating prediction task on this dataset (see Section 7.4.1).

Results are collected in Tables 7.9, 7.10, 7.11 and 7.12. As expected, adding contextual information into the recommendation process increases *precision* and *recall* values for all algorithms and datasets that were used. Moreover, the results show that our ontology-based contextual pre-filtering also improves serendipity of all of the algorithms, i.e. it obtained the smallest values of expectedness and unserendipity. For all of the algorithms on all of the datasets except for the `ConcertTweets` with numerical rating scale, this method does not decrease *diversity* and *novelty* of recommendations.

The second considered method is re-rankCCP. As it could be seen from data in Tables 7.9, 7.10, 7.11, 7.13, 7.14, 7.15, 7.16, 7.17, 7.18 and 7.19, the method is algorithm dependent. It is impossible to identify one algorithm that is better than others in all of the cases for all of the datasets.

It is worth noting that different algorithms which we combined with our method, are good for the typical scenario on the `LDOS-CoMoDa` dataset. In this case, the best algorithm to work with our approach is WRMF, which improves all metrics besides *unserendipity* and *diversity*. As it is seen in Tab. 7.9, all other algorithms combined with our re-rankCCP method improve at least some measures - mostly *nDCG* and *MRR*, which means that good recommendations are usually higher in the ranking than without reshuffling, even if the number of good recommendations in the top 10 list is the same or smaller.

As it could be seen in Tables 7.13 and 7.14, our re-rankCCP method performs best in the typical scenario when combined with BPR and SLIM algorithms for the `Unibiz-STS` and `Restaurant & consumer` datasets, i.e. it obtained the biggest increase in precision and recall values in comparison with other algorithms. For the `Unibiz-STS` dataset, our method with BPR algorithm gives better results for the *novelty* measure than UI Splitting with BPR, which is surprising, since UI Splitting improves *novelty* for almost all of the cases for all of the datasets.

In Tables 7.15, 7.16, 7.17, 7.18 and 7.19 we omitted unserendipity measure since it is

Table 7.9: Measures for the typical scenario for `LDOS-CoMoDa` dataset.

Algorithm	Precision	Recall	MAP	nDCG	MRR	Expectedness	Unserendipity	Novelty	Diversity
CSLIM	0.0000	0.0000	0.0000	0.0000	0.0000	0.0013	0.1980	9.94	0.4099
ontoBPR	**0.0115**	**0.1154**	0.0000	0.0000	0.0000	**0.0015**	***0.1701***	9.76	0.4047
ccp-BPR	0.0075	0.0259	**0.0075**	**0.0151**	0.0209	**0.0015**	0.3206	**9.77**	0.3103
UIS-BPR	0.0014	0.0071	0.0021	0.0041	0.0042	0.0018	0.1980	9.53	**0.4065**
BPR	0.0075	0.0235	0.0070	0.0144	**0.0213**	0.0016	0.3063	9.73	0.3427
ontoFISM	***0.0308***	***0.3077***	0.0192	0.0243	0.0192	0.0041	**0.1920**	8.15	**0.3930**
ccp-FISM	0.0123	0.0823	**0.0473**	0.0601	0.0659	**0.0034**	0.3304	**8.33**	0.2866
FISM	0.0130	0.0897	0.0462	**0.0615**	**0.0675**	0.0046	0.3148	7.82	0.3174
ontoLDA	**0.0200**	**0.2000**	0.0067	0.0142	0.0067	0.0045	**0.1953**	7.93	**0.3764**
ccp-LDA	0.0130	0.0891	***0.0504***	0.0641	***0.0691***	**0.0034**	0.3301	**8.32**	0.2864
LDA	0.0137	0.0965	0.0481	***0.0645***	0.0686	0.0046	0.3165	7.80	0.3189
ccp-SLIM	**0.0062**	**0.0377**	**0.0153**	**0.0226**	0.0204	0.0016	0.3186	9.80	0.3128
UIS-SLIM	0.0000	0.0000	0.0000	0.0000	0.0000	***0.0000***	**0.1945**	***Infinity***	**0.4098**
SLIM	0.0055	0.0360	0.0147	0.0217	**0.0216**	0.0016	0.2968	9.8361	0.3527
ccp-UserKNN	**0.0068**	**0.0438**	**0.0249**	**0.0314**	**0.0322**	0.0020	0.3248	9.30	0.2995
UIS-UserKNN	0.0012	0.0122	0.0030	0.0052	0.0030	***0.0000***	**0.1776**	***Infinity***	***0.4136***
UserKNN	0.0062	0.0386	0.0136	0.0215	0.0206	0.0026	0.3128	8.84	0.3247
ontoWRMF	**0.0269**	**0.2692**	0.0103	0.0248	0.0103	**0.0017**	**0.1782**	**9.68**	**0.3988**
ccp-WRMF	0.0075	0.0512	**0.0281**	**0.0348**	**0.0307**	0.0020	0.3280	9.41	0.3020
WRMF	0.0048	0.0324	0.0070	0.0140	0.0101	0.0026	0.3190	8.89	0.3260

Table 7.10: Measures for the typical scenario for `MovieTweetings` dataset.

Algorithm	Precision	Recall	MAP	nDCG	MRR	Expectedness	Unserendipity	Novelty	Diversity
ontoBPR	**0.0261**	0.2607	**0.0563**	**0.0400**	0.0042	**0.0037**	**0.2696**	**8.48**	**0.3002**
ccp-BPR	0.0075	**0.0491**	0.0161	0.0259	0.0240	0.0046	0.2969	7.84	0.2872
UIS-BPR	0.0052	0.0075	0.0095	0.0261	**0.0641**	0.0041	0.3071	8.07	0.2938
BPR	0.0075	0.02713	0.0179	0.0275	0.0262	0.0051	0.3057	7.74	0.2964
ontoFISM	**0.0094**	**0.0942**	**0.0167**	0.0105	0.0010	***0.0011***	***0.2123***	***10.98***	***0.3824***
ccp-FISM	0.0048	0.0313	0.0075	0.0141	0.0113	0.0015	0.2744	10.60	0.3437
FISM	0.0049	0.0320	0.0080	**0.0147**	**0.0116**	0.0019	0.3030	10.63	0.3273
ontoLDA	**0.0267**	**0.2670**	***0.0597***	**0.0425**	0.0048	0.0042	**0.2735**	7.96	0.2866
ccp-LDA	0.0087	0.0576	0.0172	0.0288	0.0248	**0.0037**	0.3102	**8.26**	0.2860
LDA	0.0094	0.0628	0.0205	0.0327	**0.0288**	0.0042	0.3139	8.08	**0.2986**
ontoWRMF	**0.0305**	**0.3046**	**0.0505**	***0.0707***	***0.0505***	0.0053	**0.2865**	**9.15**	**0.3156**
ccp-WRMF	0.0082	0.0546	0.0164	0.0272	0.0229	**0.0032**	0.3147	8.51	0.2924
WRMF	0.0086	0.0581	0.0194	0.0305	0.0267	0.0035	0.3221	8.33	0.2963

Table 7.11: Measures for the typical scenario for `ConcertTweets` dataset with numerical rating scale.

Algorithm	Precision	Recall	MAP	nDCG	MRR	Expectedness	Unserendipity	Novelty	Diversity
ontoBPR	**0.0853**	**0.8533**	**0.3953**	**0.4894**	**0.3953**	***0.00002***	***0.0346***	11.77	0.2773
ccp-BPR	0.0006	0.0063	0.0022	0.0032	0.0022	0.00003	0.1136	16.28	0.4274
UIS-BPR	0.0021	0.0213	0.0099	0.0125	0.0099	0.00003	0.1151	16.37	***0.4518***
BPR	0.0003	0.0031	0.0010	0.0016	0.0010	0.00003	0.0985	***16.45***	0.4510
ontoFISM	**0.0853**	**0.8533**	**0.4214**	**0.5096**	**0.4214**	***0.00002***	**0.0377**	11.67	0.2741
ccp-FISM	0.0025	0.0236	0.0077	0.0115	0.0079	0.00004	0.1396	**15.48**	0.4037
FISM	0.0031	0.0299	0.0082	0.0132	0.0085	0.00004	0.1419	15.44	**0.4234**
ontoLDA	**0.0233**	**0.2328**	**0.1826**	**0.2313**	**0.1826**	**0.00005**	**0.0745**	**14.36**	**0.4351**
ccp-LDA	0.0000	0.0000	0.0000	0.0000	0.0000	0.00007	0.1238	14.25	0.4061
LDA	0.0009	0.0079	0.0008	0.0024	0.0010	0.00008	0.1188	14.28	0.3613
ontoWRMF	***0.0867***	***0.8667***	***0.4574***	***0.5430***	***0.4574***	***0.00002***	***0.0346***	11.75	0.2764
ccp-WRMF	0.0013	0.0110	0.0015	0.0037	0.0017	0.00006	0.1331	**14.60**	0.3601
WRMF	0.0016	0.0142	0.0040	0.0067	0.0056	0.00008	0.1338	14.20	**0.3922**

Table 7.12: Results obtained for `ConcertTweets` subset with a descriptive rating scale.

Algorithm	Precision	Recall	MAP	nDCG	MRR	Expectedness	Unserendipity	Novelty	Diversity
ontoBPR	**0.0366**	**0.3665**	**0.1453**	**0.1892**	**0.1453**	***0.00002***	**0.0515**	15.63	0.4116
UIS-BPR	0.0004	0.0019	0.0014	0.0022	0.0028	0.00003	0.0936	16.44	0.4523
BPR	0.0006	0.0039	0.0017	0.0025	0.0021	0.00003	0.0889	***16.51***	***0.4524***
ontoFISM	**0.0336**	**0.3365**	**0.1339**	**0.1744**	**0.1339**	***0.00002***	***0.0508***	**15.30**	**0.4134**
FISM	0.0002	0.0014	0.0004	0.0008	0.0006	0.00009	0.0687	13.77	0.3947
ontoLDA	**0.0218**	**0.2182**	**0.0865**	**0.1113**	**0.0865**	**0.00004**	**0.0668**	**15.08**	**0.4377**
LDA	0.0003	0.0012	0.0004	0.0008	0.0006	0.00008	0.0968	13.88	0.4309
ontoWRMF	***0.0381***	***0.3806***	***0.1575***	***0.2010***	***0.1575***	***0.00002***	**0.0548**	**15.80**	0.4106
WRMF	0.0009	0.0022	0.0014	0.0024	0.0029	0.00003	0.0878	15.48	**0.4505**

Table 7.13: Measures for the typical scenario for `Restaurant & consumer` dataset.

Algorithm	Precision	Recall	MAP	nDCG	MRR	Expectedness	Unserendipity	Novelty	Diversity
CSLIM	0.0581	0.4068	0.1413	0.2173	0.1806	0.0091	0.3225	7.06	0.3393
ccp-BPR	***0.1129***	***0.7473***	***0.4723***	***0.4527***	0.2938	0.1001	0.6137	3.57	0.1715
UIS-BPR	0.0720	0.5000	0.1869	0.2781	0.2411	**0.0110**	**0.3281**	**6.76**	**0.3352**
BPR	0.0817	0.5448	0.2636	0.3519	***0.3306***	0.0971	0.6063	3.64	0.1976
ccp-FISM	**0.0538**	**0.3082**	**0.1960**	**0.1951**	0.1400	0.1671	0.6065	2.58	0.1687
FISM	0.0409	0.2312	0.1220	0.1630	**0.1715**	**0.1568**	**0.5906**	**2.71**	**0.2060**
ccp-LDA	**0.0516**	**0.2975**	**0.1982**	**0.1927**	0.1398	0.1673	0.6070	2.57	0.1676
LDA	0.0387	0.2222	0.1222	0.1601	**0.1698**	**0.1575**	**0.6072**	**2.70**	**0.1854**
ccp-SLIM	**0.1000**	**0.6703**	**0.3490**	**0.3732**	0.2280	0.0939	0.6140	3.65	0.1722
UIS-SLIM	0.0194	0.1165	0.0524	0.0758	0.0824	***0.0081***	***0.2930***	***7.27***	***0.3593***
SLIM	0.0860	0.5824	0.2469	0.3452	**0.2987**	0.0907	0.6039	3.75	0.1985
ccp-UserKNN	**0.0452**	**0.2885**	**0.1824**	**0.1795**	0.1258	0.0806	0.6063	3.87	0.1736
UIS-UserKNN	0.0183	0.1022	0.0302	0.0543	0.0505	**0.0086**	**0.3027**	**7.15**	**0.3520**
UserKNN	0.0366	0.2240	0.0935	0.1397	**0.1489**	0.0789	0.5872	3.93	0.2077
ccp-WRMF	**0.0892**	**0.5502**	**0.2931**	**0.3418**	0.2720	0.0943	0.6107	3.58	0.1706
WRMF	0.0828	0.5287	0.2376	0.3298	**0.3108**	**0.0884**	**0.6009**	**3.73**	**0.2004**

Table 7.14: Measures for the typical scenario for `Unibz-STS` dataset.

Algorithm	Precision	Recall	MAP	nDCG	MRR	Expectedness	Unserendipity	Novelty	Diversity
CSLIM	0.0615	0.5426	0.1927	0.2773	0.2007	0.0237	0.2353	8.14	*0.4321*
ccp-BPR	***0.1129***	***0.7473***	***0.4723***	***0.4527***	0.2938	0.0064	0.6137	7.49	0.1715
UIS-BPR	0.0844	0.7393	0.2714	0.3859	0.2789	0.0553	**0.3122**	4.64	**0.3727**
BPR	0.0817	0.5448	0.2636	0.3519	***0.3306***	**0.0062**	0.6063	**7.60**	0.1976
ccp-FISM	**0.0538**	**0.3082**	**0.1960**	**0.1951**	0.1400	0.0107	0.6065	6.49	0.1687
FISM	0.0409	0.2312	0.1220	0.1630	**0.1715**	**0.0101**	**0.5906**	**6.67**	**0.2060**
ccp-LDA	**0.0516**	**0.2975**	**0.1982**	**0.1927**	0.1398	0.0107	0.6070	6.49	0.1676
LDA	0.0387	0.2222	0.1222	0.1601	**0.1698**	**0.0101**	**0.6072**	**6.66**	**0.1854**
ccp-SLIM	**0.1000**	**0.6703**	**0.3490**	**0.3732**	0.2280	0.0060	0.6140	7.56	0.1722
UIS-SLIM	0.0728	0.6452	0.2787	0.3701	0.2900	0.0365	**0.3211**	6.16	**0.3744**
SLIM	0.0860	0.5824	0.2469	0.3452	**0.2987**	**0.0058**	0.6039	**7.71**	0.1985
ccp-UserKNN	**0.0452**	**0.2885**	**0.1824**	**0.1795**	0.1258	0.0052	0.6063	7.78	0.1736
UIS-UserKNN	0.0095	0.0906	0.0230	0.0385	0.0234	0.0097	***0.1678***	***9.05***	**0.4304**
UserKNN	0.0366	0.2240	0.0935	0.1397	**0.1489**	***0.0051***	0.5872	7.89	0.2077
ccp-WRMF	**0.0892**	**0.5502**	**0.2931**	**0.3418**	0.2720	0.0061	0.6107	7.50	0.1706
WRMF	0.0828	0.5287	0.2376	0.3298	**0.3108**	**0.0057**	**0.6009**	**7.69**	**0.2004**

Table 7.15: Measures for the new user cold-start scenario for `LDOS-CoMoDa` dataset.

Algorithm	Precision	Recall	MAP	nDCG	MRR	Expectedness	Novelty	Diversity
CSLIM	0.0032	0.0117	0.0045	0.0075	0.0077	*0.0000*	*Infinity*	*0.4257*
ccp-BPR	**0.0026**	**0.0134**	0.0019	**0.0049**	0.0037	0.0013	9.85	0.3117
UIS-BPR	0.0008	0.0025	**0.0022**	0.0029	**0.0050**	***0.0000***	***Infinity***	**0.3979**
BPR	0.0013	0.0006	0.0001	0.0006	0.0018	0.0013	9.77	0.3377
ccp-FISM	***0.0218***	**0.0992**	**0.0382**	**0.0615**	***0.0765***	**0.0037**	**8.23**	0.2843
FISM	0.0179	0.0605	0.0298	0.0455	0.0702	0.0051	7.71	**0.2996**
ccp-LDA	***0.0218***	***0.1114***	***0.0383***	***0.0637***	***0.0765***	**0.0039**	**8.18**	0.2874
LDA	0.0154	0.0471	0.0280	0.0409	0.0682	0.0054	7.61	**0.2997**
ccp-SLIM	**0.0077**	**0.0330**	**0.0086**	**0.0176**	**0.0238**	0.0016	9.83	0.3180
UIS-SLIM	0.0032	0.0117	0.0045	0.0074	0.0077	***0.0000***	***Infinity***	***0.4257***
SLIM	0.0064	0.0202	0.0085	0.0140	0.0173	0.0019	9.42	0.3661
ccp-UserKNN	**0.0077**	**0.0330**	**0.0086**	**0.0176**	**0.0238**	0.0016	9.83	0.3180
UIS-UserKNN	0.0032	0.0117	0.0045	0.0074	0.0077	***0.0000***	***Infinity***	***0.4257***
UserKNN	0.0064	0.0202	0.0085	0.0140	0.0173	0.0019	9.42	0.3661
ccp-WRMF	**0.0077**	**0.0330**	**0.0086**	**0.0176**	**0.0238**	**0.0016**	**9.83**	0.3180
WRMF	0.0064	0.0202	0.0085	0.0140	0.0173	0.0019	9.42	**0.3661**

always equal to zero. It is because unserendipity checks the similarity of items in the recommendations list to items in the user history/profile. Here, we consider only new users who have not rated any item yet.

For the new user scenario with the `LDOS-CoMoDa` dataset, our re-rankCCP method works best with FISM and LDA algorithms. They improve all of the measures besides *diversity*. The improvements vary for different measures but they are greater than 35% in comparison with pure algorithms for the first four measures. The re-rankCCP with other algorithms also gives slightly better results than the pure algoritms in the new user scenario. Surprisingly, baseline context-aware algorithms perform pretty weak according to the accuracy measures. However, they obtained the best values for *expectedness*, *novelty* and *diversity* measures, which is shown in Tab. 7.15.

For the new user scenario with the `Unibiz-STS` dataset, the UI Splitting method with BPR algorithm outperforms all other methods according to all measures. For the re-rankCCP method, the best algorithms are SLIM, User kNN and WRMF, which improve all of accuracy measures and *expectedness* and only slightly decrease *diversity*, which is presented in Tab. 7.16.

For the new user scenario with the `Restaurant & consumer` dataset, our re-rankCCP method outperforms all other algorithms according to the accuracy measures when combined with BPR, FISM and LDA algorithms, as shown in Tab. 7.17. For `ConcertTweets` dataset it performs pretty weak (see Tab. 7.18) and for `MovieTweetings` it works well with FISM and WRMF algorithms (see Tab. 7.19). Thus, we could conclude that there is no algorithm that always performs best with our re-rankCCP method. It depends on the scenario and the dataset that the experiments are performed on.

From Tables 7.15, 7.16 and 7.17 we could observe that CSLIM and UI Splitting with SLIM and User KNN give exactly the same results for all the datasets in the new user cold-start scenario. However, this never occurs for the typical scenario.

CSLIM and UI Splitting almost always gave better values of *expectedness*, *unserendipity*, *novelty* and *diversity*. Nevertheless, they gave the worst *precision* and *recall* values for all

Table 7.16: Measures for the new user cold-start scenario for `Unibz-STS` dataset.

Algorithm	Precision	Recall	MAP	nDCG	MRR	Expectedness	Novelty	Diversity
CSLIM	0.1121	0.2868	0.0706	0.1571	0.1454	*0.0000*	*Infinity*	0.1889
ccp-BPR	0.1165	0.3057	0.1836	0.2623	0.3573	0.0417	3.95	0.2931
UIS-BPR	***0.2664***	***0.6596***	***0.4328***	***0.5186***	***0.5217***	***0.0000***	***Infinity***	***0.3748***
BPR	0.2055	0.5761	0.3583	0.4397	0.4454	0.0634	4.29	0.3707
ccp-FISM	0.1174	0.3103	0.1883	0.2673	0.3629	**0.0417**	3.97	0.2894
FISM	**0.2055**	**0.5728**	**0.3557**	**0.4362**	**0.4358**	0.0674	**4.12**	**0.3667**
ccp-LDA	0.1174	0.3103	0.1903	0.2688	0.3637	**0.0417**	3.97	0.2894
LDA	**0.2055**	**0.5728**	**0.3630**	**0.4427**	**0.4498**	0.0674	**4.12**	**0.3667**
ccp-SLIM	0.1083	0.2610	**0.0790**	**0.1578**	**0.1656**	0.0465	3.76	0.1864
UIS-SLIM	**0.1121**	**0.2868**	0.0706	0.1571	0.1454	***0.0000***	***Infinity***	**0.1889**
SLIM	0.0899	0.2685	0.0582	0.1354	0.1212	0.0551	5.44	**0.1889**
ccp-UserKNN	0.1083	0.2610	**0.0790**	**0.1578**	**0.1656**	0.0465	3.76	0.1864
UIS-UserKNN	**0.1121**	**0.2868**	0.0706	0.1571	0.1454	***0.0000***	***Infinity***	**0.1889**
UserKNN	0.0899	0.2685	0.0582	0.1354	0.1212	0.0551	5.44	**0.1889**
ccp-WRMF	**0.1083**	0.2610	**0.0790**	**0.1578**	**0.1656**	**0.0465**	3.76	0.1864
WRMF	0.0899	**0.2685**	0.0582	0.1354	0.1212	0.0551	**5.44**	**0.1889**

Table 7.17: Measures for the new user cold-start scenario for `Restaurant & consumer` dataset.

Algorithm	Precision	Recall	MAP	nDCG	MRR	Expectedness	Novelty	Diversity
CSLIM	0.0958	0.1233	0.0671	0.1282	0.2472	*0.0131*	*6.71*	*0.3339*
ccp-BPR	**0.2000**	**0.1962**	**0.1518**	**0.2192**	0.3444	0.1588	2.70	0.1325
UIS-BPR	0.1167	0.1437	0.0858	0.1529	0.2903	**0.0178**	**5.91**	**0.3214**
BPR	0.1750	0.1703	0.1120	0.1952	***0.3869***	0.1520	2.76	0.1753
ccp-FISM	**0.2000**	**0.1897**	***0.1578***	**0.2185**	0.3304	0.1717	2.58	0.1684
FISM	0.1714	0.1680	0.1074	0.1859	**0.3533**	**0.1562**	**2.71**	**0.1918**
ccp-LDA	***0.2071***	***0.1965***	**0.1535**	***0.2209***	0.3299	0.1724	2.57	0.1637
LDA	0.1571	0.1502	0.0965	0.1732	**0.3474**	**0.1569**	**2.70**	**0.1933**
ccp-SLIM	**0.1571**	**0.1591**	**0.1271**	**0.1833**	0.3191	0.1352	3.22	0.1358
UIS-SLIM	0.0958	0.1233	0.0671	0.1282	0.2472	***0.0131***	***6.71***	***0.3339***
SLIM	0.1179	0.1290	0.0790	0.1558	**0.3726**	0.0938	3.87	0.1844
ccp-UserKNN	**0.1571**	**0.1591**	**0.1271**	**0.1833**	0.3191	0.1352	3.22	0.1358
UIS-UserKNN	0.0958	0.1233	0.0671	0.1282	0.2472	***0.0131***	***6.71***	***0.3339***
UserKNN	0.1179	0.1290	0.0790	0.1558	**0.3726**	0.0938	3.87	0.1844
ccp-WRMF	**0.1571**	**0.1591**	**0.1271**	**0.1833**	0.3191	0.1352	3.22	0.1358
WRMF	0.1179	0.1290	0.0790	0.1558	**0.3726**	**0.0938**	**3.87**	**0.1844**

Table 7.18: Measures for the new user cold-start scenario for `ConcertTweets` dataset.

Algorithm	Precision	Recall	MAP	nDCG	MRR	Expectedness	Novelty	Diversity
ccp-BPR	0.0004	**0.0039**	**0.0030**	**0.0032**	0.0030	0.00007	14.32	0.4003
UIS-BPR	**0.0008**	0.0025	0.0022	0.0029	**0.0050**	**0.00003**	***Infinity***	0.3979
BPR	0.0004	0.0039	0.0009	0.0016	0.0009	0.00007	14.22	**0.4255**
ccp-FISM	0.0001	0.0013	0.0002	0.0004	0.0002	0.00006	14.89	0.4053
FISM	**0.0003**	**0.0019**	**0.0004**	**0.0008**	**0.0005**	***0.00005***	**15.39**	**0.4356**
ccp-LDA	0.0005	0.0045	**0.0045**	0.0047	***0.0051***	**0.00009**	13.89	**0.3954**
LDA	***0.0009***	***0.0090***	0.0038	***0.0050***	0.0038	0.00010	**13.93**	0.3889
ccp-WRMF	0.0000	0.0000	0.0000	0.0000	0.0000	0.00003	16.11	0.4402
WRMF	0.0000	0.0000	0.0000	0.0000	0.0000	***0.00002***	**17.37**	***0.4501***

Table 7.19: Measures for the new user cold-start scenario for `MovieTweetings` dataset.

Algorithm	Precision	Recall	MAP	nDCG	MRR	Expectedness	Novelty	Diversity
ccp-BPR	0.0127	0.0629	0.0200	0.0345	0.0372	0.0055	7.55	0.2797
UIS-BPR	***0.0162***	0.0151	***0.0289***	***0.0703***	***0.2633***	**0.0041**	**8.08**	**0.3051**
BPR	0.0135	**0.0699**	0.0238	0.0393	0.0423	0.0056	7.53	0.2893
ccp-FISM	**0.0065**	**0.0320**	**0.0108**	**0.0183**	**0.0222**	0.0029	9.32	0.3012
FISM	0.0025	0.0124	0.0023	0.0052	0.0048	***0.0010***	***10.92***	**0.3319**
ccp-LDA	0.0124	0.0611	0.0204	0.0345	0.0381	**0.0055**	**7.53**	0.2521
LDA	**0.0142**	***0.0729***	**0.0247**	**0.0411**	**0.0440**	0.0062	7.35	**0.2895**
ccp-WRMF	**0.0068**	**0.0301**	**0.0096**	**0.0172**	**0.0210**	0.0027	9.43	***0.3470***
WRMF	0.0050	0.0230	0.0062	0.0117	0.0118	**0.0019**	**10.10**	0.3396

of the cases besides the new user cold-start scenario for the `Unibiz-STS` dataset, when UI Splitting with BPR performed best.

The value of *diversity* always decreases after reshuffling the primary recommendations list with the proposed re-rankCCP algorithm.

Experiments for re-rankCCP for both scenarios partially prove the second auxiliary thesis. They show that:

> *User modelling in the form of contextual conditional preferences allows to generate contextual recommendations in new user cold-start situations as well as in typical scenarios.*

7.4.3 Multi-domain Recommendation

`MovieTweetings` and `ConcertTweets` datasets are created based on publicly available tweets from Twitter. Thus, we decided to check whether these datasets contain the same users. We found 296 Twitter users that appear in both datasets and therefore can be used for testing multi-domain recommendations with COUP.

As it was mentioned in Chapter 5, the method of ontology-based contextual pre-filtering works in the same way independently of an application. However, the most important thing is how context types are connected with each other in COUP. In our case, we have two domains: music and movies, and two contextual parameters: time and location, which appear in at least one of the datasets. Figure 7.18 presents terminological part of COUP and connections between modules.

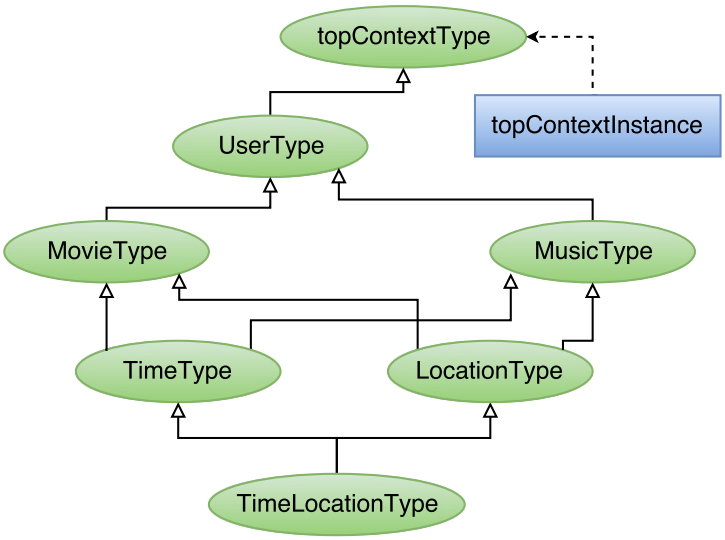


Figure 7.18: Connections between context types in COUP for multi-domain recommendations.

Table 7.20: Results obtained for multi-domain recommendations with ontology-based contextual pre-filtering.

Algorithm	Precision	Recall	MAP	nDCG	MRR	Expectedness	Unserendipity	Novelty	Diversity
BPR	0.0172	0.1724	0.0275	0.0374	0.0275	0.0015	0.2074	10.09	0.3696
FISM	0.0161	0.1609	0.0336	0.0418	0.0336	0.0019	0.2344	9.72	0.3566
LDA	0.0241	0.2414	0.0295	0.0437	0.0295	0.0033	0.2593	8.31	0.3015
WRMF	0.0230	0.2299	0.0455	0.0617	0.0455	0.0019	0.2413	9.59	0.3450

Results obtained for multi-domain recommendations are shown in Tab. 7.20. It is hard to compare them with other methods since, to the best of our knowledge, there is no other context-aware method for generating multi-domain recommendations. However, comparing the results with those obtained by the method for single domain on the same datasets, we can observe improvement of *diversity*. Ontology-based contextual pre-filtering method with BPR algorithm obtained *diversity* values for single domain equal to 0.3002 and 0.2773 on `MovieTweetings` and `ConcertTweets` datasets, respectively (see Tables 7.10 and 7.11), while for multi-domain *diversity* is equal to 0.3696 (see Tab. 7.20). *Diversity* value obtained for multi-domain recommendations with WRMF algorithm is equal to 0.3450 in comparison to values 0.3156 and 0.2764 obtained for single domain on `MovieTweetings` and `ConcertTweets` datasets, respectively.

This completes the proof of the first auxiliary thesis, which reads as follows:

> *User modelling in the form of contextual ontology and usage of an ontology of context enable us to generate multi-domain recommendations and allow dynamic generalization of contextual parameters values.*

7.4.4 On-line Survey

Besides offline experiments described in Sections 7.4.1, 7.4.2 and 7.4.3 we also conducted an experiment with real users to prove the usability of our methods. Similarly to the experiments presented in Section 7.4.2, we wanted to test ontology-based contextual pre-filtering in the typical scenario and re-rankCCP in two scenarios: typical and for new users. We chose User Item Splitting with BPR as the method for comparison because it performed pretty well in previous experiments, outperforming other algorithms in some cases. It is also the only one context-aware method that we were able to apply on every dataset that we used. Because our methods need to be combined with a non-context aware algorithm, we decided to use BPR algorithm for better comparability of results.

It would be hard to build and manage an RS that exploits four different algorithms. So, instead of developing a working system, we created an interactive questionnaire in the movie domain in four versions - one per each algorithm. We decided to use LDOS-CoMoDa dataset as the base of movies data and as the dataset to compute models (e.g. the set of GCCPs) that were further used to produce recommendations. Since the dataset contains small amount of information about movies themselves, we extended it with data from DBpedia and IMDb[14] to present them to users, e.g. a movie poster.

In three versions of questionnaire (those designed for the typical scenario) users were asked to provide their age and sex and to fill in their profile by rating about 10 movies in the 5-star scale, accompanying it with the corresponding contextual information. The step of filling in a user profile was skipped in the version for the new user scenario. However, they also provided information about age and sex. The next step in the questionnaire was evaluating the top 5 recommendations list received for given contextual situation by answering the following questions:

1. Which of the above movies have you watched before?
2. Which of the above movies have you never heard of?
3. Which of the above recommendations surprised you?
4. Which of the above recommendations are serendipitous for you?
5. Which of the above recommendations are useful for you?
6. In what order would you watch the above movies in a given situation? If you do not take advantage of any recommendation, leave blank. If only one movie is appropriate, fill in box 1, etc.
7. How much diverse the above recommendations are?

All questions were closed and a user had to choose one of possible options. Each user had to evaluate at least three lists of recommendations in the typical scenario. If she wished to, she could evaluate more. For the new user scenario, the minimal number of recommendation lists to evaluate was five, since users did not have to fill in the user profile. A user could possibly evaluate less recommendation lists than minimum if she closed the web browser before the end of the survey.

[14] http://www.imdb.com/

Table 7.21: The number of respondents and evaluated recommendation list for each version of the questionnaire.

Version	Algorithm	Scenario	Respondents	Replies
1	reshuffling with CCPs	new user	79	371
2	reshuffling with CCPs	typical	78	227
3	ontology-based CPF	typical	79	321
4	User-Item Splitting	typical	79	230
	Total		315	1149

In two versions for the re-rankCCP algorithm, we generated explanations for users on how recommendations were created. Exemplary explanations are given below.

People usually:
- in situation: *weekend, winter* prefer genres *Action, Comedy, Crime, Mystery* than *Sci-Fi, Thriller, War* and directors similar to *Clint Eastwood, Tony Scott, Roman Polanski* than directors similar to *Woody Allen* and movies with budget < 10 mln than those with budget > 100 mln.

You usually:
- in situation: *night, weekend, autumn* prefer genres *Comedy, Crime, Thriller* than *Action, Adventure, Drama* and actors similar to *Hugh Jackman, Morgan Freeman, Liam Neeson* than actors similar to *Michael Caine, Angelina Jolie.*

The second explanation can only occur in the typical scenario, while the first one appears in both scenarios. Our intention was to evaluate usefulness of explanations. Thus, we added two additional questions in two versions of the questionnaire for the re-rankCCP algorithm:

1. Is this explanation satisfactory to you?
2. Did this explanation change your willingness to watch a movie?

Assignment of a user to the version of questionnaire was done in a random way. We wrote simple web application which redirects a user to a version of questionnaire with the smallest number of users who submitted their replies so far. Users were unaware of the algorithm that they tested. Respondents were typically students of Informatics and Electronics as well as academic teachers from Gdańsk University of Technology. They were also people from other fields and from abroad. Table 7.21 shows number of users and evaluated recommendation lists for each version of the questionnaire. In Tab. 7.22 similar numbers are shown for these questionnaires that contain explanations.

Respondents were satisfied with explanations they got in 2/3 of the cases, independently of the scenario. It should be noticed that the scenario is strongly correlated with the type of CCPs used. In the new user scenario we can use only GCCPs while in the typical scenario we use both, ICCPs and GCCPs. Explanations change a user's willingness to watch at least one movie in two of three cases for the typical scenario. For the new user scenario it was every second case. Thus, we can deduce that explanations are useful and somehow change a

Table 7.22: The number of respondents and evaluated recommendation list for questionnaires that contained explanations.

Version	Algorithm	Scenario	Respondents	Replies
1	reshuffling with CCPs	new user	32	154
2	reshuffling with CCPs	typical	24	64
	Total		56	218

Table 7.23: Results obtained with interactive questionnaires.

	Version			
	1	2	3	4
Average no of seen items	1.67	1.75	1.83	3.40
Average no of unknown items	2.90	2.11	1.96	1.23
Average no of surprising items	1.40	1.48	1.33	1.41
Average no of serendipitous items	1.04	1.21	1.14	1.26
Average no of useful items	1.61	1.46	1.39	1.60
Percentage of diverse lists	56	75	71	51

user perception of received recommendations. This is also confirmed by the verbal feedback that we obtained from respondents. All of them said that it was really great to receive explanations, even if they not always agreed with the sentence that they had read.

This justifies the second auxiliary thesis, which reads as follows:

> *User modelling in the form of contextual conditional preferences allows to generate explanations and contextual recommendations in new user cold-start situations as well as in typical scenarios.*

Results obtained with the interactive questionnaire are presented in Tab. 7.23 (numbers of versions taken from Tab. 7.21). Our methods for the typical scenario generate the biggest number of diverse recommendation lists. However, they still recommend many items unknown to a user. The re-rankCCP in the new user scenario obtained worst value of diversity, but better results for usefulness and the average number of unknown items. We can see from Tab. 7.23 that User Item Splitting method recommends many items already seen by a user and a few items that were unknown to her. It also obtained the worst value of diversity. Indeed, responders verbal feedback was that the method generates very similar recommendation lists that also contain items that were already rated by them while creating their profile. There was also a positive feedback about ontology-based contextual pre-filtering. For instance, one of the surveyed users admitted that she received many interesting recommendations and that for sure she will watch most of the recommended movies.

Thus, we have shown that:

> *Context-aware user models and recommendation approaches proposed in this dissertation allow to create context-aware RSs that can be successfully applied in real-life scenarios giving satisfactory results considering their quantitative and qualitative properties.*

This ends justification of the main thesis of this dissertation.

Chapter 8

Summary

This chapter summarizes the dissertation, emphasizing the original work presented in it. It also provides possible directions of future works in the topic of context-aware recommender systems.

8.1 Outcomes

In this dissertation we presented novel context-aware user models and recommendation approaches that can be successfully applied in context-aware recommender systems. Besides context-awareness we addressed other aspects and problems that appear in recommender systems: the new user cold-start problem, the multi-domain recommendation and generating explanations for users.

After providing motivations for the topic of this dissertation and formulating theses and goals of this work in Chapter 1, in Chapter 2 we provide intuitions and formal definitions of a context and explain how we can decide whether a factor is a contextual parameter.

Chapter 3 presents different definitions of ontology and classification criteria as well as basic information about description logics. It describes the SIM method for building contextual ontologies and the RSCtx ontology which is the ontology of contexts that we used in our method.

In Chapter 4 we present a systematic review of recommendation algorithms and evaluation measures for recommender systems. We also explain context-awareness of recommendation methods and describe ways in which user preferences can be modelled.

Chapters 5, 6 and 7 contain original contribution to the research area of recommender systems.

Chapter 5 introduces the proposed Ontology-based Contextual Pre-filtering method which is very universal and can be used with non-context-aware algorithms to obtain context-aware recommendations. Because of special properties of the Contextual Ontological User Profile, our method allows generating context-aware and multi-domain recommendations, which makes it, to the best of our knowledge, the first context-aware multi-domain recommendation approach.

Chapter 6 introduces Contextual Conditional Preferences and shows how they can be used as a user model. It also presents details of two proposed recommendation algorithms

that utilize CCPs in the recommendation process, i.e. the Rating Prediction with CCPs and the re-rankCCP. The re-rankCCP supports generation of context-aware recommendations in new user cold-start scenarios as well as in typical situations. It also enables us to easily produce explanations for users.

Chapter 7 provides detailed description of several series of conducted offline experiments for different kinds of recommendation tasks in different scenarios. It also contains information about an on-line experiment with real users. We applied the methods proposed in this dissertation in an interactive questionnaire which returns context-aware recommendations based on provided by respondents movie ratings and corresponding contextual information. Results of each experiment show that our methods can be successfully applied in practical context-aware recommender systems.

The most important contributions of this dissertation are:

- Definition of Contextual Conditional Preferences,
- Proposing two recommendation algorithms that apply Contextual Conditional Preferences as a user model: the Rating Prediction with CCPs and the re-rankCCP algorithm,
- Definition of measure of satisfiability of CCP by an item,
- Proposing Ontology-based Contextual Pre-filtering approach that utilizes contextual ontology based on the SIM method and allows for dynamic generalization of contextual parameters values,
- Showing that the re-rankCCP algorithm is applicable also for new users,
- Finding the "perfect" value for unalikeability as a measure of relevance for contextual parameters.

All of this led us to justification of the main thesis of this dissertation:

> *Context-aware user models and recommendation approaches proposed in this dissertation allow to create context-aware recommender systems that can be successfully applied in real-life scenarios giving satisfactory results considering their quantitative and qualitative properties.*

8.2 Further work

This dissertation does not complete the work on context-aware user models and recommendation methods proposed here. We believe that contextual conditional preferences can also be used for multi-domain or even cross-domain recommendation. However, further investigation and many experiments are needed to prove this.

Indeed, people have knowledge about relationships between items from the same domain or even from different domains, e.g. the movie "Hunger Games" is based on a novel by Suzanne Collins. Usually, it affects users choices. If someone likes the movie, she probably may want to read the book. This kind of information is usually represented in a form of a knowledge graph or an ontology in information systems. Contextual ontological user profile has a big potential to be successfully enriched with those relationships between items. Thus,

it is possible to extend the ontology-based contextual pre-filtering method to utilize this additional information. This could lead to obtaining better precision of recommendations. It will also be useful for generating cross-domain recommendations.

Nowadays, the human emotion recognition is a popular topic of research. It can be combined with our work on context-aware recommendations by automatically recognizing a user's mood, which is one of important contextual factors for many domains. It is very important to make context-aware systems as little intrusive as possible because people do not like to be asked too many personal questions.

List of Figures

List of Tables

List of Algorithms

Bibliography

[1] P. Adamopoulos and A. Tuzhilin. Estimating the Value of Multi-Dimensional Data Sets in Context-based Recommender Systems. In *8th ACM Conference on Recommender Systems (RecSys 2014)*, 2014.

[2] P. Adamopoulos and A. Tuzhilin. On unexpectedness in recommender systems: Or how to better expect the unexpected. *ACM Trans. Intell. Syst. Technol.*, 5(4):54:1–54:32, Dec. 2014.

[3] G. Adomavicius, B. Mobasher, F. Ricci, and A. Tuzhilin. Context-aware recommender systems. *AI Magazine*, 32(3):67–80, 2011.

[4] G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. In *IEEE Trans. on Knowledge and Data Engineering*, volume 17(6), pages 734–749, 2005.

[5] G. Adomavicius and A. Tuzhilin. Handbook on recommender systems. chapter Context-Aware Recommender Systems, pages 217–256. Springer, 2011.

[6] C. C. Aggarwal. *Recommender Systems: The Textbook*. Springer Publishing Company, Incorporated, 1st edition, 2016.

[7] F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, and P. F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, New York, NY, USA, 2003.

[8] L. Baltrunas and X. Amatriain. Towards time-dependant recommendation based on implicit feedback. In *Proceedings of 1st Workshop on Context-Aware Recommender Systems*, 2009.

[9] L. Baltrunas and F. Ricci. Experimental evaluation of context-dependent collaborative filtering using item splitting. *User Model. User-Adapt. Interact.*, 24(1-2):7–34, 2014.

[10] M. Benerecetti, P. Bouquet, and C. Ghidini. Contextual reasoning distilled. *Philosophical Foundations of Artificial Intelligence. A special issue of the journal of Experimental and Theoretical AI (JETAI)*, 12(3):279–305, 2000.

[11] M. Benerecetti, P. Bouquet, and C. Ghidini. *On the Dimensions of Context Dependence: Partiality, Approximation, and Perspective*, pages 59–72. Springer Berlin Heidelberg, Berlin, Heidelberg, 2001.

[12] C. Bettini, O. Brdiczka, K. Henricksen, J. Indulska, D. Nicklas, A. Ranganathan, and D. Riboni. A survey of context modelling and reasoning techniques. *Pervasive and Mobile Computing*, 6(2):161 – 180, 2010. Context Modelling, Reasoning and Management.

[13] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022, Mar. 2003.

[14] C. Bolchini, C. A. Curino, E. Quintarelli, F. A. Schreiber, and L. Tanca. A data-oriented survey of context models. *SIGMOD Rec.*, 36(4):19–26, Dec. 2007.

[15] W. Borst. *Construction of Engineering Ontologies for Knowledge Sharing and Reuse*. PhD thesis, 9 1997.

[16] C. Boutilier, R. I. Brafman, C. Domshlak, H. H. Hoos, and D. Poole. Cp-nets: A tool for representing and reasoning with conditional ceteris paribus preference statements. *Journal of Artificial Intelligence Research*, 21:135–191, 2004.

[17] M. Braunhofer, M. Elahi, F. Ricci, and T. Schievenin. Context-aware points of interest suggestion with dynamic weather data management. In Z. Xiang and I. Tussyadiah, editors, *Information and Communication Technologies in Tourism 2014*, pages 87–100. Springer International Publishing, 2013.

[18] I. Cantador, A. Bellogín, and P. Castells. Ontology-based personalised and context-aware recommendations of news items. In *Proc. of the 2008 IEEE/WIC/ACM Int. Conf. on Web Intelligence and Intelligent Agent Technology - Volume 01*, WI-IAT '08, pages 562–565, Washington, DC, USA, 2008. IEEE Computer Society.

[19] I. Cantador and P. Cremonesi. Tutorial on cross-domain recommender systems. In *Proceedings of the 8th ACM Conference on Recommender Systems*, RecSys '14, pages 401–402, New York, NY, USA, 2014. ACM.

[20] P. Castells and S. Vargas. Novelty and diversity metrics for recommender systems: Choice, discovery and relevance. In *In Proceedings of International Workshop on Diversity in Document Retrieval (DDR)*, pages 29–37, 2011.

[21] J. Cendrowska. PRISM: an algorithm for inducing modular rules. *International Journal of Man-Machine Studies*, 27(4):349–370, 1987.

[22] H. Chen, T. Finin, and A. Joshi. *Ontologies for Agents: Theory and Experiences*, chapter The SOUPA Ontology for Pervasive Computing, pages 233–258. Birkhäuser Basel, Basel, 2005.

[23] L. Costabello. *Context-Aware Access Control and Presentation for Linked Data*, chapter A Declarative Model for Mobile Context, pages 21–32. 2013.

[24] P. Cremonesi, A. Tripodi, and R. Turrin. Cross-domain recommender systems. In *2011 IEEE 11th International Conference on Data Mining Workshops*, pages 496–503, Dec 2011.

[25] M. Deshpande and G. Karypis. Item-based top-n recommendation algorithms. *ACM Trans. Inf. Syst.*, 22(1):143–177, Jan. 2004.

[26] A. K. Dey. Understanding and using context. *Personal Ubiquitous Comput.*, 5(1):4–7, Jan. 2001.

[27] A. K. Dey and G. D. Abowd. Towards a better understanding of context and context-awarness. In *Proc. of Conference on Human Factors in Computing Systems*, pages 304–307, 2000.

[28] S. Dooms, T. De Pessemier, and L. Martens. Movietweetings: a movie rating dataset collected from twitter. In *Workshop on Crowdsourcing and Human Computation for Recommender Systems, CrowdRec at RecSys 2013*, 2013.

[29] M. Elahi, M. Braunhofer, F. Ricci, and M. Tkalcic. *Personality-Based Active Learning for Collaborative Filtering Recommender Systems*, pages 360–371. Springer International Publishing, Cham, 2013.

[30] A. Felfernig, G. Friedrich, D. Jannach, and M. Zanker. Developing constraint-based recommenders. In F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, editors, *Recommender Systems Handbook*, pages 187–215. Springer US, Boston, MA, 2011.

[31] A. Felfernig, M. Mairitsch, M. Mandl, M. Schubert, and E. Teppan. *Utility-Based Repair of Inconsistent Requirements*, pages 162–171. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[32] M. Fernández-López, A. Gómez-Pérez, and N. Juristo. Methontology: from ontological art towards ontological engineering. In *Proc. Symposium on Ontological Engineering of AAAI*, 1997.

[33] I. Fernández-Tobías, I. Cantador, M. Kaminskas, and F. Ricci. Cross-domain recommender systems: A survey of the state of the art. In *Proceedings of the 2nd Spanish Conference on Information Retrieval*, pages 187–198, 01 2012.

[34] F. Giunchiglia and P. Bouquet. Introduction to contextual reasoning. an artificial intelligence perspective. In B. Kokinov, editor, *Perspectives on Cognitive Science*, pages 138–159. NBU Press, 1997.

[35] K. Goczyła. *Ontologie w systemach informatycznych*. EXIT, 2011.

[36] K. Goczyła and **A**. **Karpus**. Problemy oceny jakości ontologii. *Studia Informatica*, 34(2A):173–184, 2013.

[37] K. Goczyła, A. Waloszek, and W. Waloszek. Contextualization of a dl knowledge base. *Proc. of the 20th International Workshop on Description Logics DL'07*, pages 291–298, 2007.

[38] K. Goczyła, A. Waloszek, W. Waloszek, and T. Zawadzka. Modularized knowledge bases using contexts, conglomerates and a query language. *Intelligent Tools for Building a Scientific Information Platform*, 390:179–201, 2012.

[39] A. Gómez-Pérez, M. Fernández-López, and O. Corcho. *Ontological Engineering with examples from the areas of Knowledge Management, e-Commerce and the Semantic Web*. Advanced Information and Knowledge Processing. Springer, 1st edition, 2004.

[40] T. R. Gruber. A translation approach to portable ontology specifications. *Knowl. Acquis.*, 5(2):199–220, June 1993.

[41] R. Guha. *Contexts: A Formalization and Some Applications*. PhD thesis, Stanford, CA, USA, 1992. UMI Order No. GAX92-17827.

[42] G. Guo, J. Zhang, Z. Sun, and N. Yorke-Smith. Librec: A java library for recommender systems. In A. I. Cristea, J. Masthoff, A. Said, and N. Tintarev, editors, *Posters, Demos, Late-breaking Results and Workshop Proceedings of the 23rd Conference on User Modeling, Adaptation, and Personalization (UMAP 2015), Dublin, Ireland, June 29 - July 3, 2015.*, volume 1388 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2015.

[43] S. O. Hansson. What is ceteris paribus preference? *J. Philosophical Logic*, 25(3):307–332, 1996.

[44] A. Hawalah and M. Fasli. Utilizing contextual ontological user profiles for personalized recommendations. *Expert Systems with Applications*, 41(10):4777 – 4797, 2014.

[45] P. Hitzler, M. Krötzsch, B. Parsia, P. F. Patel-Schneider, and S. Rudolph, editors. *OWL 2 Web Ontology Language: Primer*. W3C Recommendation, 27 October 2009. Available at `http://www.w3.org/TR/owl2-primer/`.

[46] J. R. Hobbs and F. Pan. Time ontology in owl. World Wide Web Consortium, Working Draft WD-owl-time-20060927, September 2006.

[47] Y. Hu, Y. Koren, and C. Volinsky. Collaborative filtering for implicit feedback datasets. In *Proceedings of the 2008 Eighth IEEE International Conference on Data Mining*, ICDM '08, pages 263–272, Washington, DC, USA, 2008. IEEE Computer Society.

[48] L. Iaquinta, M. de Gemmis, P. Lops, G. Semeraro, and P. Molino. *E-Commerce*, chapter Can a recommender system induce serendipitous encounters?, pages 1–17. IN-TECH, Vienna, 2009.

[49] H. Imran, M. Belghis-Zadeh, T.-W. Chang, Kinshuk, and S. Graf. *A Rule-Based Recommender System to Suggest Learning Tasks*, pages 672–673. Springer International Publishing, Cham, 2014.

[50] P. Jaccard. Lois de distribution florale dans la zone alpine. *Bulletin de la Société vaudoise des sciences naturelles*, 38:69–130, 01 1902.

[51] P. Jaccard. The distribution of flora in the alpine zone. *New Phytologist*, 11:37 – 50, 02 1912.

[52] D. Jannach, M. Zanker, A. Felfernig, and G. Friedrich. *Recommender Systems: An Introduction*. Cambridge University Press, New York, NY, USA, 1st edition, 2010.

[53] D. Jannach, M. Zanker, A. Felfernig, and G. Friedrich. *Recommender Systems: An Introduction*, chapter Explanations in recommender systems, pages 143–165. Cambridge University Press, New York, NY, USA, 1st edition, 2010.

[54] D. Jannach, M. Zanker, A. Felfernig, and G. Friedrich. *Recommender Systems: An Introduction*, chapter Collaborative recommendation, pages 13–50. Cambridge University Press, New York, NY, USA, 1st edition, 2010.

[55] D. Jannach, M. Zanker, A. Felfernig, and G. Friedrich. *Recommender Systems: An Introduction*, chapter Knowledge-based recommendation, pages 81–123. Cambridge University Press, New York, NY, USA, 1st edition, 2010.

[56] D. Jannach, M. Zanker, A. Felfernig, and G. Friedrich. *Recommender Systems: An Introduction*, chapter Hybrid recommendation approaches, pages 124–142. Cambridge University Press, New York, NY, USA, 1st edition, 2010.

[57] K. Järvelin and J. Kekäläinen. Cumulated gain-based evaluation of ir techniques. *ACM Trans. Inf. Syst.*, 20(4):422–446, Oct. 2002.

[58] S. Kabbur, X. Ning, and G. Karypis. Fism: Factored item similarity models for top-n recommender systems. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '13, pages 659–667, New York, NY, USA, 2013. ACM.

[59] G. D. Kader and M. Perry. Variability for categorical variables. *Journal of Statistics Education*, 15(2), 2007.

[60] D. Kaplan. On the logic of demonstratives. *Journal of Philosophical Logic*, 8(1):81–98, 1978.

[61] **A. Karpus**. Jakość w inżynierii ontologii. In *I Podkarpacka Konferencja Naukowa Doktorantów*, pages 7–16. Oficyna Wydawnicza Politechniki Rzeszowskiej, 2014.

[62] **A. Karpus**. A context in recommender systems. In *Zagadnienia Aktualnie Poruszane Przez Młodych Naukowców*, volume 6, pages 367–371. Creativetime, 2016.

[63] **A. Karpus**, T. di Noia, and K. Goczyla. Top k recommendations using contextual conditional preferences model. In M. Ganzha, L. A. Maciaszek, and M. Paprzycki, editors, *Proceedings of the 2017 Federated Conference on Computer Science and Information Systems, FedCSIS 2017, Prague, Czech Republic, September 3-6, 2017.*, pages 19–28, 2017.

[64] **A. Karpus**, T. di Noia, P. Tomeo, and K. Goczyla. Rating prediction with contextual conditional preferences. In A. L. N. Fred, J. L. G. Dietz, D. Aveiro, K. Liu, J. Bernardino, and J. Filipe, editors, *Proceedings of the 8th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (IC3K 2016) - Volume 1: KDIR, Porto - Portugal, November 9 - 11, 2016*, pages 419–424. SciTePress, 2016.

[65] **A. Karpus**, T. di Noia, P. Tomeo, and K. Goczyła. Using contextual conditional preferences for recommendation tasks: a case study in the movie domain. *Studia Informatica*, 37(1):7–18, 2016.

[66] **A. Karpus** and K. Goczyła. A context-aware recommender system based on an ontological user profile. In *ICT Young*, 2014.

[67] **A. Karpus** and K. Goczyła. A multi-domain hybrid recommender systems based on a dynamic contextual ontological user profile. In *Doctoral Consortium - DC3K, (IC3K 2014)*, pages 83–87. INSTICC, SciTePress, 2014.

[68] **A. Karpus**, I. Vagliano, K. Goczyła, and M. Morisio. An ontology-based contextual pre-filtering technique for recommender systems. In *2016 Federated Conference on Computer Science and Information Systems (FedCSIS)*, pages 411–420, Sept 2016.

[69] **A. Karpus**, I. Vagliano, and K. Goczyła. Serendipitous recommendations through ontology-based contextual pre-filtering. In S. Kozielski, D. Mrozek, P. Kasprowski, B. Małysiak-Mrozek, and D. Kostrzewa, editors, *Beyond Databases, Architectures and Structures. Towards Efficient Solutions for Data Analysis and Knowledge Representation: 13th International Conference, BDAS 2017, Ustroń, Poland, May 30 - June 2, 2017, Proceedings*, pages 246–259, Cham, 2017. Springer International Publishing.

[70] J. A. Konstan, B. N. Miller, D. Maltz, J. L. Herlocker, L. R. Gordon, and J. Riedl. Grouplens: Applying collaborative filtering to usenet news. *Commun. ACM*, 40(3):77–87, Mar. 1997.

[71] Y. Koren. Factorization meets the neighborhood: A multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '08, pages 426–434, New York, NY, USA, 2008. ACM.

[72] Y. Koren. Collaborative filtering with temporal dynamics. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '09, pages 447–456, New York, NY, USA, 2009. ACM.

[73] Y. Koren and R. Bell. *Advances in Collaborative Filtering*, pages 145–186. Springer US, Boston, MA, 2011.

[74] A. Kosir, A. Odic, M. Kunaver, M. Tkalcic, and J. F. Tasic. Database for contextual personalization. *Elektrotehniski vestnik [English print ed.]*, 78(5):270–274, 2011.

[75] R. Krummenacher and T. Strang. Ontology-based context modeling. In *In Workshop on Context-Aware Proactive Systems*, 2007.

[76] O. Lassila and D. McGuinness. The role of frame-based representation on the semantic web. Technical report, Knowledge Systems Laboratory Report KSL-01-02, Stanford University, Stanford (USA), 2001.

[77] D. Lenat. The dimensions of context space. Technical report, Cycorp, 1998.

[78] V. Maidel, P. Shoval, B. Shapira, and M. Taieb-Maimon. Evaluation of an ontology-content based filtering method for a personalized newspaper. In *Proc. of the 2008 ACM Conf. on Recommender Systems*, RecSys '08, pages 91–98, New York, NY, USA, 2008. ACM.

[79] A. Maksai, F. Garcin, and B. Faltings. Predicting online performance of news recommender systems through richer evaluation metrics. In *Proceedings of the 9th ACM Conference on Recommender Systems*, RecSys '15, pages 179–186, New York, NY, USA, 2015. ACM.

[80] J. McCarthy. Notes on formalizing context. In *Proceedings of the 13th International Joint Conference on Artifical Intelligence - Volume 1*, IJCAI'93, pages 555–560, San Francisco, CA, USA, 1993. Morgan Kaufmann Publishers Inc.

[81] S. E. Middleton, N. R. Shadbolt, and D. C. De Roure. Ontological user profiling in recommender systems. *ACM Trans. Inf. Syst.*, 22(1):54–88, Jan. 2004.

[82] X. Ning and G. Karypis. Slim: Sparse linear methods for top-n recommender systems. In *2011 IEEE 11th International Conference on Data Mining*, pages 497–506, Dec 2011.

[83] A. Odic, M. Tkalcic, J. F. Tasic, and A. Kosir. Predicting and detecting the relevant contextual information in a movie-recommender system. *Interacting with Computers*, 25(1):74–90, 2013.

[84] W. OWL Working Group. *OWL 2 Web Ontology Language: Document Overview.* W3C Recommendation, 27 October 2009. Available at `http://www.w3.org/TR/owl2-overview/`.

[85] K. Pal and S. Michel. A data mining approach to choosing categorical attributes for ranked lists. In E. Pitoura, S. Maabout, G. Koutrika, A. Marian, L. Tanca, I. Manolescu, and K. Stefanidis, editors, *Proceedings of the 19th International Conference on Extending Database Technology, EDBT 2016, Bordeaux, France, March 15-16, 2016, Bordeaux, France, March 15-16, 2016.*, pages 664–665. OpenProceedings.org, 2016.

[86] K. Pearson. Note on regression and inheritance in the case of two parents. *Proceedings of the Royal Society of London*, 58:240–242, 1895.

[87] M. Perry and G. D. Kader. Variation as unalikeability. *Teaching Statistics*, 27(2):58–60, 2005.

[88] D. Preuveneers, J. Bergh, D. Wagelaar, A. Georges, P. Rigole, T. Clerckx, Y. Berbers, K. Coninx, V. Jonckers, and K. Bosschere. *Ambient Intelligence: Second European Symposium, EUSAI 2004, Eindhoven, The Netherlands, November 8-11, 2004. Proc.*, chapter Towards an Extensible Context Ontology for Ambient Intelligence, pages 148–159. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004.

[89] C. Rack, S. Arbanowski, and S. Steglich. Context-aware, Ontology-based Recommendations. In *SAINT-W '06: Proc. of the Int. Symposium on Applications on Internet Workshops*, pages 98–104, Washington, DC, USA, 2006. IEEE Computer Society.

[90] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, UAI '09, pages 452–461, Arlington, Virginia, United States, 2009. AUAI Press.

[91] F. Ricci, L. Rokach, and B. Shapira. Introduction to recommender systems handbook. In F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, editors, *Recommender Systems Handbook*, pages 1–35. Springer US, 2011.

[92] J. Rodríguez, M. Bravo, and R. Guzmán. Multidimensional ontology model to support context-aware systems. In *AAAI Workshops*, 2013.

[93] B. Schilit, N. Adams, and R. Want. Context-aware computing applications. In *Proceedings of the 1994 First Workshop on Mobile Computing Systems and Applications*, WMCSA '94, pages 85–90, Washington, DC, USA, 1994. IEEE Computer Society.

[94] L. Serafini and P. Bouquet. Comparing formal theories of context in ai. *Artif. Intell.*, 155(1-2):41–67, May 2004.

[95] G. Shani and A. Gunawardana. Handbook on recommender systems. chapter Evaluating Recommendation Systems, pages 257–298. Springer, 2011.

[96] A. Singhal. Modern information retrieval: a brief overview. *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering*, 24(4):35–42, 2001.

[97] M. K. Smith, C. Welty, and D. L. McGuinness. Owl web ontology language guide. World Wide Web Consortium, Recommendation REC-owl-guide-20040210, February 2004.

[98] B. Smyth. Case-based recommendation. In P. Brusilovsky, A. Kobsa, and W. Nejdl, editors, *The Adaptive Web: Methods and Strategies of Web Personalization*, pages 342–376, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[99] B. Smyth and P. McClave. Similarity vs. diversity. In D. W. Aha and I. Watson, editors, *Case-Based Reasoning Research and Development: 4th International Conference on Case-Based Reasoning, ICCBR 2001 Vancouver, BC, Canada, July 30 – August 2, 2001 Proceedings*, pages 347–361, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.

[100] H. E. Soper, A. W. Young, B. M. Cave, A. Lee, and K. Pearson. On the distribution of the correlation coefficient in small samples. appendix ii to the papers of "student" and r. a. fisher. a cooperative study. *Biometrika*, 11(4):328–413, 1917.

[101] J. B. Spira. *Overload!: How Too Much Information is Hazardous to your Organization.* John Wiley & Sons Inc., 2012.

[102] R. Studer, V. R. Benjamins, and D. Fensel. Knowledge engineering: Principles and methods. *Data Knowl. Eng.*, 25(1-2):161–197, Mar. 1998.

[103] P. Tomeo, T. D. Noia, M. de Gemmis, P. Lops, G. Semeraro, and E. D. Sciascio. Exploiting regression trees as user models for intent-aware multi-attribute diversity. In T. Bogers and M. Koolen, editors, *Proceedings of the 2nd Workshop on New Trends on Content-Based Recommender Systems co-located with 9th ACM Conference on Recommender Systems (RecSys 2015), Vienna, Austria, September 16-20, 2015.*, volume 1448 of *CEUR Workshop Proceedings*, pages 2–9. CEUR-WS.org, 2015.

[104] A. Turpin and F. Scholer. User performance versus precision measures for simple search tasks. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '06, pages 11–18, New York, NY, USA, 2006. ACM.

[105] B. Vargas-Govea, G. Gonzalez-Serna, and R. Ponce-Medellin. Effects of relevant contextual features in the performance of a restaurant recommender system. In *Proceedings of 3rd Workshop on Context-Aware Recommender Systems*, 2011.

[106] A. Waloszek. *Hierarchiczna kontekstualizacja baz wiedzy.* PhD thesis, Gdańsk University of Technology, 2010.

[107] X. H. Wang, D. Q. Zhang, T. Gu, and H. K. Pung. Ontology based context modeling and reasoning using owl. In *Proc. of the Second IEEE Annual Conf. on Pervasive Computing and Communications Workshops*, PERCOMW '04, pages 18–, Washington, DC, USA, 2004. IEEE Computer Society.

[108] M. Wooldridge. *An Introduction to MultiAgent Systems*. Wiley Publishing, 2nd edition, 2009.

[109] Y. C. Zhang, D. O. Séaghdha, D. Quercia, and T. Jambor. Auralist: Introducing serendipity into music recommendation. In *Proceedings of the Fifth ACM International Conference on Web Search and Data Mining*, WSDM '12, pages 13–22, New York, NY, USA, 2012. ACM.

[110] Y. Zheng, R. Burke, and B. Mobasher. Splitting approaches for context-aware recommendation: An empirical study. In *Proceedings of the 29th Annual ACM Symposium on Applied Computing*, SAC '14, pages 274–279, New York, NY, USA, 2014. ACM.

[111] Y. Zheng, B. Mobasher, and R. Burke. Cslim: Contextual slim recommendation algorithms. In *Proceedings of the 8th ACM Conference on Recommender Systems*, RecSys '14, pages 301–304, New York, NY, USA, 2014. ACM.

[112] Y. Zheng, B. Mobasher, and R. D. Burke. Carskit: A java-based context-aware recommendation engine. In *ICDM Workshops*, pages 1668–1671. IEEE Computer Society, 2015.

[113] C.-N. Ziegler, S. M. McNee, J. A. Konstan, and G. Lausen. Improving recommendation lists through topic diversification. In *Proceedings of the 14th International Conference on World Wide Web*, WWW '05, pages 22–32, New York, NY, USA, 2005. ACM.