Ontologies in Recommender Systems

by

Ing. Stanislav Kuznetsov

A dissertation thesis submitted to
the Faculty of Information Technology, Czech Technical University in Prague,
in partial fulfilment of the requirements for the degree of Doctor.

Dissertation degree study programme: Informatics

Department of Applied Mathematics

Prague, September 2020

Supervisor:
doc. Ing. Kordík Pavel, Ph.D.
Department of Applied Mathematics
Faculty of Information Technology
Czech Technical University in Prague
Thákurova 9
160 00 Prague 6
Czech Republic

Abstract

The cold-start problem is typical in recommender systems (RS). There are several ways to reduce this problem. There are naive approaches, or there are algorithms that try to reduce the problem by using attributes or generating additional information. We find a way to automatically or semi-automatically generate more in-depth knowledge of the domain and use that knowledge to minimise the cold-start problem. This knowledge is, in our case, represented by a semantic level that can add essential information about items and also be understandable to users. Our approach is to use ontology in the form of a graph for semantic purposes. We use the benefits of graph architecture to automatically generate an ontological profile for the items, and if needed, we can extend profiles with the most similar semantic nodes, and use these profiles for recommendation purposes.
In this thesis, we describe the principle of automatically generating ontologies and improving them with explicit word embeddings. We present the general methodology for evaluating algorithms for cold-user and cold-item problems. Next, we show our ontology-based algorithms for reducing the cold-start problem (OBACS). The algorithm is a combination of ontology, and collaborative filtering (CF) approaches. It is a universal algorithm that could reduce the cold-start problem in an earlier phase where more of the items are cold, and also in the transition phase, where the number of cold-start items decreases. Also, these algorithms could be used for diversification and to help CF approaches with the "filter bubble" problem. We prove our approach with an offline experiment in the area of movie recommendation and discuss the results.

In particular, the main contributions of the dissertation thesis are as follows:

1. We describe the cold-start problem in-depth, its main characteristics and illustrate it in experiments.

2. We present the most common approaches – naive, CB and CF – to reducing the cold-start problem and describe the advantages and disadvantages of each method.

3. We explain how to use leave-one-out cross-validation methods and how to prepare a dataset to test cold-user and cold-item problems.

4. We describe the process of semi-automatic developing ontology and describe how to improve ontology with implicit and explicit embeddings.

5. We present our universal ontology-based algorithms for reducing the cold-start problem (OBACS).

6. We show how OBACS algorithms could help CF algorithms control the "filter bubble".

7. We summarize the ideas and results of this thesis and present our future work

Keywords:
recommender systems, ontology, cold-start problem, diversity, serendipity

Acknowledgements

First of all, I would like to express my gratitude to my friend and dissertation thesis supervisor, doc. Ing. Kordík Pavel, Ph.D. He has been a constant source of encouragement and insight during my research and helped me with numerous problems and professional advancements. I would like to thank Dr. Ing. Petr Kroha, CSc. for helping me and motivating me to work on this thesis.

Special thanks go to the staff of the Department of Applied Mathematics, who maintained a pleasant and flexible environment for my research. I would like to express special thanks to the department management for providing most of the funding for my research.

This research has been partially supported by grants SGS13/098/OHK3/1T/18, SGS14/102/OHK3/1T/18, SGS15/117/OHK3/1T/18, SGS16/119/OHK3/1T/18 and SGS17/210/OHK3/3T/18 of the Czech Technical University in Prague. The Computational resources were partially supplied by the project "e-Infrastruktura CZ" (e-INFRA LM2018140) provided within the Projects of Large Research, Development and Innovations Infrastructures programme.

I would like to express thanks to my colleagues from CIG group, for their valuable comments and proofreading.

Finally, my greatest thanks go to my family and my wife Jane,
for their infinite patience and care.
Thank you, guys, we did it together!

Dedication

To my father

Contents

Abbreviations **xv**

1 Introduction **1**
1.1 Motivation . . . 1
1.2 Problem Statement . . . 2
1.3 Goals of the Dissertation Thesis . . . 3
1.4 Structure of the Dissertation Thesis . . . 3

2 Background and State-of-the-Art **5**
2.1 Theoretical Background . . . 5
2.1.1 Recommender Approaches . . . 5
2.1.2 Cold-Start Problem . . . 6
2.1.3 Evaluation of Recommender Systems . . . 7
2.1.4 Pareto Optimality . . . 10
2.1.5 Algorithms for text representation . . . 11
2.1.6 Matrix Factorisation . . . 13
2.1.7 Diversity and Serendipity . . . 14
2.2 Previous Results and Related Work . . . 15
2.2.1 Ontology-Based Recommender Systems . . . 15
2.2.2 State-of-the-Art . . . 16

3 Overview of Our Approach **19**
3.1 Ontology Development . . . 19
3.1.1 Text Mining Algorithms for Keyword Extraction . . . 20
3.1.2 Implicit Word Embedding Features . . . 21
3.1.3 Explicit Word Embedding Features . . . 22
3.1.4 Onto Architecture . . . 23
3.1.5 Onto Profiles . . . 24
3.2 Ontology Recommendation . . . 26

3.2.1 Ontological similarity . . . 26
3.2.2 Universal Ontological-Based Algorithm (OBACS) . . . 28
3.2.3 Enhance Collaborative Filtering with OBACS . . . 29
3.2.4 Diversity . . . 29
3.2.5 RUN Boosting (serendipity) . . . 29

4 Main Results **31**
4.1 Experimental Setup . . . 31
4.1.1 Experimental Dataset . . . 31
4.1.2 Cold-Start Dataset . . . 32
4.1.3 Algorithms . . . 32
4.2 Cold-User Recommendation Experiment . . . 37
4.2.1 Results . . . 39
4.2.2 Discussion . . . 43
4.3 Cold-Item Recommendation Experiment . . . 45
4.3.1 Results . . . 45
4.3.2 Discussion . . . 47
4.4 Enhanced Collaborative Filtering Experiment . . . 50
4.4.1 Results . . . 51
4.4.2 Discussion . . . 52
4.5 Summary . . . 52

5 Conclusions **55**
5.1 Summary . . . 55
5.2 Contributions of the Dissertation Thesis . . . 56
5.3 Future Work . . . 56

Bibliography **59**

Bibliography **59**

Reviewed Publications of the Author Relevant to the Thesis **65**

A Education Data Mining Case Study **67**
A.1 Education Data Mining Case Study . . . 67
A.2 Data Preprocessing . . . 67
A.2.1 OntoSkill graph . . . 68
A.2.2 Computation of student profiles . . . 73
A.2.3 Application Domain - SSP Portal . . . 73
A.2.4 Recommendation Models . . . 73
A.2.5 k-Nearest Neighbours Algorithm . . . 73
A.2.6 Offline Experiments . . . 75
A.2.7 Discussion . . . 80
A.2.8 Conclusion . . . 81

B Future work **83**
B.1 Rising Star 83
B.2 Keyword Context Detection 84
B.3 Final Score Cold Start 85

List of Figures

2.1 Picture of the main LASER approach. We take this picture from the website [41] . . . 13
2.2 Picture of matrix factorisation. . . . 14
2.3 Euler diagram of items from a user's point of view at a given moment in time [25] . . . 15

3.1 Bootstrapped learning algorithm process. . . . 21
3.2 Our adaptation of bootstrapped learning algorithm. . . . 22
3.3 Ontology generates for movie domain from a higher perspective. . . . 25
3.4 Detail view of one of the parts of the ontology that is generated by movies with Rock N Roll themes. . . . 26

4.1 AutoRec architecture. . . . 36
4.2 The basic ontological profile of the Iron Man (2008) movie. . . . 37
4.3 The ontological profile of the Iron Man (2008) movie, with extended nodes. For each node from a basic profile, we add most semantically similar nodes. . . . 38
4.4 3D projection of all algorithms results, where the X-axis contains recall, the Y-axis contains catalogue coverage, the Z-axis contains onto-similarity 40
4.5 The result of the experiment, where the X-axis contains recall, the Y-axis contains catalogue coverage. . . . 41
4.6 The result of the experiment, where the X-axis contains recall, the Y-axis contains onto-similarity. . . . 42
4.7 The comparison of similarities, inside CB algorithms in each dataset. . . . 42
4.8 3D projection of the algorithm results, where the X-axis contains recall, the Y-axis contains catalogue coverage, and the Z-axis contains onto-similarity. . . . 46
4.9 The result, where the X-axis contains recall and the Y-axis contains catalogue coverage. . . . 47
4.10 The result, where the X-axis contains recall, the Y-axis contains onto similarity 49
4.11 The results of all diversifications inside OBACS algorithms, where the X-axis show recall and Y-axis catalogue coverage. . . . 50

4.12 The results of all diversifications inside OBACS algorithms for the "20-50" dataset, where the X-axis show recall and Y-axis catalogue coverage 51

A.1 Skills are represented by the tree, so higher level nodes generalise skill of leaf nodes. 69
A.2 The principle of ontology based on the graph. 70
A.3 Ontology based on a graph that we generated based on lecturer knowledge at the Faculty of Information Technology, CTU. 71
A.4 Ontology based on a graph that we generated based on lecturer knowledge in more detail. 72
A.5 The frequency of the regular dataset. Each value on the x-axis represents the class with number of interactions. The y-axis shows the number of users in each class. 77
A.6 Results for the *Onto-best* model and *best seller* model on the cold-start dataset. The *best seller* model has no parameters; it is represented by a single dot. On the other hand, the Onto-best model has a γ parameter; on the chart, it is represented by a curve that shows the behaviour of the model, depending on the γ setting. 78
A.7 Results for the *User-kNN* model on the regular dataset. Each curve on the chart represents the setting of the β value. The values on the curves are the settings of the k parameter. 79
A.8 Results for the *association rules* model on the regular dataset. Each curve on the chart represents the setting of the β value. The values on the curves are the settings of the s_{min} parameter. 80
A.9 Results for the *Onto-best model* on the regular dataset. The curve on the graph represents the behaviour of the model depending on the γ parameter. 81
A.10 Comparisons of all models on the regular dataset. The upper green curve is the behaviour of the *User-kNN* model with fixed $\beta = 0.4$ and various k. The curve in the middle represents an AR model with fixed $\beta = 0.6$ and various settings of s_{min}. The lower curve represents the results of the Onto-best model depending on γ. At the bottom of the chart is a dot that represents the best seller model. 82

B.1 Rising star concept. 84

List of Tables

2.1 Test results. 8

3.1 Basic implicit metrics in our Ontology 23
3.2 Properties that we use at ConceptNet as explicit metrics 23

4.1 Datasets, files and metadata used 32
4.2 Cold-user datasets and characteristics 32
4.3 Results of the Cold-User Recommendation Experiment. 44
4.4 Results of the OBACS algorithm in Cold-Item Recommendation Experiment . 48

A.1 Summary of the results for *Onto-best* and *best seller* models on cold-start dataset. 78
A.2 Summary of the results for all models on the regular dataset. 79

List of Algorithms

4.1 Similarity-Based CB Algorithm 34
4.2 Matrix Factorisation 35

Abbreviations

Miscellaneous Abbreviations

CB	Content-Based
CF	Collaborative Filtering
DNN	Deep Neural Networ
LASER	Language-Agnostic SEntence Representations
LSTM	Long short-term memory
MAE	Mean Absolute Error
MF	Matrix Factorisation
NMAE	Normalize MeanAverage Error
OBACS	Ontology-Based Algorithms for reducing the Cold-Start problem
RMSE	Root Mean Squared Error
RUN Boosting	Relevant Unexpectness Novelty Boosting
TF-IDF	Frequency–Inverse Document Frequency

Chapter 1

Introduction

In this chapter, we briefly introduce our motivation, the main problem statement, define our goals and present the structure of the thesis

1.1 Motivation

Recommender systems (RS) are now now essential services used in all prominent digital platforms worldwide, such as YouTube [1], Netflix [2] and Amazon [3]. Along with search services, RS are critical services for helping users discover new content or products. RS can help customers find relevant items even when the customer does not know the exact name of the item, or recommend an item that is even more acceptable to the customer than what they originally searched for. This means added value for a business, both in terms of increased revenue and customer satisfaction. Even the smallest customer, like my one and half-year-old daughter, can consume and find appropriate content in YouTube by clicking on a recommended video.

That is why RS are a popular research field in Data Mining and Machine Learning. In recent years, many new approaches and algorithms have been presented. One of the important algorithms methods is collaborative filtering (CF). These algorithms work by analysing user behaviour patterns in the audience and providing a recommendation based on similar users or similar items. The biggest advantage of this method is that it can adapt to trends, or react to changes in the system. This algorithm works well in systems with a significant amount of interaction, so the above-mentioned digital platforms often use these principles .

But what about new, emerging platforms containing lots of content but few users? How

[1] https://youtube.com
[2] https://netflix.com
[3] https://amazon.com

can we integrate RS in digital platforms that have integrated a large amount of data but have only recently started?

One real business example is Experts.ai [4]. This is a platform that, in the middle of the year 2020, integrated several million academic outcomes from countries throughout central Europe. The main goal of this platform is to improve collaboration between the academic area and business area. Experts.ai does this by providing search and recommendation services for business so that they can find an appropriate expert for research or commercial projects. So, on the one hand, we have millions of items (experts), but on the other hand, the number of real users is pretty small and can be measured in the hundreds. With such extensive data and without interaction, the matrices for CF are incredibly sparse and calculating a relevant recommendation is impossible.

We call this the cold-start problem. The main principle is that RS in cold-start systems should find another type of data or additional information, such as metadata, geo data, etc. Based on this additional data, they can then provide recommendations to users. The central family used for this purpose is content-based (CB) algorithms. There are a lot of algorithms and methods, based on text data, where the main idea is to extract relevant features and then use these to compute similarities between items.

In this thesis, we introduce our approach, in which we use ontological-based algorithms to build a common framework that can help systems with cold-start problems to integrate working RS. The main difference between our approach and classical CB approaches is that that we build a semantic layer on top of the attribute data. We also use this layer to improve CF algorithms in the later phases of system growth. We use implicit and explicit embedding and sophisticated keyword extraction methods to generate our ontology. Our approach is semi-automated and can be adapted to a lot of domains. In this thesis, we provide experiments in the movie domain, while we demonstrate the use of our approach in the education domain in the appendix.

Our primary motivation is to describe methods that could help young startups and new ideas use the benefits of RS in the early stages and help them to grow.

1.2 Problem Statement

There are several problems that a researcher needs to solve.

One of these problems is evaluating RS algorithms. There is no single methodology for

[4] https://experts.ai

evaluating the algorithms, and there are differences of opinion as to what should be the appropriate comparison metrics.

Another problem is data. The main purpose of RS is to work in an online environment, but it is typically hard to get this type of data. Because the data is usually secured by businesses, it is difficult to access, and there is little chance of being able to publish the datasets for research comparisons. This means that we can evaluate the algorithms only on offline data, which could be biased.

For typical recommendation tasks, like a rating prediction or topN recommendation, there are a lot of approaches, such as CF or deep neural network (DNN), etc. All these approaches use a user x item interaction matrices for computing the prediction or recommendation.

The cold-start problem is a typical problem in RS. Systems can especially suffer from this at the beginning, but it can persist throughout the rest of their life. The cold-start problem means that there are no user x item interactions, or there is not enough interaction. This means that, for all the main problems described above, we also cannot use the most typical algorithms.

All of these problems create a challenge that we will try to solve in this thesis.

1.3 Goals of the Dissertation Thesis

1. Present the ontological approach for reducing the cold-start problem for different domains.
2. Explain the methodology for the offline evaluation of the cold-start problem and describe the evaluation metrics.
3. Describe the process of creating the ontology and its utilization for recommendation purposes and design an algorithm for recommendation with the use of ontologies.
4. Show how the ontology can improve the characteristics of a typical approach that we use for *topN* recommendation in cold-start environments and a regular environment.
5. Summarize the main contribution of the thesis and present future work.

1.4 Structure of the Dissertation Thesis

The thesis is organised into five chapters as follows:

1. Chapter 1 - *Introduction*: Describes the motivation behind our efforts together with our goals.

2. Chapter 2 - *Background and State-of-the-Art*: Introduces the reader to the necessary theoretical background and surveys the current state-of-the-art. This includes an overview of the methods, metrics, algorithms that we use in our approach, then related works and preliminary results.

3. Chapter 3 - *Overview of Our Approach*: We present a general approach based on ontologies that could reduce the cold-start problem. We describe the main difference between this approach and the typical approaches. Also, we discuss the advantages and disadvantages and show how to use this algorithm to improve other standard algorithms.

4. Chapter 4 - *Main Results*: We describe our experimental setup. We demonstrate our approach in experiments where we compare our results with state-of-the-art algorithms and algorithms based on the most typical methods. We present our ontology-based algorithms and discuss the results.

5. Chapter 5 - *Conclusions*: Summarizes the results of our research, suggests possible topics for further research, and concludes the thesis. Also, there is a list of the contribution of the thesis.

Chapter 2

Background and State-of-the-Art

This chapter contains two parts. The first part describes all concepts necessary for understanding the thesis. The issue of current recommender systems (RS) is introduced, the basic types of RS are described, and the basic comparative techniques and methodologies for comparing RS are introduced. Furthermore, the reader will be made familiar with the issues of ontologies and topics which can be solved by ontologies. The second part of this chapter describes systems that are close either technically or conceptually to this work.

2.1 Theoretical Background

In this chapter, we briefly explain the methods, metrics, approaches, etc., that are important for understanding the thesis.

2.1.1 Recommender Approaches

Recommender systems [1] help users find information, products or services that a user might be interested in. Recommender systems are abbreviated as RS throughout the entire text. Development of these systems has been underway for several decades, but there is still no universal system that would be suitable for all areas or problems. Therefore, research in this area has not yet stopped; on the contrary, the area of RS is more topical than ever, with the amount of information still growing. Due to the constant increase in the volume of information and number of users, RS are becoming increasingly complex, combining methods/algorithms from many areas of computer engineering. Adamavicius and Tuzhilini [1] divide RS into two main groups: rating-based recommender systems and preference-based filtering techniques.

Rating-based systems are focused on predicting the absolute rating value (ranking) of an item that has not yet been displayed to a user. For example, if a user has evaluated the films Saving Private Ryan as 9/10 and Beautiful Mind as 8/10, then he might evaluate The Imitation Game as 7/10.

Preference-based filtering techniques predict the top-k recommendation or the relative order of items for a given user according to the opinion of a user community. For example, assume that the order of items when selecting a phone is as follows for a given user: Samsung Galaxy XY $\succ$ Sony Experia XY $\succ$ HTC XY. Using the characteristics of phones and opinions of other users, we can assume that the user order is changed after the introduction of a new iPhone XY, to iPhone XY $\succ$ Samsung Galaxy XY $\succ$ Sony Experia XY $\succ$ HTC XY. This work focuses on rating-based recommender systems, thus the term "recommender systems (RS)" will mean precisely this kind of RS throughout the rest of the thesis. RS are, according to [5], divided into four categories: content-based methods, collaborative filtering, demographic filtering and hybrid approaches.

Content-based RS [29] make use of the assessment given to items by users in the past for a definition of a current item profile. This profile can be enhanced with information extracted from a description of the item.

Collaborative-based RS are unlike the previous systems in that they use the known rating of a group of users to predict the unknown rating of a specific user. The idea is based on the assumption that, if a group of users made the same assessment in the past, we could assume that the group's assessment will be similar in the future [21]. Wang and Blei [46] divide collaborative filtering methods into two main groups: neighbourhood methods and matrix factorization. Neighbourhood methods define a group of those users most similar to the current user and determine his interest by using the aggregation of the interests of the whole group. Matrix factorisation is a method of latent factor models in which users and items are represented in low dimension space. The new representation of users and items are commonly computed by minimising the regularized squared error.

Demographic filtering systems make recommendations by using a user's personal attributes like gender, income, age, country and so on [5].

Hybrid RS combine two or more types of the systems above. They mostly provide better results, but are more difficult to implement [1].

2.1.2 Cold-Start Problem

Generally, we can say that the larger the set of data about users and their interactions we have, the "better" the RS works [32]. The problem comes with having small amounts of these data or none at all. Then, we have to decide which attributes we should use to carry out the recommendations. If we cannot find a suitable solution, a paradox can occur from the fact that, if we cannot carry out quality recommendations, users will stop using a given system, which means that we cannot improve the quality of recommendations, with a corresponding decline in user activity. This nontrivial problem is called the cold-start problem [43]. We will look at two forms of the problem in this section.

The new-system cold-start problem is that there are no initial ratings by users, and hence no profiles of users. In this situation, most recommender systems have no basis on which to recommend, and hence perform very poorly.

The new-user cold-start problem is that the system has been running for some time and a set of user profiles and ratings exist, but no information is available about a new user. Most recommender systems also perform poorly in this situation [32].

Collaborative-based recommender systems cannot cope with this problem because they are not able to find a group of users with the same behaviour, as there is no preceding user interaction. Content-based and hybrid-based recommender systems perform a little better because they need a smaller number of user interactions to find similarities.

None of these recommender systems can manage an absolute cold-start problem on their own, as the content-based recommender system needs, at least, an initial set of interactions. One solution would be to use a RS based on ontology. Ontologies can provide us with another kind of information, based on which the system may begin recommending without having to wait for interaction from users.

2.1.3 Evaluation of Recommender Systems

As described above, there are many RS, differing in principle as well as algorithm or method used. This diversity made it necessary to find a way in which the different RS can be compared. Shani 2011 [43] describes three basic methods for testing/comparing RS. These are offline testing, user studies, and online experiments. This section describes only the basic metrics of offline testing; the other two methods are beyond the scope of this work. Offline testing is carried out on pre-built user data, with which we try to simulate real user behaviour when interacting with the RS. We assume that the user behaviour during data collection will be the same as when deploying the system. This kind of testing is attractive, since it does not require any interaction with users, enabling us to carry out far-reaching experiments with a large number of algorithms for little cost. The disadvantage of this approach is that we are able to answer only a very narrow set of questions based on it, such as the predictive power of an algorithm, but we cannot determine the real impact of RS on user behaviour. Therefore, this method is primarily used to filter each of the wrong approaches right from the start and to have only a small set of functional solutions left, enabling us to carry out a much more extensive user study or online experiments.

Performance Measures in Recommender Systems To compare individual RS with each other, we shall define the basic metrics. The literature often mentions metrics such as Mean Absolute Error (MAE), Root Mean Squared Error (RMSE) and Normalize Mean Average Error (NMAE) [43]. These metrics are used when comparing RS based on rating predictions (stars) given to an item by a user.

Mean Absolute Error MAE is a typical rating prediction measure. It averages the absolute difference between item prediction rating and the actual rating that the user

	Recommended	Not Recommended
Used	True-Positive (tp)	False-Negative (fn)
Not Used	False-Positive (fp)	True-Negative (tn)

Table 2.1: Test results.

gives to an item. Equation 2.1 shows these metrics.

$$MAE = \frac{1}{|N|} \sum_{(u,i) \in N} |\hat{r}_{ui} - r_{ui}| \tag{2.1}$$

Where N is a set of ratings from test set, $\hat{r}_{ui}$ is a predicted rating of item i by user u and $r_u i$ is an actual rating

Root Mean Squared Error RMSE is an alternative to MEA. Compared to MAE, RMSE penalises higher errors more than MAE, which add the same weight on each error. Equation 2.2 shows these metrics.

$$RMSE = \sqrt{\frac{1}{|N|} \sum_{(u,i) \in N} (\hat{r}_{ui} - r_{ui})^2} \tag{2.2}$$

Where N is a set of ratings from test set, $\hat{r}_{ui}$ is a predicted rating of item i by user u and $r_u i$ is an actual rating

Our system is primarily aimed at predicting whether an item (input) is relevant or irrelevant to a given user. The methods used are metrics known from Information Retrieval. These are Recall and Precision [43]. Another metric that interests us is Coverage [45]. These metrics will be described in more detail.

If we want to find out whether the system recommends relevant items, we have to carry out tests in which a system generates the assumption for a binary pair (user x item) to be compared with the actual value. The result of each test can be a total of four states. Table 2.1 describes all possible states.

- TP are inputs suggested by a system and selected by a user.
- FN are inputs selected by a user but not recommended by a system.
- FP are inputs recommended by a system but not selected by a user.
- TN are all other, not recommended by a system and not selected by a user.

Based on these conditions, it is possible to calculate the following metrics 2.3.

$$\text{Precision} = \frac{\#\text{TP}}{\#\text{TP} + \#\text{FP}} \qquad \text{Recall} = \frac{\#\text{TP}}{\#\text{TP} + \#\text{FN}} \tag{2.3}$$

Precision is a metric that indicates the number of relevant recommendations to the total number of recommended results. Recall is the ratio of relevant recommendations to all interactions by a user. Both metrics affect each other: if one increases as a result of the settings, it may mean a decline in the other. Another metric is coverage 2.4, also called catalogue-coverage. This metric shows whether a system can recommend variously, i.e. if it can offer items other than best sellers . The metric is calculated as the number of unique items in all tests to all items in the tests.

$$\text{Coverage} = \frac{\#\text{Unique items in all tests}}{\#\text{All items in all tests}} \tag{2.4}$$

If a system only offers mainstream items, it can achieve great accuracy, but this is at the expense of coverage. This means that such a system is in practice unable to recommend an item which is not a best seller, although it may nevertheless be of interest to a user, which leads to the fact that a large number of non-recommended items (inputs) remain in a system.

Evaluation methodology There are several ways to simulate user behaviour in a system [43]. The most common method is to use historical data in which some user interactions are hidden and then used to make an RS predict their value. Ideally, timestamp interaction is used, based on which we can simulate system behaviour over time. The most commonly used procedure for large datasets is one in which we randomly select test users, randomly choose a time interval for these users, hide all the interactions of these users from this time period, and try to recommend items for those users. The problem with this approach is that it is necessary to perform the entire process before each recommendation, which is not very effective in terms of performance.

Since our RS always recommends *topN* items N=10 for each user, a leave-one-out cross-validation methodology [6] seems like a good compromise for testing the RS. Within this methodology, cross-validation is carried out across all users in the system, and recall and coverage are calculated. The exact description of the methodology and discussion of the selection of these metrics can be found here [6].

Leave-one-out cross-validation methodology We will construct an interaction matrix for all users of the RS and all items, where each row will be a user and each column an item. A single cell of the matrix will contain the number of interactions between an item and a user. If no interaction has taken place, the cell is filled with zero. During each step of the cross-validation, we will assign 90% of the users to a training set and the remaining 10% to a test set. For each user, we will always choose the top ten items (inputs). The following procedure is divided into steps:

1. We will teach a model on training data for each cross-validation step.
2. We will hide user interaction from the system and let the top ten recommendations be generated for each user from the test set and each item (matrix cell).

3. If the current item is between the recommended items while finding out that the value in a cell is greater than zero [1], the recall value of a single test is one, otherwise it is zero.

4. After we go through all matrix cells, we will calculate total cross-validation step recall by dividing the number of successful tests (1) by the number of all tests (all matrix cells). The total coverage is thus the number of unique items to all recommended items.

5. Then, we move on to the next step of cross-validation, i.e. to point 1.

6. After completing the cross-validation, we can count the average recall and the average coverage. This procedure appears to be optimal since it allows us to calculate the average recall across all tested users as well as solve the precision-recall dilemma described here [44].

2.1.4 Pareto Optimality

In our experiments, we use a multicriteria approach, that is difficult to evaluate, because of orthogonality of the parameters. So, we chose a Pareto optimality approach to define the best algorithm setup.

Let X be a set and $f_1, \ldots, f_n$ be real functions $f_i \colon X \to R$, values of which we want to maximise (also called the *objective* functions). We say that $x \in X$ *Pareto-dominates* $\hat{x} \in X$ *iff* the following two conditions are satisfied:

1. $\forall i \in \{1, \ldots, n\} \colon f_i(x) \geq f_i(\hat{x})$,
2. $\exists i \in \{1, \ldots, n\} \colon f_i(x) > f_i(\hat{x})$.

We also say that $x^* \in X$ is *Pareto-optimal* state *iff* it is not *Pareto-dominated* by any $x \in X$ different from x^*. We call *Pareto-optimal front* the set X^* of all *Pareto-optimal* states. In our case of Recall-Coverage evaluation, X is the set of available models (by example, a set of k-NN models of different k), and there are two objective functions present: f_{rec} and f_{cov}. These stay for the *recall* and the *catalogue coverage* on the fixed dataset being examined. One model, m, may be viewed as *Pareto-dominating* another model, $\hat{m}$, if at least one of the two following conditions is met:

- $f_{rec}(m) > f_{rec}(\hat{m}) \wedge f_{cov}(m) \geq f_{cov}(\hat{m})$, i.e. m has higher recall than $\hat{m}$, but still possesses at least as high catalogue coverage as $\hat{m}$,
- $f_{cov}(m) > f_{cov}(\hat{m}) \wedge f_{rec}(m) \geq f_{rec}(\hat{m})$, i.e. m has higher catalogue coverage than $\hat{m}$, but still possesses at least as high recall as $\hat{m}$.

[1] If we have a system with a large number of interactions, we can set a certain threshold, e.g. greater than five

If we accept the set X of objective functions as the relevant measures of model quality, then m may be considered strictly better then $\hat{m}$, and there is good sense in using m instead of $\hat{m}$.

2.1.5 Algorithms for text representation

In this section, we present the tree difference approach, wich we can use to compare the text corpus. First, we briefly describe the ontology, while the main description will be present in Chapter 3. Second, we describe the most common approach, which is Td-idf. Third, we present Laser, one of the new approaches, based on encoder/decoder and developed by Facebook engineers.

Ontology Ontologies are traditionally regarded as a branch of philosophy known as metaphysics, where ontologies are used to describe existence [11]. Generally, we could say that ontology is a formal specification of a dictionary with the concepts and their mutual relationships associated with a single domain area. Ontologies represent a knowledge model with a semantic description and structure in computer science and information systems. Ontology plays an important role in areas such as knowledge engineering, object-oriented analysis, knowledge representation, informational retrieval and extraction, database design, etc. [12].

Another important area at the moment is the Semantic Web [37]. The Semantic Web aims to change the WWW [2] environment so that information would not only be in a form readable to people, but would also enable systematic machine processing. The main idea is to use ontologies to describe entities and their relationships (OWL [3] language is used for ontology modelling) and to use a generic structural format (RDF [4] is used for notation) for the description of all data and to use the query language for their processing (SPARQL [5] is used for this). This article will use Semantic Web technologies for definition, notation, and subsequent processing.

Each domain area contains information specific only to itself. On the other hand, there is information that is shared with other areas. Therefore, the decision on whether to use one's own ontologies or those already existing in the RS poses a dilemma. If we decide to use our own ontology, we can design it to fit our purposes, but we will not be able to utilise the full potential of existing ontologies relevant to the area. If we decide to use an existing ontology, it will bring us benefits in the form of less human effort in the initial draft, the quality of our ontology will be increased as we will be using already tested elements, and finally, it will facilitate information sharing between other systems. However, to find a suitable, already existing ontology is difficult, because there are a large number of them and not all are fully relevant. Therefore, we decided to build our own ontology for this thesis and to subsequently find a suitable ontology for mapping.

[2] World Wide Web
[3] The W3C Web Ontology Language
[4] Resource Description Framework
[5] Simple Protocol and RDF Query Language

Tf-Idf Tf-Idf is short for *frequency–inverse document frequency* , and is a method used in information retrieval. The main idea of this model is to assign higher weights to more meaningful words in a document. The weight of a word is proportional to the frequency of the word in the document and inversely proportional to the frequency of the word in all documents in the collection [23]. The weight value $w_{t,d}$ of the word t in document d is the product of two values $tf_{t,d}$ and idf_t. The formula presents $tf_{t,d}$:

$$tf_{t,d} = log_{10}(f_{t,d} + 1) \tag{2.5}$$

Where f_{t_d} is the frequency of word t in document d. The formula presents idf_t:

$$idf_t = log_{10}(\frac{n}{df_t}) \tag{2.6}$$

Where n is the total number of documents, and df_t is the number of documents in which word t occurs. The final weight of the word t is the product of formulas 2.5 and 2.6:

$$w_{t,d} = tf_{t,_d} \times idf_t \tag{2.7}$$

The model assigns a zero weight to the words occurring in each document and assigns a higher weight to words that occur on average across all documents. The output of the model is a sparse matrix with vectors of high dimensionality. The algorithm uses only statistic values and does not use the semantic value of the words.

LASER is Language-Agnostic SEntence Representations toolkit, used to accelerate the transfer of natural language processing (NLP) applications to many more languages [3]. The toolkit works in many languages written in different alphabets.

LASER's vector representations of sentences are generic concerning both the input language and the NLP task. The tool maps a sentence in any language to a point in a high-dimensional space with the goal that the same statement in any language will end up in the same neighbourhood. This representation could be seen as a universal language in a semantic vector space. We have observed that the distance in that space correlates very well to the semantic closeness of the sentences.

The main idea is to use the encoder/decoder approach or sequence-to-sequence processing. The encoder is a five-layer bi-directional LSTM network obtained by max-pooling over the last states of the BiLSTM. The result is the fixed-size vector that has 1024 dimension [41]. Fig 2.1 show the principle of the LASER approach.

We can use this approach for computer embedding of the text corpus. After we get the embedding, we can use some typical techniques like cosine distance to compute the similarity between two text corpus. The reader can find more information about this approach reader in paper [3]

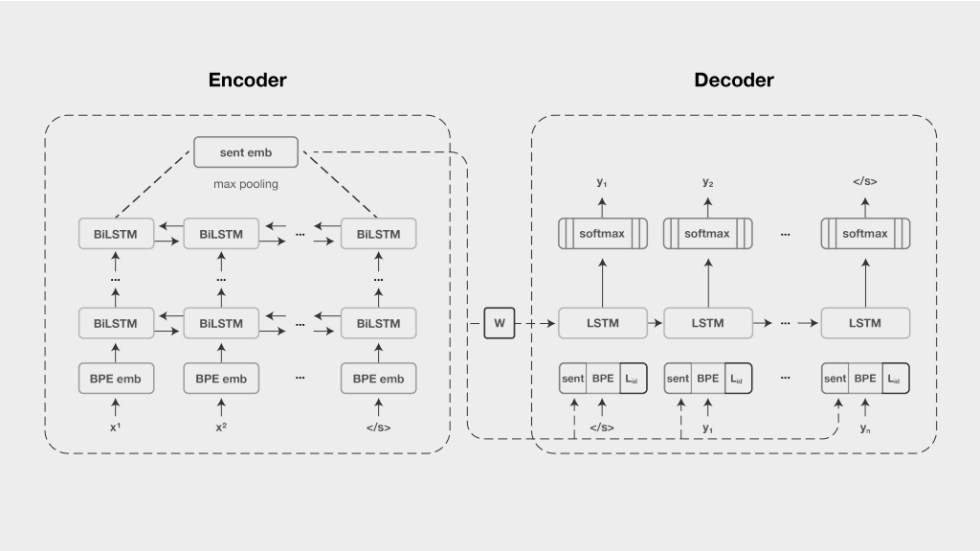

Figure 2.1: Picture of the main LASER approach. We take this picture from the website [41]

2.1.6 Matrix Factorisation

Factorisation is the operation of decomposing an object (number, matrix) into its simple components.

There are several methods with which we can decompose a matrix. The most popular are PCA (principal component analysis) [9], truncatedSVD [6] (truncated singular value decomposition) or NNMF/NMF [7](non-negative matrix factorisation).
Matrix factorisation algorithms work by decomposing the user-item interaction matrix into the product of two lower dimensionality rectangular matrices [24].
Figure 2.2 introduces the basic idea of matrix factorisation. The image is taken from this article [8]. Let there be matrix V with dimensionality (m, n) and this matrix can be viewed as a dot product between matrix $Q_(m, k)$ and $P_(k, n)$. The mathematical representation is here 2.8.

$$\hat{r}_{u,i} = q_i^T p_u \tag{2.8}$$

Then the objective function that we want to minimise is present in equation 2.9.

$$min \sum_{(u,i)\in K} (r_{ui} - q_i^T p_u)^2 + \lambda(||p_u||^2 + ||q_i||^2) \tag{2.9}$$

[6]sklearn.decomposition.TruncatedSVD.html
[7]sklearn.decomposition.NMF.html
[8]https://medium.com/@connectwithghosh/simple-matrix-factorization-example-on-the-movielens-dataset-using-pyspark-9b7e3f567536s

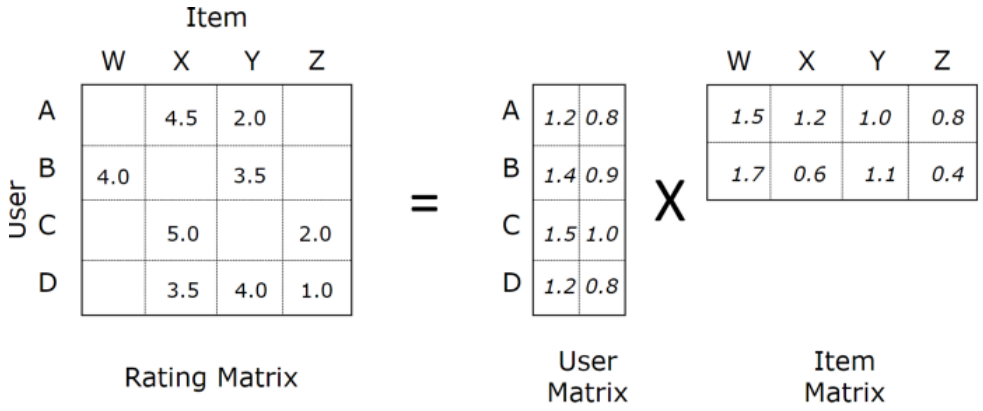

Figure 2.2: Picture of matrix factorisation.

The term on the right is the regularisation term; we added this because we do not want our decomposed matrix q and p to overfit to the original matrix. Since our goal is to generalise the previous ratings in a way that predicts future, unknown ratings, we should not overfit our model. The extent of regularisation is controlled through the tuning parameter λ, which ranges from 0 (no effect) to ∞ (maximum effect).
The proposed score function for our recommendation in [44] is:

$$score(u, i) = r_m + q_{*,i}^T . p_{*,u} \tag{2.10}$$

where $r_m \in R$ is the imputed value. More theoretical background information can be found in the linked footnote reference [9].

2.1.7 Diversity and Serendipity

Nowadays, companies with online stores offer thousands of different products, such as movies, songs, books, etc. With a wide range of goods, it is useful to recommend a product that fits user preferences. However, growth in choice does not always lead to user satisfaction. Sometimes it is difficult to find a product a user would like to purchase due to the overwhelming number of available products. Companies attempt to solve this problem by offering the user more diversity in terms of products, in the hope that there will be a greater probability that the user makes a purchase [25]. With regard to this problem, we describe two concepts in this section, diversity and serendipity.

Diversity is a desirable property in an RS, as users often become bored with recommendation lists containing items similar to each other [25]. It does not depend on information about the user, but it depends on the information inside the recommendation list. We can define it as the opposite of similarity and calculate it as a dissimilarity between the items

[9] https://www.sciencedirect.com/topics/computer-science/matrix-factorization

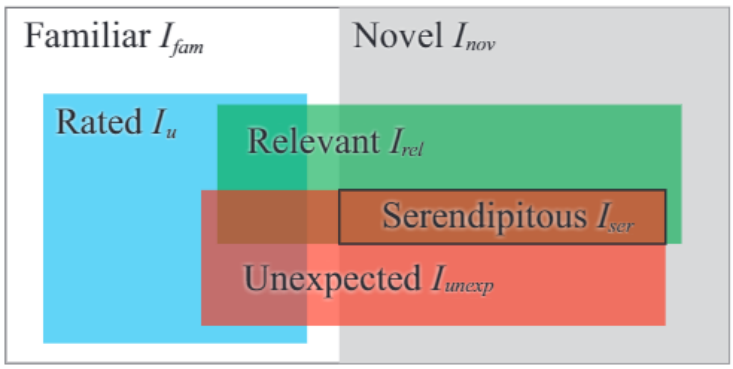

Figure 2.3: Euler diagram of items from a user's point of view at a given moment in time [25]

in the recommendation list. Diversity may be useful in scenarios where similar items may not be so relevant for the user. An example could be a holiday package, in which offering hotels from the same resort may not be as interesting as offering a list of different resorts. Also, we can use diversity to allow the user to explore our item catalogue faster.

Serendipity is also a concept related to the problem [25]. The main idea is to offer items that might surprise the user. The recommendation should be interesting (*relevant*), new (*novelty*) to the user and *unexpected*. For instance, recommending a just-released album of a favourite musician may be novel to the user but it is hardly surprising. Recommending a random item, on the other hand, may be very surprising, but not relevant. Serendipity can therefore be defined as representing delightful surprise for the user. Figure 2.3 illustrates this concept.

There are two key differences between diversity and serendipity. First, diversity is based on the comparison of recommended items, while serendipity is based on the comparison of recommended items and user profile [20]. Second, diversity does not include relevance to a user, which is a necessary component of serendipity.

2.2 Previous Results and Related Work

2.2.1 Ontology-Based Recommender Systems

Ontologies are the basis for ontology-based recommender systems. The main idea of the entire concept is to create a model that could describe the mapping between a domain area and users belonging to this area. In practice, this means creating user profiles which would include both implicit and explicit information [22]. It is very difficult to obtain explicit information because the preferences of users can often change, and not all users are willing to report these changes. One possibility is to ask users to fill out a questionnaire in the

system; unfortunately, users are not always willing to do this. Therefore, it is necessary to let the system make use of implicit information, i.e. the information that can be derived from user interaction, and modify user preferences (the user profile) based on this.

2.2.2 State-of-the-Art

This section will introduce some recommender systems using ontology either as the main tool for recommending or as a part of the recommending process. Procópio 2013 [36] described several studies involving ontology-based recommender systems with regard to the division of recommender systems into the above-mentioned groups – collaborative, content-based and hybrid – in Section IV. Furthermore, he presents a table comparing the individual studies. Since there are many studies in this area, we chose only those that are semantically close to the current thesis.

The authors [19] describe how to create a framework that would provide personalised Internet advertising in the form of a service. First, the advertising categories are created according to the commercial categories appearing on the Yahoo portal. Subsequently, keywords are extracted from the texts of adverts. Incorporating these keywords into categories is carried out using TF-IDF [10] analysis. Subsequently, the ontology of advertising words is built. NGD [11] is used to derive a feature vector for each advert. Logistic regression is then used to describe the preferences of each user; the responses to an already displayed advert will be used for this. Finally, a recommended value is calculated on the basis of whether the user sees the advert or not.

Hexin [30] describes a framework for recruiting new people for jobs where they seek to increase the likelihood of finding a suitable candidate using the ontology of skills and reciprocally to help job-seekers find suitable employment. The ontology is made of nodes with weights in the interval 0 to 1 between them. If the relationship between the nodes is defined as "is-a", the weight is set to 1; if the relationship is defined as "part-of", the weights are adjusted manually. A special sub-ontology is created for each industry.

BlindDate [38] is a recommender, a CB platform that utilises semantic technologies to describe user preferences. They enrich a generated ontology model with information from DBPedia repositories. The instances inserted into the ontology enable matches between users.

Biperdia [13] is an ontology that is used to support a search engine. It contains more than 1.6M pairs, attributes, and entities. The ontology extends the Freebase [12] database by extracting a syntactic analysis of user queries typed during a search by a user. A set of synonyms and word pattern where the synonyms were found is saved for each attribute.

[10] Term Frequency and Inverted Domain Frequency
[11] Normalized Google Distance
[12] https://www.freebase.com/

This approach makes it possible to describe the attributes in a much broader context.

OntoCommerce [8] is an e-commerce system that incorporates semantic algorithms for product recommendation. It assimilates ontologies and recommends products based on the user query, recorded user navigation as well as user profile analysis. They use parametric fuzzification to increase relevant product recommendations.

FO-SSSM [40] is an approach for mapping CO-SSSM [13] to a fuzzy context in the form of fuzzy ontology (FO)-based similarity. The main advantage of the framework is that, rather than developing a specific fuzzy equivalent for each of CO-SSSMs, it defines a generic paradigm that is applicable to all common CO-SSSMs.

Lenhart [27] present a hybrid new sports recommendation system that combines CB recommendations with CF. The result shows that a pure CB approach delivers accurate news recommendations. They create user profiles based on implicit feedback the user shares when reading the article. Using automatically created keywords, the similarity between articles can be measured, and the relevance for the user can be predicted.

The personalized semantic recommendation system (PSRS) [28] for e-learning is a system using Video Structured Description (VSD) technology to extract keywords that are included in teaching materials. It subsequently performs a lexical analysis of these keywords, generating rules (auto-updating rules) which are stored in the ontology based on the keywords. Based on this ontology and other techniques (semantic mapping), semantically coherent units are created which are then recommended to users within e-learning lessons.

Zhou [48] focuses on the definition of human skills in mechanical design. He proposes a set of skill metadata such as level of significance, experience, description, know-how and multimedia. The ontology is generated by using this metadata and their connections (component and association). The ontology meets all the standards of the Semantic Web and can be used both in recruiting and in e-learning education.

[13] Crisp Ontology-based Structural Semantic Similarity Measures

Chapter 3

Overview of Our Approach

In this chapter, we present our approach of generating ontology and using them for a recommendation task. The key idea is to use ontology as a semantic layer inside the recommendation algorithm and try to reduce the cold-start problem. If we want the process to work in the long term, this requires us to use the maximum amount of automatization. Firstly, we present the preprocessing approach and feature extraction. Secondly, we present the approach of automatically generating ontology and using implicit and explicit embeddings for edge weights. Thirdly, we describe how to use ontology for a content-based and collaborative filtering approach. Finally, we show how to use an ontology content-based algorithm to diversify our recommendation.

This chapter has two main sections:

First is Ontology Development 3.1, in which we explain our approach to developing the ontology.

Second is Ontology Recommendation 3.2, in which we present our ontological-based algorithms.Readers more interested in algorithms than ontology specification can skip the first part and come back later.

3.1 Ontology Development

In Section 2.1.5, we described the basic concept of ontology, and that is why we are going to focus on our approach here. As we have written before, the creation of ontology and its subsequent maintenance is a continuous process that we should keep improving. Ontology reflects the current information state in time, which ages considerably without maintenance.

The keywords we use in ontology have a time progression. For example, the word *"vector"*, which we use widely in data mining, is starting to be replaced by *"embedding"* for

some disciplines, even though it is about the same concept. Also, some words experience a *"take-off"* or *"hype"*, and become widely used even when it does not make as much sense; after a while, they become more meaningless.

Another essential feature of ontology is the context of the keyword. The context of the word is associated with the terms of its surroundings. As we work in real life with data that is multidisciplinary, e.g. at Experts.ai we need to distinguish words that mean something different to different fields. For example, the term *"AI"* in the context of data mining means *"artificial intelligence"*, but in the field of agriculture, it means *"artificial insemination"*.

These problems led us to begin to use both the *explicit embeddings* of words and the *implicit embeddings* of words. And to generate the final ontology as a combination of these two approaches. In the following sections, we describe the tool we used for explicit embeddings and metrics for both type of embeddings.

3.1.1 Text Mining Algorithms for Keyword Extraction

Text mining is a burgeoning new field that attempts to glean meaningful information from natural language text. It may be loosely characterized as the process of analysing text to extract information that is useful for particular purposes [47].

Nowadays, there are a number of libraries for text processing. The Stanford CoreNLP [1] library has proved the most successful for us, and the reader can find more information on this library in this [31] publication or its algorithm Bootstrapped Entity Learning [2] described in detail below.

One of the algorithms was developed as part of a dissertation by Sonal Gupta [14], where the author describes and proves the effectiveness of the entire algorithm using several studies. A concise description and evidence can be found here [15]. This work will describe the basic principle and basic steps of the algorithm. They use lexico-syntactic surface word patterns to extract entities from unlabelled text starting with seed dictionaries of entities for multiple classes. In practice, this means that four files are sent on the algorithm input. The first file contains the words or phrases whose patterns we are looking for. The second contains so-called golden words that are absolutely wanted. The third file contains words and phrases whose patterns the algorithm has to deal with. The last file contains unlabelled text. Furthermore, the parameters such as the threshold of acceptability, parser, lemmatizer, stop words, libraries for calculating the score and the number of iterations are set in the algorithm. The algorithm is shown in Figure 3.1. The output contains new searched words with patterns and their counterparts.

[1] The Stanford CoreNLP Natural Language Processing
[2] http://nlp.stanford.edu/software/patternslearning.shtml

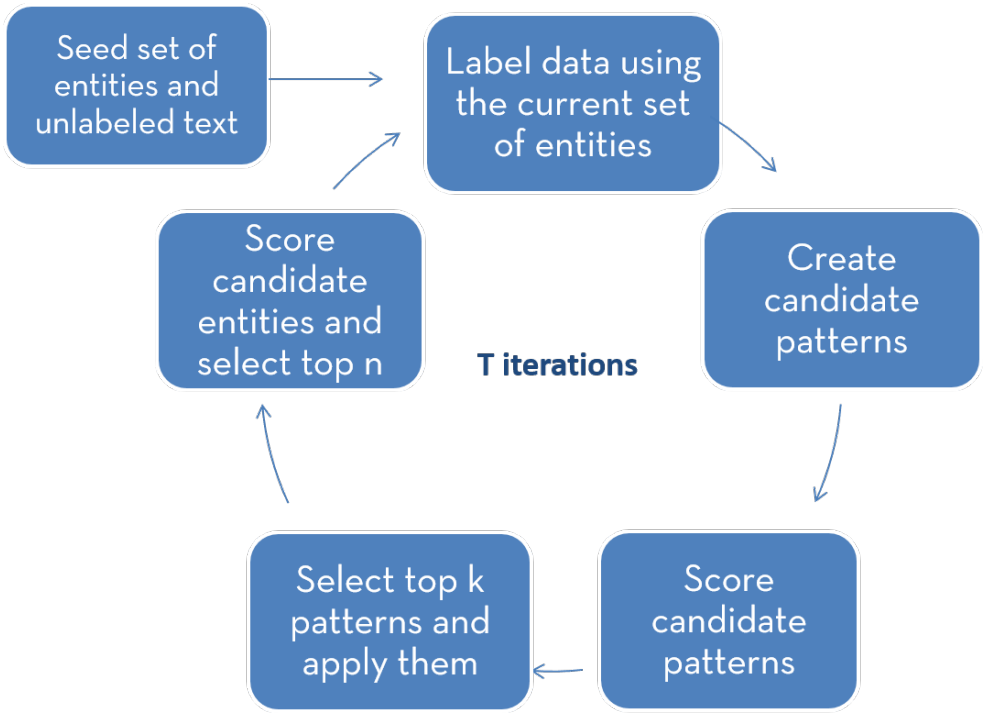

Figure 3.1: Bootstrapped learning algorithm process.

In our system, we used the keywords that each object contains within the text corpus. All acronyms are included in the golden words. In the algorithm, we used a threshold of sensitivity of ca. 0.01 (sensitive) and the number of iterations 100. This setting gave good results (about 20% noise) within an acceptable time (tens of minutes). The results of the algorithm were manually sorted, merged, and the final set of "proposed" keywords was eventually created. Figure 3.2 shows our algorithm adaptation.

Another algorithm that we use for data preprocessing is RAKE (Rapid Automatic Keyword Extraction) [39]. This algorithm works on the assumption that a keyword often consists of multiple words and only rarely contains interpunction or a stop word.

Algorithms delete stop-words from the input text corpus and prepare a words matrix, similar to GloVe [35]. Then they use metrics based on the number of co-occurrence and compute the score for each metric. The final score is calculating by the sum of all ratings. The output of the algorithms is T candidates for a new keyword.

3.1.2 Implicit Word Embedding Features

Our Ontology consists of Nodes (keywords) and Edges. Each edge presents a connection between two nodes. A relationship consists of several components or metrics, and the resulting edge value is given by weighting the individual parameters of that edge.

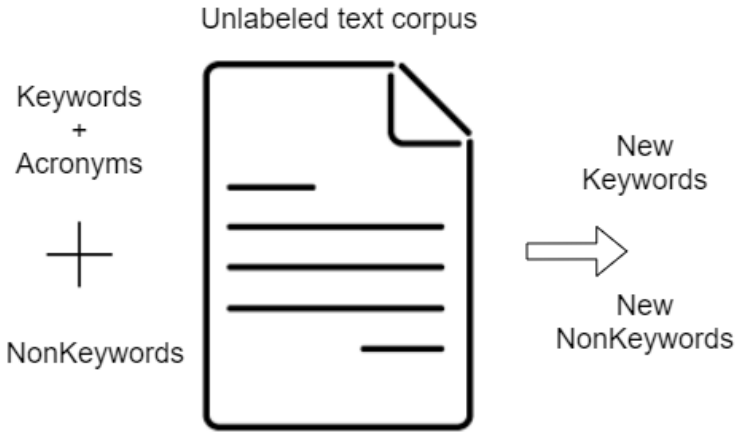

Figure 3.2: Our adaptation of bootstrapped learning algorithm.

In the beginning, we generate ontology with implicit metrics. The metrics are simple but powerful. With a combination of implicit metrics, we can develop the first implicit semantic of the keywords and we can start to use ontology for a recommendation. In 3.1 we show our most common metrics.

The primary metric is the "common occur", which represents the count of the common occurrence of the two keywords at all items in the dataset. Another important metric is the average year of the items in which the keyword mostly appears. We can compare two keywords with the same meaning by this metric and if one keyword is much younger than another one, we can assume that this is a "coming keyword" that is starting to be used instead of the old one.

Unfortunately, the implicit embedding is generated directly from the dataset, so the semantic is very biased. If our dataset is not too large, it is impossible to detect the real-life semantic connection between words. This is why we should use explicit embedding that could include the unbiased semantic.

3.1.3 Explicit Word Embedding Features

For explicit word embedding representation, we use a framework called *ConceptNet*, and the advantages of this framework are described in this section.

"ConceptNet is a freely-available semantic network, designed to help computers understand the meanings of words that people use."[3]

[3]https://conceptnet.io/

Metric	Description
weight	Final weight
wCommonOccur	Common occur of the keywords
wSourcePopularity	Popularity of source keyword
wTargetPopularity	Popularity of target keyword
wPopularityRation	Ratio between popularity of target and source keyword
wSourceAvgYear	Average year of items in which source keywords occurred
wTargetAvgYear	Average year of items in which target keywords occurred
wAvgYearRatio	Ratio between average year metrics

Table 3.1: Basic implicit metrics in our Ontology

Relation URI	Description	Examples
/r/RelatedTo	The most general relation. There is some positive relationship between A and B.	learn <-> erudition
/r/HasProperty	A has B as a property; A can be described as B.	ice -> cold
/r/HasContext	A is a word used in the context of B, which could be a topic area, technical field, or regional dialect.	retrieval -> computer science
/r/IsA	A is a subtype or a specific instance of B; every A is a B	car -> vehicle; Chicago -> city
/r/Synonym	A and B have very similar meanings	network <-> net

Table 3.2: Properties that we use at ConceptNet as explicit metrics

It is a framework that contains word embeddings, similar to word2vec [33] or GloVe [35]. Its main advantages are that it is multilingual, has a lot of useful properties, and is linked with Linked Open Data. It allows the use of the same terms in other dictionaries, such as WordNet [4] or DBPedia [5], so is very useful, because it extends the amount of information. More information about ConceptNet can find on the documentation page [6].

ConceptNet uses several properties and metrics that we can use not only for word similarity but also for word context. We can use related words and word synonyms but also associated terms with similar meaning. In table 3.2, we present some properties that we used in our ontology.

3.1.4 Onto Architecture

The keyword that represents a node in our ontology presents by n-grams where n is maximum of four. We use lemmatization to get the primary form of the word. For keywords with more than one word, we substitute white space with the underscore symbol.

[4]https://wordnet.princeton.edu/
[5]https://wiki.dbpedia.org/
[6]https://github.com/commonsense/conceptnet5/wiki

Our ontology is an oriented graph, also called a knowledge graph. The reason why we use graph architecture instead of a tree we describe in appendix A. Every edge in graph is oriented and contains a set of attributes described in table 3.1. If there is a connection between two nodes, there always exist two edges. In ontology, we use attributes that present *ratios* between nodes. For example, node A, which occurs in 1K items, and node B, which occurs in only 10 nodes, have different popularity. Therefore, the edge from A to B has a popularity ratio attribute equal to 100 (A is 100 times more popular than B) and edge from B to A has popularity ratios of only 0.01. Furthermore, we also use average year and average item votes.

This architecture allows us to provide ontology space exploration. This means that we can choose the associated word by explicit, implicit or both weights. However, we can determine the direction of our exploration, i.e. whether we explore more popular nodes or newer nodes in terms of time. This is useful, and we can use it in a context detection task or finding a rising star task.

For each item in our system, we should take the text attributes and create a text corpus. In this corpus, we should remove punctuation, tokenize the words, remove stop-words, carry out lemmatization and generate n-grams. After we have preprocessed the data, we map them to ontology and generate the item profile. In other words, we map the item text data to ontology space. Fig 3.3 present the ontology from the next chapter; This is the ontology generated for the movie domain. In the figure, we can see that, in the centre of the ontology are grouped more significant nodes with lots of edges,while less significant nodes are moved to the edges. Fig 3.4 shows one part of the ontology, the part generated by the movies with a Rock N Roll theme. We see how similar nodes make the clusters.

For the recommendation task, we convert the onto profile to embedding with the dimensionality of N, where N is the number of all nodes in the ontology. The final task is to find similarities in the ontology profile matrix, where rows are items, columns are nodes and values could be an ontology a weight of one if the profile contains a node, or zero if no.
One of the advantages of this approach compared to Tf-idf 2.1.5 or LASER 2.1.5 is profile exploration. While the ontology matrix is too sparse, we can extend the ontology profile by other related nodes, with some threshold, and make the matrix less sparse, and do it dynamically. Fig 4.2 we can see the basic profile of the Iron Man movie, from the next chapter. Fig 4.3 presents the extended profile.

In the next chapter, we demonstrate this functionality in the experiment.

3.1.5 Onto Profiles

For ontology generation, we use the above techniques such as keyword extraction, implicit and explicit embeddings. Ontology creates a single semantic space into which we can map

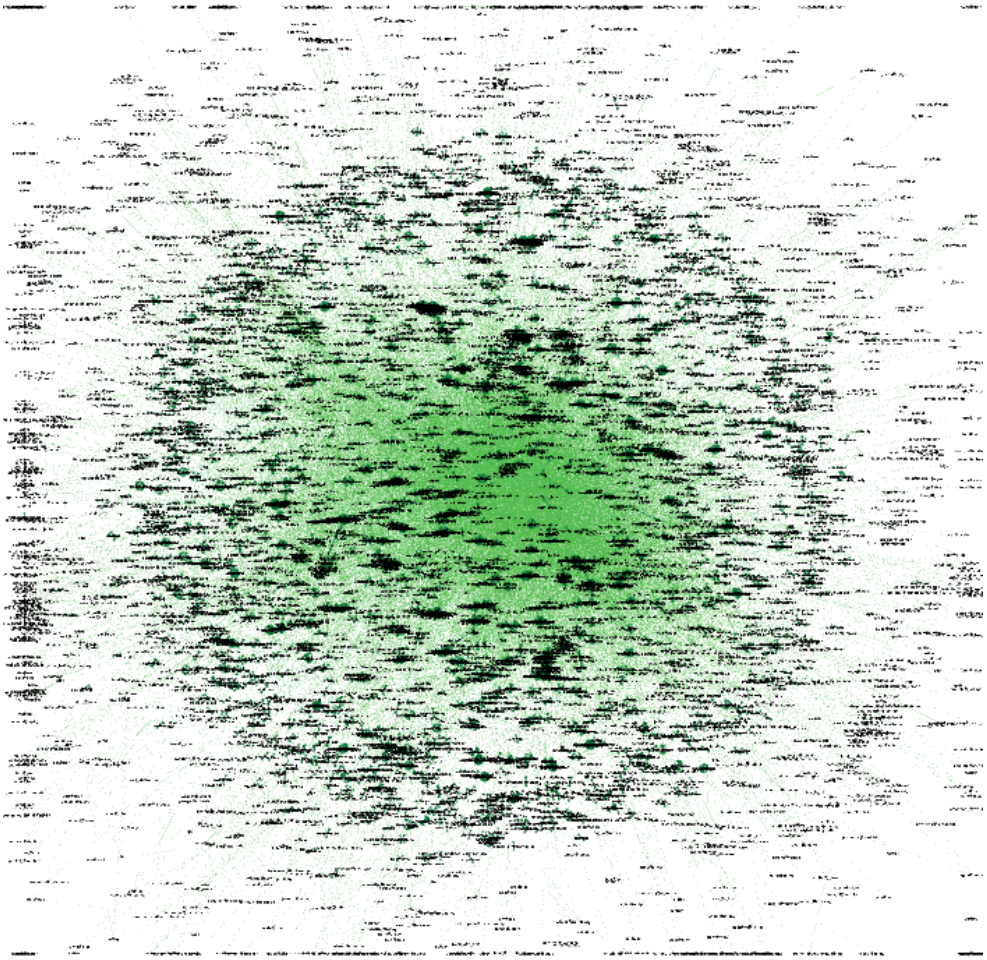

Figure 3.3: Ontology generates for movie domain from a higher perspective.

the attributes of our items. We can use ontology in many cases, such as a recommendation task or semantic search. In this thesis, we focus on recommendation tasks. Our approach is to use the maximum of automatization because we believe that this supports the long-term use of the whole system. If the data quality is high, ontology does not need any manual adjustments.

The advantage of ontology is simplicity. Each edge between two nodes can be decomposed into an implicit and explicit coefficient so that we can describe connections between nodes and then the connection inside the item profile. This feature is crucial because it adds the semantic level to our case and can help us to more deeply understand the behaviour of the system.

The disadvantage is that our approach is sensitive to low data quality. If the data quality is too low, the ontology could contain a lot of noise, i.e. words without semantic (explicit)

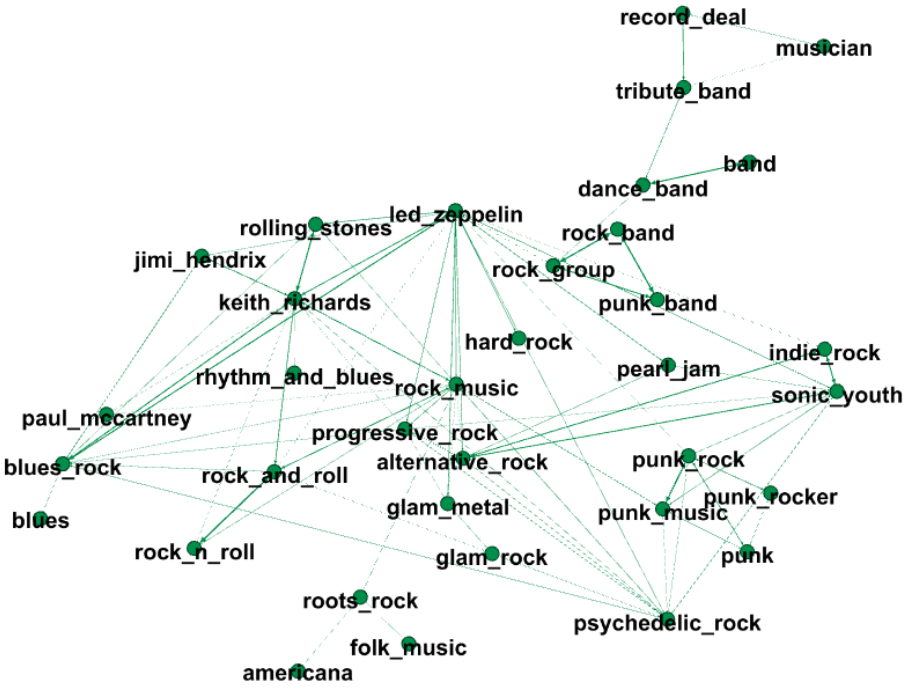

Figure 3.4: Detail view of one of the parts of the ontology that is generated by movies with Rock N Roll themes.

weights or words that have minimal implicit weights. This could be a problem because it creates problems with large dimensionality and sparsity in our matrices. We can eliminate this problem by manual correction.

3.2 Ontology Recommendation

In Section 2.1.2, we describe the main aspects of the cold-start problem. In this thesis, we present a solution for using an ontology approach to reduce the new-user cold-start, new-item, and new-system problems and, subsequently, how to use ontology in the further stages of systems. We focus on the *topN* recommendation task. Next, in this section, we describe the ontological similarity and ontological content-based algorithm.

3.2.1 Ontological similarity

The classic approach to reduce a new-system cold-start problem uses the *best seller* algorithm. The problem with a *best seller* is that it presents a small selection of items that may not be relevant to a particular user, i.e. these models are not personalised. Furthermore, in the earlier stages, the recommendation should be diverse, but the *best seller*

might only recommend a small number of items. The solution is to extend *best seller* with ontology, the approach we present in this paper [26].

Suppose our system contains attribute data; then we can use a content-based approach, as described in Section 2.1.1. If the attribute data contains text, we can use methods such as *Tf-idf*, *LASER* or *ontology*. The main principle of this algorithm family is to find similar items, and if the user has engaged in the first interaction, we can offer similar items. We present a content-based algorithm and experiment in the next chapter. In this section, we continue to focus on the ontological approach.

As noted above, the item ontology profile presents by embedding with the length of N and N is number nodes is our ontology. The resulting matrix contains a large amount of dimension, and it is very sparse. Therefore, we should reduce the dimension of the matrix. We can use PCA or $MatrixFactorization$. In our approach, we use $MatrixFactorization$. This method will create latent space for all items, reducing the dimensionality to an acceptable size.

After we use the dimension reduction method, we can use the *k Nearest Neighbours* method to find the closest neighbours (items) by computing cosine similarity, or use just cosine similarity. If we use kNN, the number of neighbours, the K parameter is one of our hyperparameters. The final recommendation is the most similar items.

The advantage or our approach over *LASER* and *Tf-idff-based* algorithms:

- If the item has few keywords, we have the option to expand the profile.
- If the item has many keywords, we have the option to reduce the profile and keep only the significant ones.
- Although we use latent space for the recommendation, we can show the user the original item profile, and explain when we recommend a particular item.
- Ontology-based on quality data instead of *Tf-idf* and *LASER* contains only those keywords that mostly represent the item; they are cleared and concentrate the maximum of information. *Tf-idf* also uses relevant words, but it tends to have a lot of noise. Ontology seems to be more stable and not susceptible to the change of the authors.

An empirical experiment that we provide in the next chapter shows that the utilisation of ontology achieves better results than the classical content-based algorithm in the chosen dataset.

3.2.2 Universal Ontological-Based Algorithm (OBACS)

Universal **O**ntological-**B**ased **A**lgorithms for reducing the **C**old-**S**tart problem - OBACS. In this section, we describe how collaborative filtering algorithms (CF) work in a cold-start system. Next, we explain how we can use these algorithms in a transition period where the system goes from a cold-start state to a normal state, with a normal state being when there are enough users and interaction. Finally, we demonstrate our approach, using ontology to improve diversity in CF recommendation algorithms and accelerate item catalogue coverage.

CF models in *topN* recommendation tasks are important because, in normal systems, they provide significantly better results in terms of recall than content-based CB algorithms. While CB algorithms work by attribute similarity that enters a $foreign$ person and does not have any connection with user preference, CF algorithms use the interactions, implicit or explicit, that a particular user has. They can dynamically adapt to current user preference and so are better suited to the role of personalised recommendation.

The disadvantage of the CF approach in a cold-start environment is the fact that, if a given item does not have user interaction, the algorithm cannot recommend it. However, when the interaction is starting to appear in our system, we can then use CF. This period, where the system is still in the cold start, but also has some interaction, we call the "transition cold-start period". In this period, we should use a hybrid combination of CB and CF. The main goal in this period is to get as much user interactions as possible with regard to *cold* items. Cold items are those items without interactions. We discuss several solutions below.

The first solution is to generate random interaction for each item and each user. This is not a good idea because we lose relevancy. CF compare users by their interactions, and the principle works on the idea that interactions are relevant. When the interactions are random, then the CF algorithm offers a random recommendation and thus discourages users from using the recommendation.

The second solution is to use CB algorithms throughout the transition period and wait until at least one relevant interaction has been recorded for each item. This is a proper solution, but as we show in the next chapter, CF has significantly better recall, so users get a more relevant recommendation. Therefore, we should start to use CF as soon as possible.

The third method is to use a hybrid approach, i.e. an ensemble of CF with good recall and CB that could recommend similar *cold* items. In this method, it is essential to define the ratio between the use of CF and CB. If we start to use CF algorithms "too soon", the problem may be that the model will recommend the same items, but the later we use CF, the later we will have relevant personalised recommendations. We should therefore generate combined recommendations from both systems. At the beginning, we

should use more items from CB in the final recommendation, but after a while, the ratio should change to the CF side.

In this thesis, we use the third method. As a CB algorithm, we use the similarity-based algorithms 4.1 that we describe in next chapter with ontological-similarity inside. As a CF algorithm, we use the UserKnn algorithms 4.1.3.3. The experiment in this area is presented in the next chapter.

Furthermore, we use the hybrid approach in those cases where we have a new item in our system. Since we assume that the new item has the same attributes as other items in our system, we can then automatically generate an ontological profile and include this item with similar, relevant items. Another approach for this task is to use multi-armed bandits; you can find more information about this concept here [10]. We do not use this technique in this thesis.

3.2.3 Enhance Collaborative Filtering with OBACS

In Section 2.1.7, we described the concept of diversity and serendipity. Both concepts are useful in *topN* recommendations, because with them we can cover as many relevant items as possible, and the user can still explore new items, both diverse or unexpected, and not end up in "filter bubbles" [34].

3.2.4 Diversity

There are different approaches for how to provide diversity. For example, the dissertation thesis of [49] offers the approach of using the β parameter to regularize the coverage of long-tail items in the catalogue. We include this kind of diversity in our experiments.

Our approach is to use an ensemble of a CF algorithm, in our case $UserKnn$ 4.1.3.3 as a primary model and a similarity-based algorithm 4.1 as a secondary model. The primary model will provide the first, relevant recommendation, while a secondary algorithm will rerank this recommendation. The reranking method works on the principle that, for each next item in the recommendation, it finds the most dissimilar item according to ontological similarity. Our diversity always provides relevance, because of the primary algorithm and novelty, and because we always recommend unseen items to the user. In each stage of the cold-start system, the requirements for diversity could change, so that we present the γ parameter for the manipulation of the composition of the final recommendation.

3.2.5 RUN Boosting (serendipity)

The serendipity concept, as described in Section 2.1.7, should provide three main properties. *Relevance* – the item should be relevant for a user. *Unexpectedness* – the item should be a surprise to the user. *Novelty* – the item should be new. The behaviour of this

concept, compared to diversity, provides a recommendation that slightly decreases recall, but increases catalogue coverage. In the experiment below, we will discuss this behaviour. We found that this type of diversity boost catalogue coverage and keep recall at high levels. So, we use the **R**elevant **U**nexpectness **N**ovelty **Boosting**, or RUN Boosting name for this diversity. We will use this name in the rest of this thesis.

Our approach to RUN Boosting is to use the same method as for diversity, as described above, but with a slight difference. RUN Boosting should, instead of diversity, provide unexpectedness. Unexpectedness, in our case, means the outlier item or the item that is most ontologically distant from the user profile. To compute this, we should first calculate the ontological profile of our user. We can do this by a combination of the ontological profile of the interacted item and by the merging of all nodes that the historical items have into a single profile. The final profile is similar to the skill cloud that we present in case study appendix A. After we calculate the profile and obtain the recommendation from the primary algorithm, we can compute the ontological similarity between the user profile and the item in the recommendation. After we get the ontological score, we can calculate the *z-score*. There are two options for the final score:

- we select outlier items by (z-score $>$ threshold) and then substitute these items with the last items in the recommendation.
- we sort our recommendation by z-score.

In our experiment, we use the second option.

Chapter 4

Main Results

In this chapter, we present experiments that confirm our proposals from the previous chapter. For that purpose, we choose offline evaluation, because we believe that, if our proposal will work offline, then it should also work online. First, we introduce our datasets and the preprocessing dataset that we prepared for a cold-start task. Second, we present algorithms in several categories and describe our offline validation method. Third, we present the experiment and the results of all the algorithms on cold-start datasets. Fourth, we present experiments with diversities. The experiment should show the main ideas, the robustness of the method and prove the correctness of our proposal.

4.1 Experimental Setup

In this section, we describe our datasets and introduce our datasets. All the necessary principles and methods used in our algorithms are described in Chapter 2. For better replicability, we also add the algorithms schema.

4.1.1 Experimental Dataset

For our experiments, we selected the Full MovieLens 20M [17] [1] dataset and its extension *The Movie Dataset*[2].MovieLens is a very well-known dataset that is used as a benchmark for most academic articles concerning recommendation systems, and is primarily designed for offline testing. This dataset comes from the GroupLens Research laboratory of Computer Science and Engineering at the University of Minnesota.

The Movie Dataset extends the MovieLens dataset with additional metadata, extracted from the TMDB [3] database. For each movie, MovieLens collects additional text data such

[1] https://grouplens.org/datasets/movielens/
[2] https://www.kaggle.com/rounakbanik/the-movies-dataset
[3] https://www.themoviedb.org/

Dataset	File	Metadata
MovieLens 20M	tags.csv	film tags
MovieLens 20M	movies.csv	film genres
The Movie Dataset	movies_metadata.csv	original title overview
The Movie Dataset	keywords.csv	keywords

Table 4.1: Datasets, files and metadata used

Dataset	Total inter.	User count	Average user inter.	Movies count
2-3	12975	5252	2.5	2735
3-5	46054	10763	4	4809
5-10	194911	26784	7	7637
10-20	1063130	70170	15	10686
20-50	1879318	58864	32	12173

Table 4.2: Cold-user datasets and characteristics

as title, overview, keywords, production company, year, director, etc. This extended information is essential, as it gave us a more significant amount of text data to build an ontology. For a full overview of the attributes and a description of the file, see table 4.1. The average ontological-similarity 3.2.1 for all datasets is around 6%.

4.1.2 Cold-Start Dataset

Our primary goal in this chapter is to show how our algorithms work in a cold-start environment. One of the main characteristics of a cold-start environment is that the transition between cold-start and non-cold start is not a jump but is gradual. This means that we have to test whether the system is still in the cold-start and our algorithms have problems, or the number of interactions enables us to use the algorithms in a proper way and the cold-start problems disappear.

For this purpose, we should prepare datasets where all users will have the same amount of interactions. MovieLens contains more than 20M interactions, and is not homogenous, so we decided to split the dataset into five datasets; before we spilt the data, we removed noisy items and items with less than 50 interactions. We present our new datasets in table 4.2.

4.1.3 Algorithms

For our experiments, we select a set of algorithms, which we separate into several groups by their approaches:

1. Naive
    a) Random
    b) Bestseller
2. Content-based
    a) Tf-idf-Based
    b) LASER-Based
    c) Ontology-Based
3. Collaborative Filtering
    a) User-Based K-Nearest Neighbours (UserKnn)
    b) Matrix Factorization
    c) AutoRec
4. Hybrid
    a) OBACS

The implementation of each algorithm is described below.

4.1.3.1 Naive

This category contains algorithms with a naive approach.

The *Random* algorithm generates a random number from 0 to N, where N is the number of items in the catalogue. Subsequently, the recommendation consists of these *topN* randomly generated items. This algorithm features small *recall* and significant *catalogue coverage*. The small *recall* is given by the chance to hit a user's preference, the probability of which is $1/N$, where N is the number of items in the item catalogue. Since the algorithm generates each recommendation randomly, it can cover 100% of the catalogue over a more extended period of time.

The best seller algorithm always selects the *topN* of the best items, where the best can mean, for example, the most-rated item. The algorithm will always offer the same set of recommendations for each user. This means that its *catalogue coverage* will always be at most $topN/N$ where *topN* is the number of recommendation and N is the number of items in the catalogue. On the other hand, recall may be higher than in the case of the random algorithm, because best seller items are relevant. The characteristic of ontological best seller we present in the article [26], so we do not show this approach in this thesis.

Both types of the algorithm are extremes, and we use them relative to other algorithms to show that our approach works better than these extremes.

4.1.3.2 Similarity-Based CB Algorithm

This algorithm is a classic content-based algorithm. We describe it in Section 3.2.1, so here we summarize.

The guiding principle is to use the text corpus that we have for each item and to calculate the similarity based on it. In our case, we chose data from all files, i.e. the algorithm will recommend movies based on the inside similarities.

The resulting recommencement is calculated based on the cosine distance between films embedding. The schema of the algorithm is here 4.1. We use the same algorithms for each similarity method, like Tf-idf 2.1.5, LASER 2.1.5 and ontology 3.2.1, to show that the similarity is better. The schema of the algorithm is here 4.1.

For each item in the dataset we compute embedding with a particular method, then we get the matrix $M_{i,\vec{j}}$ where i is a movie and $\vec{j_i}$ is a vector. From this matrix, we compute the cosine similarity for each movie pair. Finally, for each movie, we get the list of most similar movies by cosine similarity our score is a cosine similarity value.

For *topN* recommendation, we use the $ItemToItem$ approach, where for each movie in the user profile, we recommend N recommendation for those movies the user has explicitly rated. The final recommendation contains movies that we sort by item score. N is a parameter that we can use to control the quality of the recommendation.

Algorithm 4.1 Similarity-Based CB Algorithm

for Every item $i \in I$ **do** ▷ Preprocessing Phase
    Get textual attributes and prepossess
    Prepare embedding $\vec{j_i}$
end for
Merge all embedding to matrix $M_{i,\vec{j}}$ where $i \in I, \vec{j} \in \vec{E_i}$
If needs, provide dimension reduction of $M_{i,\vec{j}}$
For $\forall\ (i_n, i_m)$ compute $similarity = \cos(i_n, i_m)$, $i_n \neq i_m$

function ItemToItem($itemId, N$)
return $TopN$ most similar item i_{sim} by $similarity$
end function

function recommendation(userProfile, n) ▷ Recommendation
    for Every i in userProfile **do**
        retRecomArray.append($ItemToItem(i, N)$)
    end for
    return retRecomArray[0:n]
end function

4.1.3.3 k-Nearest Neighbours Algorithm

UserKnn is a class of collaborative filtering algorithms. We use the user-based k-Nearest Neighbours algorithm with cosine similarity and voting. In [7], such an algorithm is referred to as the *Popularity-Stratified, Non-normalised Cosine Neighbourhood*, which we will, for simplicity, refer to as *User-kNN* throughout the rest of the thesis. To rank items in a user's neighbourhood, the following formula is used:

$$\text{rank}(u,i) = \frac{\sum_{\hat{u}\in N^k(u)} \text{sim}(u,\hat{u}) \cdot (r_{\hat{u},i} - \overline{r_{\hat{u}}})}{\left(\sum_{u\in\mathcal{U}}(r_{\hat{u},i} - \overline{r_{\hat{u}}})\right)^{\beta}}, \tag{4.1}$$

where $N^k(u)$ in the set of the k nearest neighbours, $\mathcal{U}$ is the set of all the users in the database, and β is the long-tail biasing parameter as proposed in [45]. This algorithm was also presented in conference paper [26].

4.1.3.4 Matrix Factorisation

Matrix Factorisation MF is a class of collaborative filtering algorithms that work on the principle we presented in Section 2.1.6. Our implementation of MF is based on truncatedSVD [4] and the NearestNeighbours [5] algorithm. We use latent space and the nearest neighbours approach to find kNN items. We also use average item rating and popularity to compute the final score. The whole algorithm is presented here 4.2

Algorithm 4.2 Matrix Factorisation

```
1: function PREDICTITEMBYUSER(userItemProfile, N)          ▷ Recommendation
2:     transform userItemProfile to MF latent space
3:     find kNN users and cosine distance

4:     for every NN user do
5:         compute average rating

6:         for every item of NN user do
7:             compute diff as difference between average user rating and item rating
8:             compute score, by multiply user cosine distance and diff
9:         end for
10:        remove less popular items (rating is lower than average rating)
11:    end for
12:    sort recommendation by score
13: return TopN recommendation
14: end function
```

[4] sklearn.decomposition.TruncatedSVD
[5] sklearn.neighbors.NearestNeighbors

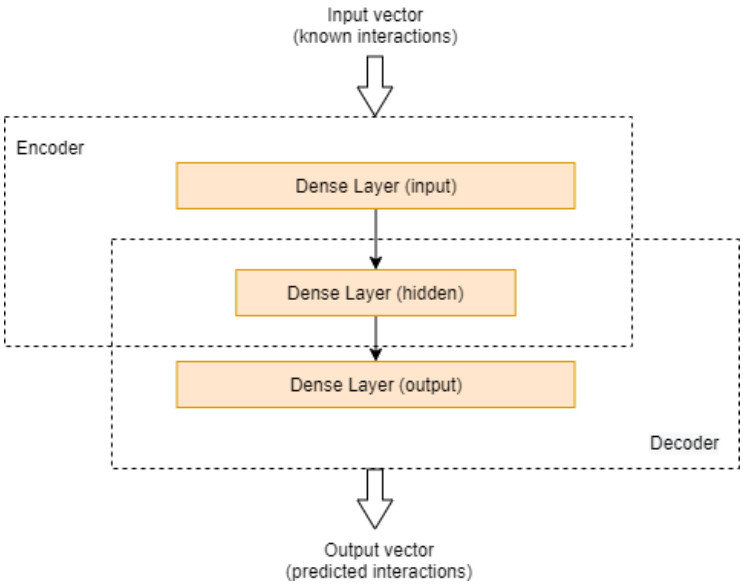

Figure 4.1: AutoRec architecture.

4.1.3.5 AutoRec

AutoRec [42] is a collaborative filtering method based on the use of autoencoders. In the base variant, the autoencoder is a forward-facing neural network composed of three layers: an encoder, a hidden layer and a decoder [18]. AutoRec uses this architecture to get latent factors from an input vector that corresponds to one row of the user-based AutoRec interaction matrix (item-based AutoRec column).

Figure 4.1 describes AtoRec architecture.

A hidden layer represents latent factors. Unknown interactions are coded as zero in the input vector and are masked in the output when training the neural network to be ignored by the backpropagation algorithm.

AutoRec is mostly used for rating prediction tasks; for the *topN* recommendation task we need to make a slight adaptation. First, for our test user, we predict his future rating for all items in the catalogue. Second, we sort the items by the new predicted rating. Third, we select N best items. The algorithm is straightforward, and maybe we can add more complexity, but, as we will see in the next section, this version works well.

4.1.3.6 OBACS

OBACS is the main algorithm that we describe in Section 3. In the beginning, we should generate ontology. We take all the text data that we presented in table 4.1, then we apply the preprocessing methods and use the keyword extraction algorithms from Section 3.1.1.

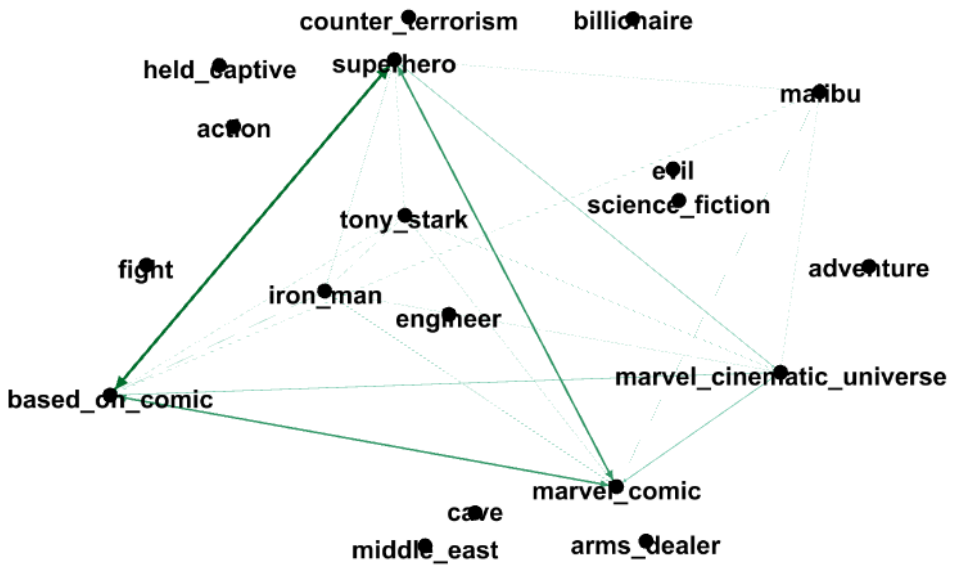

Figure 4.2: The basic ontological profile of the Iron Man (2008) movie.

Our keyword could be composed of a maximum of four words ngram. Each keyword represents a unique node in our ontology. After we have a list of nodes, we then compute implicit and explicit embedding, calculate the matrix and define the edges between all nodes. The final ontology contains 76K nodes and 1.1M edges. Figure 3.3 presents the ontology.

After we build an ontology, we prepare a profile for each movie. For the first part of the experiment, which we present below, we make multiple types of profiles. The main idea is to find the best compromise of several nodes that could represent a particular movie. We can extend the original movie profile in several ways. In Section 2.1.5, we present some options. Figure 4.2 shows the example of basic profile for the Iron Man movie and figure 4.3 shows the example of extends the profile for same movie.

OBACS uses hyperparameter γ. This parameter presents the ratio between recommendations from primary and secondary algorithms.

4.2 Cold-User Recommendation Experiment

In this section, we demonstrate the results and behaviour of each algorithm from the previous section in a cold-user environment. We use datasets from table 4.2. We use the leave-one-out cross-validation technique from Section 2.1.3 for the evaluation of our algorithms. The result is presented in the recall-coverage optimization space with model capacity parameters [6] [49]. Each dataset is split into a training and test dataset. Because

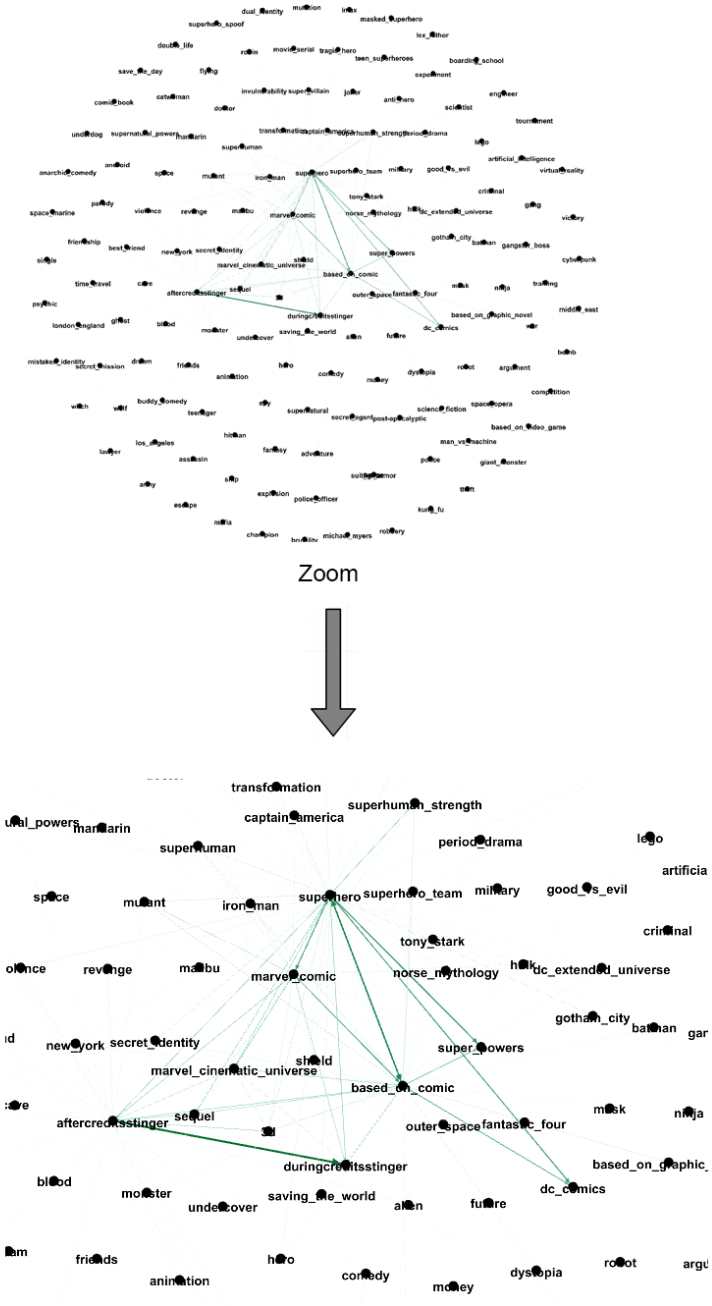

Figure 4.3: The ontological profile of the Iron Man (2008) movie, with extended nodes. For each node from a basic profile, we add most semantically similar nodes.

our test dataset contains a different number of user and movies, we do not use the same ratio for all datasets; it differs in range (20-30%). The test dataset contains shuffle data and there is a disjunction with the training dataset. We evaluate all algorithms on the same training and test datasets.
The main idea of this experiment is to show the behaviour of this algorithm in several scenarios, that presents our dataset, where the first dataset contains users with several interactions, but the latest dataset contains users with enough interaction, so this dataset no longer has a cold-start problem.

One of the primary goals of this experiment is to show how the algorithm works in our prepared datasets, discuss the results, and choose the best algorithms for the next experiment.

Content-Based Algorithms The first set of algorithms would be CB. For that purpose, we select Tf-Idf-based algorithm 2.1.5, LASER-based 2.1.5 and ontology-based algorithms 3.2 For this algorithm approach, we set the hyper-parameter N that we use for capacity manipulation. The value of this parameter means how many similar items should return for each user item. Where this parameter is one, there is always return one item for each item in user profile. While parameter is set to ten, the final recommendation will contain only the item similar to first (last) user item. All our CB algorithms use the same parameter.

Collaborate Filtering Algorithms The second set of algorithms are CF. We chose our userKnn algorithm 4.1.3.3, Matrix-Factorization (MF) based algorithm 4.2 and AutoRec algorithm 4.1.3.5.

The UserKnn algorithm use K parameter for capacity manipulation. K is the number of k nearest neighbours. The range of the parameter is set from 3 to 1597. In this expeiment, we do not use the parameter β, this parameter is set to 0; this means that there are no long-tail recommendations.

MF use hyperparameter L. This parameter presents the dimension of latent space in MF. The range of the parameter is set from 5 to 1000.

Autorec uses hyperparameter H. This parameter presents the number of neurons in the hidden layer. The range of the parameter is set from 4 to 512

4.2.1 Results

- Fig 4.4 shows the 3D projection of all algorithms results, where the X-axis contains recall, the Y-axis contains catalogue coverage, the Z-axis contains onto-similarity

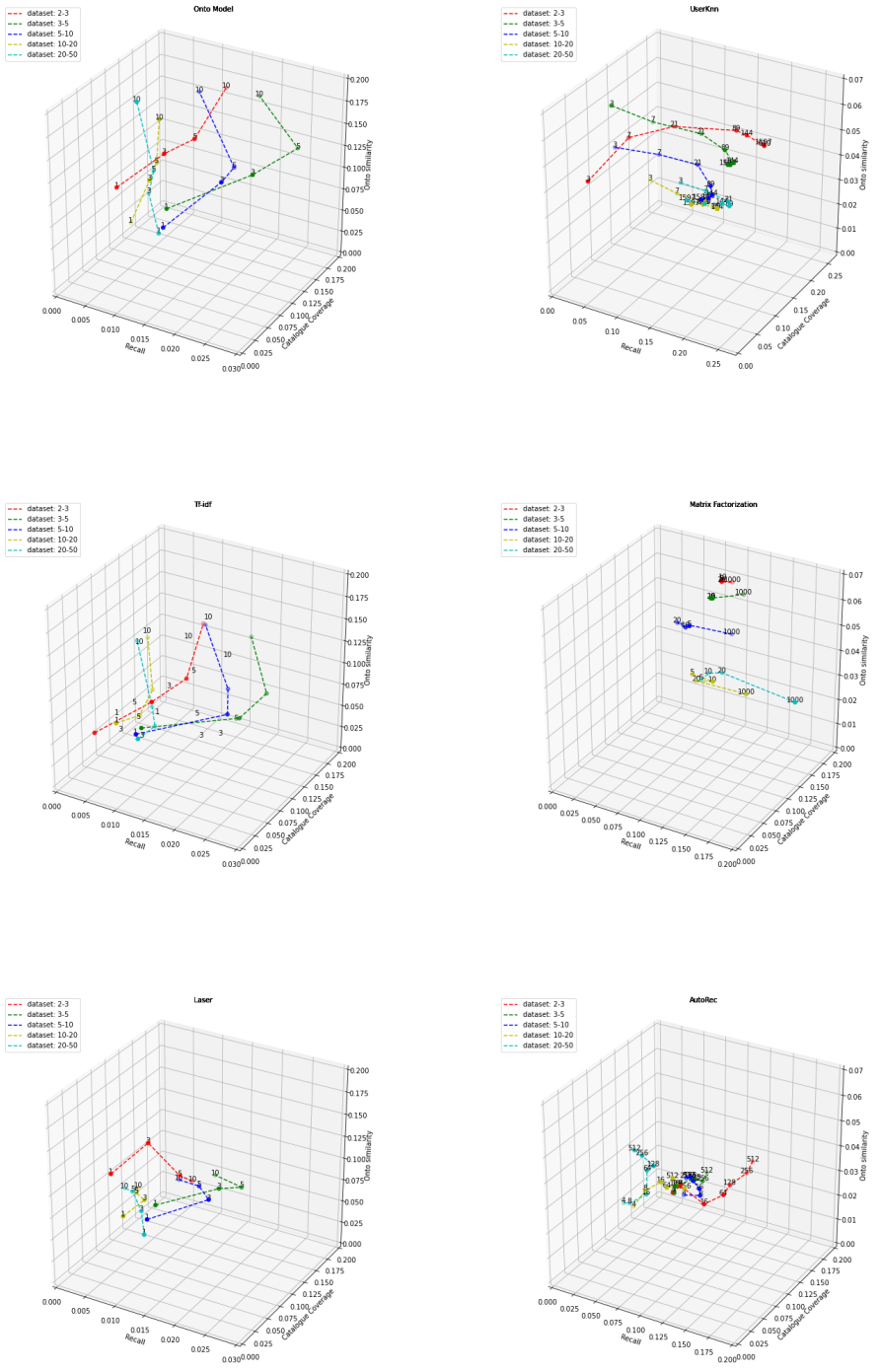

Figure 4.4: 3D projection of all algorithms results, where the X-axis contains recall, the Y-axis contains catalogue coverage, the Z-axis contains onto-similarity

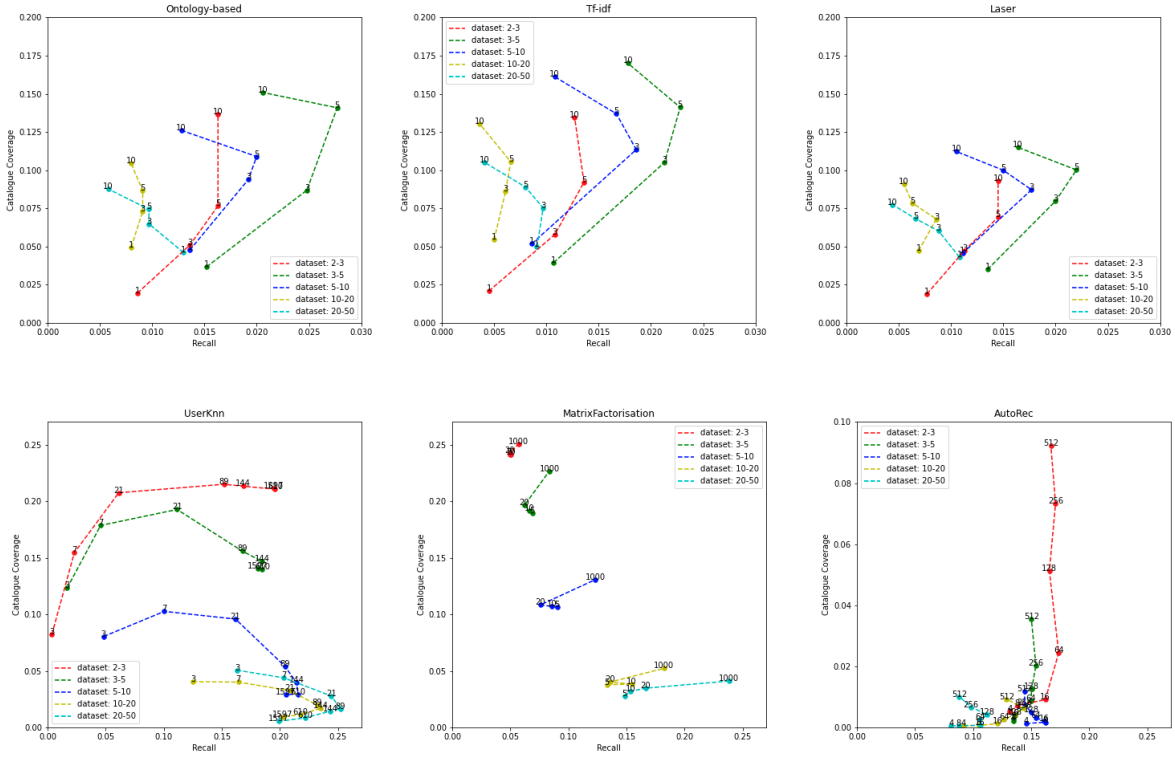

Figure 4.5: The result of the experiment, where the X-axis contains recall, the Y-axis contains catalogue coverage.

- Fig 4.5 shows the result of the experiment, where the X-axis contains recall, the Y-axis contains catalogue coverage.
- Fig 4.6 shows the same results, but on the X-axis is recall, the Y-axis is onto-similarity.
- Fig 4.7 shows the comparison of similarities, inside CB algorithms in each dataset.

At first look, there is a significant difference between CB and CF approaches. CF algorithms have significantly better recall than CB. CF algorithms work better with more interaction; this is not a surprising fact. In the case of catalogue coverage, the algorithm shows similar results; the difference is in particular algorithms.

Another interesting difference is ontology similarity. While CB algorithms could provide a recommendation with some scale, the CF algorithm scale is significantly smaller. This is typical for CF, because they provide recommendation by the interaction of the most closes users, and they do not work with any item similarity. Their values are almost the same as the average onto-similarity in datasets.

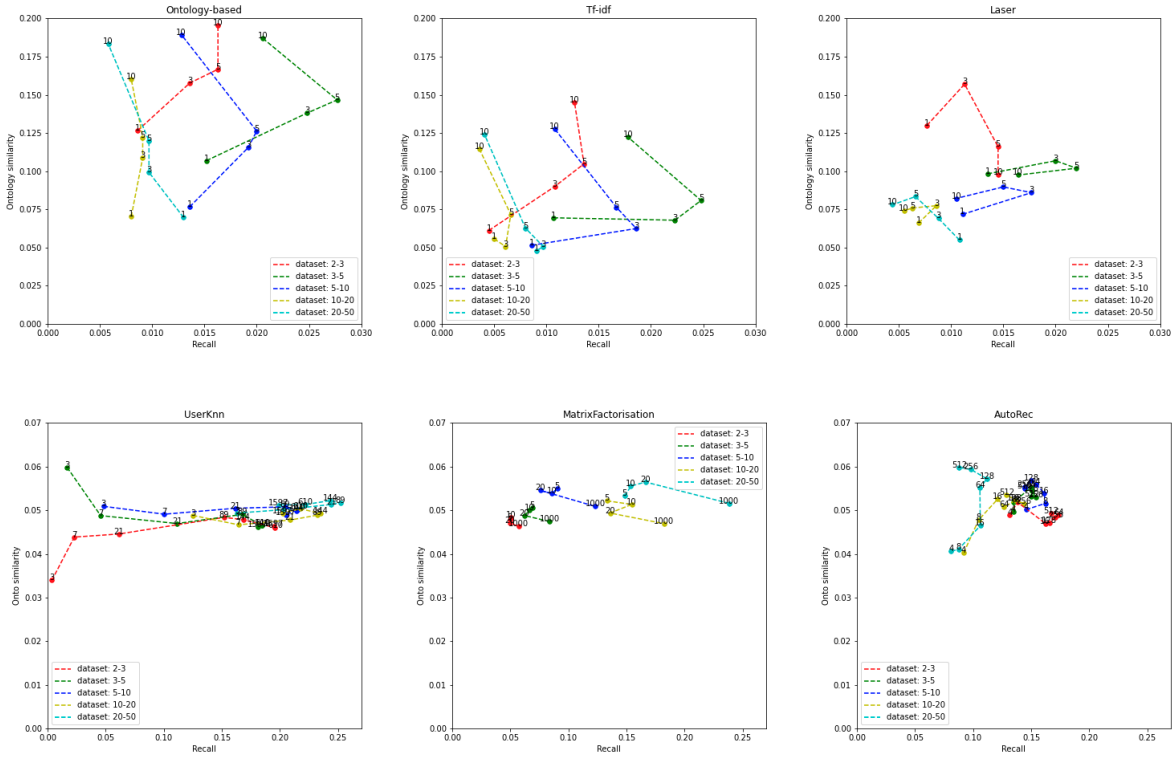

Figure 4.6: The result of the experiment, where the X-axis contains recall, the Y-axis contains onto-similarity.

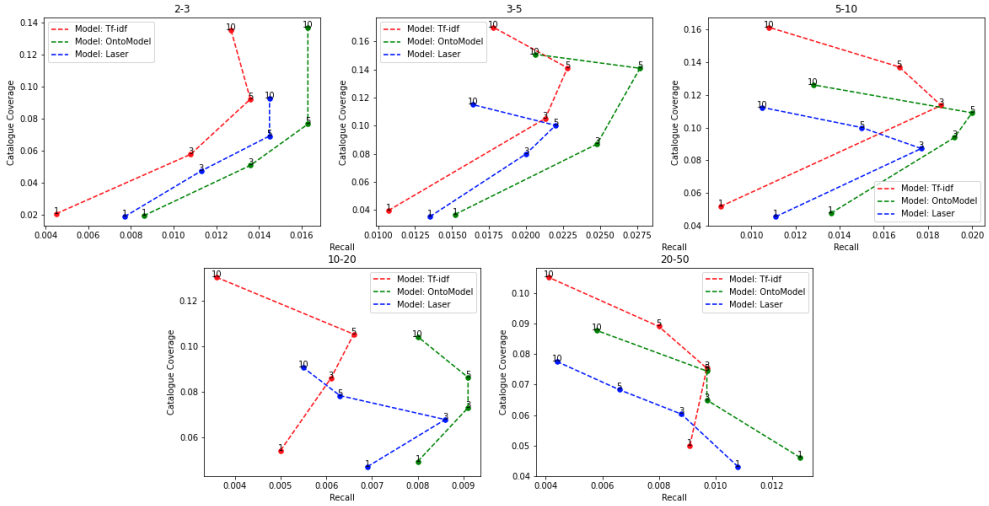

Figure 4.7: The comparison of similarities, inside CB algorithms in each dataset.

Fig 4.7 shows the comparison of CB algorithms in each dataset. In the result of the experiment, we can see that the ontology-based model is Pareto dominant 2.1.4 than other CB algorithms. The Tf-idf and LASER algorithm is Pareto not dominant compared to ontology-based. The reason is that in our approach ontology could provide more relevant keywords, and if the film has a small number of nodes, we can increase this number with semantically close words.

The best result and Pareto optimal algorithm from CF is userKnn. This algorithm provides the best recall and not significantly worse catalogue coverage compared to MF. AutoRec shows interesting behaviour in the datasets with a small amount of interaction, where for dataset of 2-3 and 3-5 interaction shows good recall and growth of catalogue coverage, but still catalogue coverage is not so significant. While the number of interactions is growing, the AutoRec seems to be working more like the best seller, where recall is increasing but catalogue coverage decreases. Table 4.3 shows the results of the experiment.

4.2.2 Discussion

One of the conclusions of this experiment is that CF algorithms work better than CB in datasets that contains a small amount of interaction, so we should switch from CB to CF algorithms as soon as possible; otherwise, we lose recall and the relevance of our recommendation.

Another conclusion is that, in this experiment, the Pareto optimal CB algorithm is onto-based algorithms, and the Pareto optimal CF algorithm is UserKnn.

We also provide this experiment because we want to demonstrate that our algorithms and the results are approx. similar to the state-of-the-art.

While this experiment shows that we can reduce the cold-start problem with CF or CB, *"the devil is in the detail"*. Our assumption for this experiment is based on a dataset that has a small amount of interaction. However, there is still a lot of interaction for each item, so the CF could make a recommendation.

The real cold-start problem assumes that a lot of items exist without any interaction, so CF does not work. Unfortunately, this is a very common state, and in real-world businesses it takes a long time to make recommendations for each item.

We show our primary approach in the next experiment, where we present algorithms that could reduce the cold-start problem from scratch; this is more general and could be used for a lot of domains.

Model	Parameter	Recall	Catalogue Coverage	Ontological Similarity
Dataset 2-3				
AutoRec	H = 256	0.17	0.073	0.05
UserKnn	K = 610	**0.19**	0.21	0.05
MF	L = 1000	0.057	**0.25**	0.05
LASER	N= 10	0.014	0.09	0.1
Tf-idf	N = 5	0.013	0.09	0.11
Ontology	N = 10	**0.016**	**0.13**	**0.2**
Dataset 3-5				
AutoRec	H = 256	0.15	0.02	0.06
UserKnn	K = 610	**0.18**	0.13	0.05
MF	L = 1000	0.084	**0.22**	0.05
LASER	N= 5	0.022	0.1	0.11
Tf-idf	N = 5	0.022	**0.14**	0.9
Ontology	N = 5	**0.027**	**0.14**	**0.15**
Dataset 5-10				
AutoRec	H = 8	0.16	0.017	0.06
UserKnn	K = 610	**0.21**	0.029	0.06
MF	L = 1000	0.12	**0.13**	0.06
LASER	N= 3	0.17	0.08	0.09
Tf-idf	N = 3	0.018	**0.11**	0.07
Ontology	N = 5	**0.02**	0.1	**0.13**
Dataset 10-20				
AutoRec	H = 256	0.14	0.0062	0.06
UserKnn	K = 89	**0.23**	0.019	0.05
MF	L = 1000	0.18	**0.05**	0.05
LASER	N= 1	**0.01**	0.04	0.07
Tf-idf	N = 5	0.006	**0.1**	0.8
Ontology	N = 5	**0.01**	0.08	**0.13**
Dataset 20-50				
AutoRec	H = 128	0.11	0.0042	0.06
UserKnn	K = 89	**0.25**	0.016	0.06
MF	L = 1000	0.23	**0.04**	0.06
LASER	N= 1	0.01	0.04	0.06
Tf-idf	N = 3	0.009	**0.0075**	0.05
Ontology	N = 1	**0.013**	0.046	**0.07**

Table 4.3: Results of the Cold-User Recommendation Experiment.

4.3 Cold-Item Recommendation Experiment

In this section, we present our algorithm for reducing the cold-start problem from scratch. The idea of the algorithm describing in Section 3.2.2. Let us combine the CB approach for the first phase of the cold start, while at the beginning, we will mostly use a CB approach. Then, in the transition state, we use the combination of CB and CF, where the final recommendation can be made with a ratio parameter. When the number of our items without interaction decreases, we then switch to mainly CF algorithms.

Our approach is OBACS 4.1.3.6 with the Similarity-based algorithm 4.1 with ontological-similarity 3.2.1 as primary, while the UserKnn algorithm 4.1.3.3 would be secondary.

Experiment setup First of all, we should prepare our datasets. We get a cold-start dataset 4.1, and for each dataset, we prepare five "cold-start-item" versions. A cold item is an item that has no interaction, so we randomly select the required number of items, and remove their interaction from the training data. Every version of the dataset presents the number of cold items in test data. So the first dataset has 100% cold items; the third dataset has 50% and the last dataset 0%.

The reason why we use all these datasets is that the number of user interactions and the number of cold items does not mean correlation. Even if the average user has more than 20 interactions, there could still be more than 80% cold items in the catalogue.

4.3.1 Results

- Fig 4.8 shows the 3D projection of the algorithm results, where the X-axis contains recall, the Y-axis contains catalogue coverage, and the Z-axis contains onto-similarity.
- Fig 4.9 presents the result, where the X-axis contains recall and the Y-axis contains catalogue coverage.
- Fig 4.10 presents the result, where the X-axis contains recall, the Y-axis contains onto similarity.

At first glance, we can see that the algorithm works similarly on all datasets. The starting point, where all items are cold, starts in the same position as our ontology-based algorithms and provide the same characteristics. While the average number of cold items decreases, the algorithm moves to better recall.

We can see interesting behaviour at nearly half cold items. In the first part, or at the first part in our transition state, recall does not grow so well, but the diversity is increasing. This is because the UserKnn algorithms are still not working well and could not provide an appropriate recommendation. After we go into the second half, catalogue coverage still increases, because of Similarity-based algorithms, but recall is also increasing. The

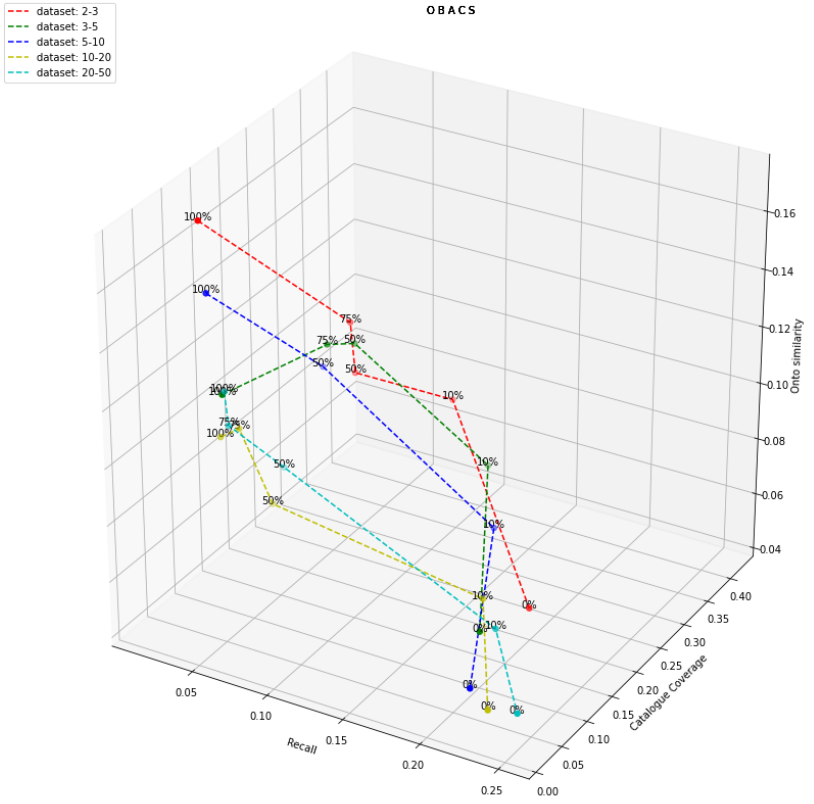

Figure 4.8: 3D projection of the algorithm results, where the X-axis contains recall, the Y-axis contains catalogue coverage, and the Z-axis contains onto-similarity.

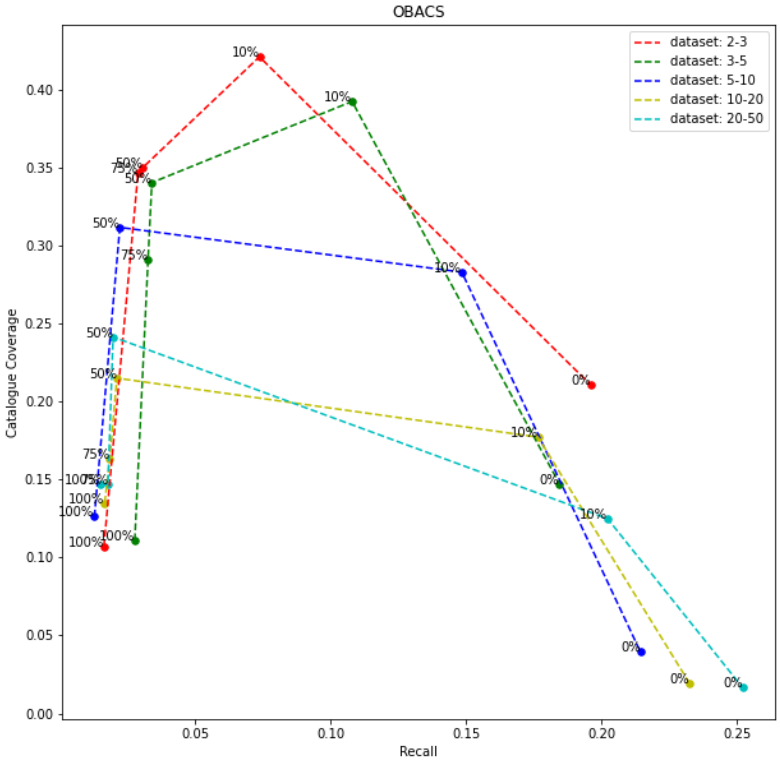

Figure 4.9: The result, where the X-axis contains recall and the Y-axis contains catalogue coverage.

reason is that our CF approach starts working, and we can recommend more relevant items.

The last phase of algorithms is when we switch to UserKnn algorithms, the behaviour is the same as in previous experiments. After we do this, recall increases, but coverage rapidly decreases. Our algorithms are in a "filter bubble".

Fig 4.10 shows the same behaviour, from the perspective of onto similarity, the more userKnn involve into a recommendation, the level of similarity decreases. At the same time, it reaches the average level in the datasets. table 4.4

4.3.2 Discussion

In this experiment, we prove our primary approach, that we can reduce the cold-start problem from scratch, then work effectively in a transition state, and after the cold-start

Cold Item	Parameter	Recall	Catalogue coverage	Ontological Similarity
Dataset 2-3				
100%	$\gamma = 0$	0.016	0.1	0.18
75%	$\gamma = 0$	0.027	0.11	0.12
50%	$\gamma = 0$	0.012	0.12	0.15
10%	$\gamma = 0$	0.016	0.16	0.1
0%	$\gamma = 0$	0.015	0.14	0.11
Dataset 3-5				
100%	$\gamma = 2$	0.028	0.34	0.11
75%	$\gamma = 2$	0.032	0.29	0.11
50%	$\gamma = 4$	0.019	0.32	0.09
10%	$\gamma = 2$	0.18	0.16	0.1
0%	$\gamma = 2$	0.17	0.14	0.1
Dataset 5-10				
100%	$\gamma = 4$	0.03	0.35	0.09
75%	$\gamma = 2$	0.034	0.34	0.1
50%	$\gamma = 4$	0.02	0.31	0.1
10%	$\gamma = 8$	0.21	0.21	0.06
0%	$\gamma = 8$	0.19	0.24	0.06
Dataset 10-20				
100%	$\gamma = 8$	0.07	0.42	0.07
75%	$\gamma = 8$	0.1	0.39	0.06
50%	$\gamma = 8$	0.14	0.28	0.06
10%	$\gamma = 8$	0.17	0.17	0.06
0%	$\gamma = 8$	0.2	0.12	0.06
Dataset 20-50				
100%	$\gamma = 10$	0.19	0.21	0.05
75%	$\gamma = 10$	0.18	0.14	0.05
50%	$\gamma = 10$	0.21	0.03	0.05
10%	$\gamma = 10$	0.23	0.019	0.05
0%	$\gamma = 10$	0.25	0.016	0.05

Table 4.4: Results of the OBACS algorithm in Cold-Item Recommendation Experiment

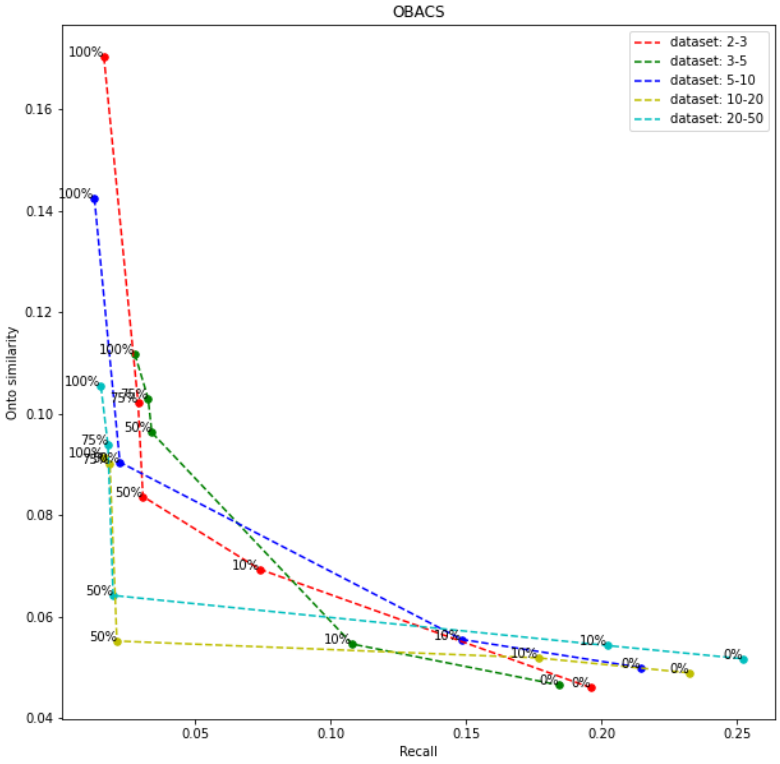

Figure 4.10: The result, where the X-axis contains recall, the Y-axis contains onto similarity

problem is gone, switch to a CF approach.

This experiment shows that a combination of these two approaches make sense. We can use it to reduce the cold-start problem, but we can also even use it in regular work. Fig 4.9 shows the difference between before the last state (10% of cold items left) and the last state is in increasing the recall but rapidly decreasing catalogue coverage. This points to the fact that an ontology-based model plays a role in algorithms that can in- crease diversity. Even though we are close to covering all the catalogue of cold items, and the role of ontology-based algorithms is minor, there is still good catalogue coverage.

In the next experiment, we prove our concept, that ontology-based algorithms could help a CF approach to increase diversity and also help it emerge from the “filter bubble”.

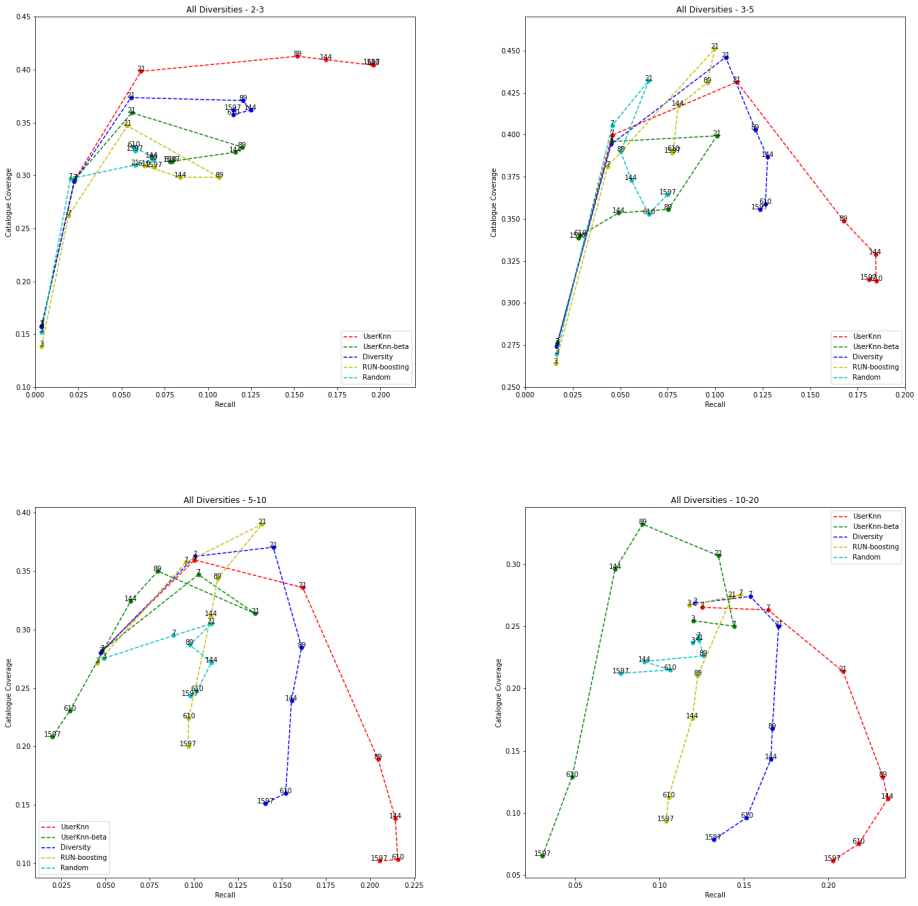

Figure 4.11: The results of all diversifications inside OBACS algorithms, where the X-axis show recall and Y-axis catalogue coverage.

4.4 Enhanced Collaborative Filtering Experiment

In this section, we present our last experiment, to prove that using ontologies, even when a system does not have a cold-start problem, makes sense and brings benefits. We discuss this idea in Section 3.2.3, where we present the possibility to use ontology to decrease the problem with the "filter bubble" [34] and increase long-tail recommendations. Our approach is to use diversification and RUN boosting.

Experiment setup For this experiment, we use our cold-start dataset 4.1 without any changes. We select our CF algorithm from the previous experiment UserKnn 4.1.3.3. We then select the re-ranking method that suggests [25]. For this purpose, we add a new feature to our OBACS algorithms, that uses ontology similarity for re-ranking.

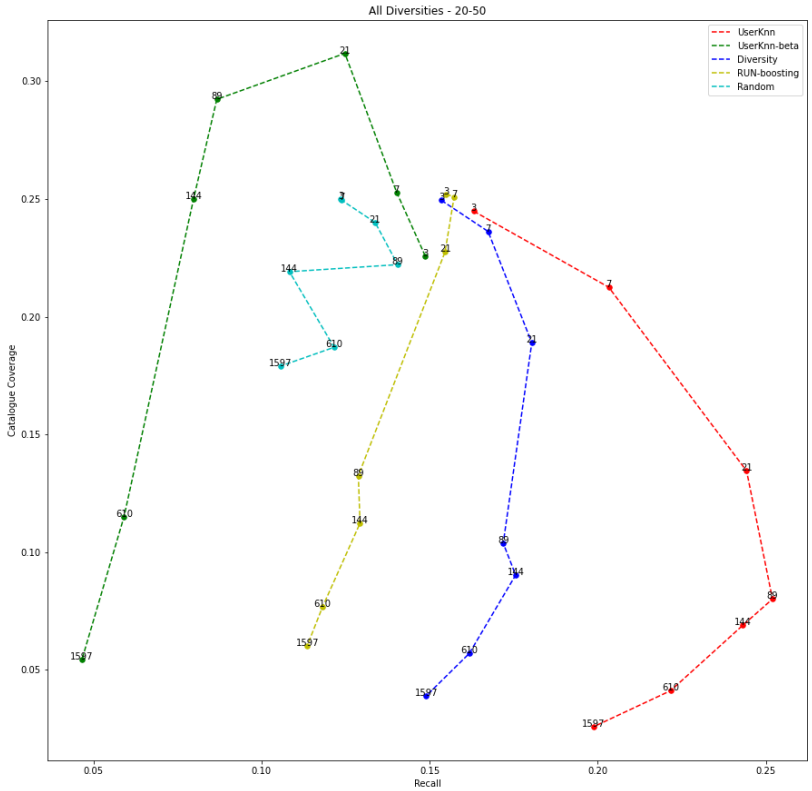

Figure 4.12: The results of all diversifications inside OBACS algorithms for the "20-50" dataset, where the X-axis show recall and Y-axis catalogue coverage

4.4.1 Results

- Fig 4.11 presents the results of all diversifications inside OBACS algorithms, where the X-axis show recall and Y-axis catalogue coverage.

In the first dataset, where users have a small amount of interaction, we can observe that the regular version of the algorithms is Pareto dominant.

In the second dataset, we can observe that the catalogue coverage of regular algorithms decreases and for the same hyperparameter K, the diversification approach starts to work.

The last three datasets show the same characteristic; the regular dataset shifts more to the recall part, while catalogue coverage is decreasing, but the diversification approach also

increases catalogue coverage.

Now we get the last dataset in fig. 4.12, where the average number of interactions is high, and shows the characteristic of each diversity.

RUN boosting as we mentioned in Section 3.2.3, increases catalogue coverage, has, compared to other diversification methods, the best recall, which is why we call it boosting, because it boosts catalogue coverage while keeping recall in the same place. Even in this method, we move the “outlier” or "unexpectedness" item to a higher position in the recommendation, but also do this based on the user profile, so we still try to be relevant to the user.

Diversity increases catalogue coverage compared to RUN boosting, but also rapidly decreases recall, compared to the original algorithms. This is because, in diversity, we compare items to each other, and we do not care about the similarity with the user profile.

Long-tail β parameter: this parameter, as mentioned [45], provides for the penalisation of the items that are similar to the user profile and k nearest neighbour and adds items that are popular in the whole catalogue. This behaviour swells our methods and pushes them to catalogue coverage maximisation. The disadvantage is that it rapidly decreases recall.

4.4.2 Discussion

This experiment proves our hypothesis that ontology-based models and ontological similarity can improve CF characteristics in cases of diversity. The goal of this experiment is not to choose the best method, but show their behaviour in a particular domain.

Each method could assist in a different case. Because all users have different preferences, we can use a different approach. For example, if a user is looking for relevant movies, we can keep the algorithm as it is. If the user is looking for relevant, but maybe older, movies, or does not find an appropriate movie for a long time, we can use RUN boosting, which could recommend a movie high in unexpectedness. If the user wants to explore the catalogue, we can use diversity, or we can turn on our beta parameter.

4.5 Summary

In this chapter, we prove our approach in the three experiments.
The first experiment shows the characteristics of our algorithms and is presented in this chapter. We use a hyperparameter for each algorithm to manipulate algorithm capacity. The best algorithm or Pareto optimal was UserKnn from the CF approach and ontology-similarity from CB.

The second experiment proves the correctness of our approach, while the combination of Similarity-based, with ontological-similarity, and UserKnn algorithms inside our OBACS algorithms reduce the cold-start problem effectively in this particular domain.

The third and last experiment shows that modification of our OBACS algorithms with reranking functionality could improve recommendation in the regular dataset, where we could help CF algorithms with diversification and provide system developers wirh tools for many use cases.

Chapter 5

Conclusions

In this thesis, we present a universal approach based on ontology for reducing the cold-start problem. We present the general approach of evaluation of testing the cold-start problem for offline *topN* recommendation task.

Because this thesis is designed for academic purposes, we use academic dataset as a proxy to evaluate algorithms designed to solve real-world tasks. We use benchmark data from MovieLens with some extension from The Movie Database and provide experiments. We have tried to describe all our approaches clearly and in-depth so that other researchers could reuse our methods.

Because we want to show that our approach could work in general, we have prepared a case study for the educational domain that we present in the appendix. Also, our OBACS algorithm is being preparing for a production version on the Experts.ai platform and could be working online from September 2020.

We have spent years attempting to figure out the challenges of the cold-start problem because we know the business need for this solution. We appreciate that finding an appropriate solution could assist both research and business, could help new ideas might emerge, and new startups could become worldwide platforms.

5.1 Summary

- The cold-start problem can be evaluated with a benchmark dataset like MovieLens with a leave-one-out cross-validation approach for offline *topN* recommendation, and Pareto optimality for choosing the best algorithm.

- An ontology-based approach could effectively reduce the cold-start problem by means of the ontological similarity of the items. Also, it provides the semantic layer for

the recommendation; this helps our recommendation to be more transparent and trustworthy.

- We designed the OBACS algorithm to help the system move through a transitional state while there are cold items in our catalogue.
- We modified the OBACS algorithm to help CF algorithms with the "filter bubble" problem, thanks to several diversification approaches such as Diversity and RUN boosting.
- We present the ontology-based solution for the educational domain in the appendix.
- Our OBACS algorithm will be used in an online production environment in Expert.ai platform from September 2020

5.2 Contributions of the Dissertation Thesis

1. We describe the cold-start problem in-depth and show its main characteristics.
2. We present the most common approaches to reducing the cold-start problem and describe the advantages and disadvantages of each method.
3. We explain how to use leave-one-out cross-validation methods to test cold-user and cold-item problems in recall, catalogue coverage and onto-similarity space.
4. We describe the process of semi-automatic developing ontology, describe how to improve ontology with implicit and explicit embeddings.
5. We present our universal ontology-based algorithms for reducing the cold-start problem (OBACS). These algorithms could effectively reduce the cold-start problem, and it also helps the system overcome the transition phase from a cold start to a stable state.
6. We show how OBACS algorithms could help CF algorithms control the "filter bubble" in regular datasets, and provide them with control over diversity for several scenarios.
7. We summarize the ideas and results of this thesis and present our future work.

5.3 Future Work

Our main goal would be the deployment of our OBACS algorithms to a production environment on the Experts.ai platform and to try to evaluate this method in an online environment with real traffic.

While working on the thesis, we found several interesting ideas to explore:

- Rising star, or the problem with the appropriate recommendation of experts in the academic domain B.1.
- Context detection, or problems with word context inside ontology B.2
- The recommendation score problem in CF methods B.3
- Using the new embedding technique to improve explicit embeddings

Bibliography

[1] Gediminas Adomavicius and Alexander Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Trans. on Knowl. and Data Eng.*, 17(6):734–749, June 2005.

[2] Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules in large databases. In *Proceedings of the 20th International Conference on Very Large Data Bases*, VLDB '94, pages 487–499, San Francisco, CA, USA, 1994. Morgan Kaufmann Publishers Inc.

[3] Mikel Artetxe and Holger Schwenk. Massively multilingual sentence embeddings for zero-shot cross-lingual transfer and beyond, 2018.

[4] Paulo J. Azevedo and Alípio M. Jorge. Comparing rule measures for predictive association rules. In *Proceedings of the 18th European Conference on Machine Learning*, ECML '07, pages 510–517, Berlin, Heidelberg, 2007. Springer-Verlag.

[5] J. Bobadilla, F. Ortega, A. Hernando, and A. GutiéRrez. Recommender systems survey. *Know.-Based Syst.*, 46:109–132, July 2013.

[6] E. Campochiaro, R. Casatta, P. Cremonesi, and R. Turrin. Do metrics make recommender algorithms? In *Advanced Information Networking and Applications Workshops, 2009. WAINA '09. International Conference on*, pages 648–653, May 2009.

[7] Paolo Cremonesi, Yehuda Koren, and Roberto Turrin. Performance of recommender algorithms on top-n recommendation tasks. In *Proceedings of the Fourth ACM Conference on Recommender Systems*, RecSys '10, pages 39–46, New York, NY, USA, 2010. ACM.

[8] Gerard Deepak and Dheera Kasaraneni. Ontocommerce: an ontology focused semantic framework for personalised product recommendation for user targeted e-commerce. *International Journal of Computer Aided Engineering and Technology*, 11(4-5):449–466, 2019.

[9] M. Einasto, L. J. Liivamägi, E. Saar, J. Einasto, E. Tempel, E. Tago, and V. J. Martínez. Sdss dr7 superclusters. *Astronomy and Astrophysics*, 535:A36, Oct 2011.

[10] Crícia Z. Felício, Klérisson V.R. Paixão, Celia A.Z. Barcelos, and Philippe Preux. A multi-armed bandit model selection for cold-start user recommendation. In *Proceedings of the 25th Conference on User Modeling, Adaptation and Personalization*, UMAP '17, page 32–40, New York, NY, USA, 2017. Association for Computing Machinery.

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

[12] N. Guarino. *Formal Ontology in Information Systems: Proceedings of the 1st International Conference June 6-8, 1998, Trento, Italy.* IOS Press, Amsterdam, The Netherlands, The Netherlands, 1st edition, 1998.

[13] Rahul Gupta, Alon Halevy, Xuezhi Wang, Steven Whang, and Fei Wu. Biperpedia: An ontology for search applications. In *Proc. 40th Int'l Conf. on Very Large Data Bases (PVLDB)*, 2014.

[14] Sonal. Gupta. *Distantly supervised information extraction using bootstrapped patterns.* PhD thesis, Stanford University, Dept. of Computer Science., https://purl.stanford.edu/nt508qx3506, 2015.

[15] Sonal Gupta and Christopher D. Manning. Improved pattern learning for bootstrapped entity extraction. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning (CoNLL)*, 2014.

[16] Jiawei Han, Jian Pei, and Yiwen Yin. Mining frequent patterns without candidate generation. *SIGMOD Rec.*, 29(2):1–12, May 2000.

[17] F. Maxwell Harper and Joseph A. Konstan. The movielens datasets: History and context. *ACM Trans. Interact. Intell. Syst.*, 5(4), December 2015.

[18] G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

[19] Wei-Hao Hwang, Yeong-Sheng Chen, and Tsang-Ming Jiang. Personalized internet advertisement recommendation service based on keyword similarity. In *Computer Software and Applications Conference (COMPSAC), 2015 IEEE 39th Annual*, volume 1, pages 29–33, July 2015.

[20] Leo Iaquinta, Marco de Gemmis, Pasquale Lops, Giovanni Semeraro, and Piero Molino. Can a recommender system induce serendipitous encounters. In *WWW*, 2010.

[21] Dietmar Jannach, Markus Zanker, Alexander Felfernig, and Gerhard Friedrich. *Recommender Systems: An Introduction.* Cambridge University Press, New York, NY, USA, 1st edition, 2010.

[22] Gawesh Jawaheer, Martin Szomszor, and Patty Kostkova. Comparison of implicit and explicit feedback from an online music recommendation service. In *Proceedings of the 1st International Workshop on Information Heterogeneity and Fusion in Recommender Systems*, HetRec '10, pages 47–51, New York, NY, USA, 2010. ACM.

[23] Daniel Jurafsky and James H. Martin. *Speech and Language Processing (2nd Edition)*. Prentice-Hall, Inc., USA, 2009.

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

[25] Denis Kotkov, Shuaiqiang Wang, and Jari Veijalainen. A survey of serendipity in recommender systems. *Knowledge-Based Systems*, 111:180 – 192, 2016.

[26] S. Kuznetsov, P. Kordík, T. Řehořek, J. Dvořák, and P. Kroha. Reducing cold start problems in educational recommender systems. In *2016 International Joint Conference on Neural Networks (IJCNN)*, pages 3143–3149, July 2016.

[27] Philip Lenhart and Daniel Herzog. Combining content-based and collaborative filtering for personalized sports news recommendations. In *CBRecSys@RecSys*, 2016.

[28] Yi Li, Lin Mei, and Jian Wang. A personalized recommendation system in e-learning environment based on semantic analysis. In *Information Science and Service Science and Data Mining (ISSDM), 2012 6th International Conference on New Trends in*, pages 802–807, Oct 2012.

[29] Pasquale Lops, Marco de Gemmis, and Giovanni Semeraro. Content-based recommender systems: State of the art and trends. In Francesco Ricci, Lior Rokach, Bracha Shapira, and Paul B. Kantor, editors, *Recommender Systems Handbook*, pages 73–105. Springer US, 2011.

[30] Hexin Lv and Bin Zhu. Skill ontology-based semantic model and its matching algorithm. In *Computer-Aided Industrial Design and Conceptual Design, 2006. CAIDCD '06. 7th International Conference on*, pages 1–4, Nov 2006.

[31] Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. The stanford corenlp natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60, 2014.

[32] Stuart E. Middleton, Harith Alani, Nigel Shadbolt, and David De Roure. Exploiting synergy between ontologies and recommender systems. In *Proceedings of the WWW2002 International Workshop on the Semantic Web, Hawaii, May 7, 2002*, 2002.

[33] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space, 2013.

[34] Tien T. Nguyen, Pik-Mai Hui, F. M. Harper, L. Terveen, and J. Konstan. Exploring the filter bubble: the effect of using recommender systems on content diversity. In *WWW*, 2014.

[35] Jeffrey Pennington, Richard Socher, and Christopher Manning. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar, October 2014. Association for Computational Linguistics.

[36] F.A. Procopio de Paiva, J.A. Ferreira Costa, and C. Rodrigues Muniz Silva. A hierarchical architecture for ontology-based recommender systems. In *Computational Intelligence and 11th Brazilian Congress on Computational Intelligence (BRICS-CCI CBIC), 2013 BRICS Congress on*, pages 362–367, Sept 2013.

[37] Achim Rettinger, Uta Lösch, Volker Tresp, Claudia d'Amato, and Nicola Fanizzi. Mining the semantic web. *Data Mining and Knowledge Discovery*, 24(3):613–662, 2012.

[38] Miguel Angel Rodriguez-Garcia, Rafael Valencia-Garcia, Ricardo Colomo-Palacios, and Juan Miguel Gomez-Berbis. Blinddate recommender: A context-aware ontology-based dating recommendation platform. *Journal of Information Science*, 45(5):573–591, 2019.

[39] Stuart Rose, Dave Engel, Nick Cramer, and Wendy Cowley. *Automatic Keyword Extraction from Individual Documents*, chapter 1, pages 1–20. John Wiley and Sons, Ltd, 2010.

[40] H. Safaeipour, M.H. Fazel Zarandi, and S. Bastani. Mapping crisp structural semantic similarity measures to fuzzy context: A generic approach. *Int. J. Fuzzy Syst.*, 22:1224–1242, 2020.

[41] Holger Schwenk. Zero-shot transfer across 93 languages: Open-sourcing enhanced laser library, jan 2019.

[42] Suvash Sedhain, Aditya Krishna Menon, Scott Sanner, and Lexing Xie. Autorec: Autoencoders meet collaborative filtering. In *Proceedings of the 24th International Conference on World Wide Web*, WWW '15 Companion, page 111–112, New York, NY, USA, 2015. Association for Computing Machinery.

[43] Guy Shani and Asela Gunawardana. *Evaluating Recommendation Systems*, chapter 8. Springer, 2011.

[44] Harald Steck. Training and testing of recommender systems on data missing not at random. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '10, pages 713–722, New York, NY, USA, 2010. ACM.

[45] Harald Steck. Item popularity and recommendation accuracy. In *Proceedings of the Fifth ACM Conference on Recommender Systems*, RecSys '11, pages 125–132, New York, NY, USA, 2011. ACM.

[46] Chong Wang and David M. Blei. Collaborative topic modeling for recommending scientific articles. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '11, pages 448–456, New York, NY, USA, 2011. ACM.

[47] Ian H. Witten. *The Practical Handbook of Internet Computing*, chapter Text Mining. CRC Press, Inc, 2004. ISBN 1584883812.

[48] Jun Zhou and Keiichi Watanuki. Skill ontology for mechanical design of learning contents. *Journal of Software*, 7(1):61–67, 2012.

[49] Tomáš Řehořek. *Manipulating the Capacity of Recommendation Models in Recall-Coverage Optimization.* PhD thesis, Faculty of Information Technology, Czech Technical University in Prague, 2018.

Reviewed Publications of the Author Relevant to the Thesis

[A.1] Kordík,Pavel and Kuznetsov, Stanislav *Mining Skills from Educational Data for Project Recommendations.* International Joint Conference - CISIS'15 and ICEUTE'15 p. 617-627 Switzerland: Springer International Publishing, 2015/ Álvaro Herro. ISBN: 978-3-319-19712-8 ISSN: 2194-5357

The paper has been cited in:

- Hanan Aldowah and Hosam Al-Samarraie and Wan Mohamad Fauzy, "Educational data mining and learning analytics for 21st century higher education: A review and synthesis" *Telematics and Informatics 2019.* vol 37 pp. 13-49, 2019. *doi = "https://doi.org/10.1016/j.tele.2019.01.007"*

[A.2] S. Kuznetsov and P. Kordík and T. Řehořek and J. Dvořák and P. Kroha *Reducing cold start problems in educational recommender systems.* 2016 International Joint Conference on Neural Networks (IJCNN) p. 3143-3149 Vancouver, BC, Canada, 2016 IEEE. ISBN: 978-1-5090-0620-5 ISSN: 2161-4407 DOI: 10.1109/IJCNN.2016.7727600

- Izzah Fadhilah Akmaliah, Adila Alfa Krisnadhi, Dana Indra Sensuse, Puji Rahayu, Ika Arthalia Wulandari, "Role of Ontology and Machine Learning in Recommender Systems" *Electrical Power Electronics Communications Controls and Informatics Seminar (EECCIS) 2018.* pp. 371-376, 2018.

- Carla Barvinski, Gislaine Ferreira, Leticia Machado, Magali Longhi, Patricia Behar *Smart Education and e-Learning 2019.* vol. 144, pp. 159, 2019.

Appendix A

Education Data Mining Case Study

A.1 Education Data Mining Case Study

A.2 Data Preprocessing

Before we can build ontologies, we must collect data and carry out preprocessing. The basic idea of the entire system is not only the resulting recommender algorithm but primarily the creation of the entire process from the acquisition of data through their processing to their subsequent use. This process should be maximally automated and should use only minimal human intervention. It should be further divided into independent components so that we can always replace any component. The final process for a particular case study could be different, but the general best practise for it is as follows:

- Carefully track data quality.
- Use a data warehouse approach for storing data rather than a conventional database.
- Maximum effort made to clear data because the final datasets are always big and very sparse.
- Try to implement a mechanism for clearing data by end-users.

The process consists of extracting accreditation materials from a data warehouse and their automatic classification into a set of keywords, acronyms, and text. Text mining is performed next, thanks to which we will obtain proposed skills. Additionally, we created a website to collect information from teachers, where each teacher indicates what skills belong to a subject based on the proposed skills. Finally, this information will be written into the matrix from which the ontology is generated. Besides a list of skills that cover a given subject, we assigned a value to each skill (from 0 to 1); this is the level to which the given subject covers a specific skill.

Example A.2 shows the mapping of subjects with code MI-PDD [1] to skills.

- MI-PDD:
  - Datamining weight = 1.0
  - Mathematics weight = 0.2
  - Database weight = 0.4
  - ...

A.2.0.1 OntoSkill tree

Our ontology represents a simplified "skill tree" inspired by the ACM tree. We simplified the tree by removing skills that are not taught by the faculty. We did not want the tree to be excessively deep (maximum of two levels) so that clients (people without experience in the education system) can easily understand it. The higher levels of the tree contain skills that represent abstract levels of knowledge (centroid) and which are made up of more than one specific skill. Around each of these skills, we have a cluster of specific skills. The representation of a skill tree is shown in Figure A.1. The advantage of this ontology is the fact that we can work with skills at different levels of abstraction thanks to a precise hierarchy. With this feature, we can easily display only the first level of skills in the profiles for users, thus making the profiles well arranged. The biggest drawback is that the tree structure is not flexible and does not allow us to combine the skills directly, thus creating a problem with the fact that some skills can belong to multiple sub-trees (i.e. security belongs in many areas). Another disadvantage is the fact that the decision about the superiority of one skill over the other cannot be determined automatically, it must be decided by a specialist in this area, thereby greatly slowing down development.

A.2.1 OntoSkill graph

This ontology is presented by an incoherent graph, with the nodes of the graph consisting of individual skills. The edge shows the relationship between two nodes. Each edge has a weight that indicates the level of connection between skills. Weight calculation is done so that if, e.g. the skills such as "matlab" [2] and "svm" [3] are taught in the subject of "MI-PDD", providing that "matlab" be paid a lot of attention to within the subject (coefficient 0.8) and "svm" only a little (coefficient 0.4), then the weight between the two skills is calculated as the product of both the coefficients. The total weight is the sum of these products. Figure A.2 illustrates the cornerstone of ontology, with the green node and its arrows indicating a connection of the subject with skills, the blue ones are nodes, edges and calculated weights of the ontology. Figure A.3 illustrates the whole generated ontology. Figure A.4 illustrates

[1] Data Preprocessing for master's degree
[2] http://www.mathworks.com/products/matlab/
[3] Support vector machines

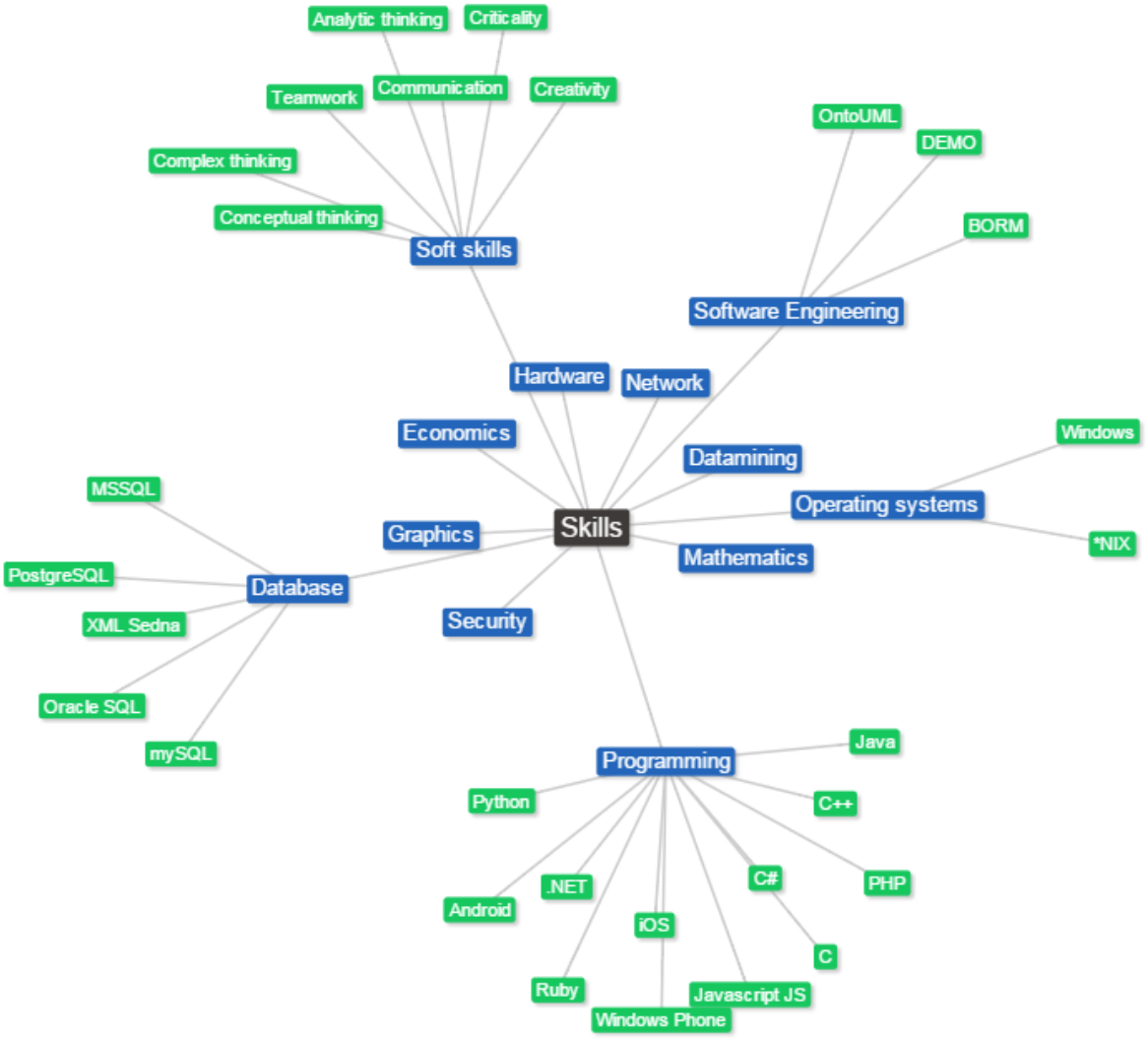

Figure A.1: Skills are represented by the tree, so higher level nodes generalise skill of leaf nodes.

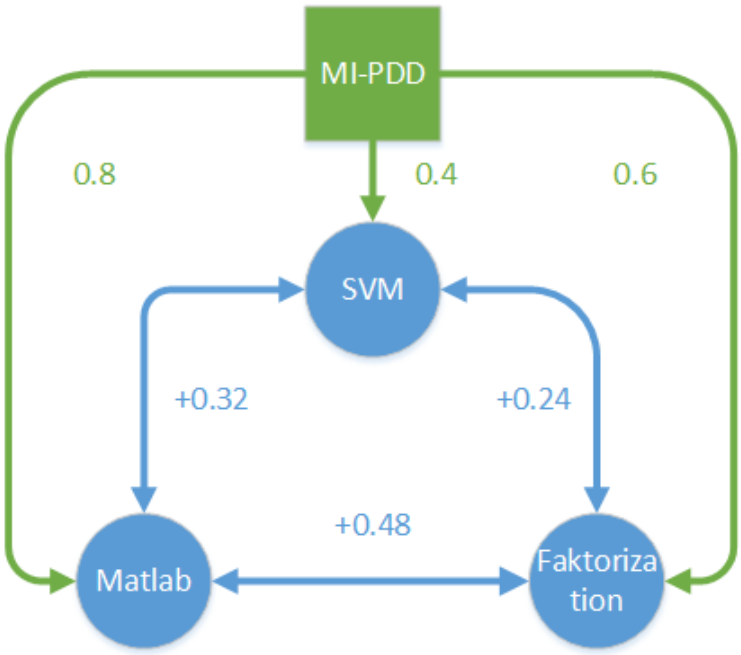

Figure A.2: The principle of ontology based on the graph.

one part of the ontology in more detailed form. The main advantage of this ontology is its flexibility, with each skill representing one node with any number of connections with other skills. Another advantage is that the ontology is generated completely automatically, and the relationships are generated solely from the skill matrix. Furthermore, the weights may change over time, new connections can be added, created by other information (see the section Future Works). The final advantage is that we can use the skill itself as well as its surroundings for recommendation thanks to these connections. For example, if I wanted to find a student who is mastering C++ but do not have one, I can recommend a student who has different characteristic associated with the search criterion, e.g. C#.

The disadvantage is that the ontology consists of several hundreds of skills which are not suitable for visualizing the final profile. Therefore, it is necessary to perform clustering with the selection of a representative node that will represent this area and will be subsequently presented to a user over the ontology; the process is not trivial.

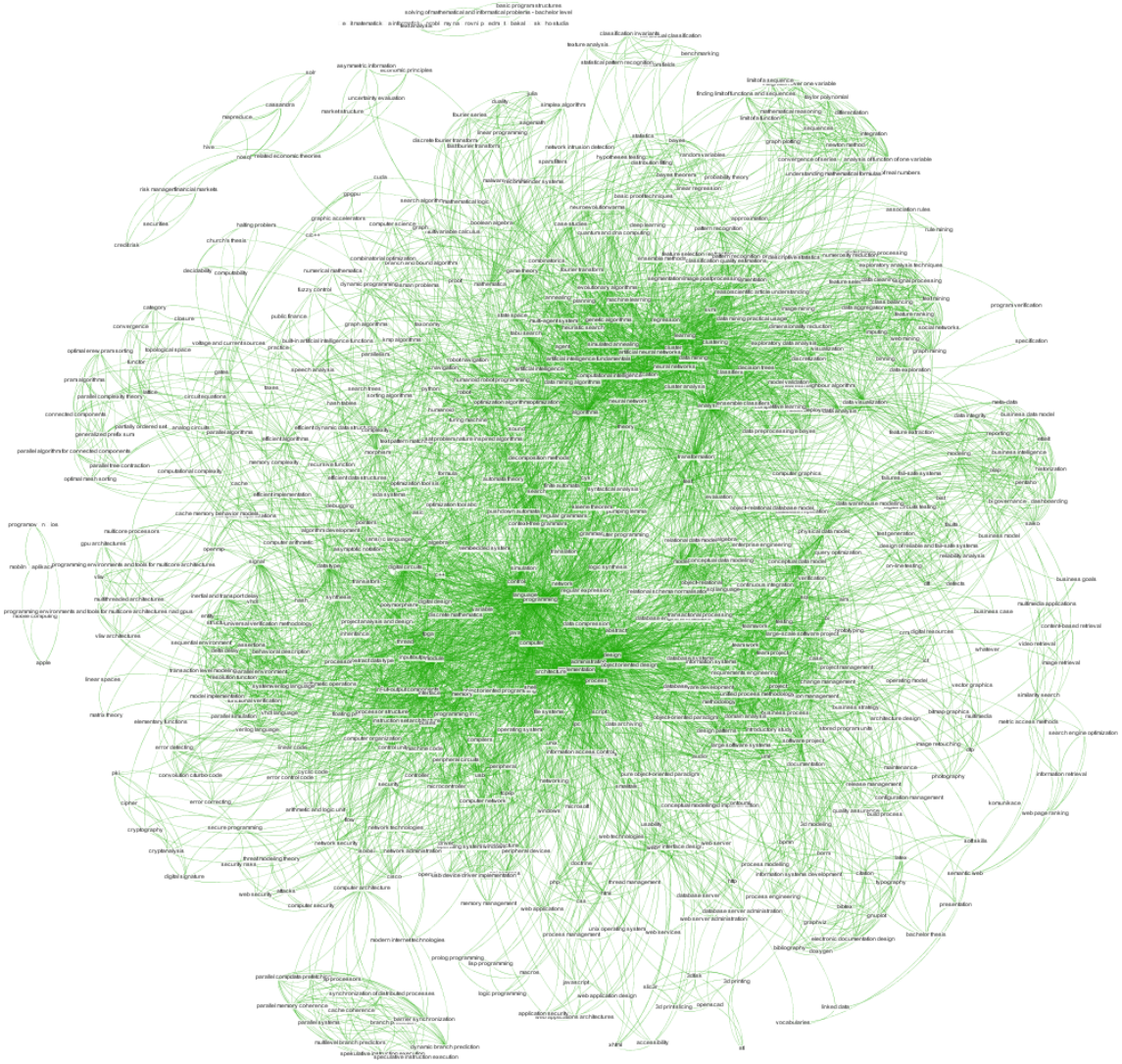

Figure A.3: Ontology based on a graph that we generated based on lecturer knowledge at the Faculty of Information Technology, CTU.

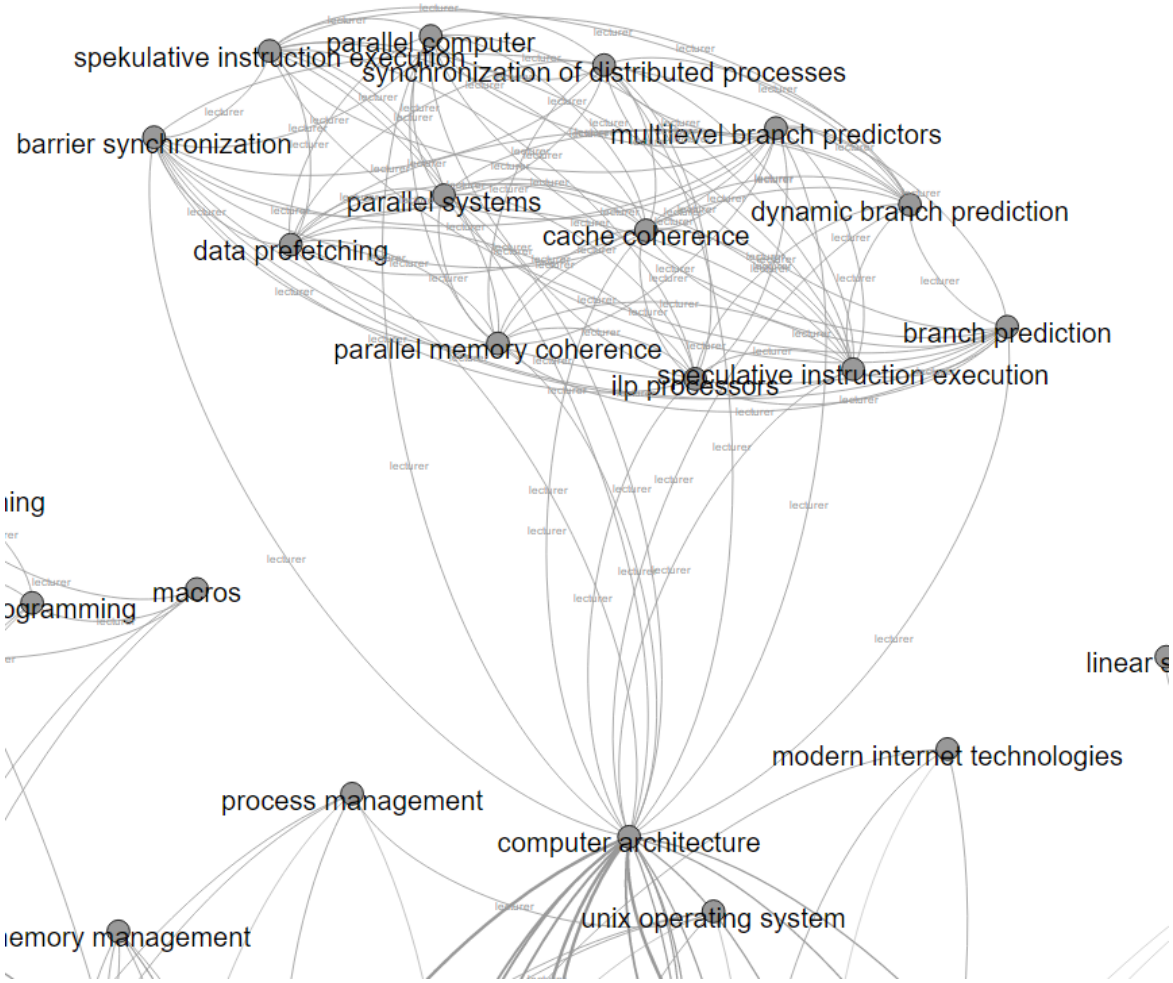

Figure A.4: Ontology based on a graph that we generated based on lecturer knowledge in more detail.

A.2.2 Computation of student profiles

First, we created a process that would be able to generate a student's profile based on the student's study results. Then we compare these profiles against each other and recommend suitable students. The final process contains three phases.

Score Calculation Phase In this phase, we work with the grades of the student using algorithms; we transfer them into absolute values which we call scores. All calculations made are done individually for each student, i.e. no comparison is made.

Comparison and Normalisation Phase This phase compares the students and then calculates the number of stars using a normalisation function. The faculty has two main groups of students – bachelor's degree students and master's degree students; we call these programs. The algorithms compare two students in the same programs of study and the same year.

Practical applications In the last phase of our process, we assign students to an assignment. Any assignment may contain a random number of positions, e.g. team researcher, database analyst, C++ programmer, etc. The client may select the necessary amount and level of skills for each position. We are then able to calculate the most suitable students based on these values.

A.2.3 Application Domain - SSP Portal

The Student Cooperation Portal (SSP) is an information system designed and developed by the Faculty of Information Technology, CTU, Prague. This portal provides a platform for industrial partners (referred to as "clients"), who input their projects into the system and for students, who look through these projects and apply if interested.

Because a large number of students are involved in the portal, it is critical to recommend the "correct" students for positions in projects. This approach also applies vice versa: it is necessary to offer the projects to the right students. The calculated skills also enable us to generate a student's CV and their profile specialisation automatically.

A.2.4 Recommendation Models

In our experiments, we use several algorithms. We briefly describe them in this section.

A.2.5 k-Nearest Neighbours Algorithm

We use a user-based k-Nearest Neighbours algorithm with cosine similarity and voting. In [7], such an algorithm is referred to as the *Popularity-Stratified, Non-normalised Cosine Neighbourhood*, which we will, for simplicity, keep referring to as the *User-kNN* throughout

the rest of this case study. To rank items in a user's neighbourhood, the following formula is used:

$$\text{rank}(u,i) = \frac{\sum_{\hat{u}\in N^k(u)} \text{sim}(u,\hat{u}) \cdot (r_{\hat{u},i} - \overline{r_{\hat{u}}})}{\left(\sum_{u\in\mathcal{U}}(r_{\hat{u},i} - \overline{r_{\hat{u}}})\right)^{\beta}}, \tag{A.1}$$

where $N^k(u)$ in the set of the k nearest neighbours, $\mathcal{U}$ is the set of all the users in the database, and β is the long-tail biasing parameter as proposed in [45].

A.2.5.1 Association Rules

Other algorithms are Association Rules (AR) which allow us to construct pattern-driven recommendations. *AR* models are largely different from neighbourhood-based approaches, allowing us to explore the behaviour of multiple algorithms in the Recall-Coverage state space.

Given $(\mathcal{I},\mathcal{U}, s_{min})$, we are able to find a set of association rules with minimal support. An association rule is an implication $X \Rightarrow Y$ such that $X \cup Y \in \mathcal{U}$, $X \neq \emptyset$, $Y \neq \emptyset$, $X \cap Y = \emptyset$. We say that association rule $X \Rightarrow Y$ holds the *minimal support iff*

$$\frac{|\{T \in \mathcal{U} \,|\, X \cup Y \in T\}|}{|\mathcal{U}|} \geq s_{min},$$

i.e. we require at least $s_{min} \cdot 100$ % of users to contain $X \cup Y$ as relevant items in their rating history.

One of the most popular algorithms to solve the AR-mining task is the APRIORI algorithm also proposed in [2]. APRIORI first searches for frequent itemsets in a bottom-up manner. From the frequent itemsets, association rules are generated exhaustively. Since the original publication of APRIORI, many other algorithms have been proposed to mine association rules more efficiently, such as FP-growth [16], and many others. In our experiments, we use a modified, lazy version of the APRIORI algorithm, which finds the set of rules at the time of recommendation, pruning the search space of frequent itemsets on a test user

In our case study, we use the β parameter to provide a smooth transition from confidence to lift based on this formula:

$$\mathit{lift}(X \Rightarrow Y) = \frac{\mathit{conf}(X \Rightarrow Y)}{\mathit{supp}(Y)^{\beta}}, \tag{A.2}$$

we obtain the *confidence* for $\beta = 0.0$, and the *lift* for $\beta = 1.0$.

A more thorough description about *confidence* and *lift* can be found in [4] [2].

A.2.5.2 Ontology best seller

The last algorithm is an ontology best seller. This algorithm is based on our ontology and has two parts. The first part is computed by the Euclidean distance between the vector of assignment skill and the vector of student skills (computed profile). If $\vec{v_1}$ is the vector of

assignment skill and $\vec{v_2}$ is the vector of student skills, then the Euclidean distance is given by the Pythagorean formula:

$$d(\vec{v_1}, \vec{v_2}) = \sqrt{\sum_{i=1}^{n} (\vec{v_{1i}} - \vec{v_{2i}})^2} \tag{A.3}$$

This distance is computed for all students and assignments in the system. The second parameter is the number of unique visitation of students from a web page with assignment description; we call it $visit(x)$. For final rank, we normalise both parameters and use this formula:

$$rank(u, a) = (1 - d_i)^{\gamma} \cdot visit(i)^{1-\gamma} \tag{A.4}$$

Where u is a user, a is an assignment and γ is the parameter designed for weight distribution between these two components. Note that we subtract the d value from one, because we want to maximise the rank, but the Euclidean distance works in reversal. The more similar vectors are, the less the distance.

A.2.6 Offline Experiments

In this section, we present offline experiments on two datasets. All datasets are derived from the SSP [4] database, which contains approximately 1,000 students, 300 assignments and more than 7,000 interactions (assignment page visits). In our portal, we do not have heavy traffic for online experiments, so we decided to use offline experiments and analyse the behaviour of students over a longer period (one and a half years). All graphs which will be introduced have the x-value shifted to better illustrate the course of the curves.

A.2.6.1 Metrics

There are several ways to simulate user behaviour in a system [43]. The most common method is to use historical data in which some user interactions are hidden and then used to make an RS predict their value. Ideally, a timestamp interaction is used, based on which we can simulate the system behaviour over time. The most commonly used procedure for large datasets is one in which we randomly select test users, randomly choose a time for these users, hide all the interactions of these users from this time, and try to recommend items for those users. The problem with this approach is that it needs to perform the entire process before each recommendation, which is not very efficient in terms of performance.

Since our RS always recommends Top-N items (N = 5) for each user, a leave-one-out cross-validation methodology [6] seems like a good compromise for testing the RS. Within this method, cross-validation is carried out across all users in the system.

For model comparisons, we use recall and coverage [5] (also called catalogue-coverage) metrics. We chose these metrics because users prefer more diversified recommendation

[4] The Student Cooperation Portal

[5] This metric shows whether a system can recommend variously, i.e. if it can offer other items than best sellers.

with greater coverage. This is why we optimise both metrics. The exact description of the methodology and discussion of the selection of these metrics can be found in [6].

A.2.6.2 Leave-one-out cross-validation methodology

We will construct an interaction matrix for all users of the RS and all items, where each row will be a user and each column an item. A single cell of the matrix will contain the number of interactions between an item and a user. If no interaction has taken place, the cell is filled out with zero. During each step of the cross-validation, we will assign 90% of the users to a training set and the remaining 10% to a test set. For each user, we will always choose the top five items (inputs). The following procedure is divided into steps:

1. We will teach a model on training data for each cross-validation step.
2. We will hide user interaction from the system and let the top five recommendations be generated for each user from the test set and each item (matrix cell).
3. If the current item is between the recommended items while finding out that the value in a cell is greater than zero [6], the recall value of a single test is one. Otherwise, it is zero.
4. After we go through all matrix cells, we will calculate the total cross-validation step recall by dividing the number of successful tests by the number of all tests (all matrix cells). The total coverage is thus the number of unique items to all recommended items.
5. Then, we move on to the next step of cross-validation, i.e. to point 1.
6. After completing cross-validation, we can count the average recall and the average coverage. This procedure appears to be optimal since it allows us to calculate the average recall across all tested users as well as to solve the precision-recall dilemma described in [44].

A.2.6.3 Datasets

There are two datasets. The first dataset contains student data that entered the portal and made exactly one interaction; we call this the "cold-start dataset". These students are new, so we do not have any information regarding their interactions. This means that we can use neither the *User-kNN* nor the *Association rules* algorithms because of the cold-start problem described in Section 2.1.2. In practice, this means that, if an algorithm fails to make a recommendation, instead of the empty set, the system returns the *Top N* best sellers. In our case, the *Top N* most visited assignments where emphN is the usual number of recommendations.

[6] If we have a system with a large number of interactions we can set a certain threshold, e.g. greater than 5

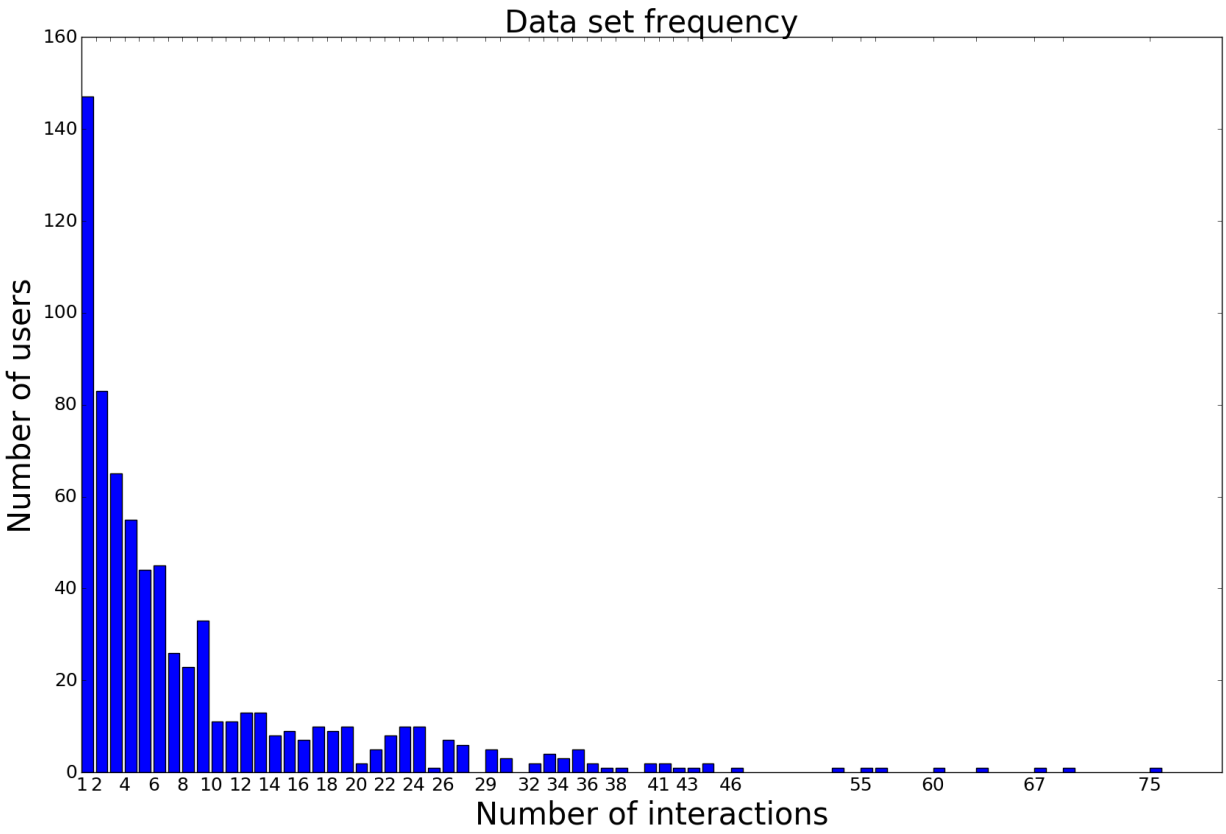

Figure A.5: The frequency of the regular dataset. Each value on the x-axis represents the class with number of interactions. The y-axis shows the number of users in each class.

The second dataset is the "regular dataset". This dataset contains all data and includes previous dataset so that we can demonstrate the behaviour of all models described in Section refsec:recom. Fig. A.5 shows the frequency of the regular dataset.

A.2.6.4 Cold-Start dataset results

As shown in Fig. A.6, the *best seller* model is *Pareto dominated* by the *Onto-best* model. The highest recall and coverage are obtained for the *Onto-best* model with $\gamma = 0.47$. That means that component, ontology weight, and most visited weight are important for the algorithm. The *best seller* model represented by a dot in a figure does not have the parameters to influence model settings. The best results are summarised in Table A.1.

A.2.6.5 Regular dataset results

As shown in Fig. A.8, for the AR model without the β term, the highest recall is obtained for the $s_{min} = 0.0141$. The lower value of s_{min} has negative impact on the recall; the higher

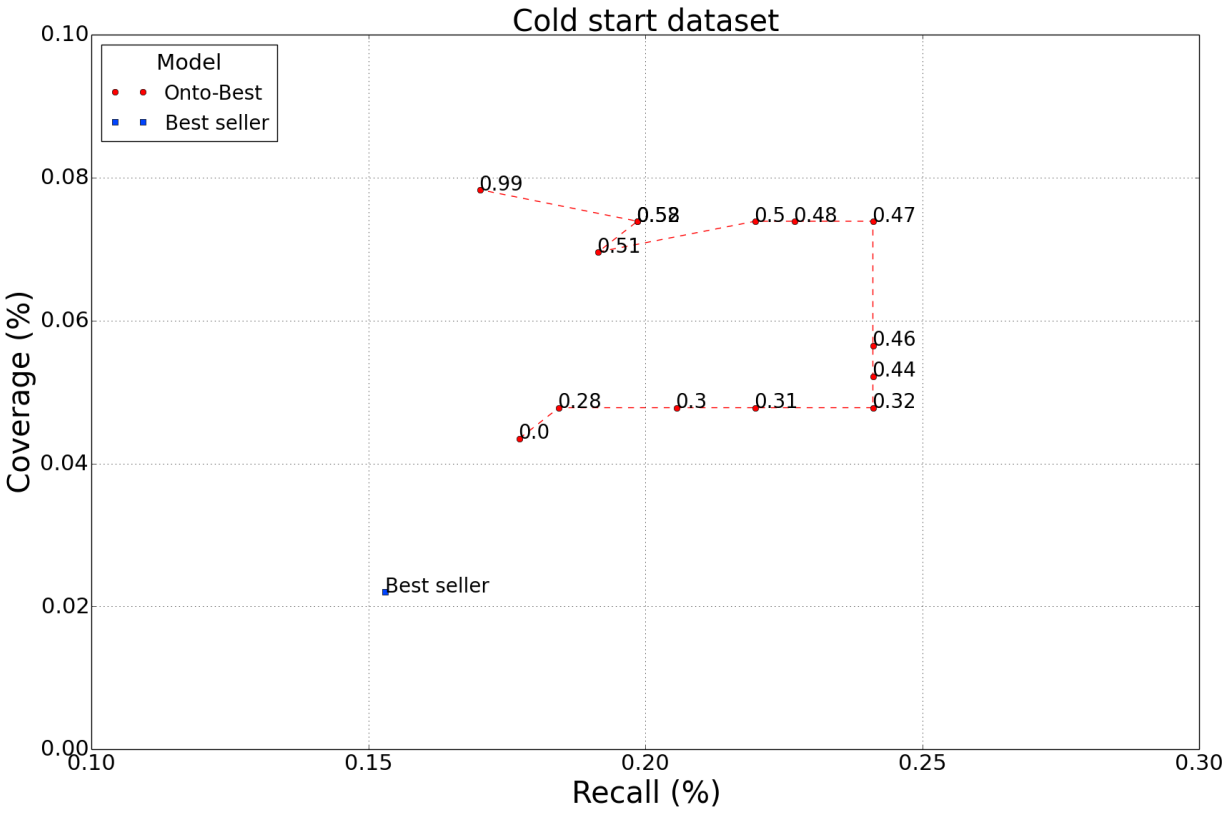

Figure A.6: Results for the *Onto-best* model and *best seller* model on the cold-start dataset. The *best seller* model has no parameters; it is represented by a single dot. On the other hand, the Onto-best model has a γ parameter; on the chart, it is represented by a curve that shows the behaviour of the model, depending on the γ setting.

Table A.1: Summary of the results for *Onto-best* and *best seller* models on cold-start dataset.

model	recall	coverage
Onto-best	24.11%	7.39%
Best seller	15.33%	2.17%

Table A.2: Summary of the results for all models on the regular dataset.

model	recall	coverage
User-kNN	30.03%	46.13%
Association Rules	27.83%	45.20%
Onto-best	14.32%	4.78%
Best seller	13.7%	2.2%

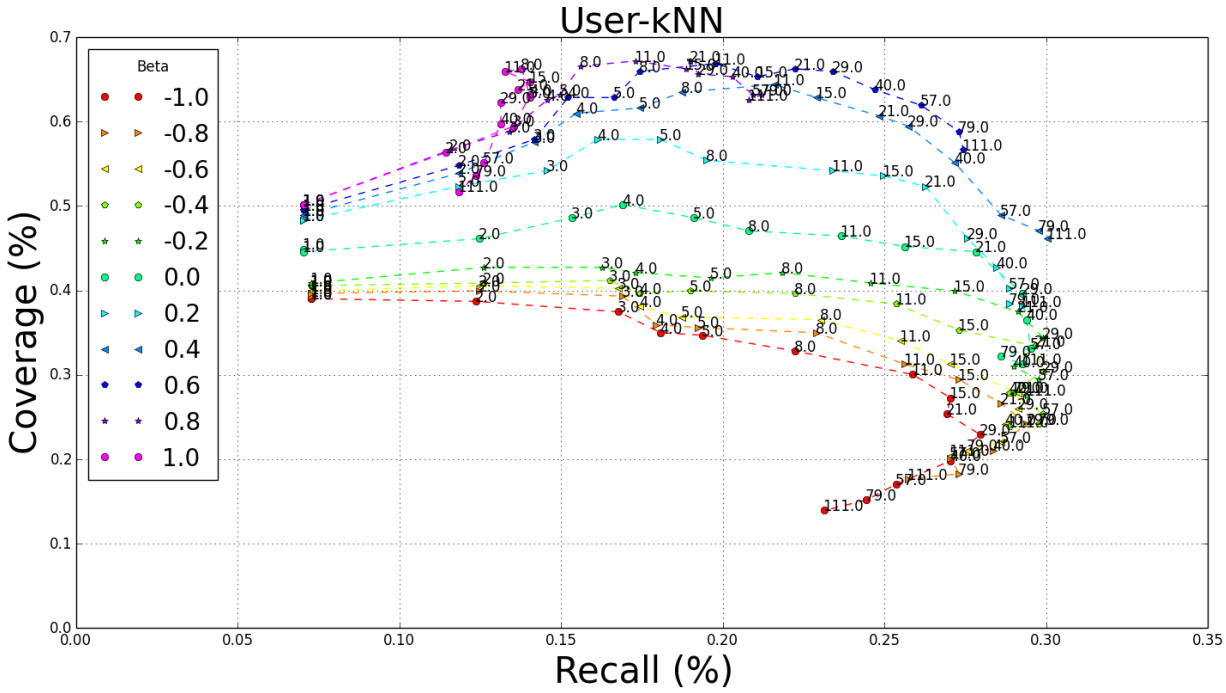

Figure A.7: Results for the *User-kNN* model on the regular dataset. Each curve on the chart represents the setting of the β value. The values on the curves are the settings of the k parameter.

value has negative impact on the coverage. As with the previous algorithm, β provides plasticity for the model. Recall and coverage are maximised for $s_{min} = 0.0001$ with $\beta = 0.6$.

As shown in Fig. A.9, for the *Onto-best* model, the higher γ term deteriorates recall in the same way as a lower value. The golden mean seems to be a value from 0.33 to 0.47, where we have higher recall and higher coverage. This alludes to the fact that students are interested in both kinds of assignments, the most popular (best seller) ones and the most appropriate ones for their profile.

Fig. A.10 shows the comparisons of all models in their best configuration. The *Pareto optimal* model is a *User-kNN* model that dominates all models. Table A.2 summarises the best results.

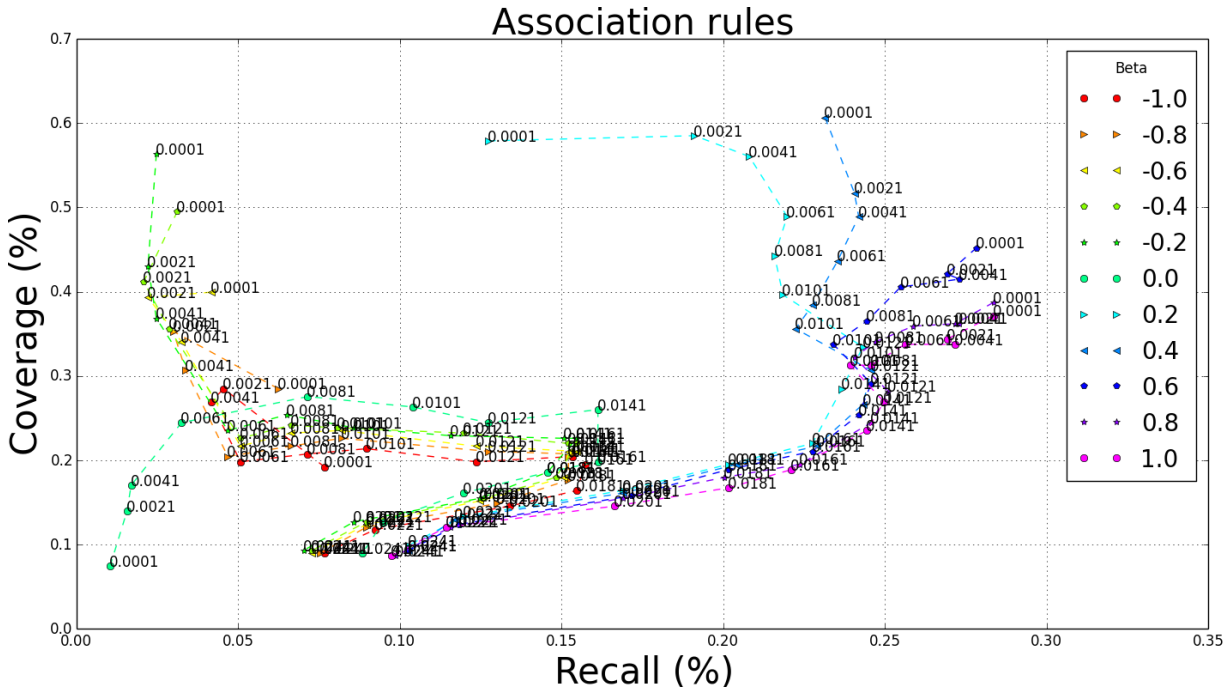

Figure A.8: Results for the *association rules* model on the regular dataset. Each curve on the chart represents the setting of the β value. The values on the curves are the settings of the s_{min} parameter.

A.2.7 Discussion

This case study presents processes that transfers final student grades into scores and then the transformation of these scores into levels of skills (stars).

We introduced a total of three algorithms as described in Section A.2.4. Measurement was carried out using parameters such as k (number of nearest neighbours), β (popularity biasing parameter), s_{min} (minimal support for all algorithms), and γ (ratio between ontology weight and visited weight).

The biggest issue we have to solve is the sparsity of the data or the small number of interactions. This problem is not related to the popularity of portals but to their use. Students use a portal irregularly after they find an appropriate assignment and have no need to visit the portal. We supposed that we could improve this by adding a semantic layer to the descriptions of student interactions. Ontology could represent this semantic layer. Experiments on the cold-start dataset show that the use of ontology and personal profile increase both recall and coverage.

The best algorithm in the regular dataset is *User-kNN* with parameter $k = 79$ and $\beta = 0.4$. This algorithm searches for similarities between users based on their interaction with the input. The reason for this algorithm being the best is probably that the system has more users, i.e. finding similarities between them is simpler than in the case of the input.

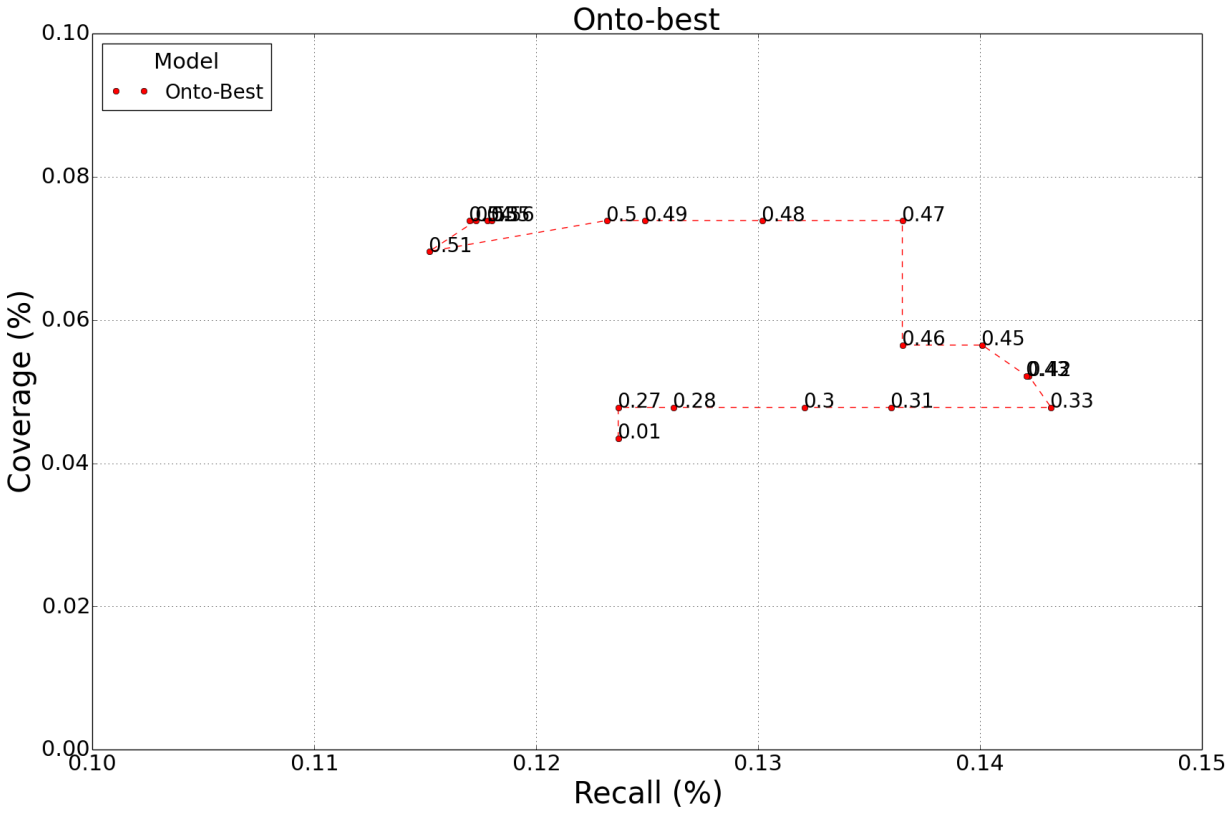

Figure A.9: Results for the *Onto-best model* on the regular dataset. The curve on the graph represents the behaviour of the model depending on the γ parameter.

The reason why the AR did not work the best is that the matrix between all students and all inputs is very sparse, making patterns of behaviour hard to find.

A.2.8 Conclusion

This case study shows how student skills ontology can reduce the cold-start problem for educational recommender systems. Student profiles are created using their results in university courses.

Our task is to recommend students for specific industrial projects and positions.

We employed collaborative filtering algorithms and rule-based recommendations from historical user interactions in our systems. We solved the cold-start problem by profile-based recommendations and showed that this approach outperforms best seller recommendations. Extensive experiments with parameters of the recommender system show that the best recommendations in terms of maximising recall and coverage are those produced by neighbourhood-based collaborative filtering. User profiles and skill ontology improve recommendations for cold-start users.

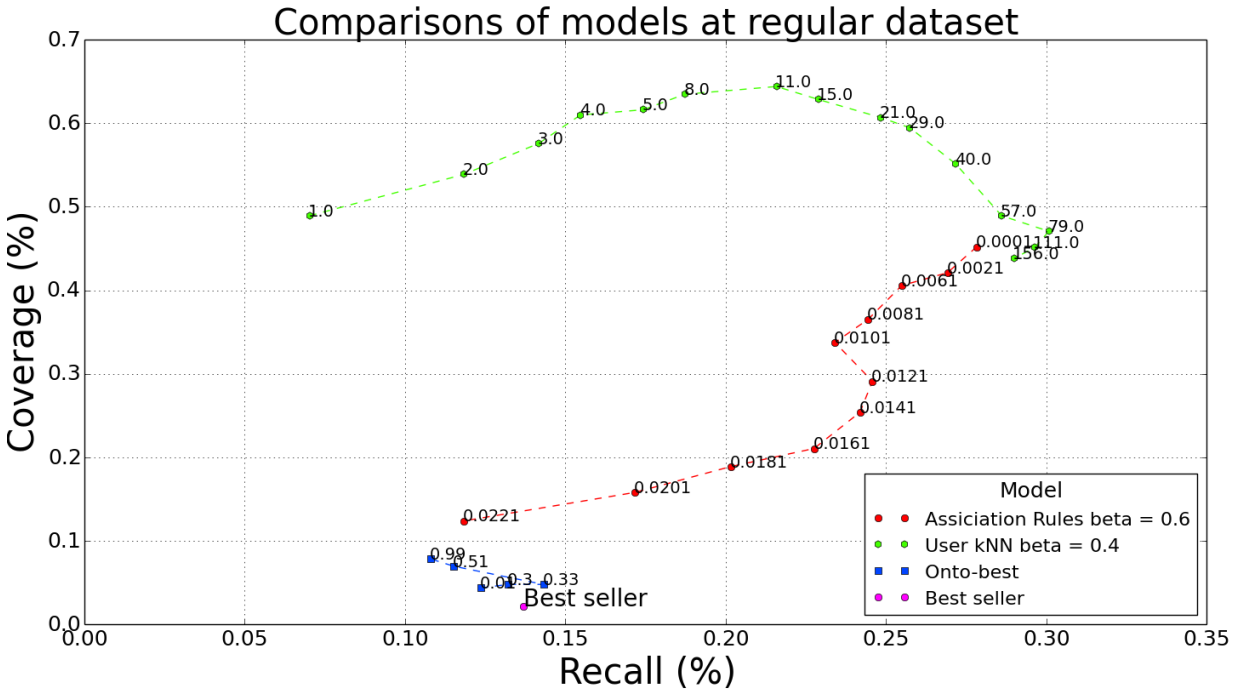

Figure A.10: Comparisons of all models on the regular dataset. The upper green curve is the behaviour of the *User-kNN* model with fixed $\beta = 0.4$ and various k. The curve in the middle represents an AR model with fixed $\beta = 0.6$ and various settings of s_{min}. The lower curve represents the results of the Onto-best model depending on γ. At the bottom of the chart is a dot that represents the best seller model.

Appendix B

Future work

B.1 Rising Star

We detected this problem in the academic domain, where the items are human experts. When a user searches for a suitable expert in some field, search algorithms find the best expert in a field, let us call him a *star*. However, what does it mean to be the best expert? It means the experts with the most outcomes in this field, and this could be a problem.

This is because, in practice, if we need to engage with this expert, and he is the best, he likely does not have time, already has a project, or is retired. Their profile contains many nodes with high frequency. So, from a business point of view, we should find a suitable candidate, who has a similar or almost similar profile as our star, has fewer outcomes, or significantly less, but their outcomes have the appropriate quality. Also, we should use the time dimension; this means that we should find an expert whose outcomes are not old. This expert is usually at the beginning of his career, younger (Ph.D.) than our *star*, and has much potential, so we call them a *rising star*

Ontology could be helpful in this task because we can find these kinds of experts with these criteria:

1. Cosine angle of the profile is smaller than our threshold. (The profiles are similar).

2. Euclidean distance is bigger than our threshold. (Significant difference in the number of outcomes).

Figure B.1 presents this concept. For this thesis, we chose a movie domain for our experiment, and this domain does not have this problem, because a user can typically play every movie without constraints. The author therefore retains this problem for future work.

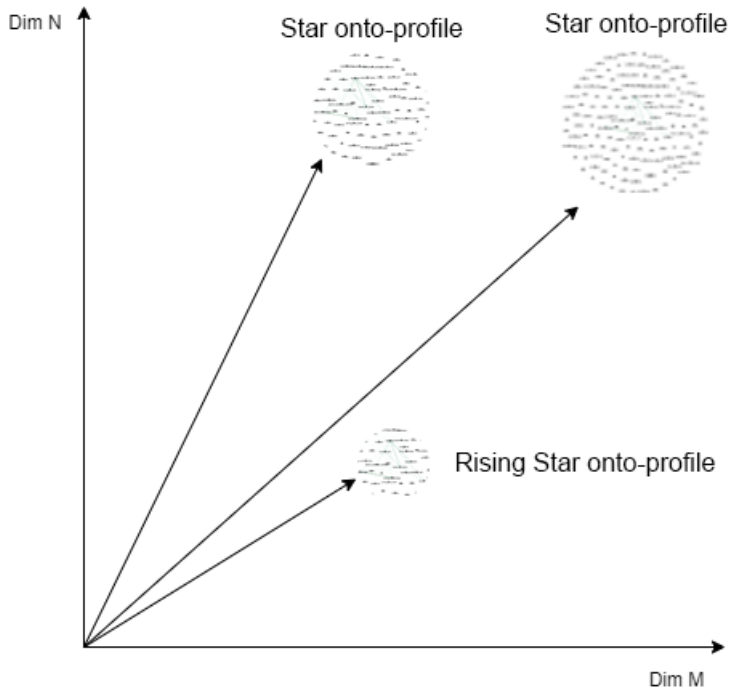

Figure B.1: Rising star concept.

B.2 Keyword Context Detection

The definition of a keyword is: *a word that you type into a computer, for example, when you are using the internet, so that the computer will find information that contains that word* [1].

Since a large number of keywords nowadays have different meanings, for example, as mentioned above, the keyword *AI* has a different meaning in computer science (artificial intelligence) and agriculture (artificial insemination), it is necessary to distinguish in what context the word is used.

The keyword itself, without the context of sentence, paragraph, etc., presents a small amount of semantic information. Ontology connects keywords with the edges, which define the semantic affiliation of two keywords. This feature is useful because it means that we can group the connecting words into a cluster of affiliation words. Then we can look at the keyword neighbourhood or cluster as a context of the keyword. We can then use the keyword and context to generate more accurate item profiles, or extend the regular search with semantic information.

[1] https://dictionary.cambridge.org/dictionary/english/keyword

This area is interesting and deserves independent research. The author keeps it for future work.

B.3 Final Score Cold Start

During our research, we encountered a problem with the final item score in CF algorithms. The algorithms use several methods to compute the final score for each item candidate. The final recommendation is composed of items that are sorted by the final score. We think that this sorting can be problematic in the following cases:

- When items have the same score, then sorting is random.
- When the score between items has no significant difference, i.e. the little difference could be made by computational noise, or one (maybe several) unimportant interactions.

In both cases, we see a pseudo cold-start problem, i.e. we do not have enough information to perform optimal sorting. One of the solutions could use a CB approach – in our case, ontological content-based. We can then decide what are the N most relevant items by semantic similarity. We do not attempt to solve this problem in this thesis but leave it for future work.