UNIVERSIDADE DE SÃO PAULO
Instituto de Ciências Matemáticas e de Computação

Enhancing recommender systems by enrichment with pre-processing approaches supported by users' feedback

Arthur Fortes da Costa

Tese de Doutorado do Programa de Pós-Graduação em Ciências de Computação e Matemática Computacional (PPG-CCMC)

ICMC USP SÃO CARLOS

Arthur Fortes da Costa

Enhancing recommender systems by enrichment with pre-processing approaches supported by users' feedback

Thesis submitted to the Institute of Mathematics and Computer Sciences – ICMC-USP – in accordance with the requirements of the Computer and Mathematical Sciences Graduate Program, for the degree of Doctor in Science. *FINAL VERSION*

Concentration Area: Computer Science and Computational Mathematics

Advisor: Prof. Dr. Ricardo José Gabrielli Barreto Campello
Co-advisor: Prof. Dr. Marcelo Garcia Manzato

USP – São Carlos
January 2020

Ficha catalográfica elaborada pela Biblioteca Prof. Achille Bassi
e Seção Técnica de Informática, ICMC/USP,
com os dados inseridos pelo(a) autor(a)

C837e

Costa, Arthur Fortes da
Enhancing recommender systems by enrichment with pre-processing approaches supported by users' feedback / Arthur Fortes da Costa; orientador Ricardo José Gabrielli Barreto Campello; coorientador Marcelo Garcia Manzato. -- São Carlos, 2019.
159 p.

Tese (Doutorado - Programa de Pós-Graduação em Ciências de Computação e Matemática Computacional) -- Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, 2019.

1. Recommender Systems. 2. Data Minining. 3. Pre-processing approaches. I. Campello, Ricardo José Gabrielli Barreto, orient. II. Manzato, Marcelo Garcia, coorient. III. Título.

Bibliotecários responsáveis pela estrutura de catalogação da publicação de acordo com a AACR2:
Gláucia Maria Saia Cristianini - CRB - 8/4938
Juliana de Souza Moraes - CRB - 8/6176

Arthur Fortes da Costa

Otimização dos sistemas de recomendação por enriquecimento com abordagens de pré-processamento baseadas em feedback de usuários

Tese apresentada ao Instituto de Ciências Matemáticas e de Computação – ICMC-USP, como parte dos requisitos para obtenção do título de Doutor em Ciências – Ciências de Computação e Matemática Computacional. *VERSÃO REVISADA*

Área de Concentração: Ciências de Computação e Matemática Computacional

Orientador: Prof. Dr. Ricardo José Gabrielli Barreto Campello
Coorientador: Prof. Dr. Marcelo Garcia Manzato

USP – São Carlos
Janeiro de 2020

This thesis is dedicated to my parents, my sister and all friends that contribute directly and indirectly with this work.

ACKNOWLEDGEMENTS

Foremost, I wish to give my heartfelt thanks to a very special person, my mother Rosário Fortes. Thanks for her unconditional love, her unflagging encouragement, patience, and continued support that made the culmination of this thesis possible. I would like to extend my sincerest gratitude to the people who mean a lot to me, my family, my friends and my girlfriend. I greatly value their insightful comments, the stimulating discussions, that helped me to keep things in perspective and for all the fun we have had on this journey.

I would like to express my special appreciation and my sincere gratitude to my advisor Prof. Ricardo José Gabrielli Barreto Campello. Thanks for the continuous support and patience during all my Ph.D. study and research, especially in the most critical periods; for his helpful and priceless advice; and for his insightful comments. I would also like to thank my co-advisor Prof. Marcelo Garcia Manzato. Thanks for his guidance, for supporting my research ideas, and for his insightful discussions about the research in the last months. Both of them, Prof. Campello and Prof. Manzato, were and remain my best role models for a scientist and mentor.

I also want to thank to all professors and faculty members from ICMS-USP who, directly or indirectly contributed to this work. Last but not at least, I would like to thank CAPES that fully financed my doctorate's degree and FAPESP and Eldorado Research Institute for the financial support for the publication and participation of scientific events.

"To dream, to seek the unknown, to look for what is beautiful is its our only reward. And I beg you to remember those words so easy to forget: A man's reach should exceed his grasp, or what's a Heaven for?" - Nina Fawcett

RESUMO

DA COSTA, A. F. **Otimização dos sistemas de recomendação por enriquecimento com abordagens de pré-processamento baseadas em feedback de usuários**. 2020. 159 p. Tese (Doutorado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, 2020.

Sistemas de recomendação utilizam informações sobre preferências de usuários para inferir o seu gosto em relação a novos itens. Um grande problema nessa área é o número de informações que os algoritmos precisam para calcular pontuações para uma grande quantidade de itens desconhecidos no banco de dados. Além disso, problemas tradicionais como esparsidade, alta dimensionalidade e partida fria, dificultam ainda mais a tarefa de predição desses algoritmos. Atualmente, vários trabalhos tentam lidar com esses problemas, utilizando soluções dentro do próprio algoritmo de recomendação, o que aumenta o tempo e o custo computacional dos algoritmos utilizados nessa tarefa. Nesta tese de doutorado, propomos abordagens de pré-processamento de dados para sistemas de recomendação que reduzem e/ou enriquecem o número de pares usuário-item desconhecidos que o recomendador deve processar para obter um conjunto de dados com informações mais confiáveis e robustas. Nossas abordagens concentram-se no feedback de usuários, tentando extrair os gostos e comportamentos de cada um usando informações disponíveis nos conjuntos de dados. Avaliamos a qualidade dessas abordagens aplicando-as em algoritmos de recomendação tradicionais e conhecidos na literatura, além de comparar os resultados com os mesmos recomendadores sem a etapa de pré-processamento e com outros algoritmos do estado da arte. Os resultados mostram uma melhora significativa na acurácia dos recomendadores tradicionais e na redução do impacto de problemas conhecidos na área de recomendação.

Palavras-chave: Sistemas de recomendação, abordagens de pré-processamento, interações de usuários, dimensionalidade, esparsidade, partida fria.

ABSTRACT

DA COSTA, A. F. **Enhancing recommender systems by enrichment with pre-processing approaches supported by users' feedback**. 2020. 159 p. Tese (Doutorado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, 2020.

Recommender systems use information about the users' preferences to define scores of interests towards items. Regardless of the method, a noticeable problem is that the system is required to compute scores for a large amount of unknown items in the database, even though these items may not be related to a determined user. Besides that, traditional problems, such as sparsity, high dimensionality and cold-start make the prediction task even more difficult. Currently, several works try to deal with these problems, using solutions within the recommendation algorithm itself, which increases the time and computational cost of them. In this doctoral thesis, we propose many pre-processing techniques for recommender systems that reduce and/or enrich the number of unknown user-item pairs the recommender must process to obtain a dataset with more reliable and robust information. Our approaches focus on users' feedback, trying to extract tastes and behaviors from each user from the information available in the datasets. We assess the quality of these approaches by applying them into some well-known RS and comparing the results against the same recommenders without our pre-processing step, as well as against other related baselines and state-of-art works. Results show a significant improvement in the accuracy of the recommenders and the reduction of the impact of the traditional recommendation problems.

Keywords: Recommender Systems, pre-processing approaches, users' feedback, dimensionality, sparsity, cold-start.

LIST OF FIGURES

LIST OF ALGORITHMS

LIST OF TABLES

LIST OF ABBREVIATIONS AND ACRONYMS

CBF	Content-based filtering
CF	Collaborative filtering
CoRec	Co-training approach for recommender systems
ECoRec	Ensemble approach using co-training for recommender systems
GB	Group-based apporach
HF	Hybrid filtering
RS	Recommender systems
TWCRS	Two-way clustering approach for recommender systems

LIST OF SYMBOLS

u — represents an user in a dataset

Nu — is the popularity of user u in a dataset

b_u — is observed deviations of user u based on his interactions in the dataset

i — represents an item in a dataset

b_i — is observed deviations of item i based on the interactions made with it

Ni — is the popularity of item i in a dataset

b_{ui} — is a baseline prediction made using b_u and b_i

μ — is the global average rating in the dataset

r_{ui} — is used to refer to a feedback from a user u to an item i

$\hat{r}_{ui}$ — is the final prediction of the system about the preference of user u to item i

$\mathscr{C}_{ui}$ — is a confidence metric for a prediction feedback $\hat{r}_{ui}$

CONTENTS

CHAPTER

1

INTRODUCTION

Recommender systems (RS) have emerged in the last two decades as an alternative to mitigate the information overload problem (AGGARWAL, 2016; RICCI; ROKACH; SHAPIRA, 2015). By delivering personalized content, it allows the user to avoid the tedious process of browsing through several unrelated content that is made available every day on the Web. Its main premise is to capture the users' tastes and, with them, elect the most related items from the collection and return them as suggestions. RS are able to infer preferences and interests of users based on their interactions, also known as feedback, which can be implicit (purchase history, clicks, among others) and/or explicit (e.g, ratings, tag attribution, text reviews) (RICCI; ROKACH; SHAPIRA, 2015; KUMAR; , 2018).

Currently, two different recommendation practices are used to assess the items' relatedness to the user (AGGARWAL, 2016): the first, called rating prediction, aims to predict the score a user would give to determined item; the second, called top-k recommendation, produces a relatedness score which is used to rank the items accordingly. In this second scenario, the ranking returned by the recommender is usually small, since users tend to only browse through the top results (AGGARWAL, 2016; RENDLE *et al.*, 2009). Both tasks are aided by filtering techniques, such as Content-based filtering (CBF), Collaborative filtering (CF), Hybrid filtering (HF), demographic filtering and many others. The first three approaches are the most well-known and most used nowadays, in which CBF uses characteristics extracted from items to construct a users' preference profile and achieve recommendations based on such information, and CF uses previous interactions of similar users or items to compute recommendations. The two paradigms are not exclusive, and often their are intermingled into a hybrid filtering, in order to mitigate the weaknesses of each individual filtering and improve overall performance by combining their strengths (AGGARWAL, 2016; RICCI; ROKACH; SHAPIRA, 2015; KUMAR; , 2018).

Regardless of the filtering approaches adopted, RS face well-known problems, such as cold-start, sparsity and high dimensionality of data (AGGARWAL, 2016; KUMAR; , 2018). In cold-start problem, items or users are considered cold if they have few or none interactions, lead-

ing to the system not being capable to recommend a cold item or to provide proper suggestions to a cold user (RICCI; ROKACH; SHAPIRA, 2015). Sparsity means that much of the information about the users' preferences is missing, given that users generally do not interact with most available items (RICCI; ROKACH; SHAPIRA, 2015; AGGARWAL, 2016), e.g. usually one does not listen or provide feedback to most songs from a radio. High dimensionality, on the other hand, occurs because there are too many attributes to describe users, assuming users are described by their interactions with each item, and typically there is a large number of items (RICCI; ROKACH; SHAPIRA, 2015).

In this context, some efforts have been made in data mining approaches (ZHANG *et al.*, 2014; VLACHOS *et al.*, 2014; ZHANG; WANG, 2015; GUPTA; PATIL, 2015; ZHANG *et al.*, 2016a; XIAOJUN, 2017; KATARYA; VERMA, 2017) to improve the accuracy of recommendations and to deal/ alleviate the previously mentioned problems. These approaches make it possible to better describe users and items representations by enriching them with extra information (e.g. inclusion of item metadata, inferring user's feedback) and to group users and/or items with similar descriptions, discovering topics of interest while possibly also detecting and removing useless information. However, these works require a large number of explicit feedback and extra item information (e.g. reviews, synopses, actors, etc), which may not be available for every application domain, besides not considering other types of user's feedback in the recommendation process. Furthermore, the computational cost to extract, treat and process the extra information within the recommender algorithms is high, which can make these algorithms extremely slow.

In this research, we focus on the developed of high-quality and semantic pre-processing approaches to improve the performance of traditional recommenders in cold-start, sparsity and/ or high dimensionality scenarios. These approaches offer to recommenders the opportunity to work with enriched and personalized information to give more semantic and accuracy to them.

1.1 Problem statement

In many real-life applications, both the number of items and the number of consumers are large. In such cases, even when many events have been recorded, the consumer-product interaction matrix can still be extremely sparse, that is, there are very few elements in the matrix whose value is known. These problems, commonly referred to as the sparsity and high dimensionality problems, have a major negative impact on the effectiveness of collaborative filtering approaches (BOBADILLA *et al.*, 2013). Because of sparsity, it is very likely that the similarity (or correlation) between two given users cannot be properly measured or may not be reliable, rendering poor collaborative filtering-based recommendations (AGGARWAL, 2016; BOBADILLA *et al.*, 2013).

Another well-known problem is cold-start, which is when a recommender system does

not have sufficient historical information about a user or an item to generate predictions for them (BOBADILLA *et al.*, 2013; AGGARWAL, 2016; KUMAR; , 2018). For instance, as opposed to popular items, it is difficult for standard recommendation approaches to provide high-quality recommendations involving newly released items or items that are not so popular, for which very few ratings are available. In other words, once the final list of predictions is generated based on the preferences of undistinguished users, the system may recommend items for users who have no interest in that selected content (e.g. in a movie domain, the system could recommend a terror movie for a user that likes only romance).

From the machine learning perspective, the cold-start and sparsity problems have a common root in the shortage of labeled data to train recommender algorithms in a traditional, fully supervised (i.e., informed/guided) way. In the realm of collaborative filtering, "labeled" data means known user/item interactions, such as ratings. However, very often labeled data are limited and expensive to obtain, since labeling typically requires users' feedback or human expertise (e.g. conducting user study). It can be even more critical if one wants to train personalized models, which require large amounts of labeled data from each individual. In addition, high dimensionality occurs because there are too many attributes to describe users, assuming users are described by their interactions with each item, and typically there is a large number of items (AGGARWAL, 2016; KUMAR; , 2018).

In fact, recommender systems algorithms need to work on data, appropriately processed by a set of operations which make assumptions and choices on the inclusion of features in user/ items representations. Nowadays, most of the works in the area try to deal with the well-known challenges directly in the algorithms (AGGARWAL, 2016; KUMAR; , 2018), increasing their computational time complexity. In this context, pre-processing approaches are fundamental for these systems to obtain good results and performance, relieving the data overload and possible problems. Normally it includes methods for data cleaning/enriching and feature extraction and selection.

1.2 Research questions and hypotheses

The main research question is:

- Research Question 1: How can we improve the accuracy of existing recommender algorithms using pre-processing approaches?
  - Hypothesis 1: Enriching the input data by adding extra information and removing useless data will improve the recommenders, since they will work with more robust and confident data.

As previously mentioned, the algorithms of recommender systems work with data, in this way, the better the data are processed and the larger the amount of information, the better

the recommendation will be. In this research we use as extra information, different type of user feedback, such as ratings, tags and browsing history. Furthermore, to answer the main question, we also need to answer and validate the following questions and hypotheses:

- Research Question 2: How can we integrate different types of user's feedback to improve the quality of my input data?
  - Hypothesis 2: Combining user's feedback in a enriched profile using machine learning techniques (e.g. ensemble and data mining approaches), heuristics or statistical information will improve the integration and quality of data in a unique vector, which will better represent the users' behavior and improve quality of recommendations.
- Research Question 3: How can we pre-process and prepare the input data to alleviate traditional recommender systems problems, such as cold-start, dimensionality and sparsity?
  - Hypothesis 3: Using data clustering and active learning techniques, we can filter only the relevant information to be processed by recommender and increase the amount of useful data in the dataset, respectively.

1.3 Objectives

The main objective of this research is to develop pre-processing approaches based on data-mining techniques (e.g. data clustering and active learning) for recommender systems that can be applied prior to any recommender algorithm, categorizing the users and items into groups and enriching their representation in order to generate more concise and reliable data as input. The proposed approaches in principle should require very little information to operate, and they can be extended in order to use more information, when available.

1.4 Main original contributions

The main original contributions of this thesis can be summarized as follows:

- We have proposed a novel user's profile based on descriptive statistics from different users' feedback, in order produce a vector-based user profile with a smaller set of attributes, reducing data sparseness and dimensionality and improving representation of each user's preferences. Additionally, the proposed enriched user's profile conveys more detailed information, such as which categories he/she likes/dislikes the most, or which categories he/she maintains a certain rating pattern. This approach is extensible and flexible for different domain datasets, since the pre-processing premise for user profiling using only statistics can be easily extended to aggregate different types of interactions.

- We have developed two clustering-based approaches for recommender systems that yield better recommendation accuracy and alleviate high dimensionality and sparsity problems, using users' groups, generated from the similarity between them. It supports different types of interactions (e.g ratings, tags and history browsing) of each user for customer's preferences and low similarities as well. In this approach, we have been able to avoid bad recommendations for users, since we only look at the items that other users with similar preferences have accessed, e.g. a user who likes metal will never receive recommendations from Britney Spears songs.

- We have developed two pre-processing approaches (CoRec and ECoRec) based on co-training, in order to solve the main problem of the two previous approaches: new users and new items (cold-start). These algorithms drive two or more recommenders to agree with each others' predictions to generate their own, in order to predict scores for unknown $(user, item)$ pairs in the original user-item matrix. Additionally, the proposed pre-processing approaches significantly reduce the sparsity problem by enriching the original user-item ratings matrix, during the co-training step.

- We made freely available a recommender system framework, named Case Recommender[1], which contains a variety of content-based and collaborative recommender algorithms, as well as pre and post processing approaches (e.g. clustering, ensemble) for support and combining multiple algorithms and data sources. In addition, it provides a set of popular evaluation methods and metrics for rating prediction and top-N recommendation.

- We also built and made available a recommender system dataset repository[2] with many databases of different domains. We list all datasets divided by domain with their respective download links, and we also offer pre-processed datasets used in our research with their respective credits.

- This work generated 4 complete refereed papers published in conferences and 2 articles published in journals.

1.5 Outline

This thesis is an articles collection, based on research articles and papers written during this doctoral study. Each article is presented as chapter, which contains an introduction and conclusion sections placing the scope and results of the article in the thesis context. The chapters are structured as follows:

- Chapter 2 describes a pre-processing approach to enrich the users' profiles using different types of user feedback. This chapter contains the following publication "Exploiting Differ-

[1] https://github.com/caserec/CaseRecommender
[2] https://github.com/caserec/Datasets-for-Recommneder-Systems

ent Users' Interactions for Profiles Enrichment in Recommender Systems" (COSTA *et al.*, 2016).

- Chapter 3 describes a collaborative filtering approach based on clustering algorithm using users' feedback to improve the accuracy of recommendations. It is based on the publication "Group-based Collaborative Filtering Supported by Multiple Users' Feedback to Improve Personalized Ranking" (COSTA; MANZATO; CAMPELLO, 2016).

- Chapter 4 presents an extension of the previous chapter, in which we implement a pre-processing approach based on clustering ensemble techniques. The article in this chapter is entitled "Ensemble Clustering Approaches Applied in Group-based Collaborative Filtering Supported by Multiple Users' Feedback" (COSTA; MANZATO; CAMPELLO, 2017).

- Chapter 5 reports the results obtained in a experiment based on two-way-clustering approach applied on recommender systems, in order to alleviating the dimensionality and sparsity problems.

- Chapter 6 reports an pre-processing approach, named CoRec, which is based on a co-training technique that drives two recommenders to agree with each other's predictions to generate their own. In this chapter, we present the paper "CoRec: A Co-Training Approach for Recommender Systems" (COSTA; MANZATO; CAMPELLO, 2018).

- Chapter 7 describes an ensemble scheme based on the last chapter, named ECoRec, that drives two or more recommenders to agree with each others' predictions to generate their own, generating a robust output. This chapter contains the following publication "Boosting collaborative filtering with an ensemble of co-trained recommenders" (COSTA; MANZATO; CAMPELLO, 2019).

- Chapter 8 presents the paper titled "Case Recommender: A Recommender Framework", which reports a RS framework developed during my master and doctoral studies. Case Recommender contains a rich set of components and pre-processing techniques from which developers and researchers can construct and evaluate customized recommender systems (COSTA *et al.*, 2018).

- Chapter 9 reports the paper titled "Case Recommender: a flexible and extensible Python framework for recommender systems", which presents the last features added to framework, such as clustering algorithms and pre-processing approaches, besides that new evaluation and validation metrics metrics (COSTA *et al.*, 2018). This paper was published and presented at the most relevant conference in our research area, called RecSys[3].

Finally, in Chapter 10, we provide conclusions and discuss limitations and future works.

[3] https://recsys.acm.org/

CHAPTER

2

EXPLOITING DIFFERENT USERS' INTERACTIONS FOR PROFILES ENRICHMENT IN RECOMMENDER SYSTEMS

2.1 Contextualization

In order to recommend products, services or people to a user, it is necessary to obtain knowledge about his/her behavior, preferences and affinities. It is important to define and identify what kind of information will be relevant to the generation of an efficient recommendation, to then capture and store a user's personal and behavioral information in the user profile (NADEE, 2016; AGGARWAL, 2016; SAHU; DWIVEDI, 2019). Each term in this representation expresses a characteristic of a particular user including all the information directly requested to them and implicitly learned during his interaction on the system (e.g. most watched type of movie and how long he/she takes to interact with the system). Pre-processing techniques can be used to generate these representations and are of great importance for the recommender systems, since they are responsible to clean, enrich existing data and extract relevant information, which in turn, considerably improve the accuracy of the recommendations and alleviate traditional problems like cold-start and high dimensionality (AGGARWAL, 2016; SAHU; DWIVEDI, 2019).

In this chapter, we introduce the development of a user profile, in which we used a set of heuristics, text mining and sentiment analysis tools to model and extract different user's feedback, such as ratings, history browsing and tags. Furthermore, this work aims to provide a full architecture that is capable of producing representations of users based on his/her interactions, and further semantically enhance those representations so that a recommender system can better define the similarity between users. Such similarity is applied to the recommendation itself, aiming to provide better suggestions to users.

Exploiting different users' interactions for profiles enrichment in recommender systems

Arthur F. da Costa, Rafael M. D'Addio, Marcelo G. Manzato and Ricardo J. G. B. Campello
Institute of Mathematical and Computer Sciences
University of São Paulo, São Carlos, SP, Brazil
{fortes, rdaddio, mmanzato, campello}@icmc.usp.br

ABSTRACT

User profiling is an important aspect of recommender systems. It models users' preferences and is used to assess an item's relevance to a particular user. In this paper we propose a profiling approach which describes and enriches the users' preferences using multiple types of interactions. We show in our experiments that the enriched version of users' profiles is able to provide better recommendations.

CCS CONCEPTS

• **Information systems → Personalization**.

PUBLICATION INFORMATION

Published in proceedings of the 31st Annual ACM Symposium on Applied Computing (SAC '16). 2016. Pages 1080-1082.
DOI:10.1145/2851613.2851923

KEYWORDS

Recommender system, User profiling, Multiple interactions

1 INTRODUCTION

Usually, in recommender systems, users' profiles consist of a vector where each position represents an item of the database, and the value of each position is the rating the user has assigned or a binary value representing whether the user consumed the item or not [4, 8]. This type of representation, however, leads to sparsity because there are thousands of items available and only a small set of them was evaluated or viewed by the user.

Many systems on the Web today allow users to interact with the content in multiple ways. In fact, depending on many factors, customers may interact differently with the content, and consequently, provide different types of feedback regarding his/her preferences. The literature reports a shortage of techniques which integrate different types of user feedback into a generic model [2, 4, 6]. However, some of these techniques are limited to a small subset of users' feedback, and others, which have a generic scope [2, 6], are time consuming and/or with low accuracy.

Given this need of processing different users' interactions in order to obtain more accurate information about the users' preferences, we propose an approach that enriches users' profiles by means of a pre-processing step that analyzes different users' interactions, producing a set of statistics related to the users' behavior. Such analysis is integrated to items' related categories, in order to represent the users' interests towards particular topics.

We direct our efforts into an item recommendation scenario, where the main goal is to produce a top-N ranking of suggestions. Given that, we adapt two recommender algorithms into this scenario, and subsequently extend them so they can incorporate the enriched profiles. We compare the final rankings produced by the algorithms with and without using the enriched profiles, and analyze how much the users' interactions can influence the final recommendations.

The paper is structured as follows: Section 2 addresses the related work; Section 3 describes how the users' profiles were built; Section 4 presents the extended recommendation algorithms; Section 5 reports the evaluation conducted in the system; finally, Section 6 presents the final remarks and perspectives for future work.

2 RELATED WORK

The SVD++ algorithm proposed in [4] uses explicit (ratings) and implicit (viewing history) information from users in a factorization model. Another factorization model, called Factorization Machines (FM) [6], can consider many types of information regarding users, items and/or their interactions. These techniques have the drawback that they process only certain types of interactions, with little capability of extension to other different types. In recent studies, Costa et al. [2] developed an ensemble recommender technique, called Ensemble BPR Learning, to unify different types of feedback from users, processed in different recommendation techniques. While this model is extensible for any types of user's interaction, its learning phase has high computational cost, since it depends on the execution of several recommendation techniques beforehand.

The present work differs from the aforementioned since it explores an alternative to build enriched user profiles, in a pre-processing step, that summarizes the interactions made by users in the system, producing a statistical view.

3 USER PROFILING

We propose a pre-processing approach that considers statistics from different user interactions and produces a vector-based user profile with a smaller set of attributes, reducing data sparseness and improving representation of each user's preferences. To do that, we derived information from ratings and tags attributed by users. We used only these two interactions due to their availability in the adopted database, but the profile can be further extended to other available interactions. We also consider the statistics towards the categories of the items the user evaluated. Category can be seen as the classification of an item in its domain, e.g. the genre of a movie. By doing this, the enriched user's profile conveys more detailed information, such as which categories he/she likes/dislikes the most, or which categories he/she maintains a certain rating pattern. In this work, we considered the movie genre category,

available within the dataset used. The interactions were processed as follows.

Ratings: we consider a global average and standard deviation of all of their ratings. Also, for each category, we track which items belonged to them and then calculate the average and standard deviation using the ratings that he/she assigned to items of that category.

Tags: we derive a user x item matrix that contains the sentiment of the tags that the users attributed to the items. Tags are small labels that describe the content of an item, and may contain sentiment or not. Neutral tags are more common and tend to dominate the calculation of the average sentiment, and thus we do not consider them. We compute a float score (ranging from 1 to 5) that represents the sentiment for each user/item pair that has tag attribution. In order to do this, we process the tags in the sentiment analysis algorithm available in the Stanford CoreNLP [1], which classifies sentences into one of the following: "Very Negative", "Negative", "Neutral", "Positive", "Very Positive". We convert this classification in a range of values from 1 to 5, where 1 is equivalent to "Very Negative" and 5 is equal to "Very Positive", finally producing a user x item matrix referring to the tags' sentiment. With the tag sentiment matrix at hand, for each user we select his corresponding row and derive from that the average tag sentiment score and the standard deviation, as well as the scores for each category, in a similar fashion performed with the ratings matrix.

4 RECOMMENDER ALGORITHMS

We evaluate our proposal with two recommender algorithms: a neighborhood-based [5, 8] and a soft clustering [3] approaches. The following subsections detail the adaptations we performed in each algorithm to allow them to process the enriched profile and provide rankings of recommendation.

4.1 User k-Nearest Neighbor

The first recommendation algorithm is the well-known User k-NN, which a description can be found in [5, 8]. We adopt this algorithm because of its well-acceptance, and because it can be intuitively extended to include other information. The main goal of the algorithm is to find similar users and predict a rating based on their rating assignments.

We extended the aforementioned algorithm to consider the enriched users' profiles in its calculation, using the default similarity defined in Equation 1.

$$s_{uv} = \frac{n_{uv}}{n_{uv} + \lambda_1} p_{uv}, \tag{1}$$

where n_{uv} is the number of items that users u and v have in common, and λ_1 is a regularization constant, set as 100 according to suggestions found in the literature [5], and p_{uv} is a similarity score given by a similarity measure such as cosine or Pearson correlation.

[1] http://nlp.stanford.edu/software/corenlp.shtml

We combine this standard similarity with the similarity between users' profiles, computed in the same fashion, producing a final average similarity.

Another change that we made was to keep the set of the k most similar users to u, regardless whether they evaluated the item or not. Then, a cut is carried out in this set, keeping only those who indeed have evaluated the item. The effect is that the system considers only users that evaluated the item in question and are in fact similar to the user u, which now can be a different number for different users u.

As previously described, the User k-NN is usually used for rating prediction, however, this study aims to evaluate the system's ability to produce personalized rankings. Each ranking is obtained based on the value $\hat{s}_{ui}$ corresponding to the affinity of a user u in relation to an item i, regardless of their predicted rating. Thus, the User k-NN algorithm was adapted in this work to generate rankings of items.

4.2 Soft Clustering Recommender

The second recommendation algorithm used was based on a work proposed by Ganu et al. [3] where the users are arranged into different clusters with a degree of relevance. The main idea of this algorithm is that each user has a value that describes the probability, or the degree of relevance, that it is contained in a cluster. The full description of this algorithm can be seen in [3].

To aggregate the enriched profile, the algorithm produces a second clustering solution from their vectors, calling it $v_j^{profile}(c_k)$. The final contribution is equivalent to the sum of the relevance degrees in both solutions for every other user that has viewed this item, as presented by the equation:

$$Contr(c_k, i) = \frac{\sum_{j=1}^{n} v_j(c_k) + v_j^{profile}(c_k)}{2} \tag{2}$$

We eliminate the addition of the ratings to the contribution score, since we are dealing with ranking recommendation. The final $\hat{r}_{ui}$ value obtained this way is equivalent to the affinity level of the user towards an item, instead of being the predicted rating.

5 EVALUATION

To evaluate our approach we first check which of the interactions provided the best gain in the user profiling by comparing these interactions individually and combined. Furthermore, we compare the algorithms having the best combination of interactions with a state-of-the-art CF-based algorithm, called BPR MF [7].

5.1 Experimental Setting

Regarding the database, we used the HetRec MovieLens 2k, introduced by Cantador et al. [1]. This database consists of 800,000 ratings (ranging from 1 to 5) and 10,000 tag assignments applied by 2,113 users into 10,197 movies.

All experiments were executed using 10-fold cross-validation, and we present the average results of our experiments. We evaluated the rankings using Mean Average Precision (MAP), applying the Student t-test to check if the results are statistically significant.

Regarding the algorithms' configuration, we tested several combinations in a preliminary analysis and report only the most promising results. For the User k-NN, we used the cosine similarity and $k = 50$. For the soft clustering approach, we used the IIB algorithm with $k = 4$ clusters.

5.2 Results

We detail the results obtained in our experiments. In a first experiment, we evaluated how much each interaction is able to improve recommendation accuracy when used for profiles' enrichment, as it can be seen in Figure 2.1.

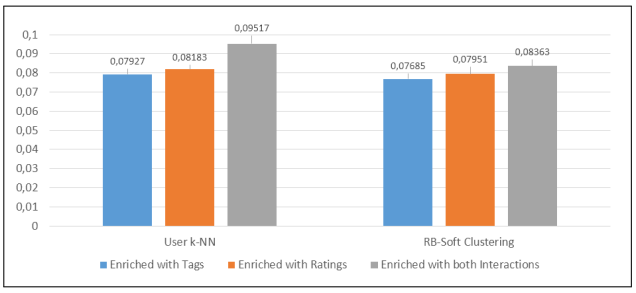


Figure 2. 1. Profiles Comparison results using MAP.

According to the results, the profiles enriched with tags performed worst, while combining both interactions provided better rankings. We argue that while tags do help increasing the user's preferences representation, their interaction matrix is generally more sparse. However, as we combine tags information with ratings into one single profile per user, the system is able to obtain more detailed information about his interests, increasing recommendation accuracy.

The second experiment consists of measuring the impact of integrating the profiles' enrichment into both recommender algorithms previously described. Figure 2.2 show comparative charts containing the results of the considered approaches.

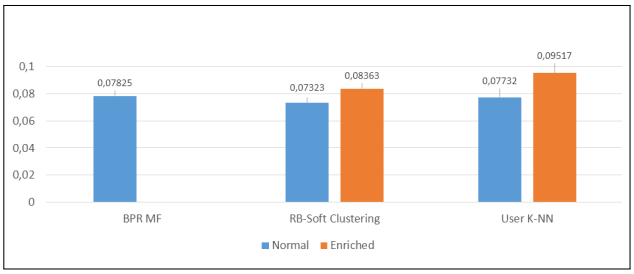


Figure 2. 2. Recommeders comparison using MAP.

It can be seen that the User k-NN and the RB-Soft Clustering achieved the worst results when not considering the enriched users' profiles. However, when considering the enriched users' profiles in their recommendation model, their accuracy was statistically better than BPR MF (with the p-value $< 0,01$). Additionally, as the results show, the enriched profiles were able to enhance the ranking generation of every algorithm considered in this study, providing a positive indication of the availability of taking into account different interactions in the user profiling.

6 FINAL REMARKS

In this paper, we proposed an approach to enrich the users' profiles using different types of user interactions. The goal of this enrichment was to describe more accurately the preferences and interests of individuals and, consequently, recommend more relevant items to them.

As future work, we aim at evaluating the proposed approach with additional data sets from other domains and checking its accuracy with more user information, since pre-processing premise for user profiling can be easily extended to aggregate different types of interactions. We also plan to extend additional algorithms, such as the BPR MF algorithm, to be able to process the enriched user profile.

7 ACKNOWLEDGMENTS

The authors would like to thank to the support from grants 2013/18698-4 and 2013/22547-1, São Paulo Research Foundation (FAPESP); and 304137/2013-8 CNPq/Brazil.

REFERENCES

[1] Iván Cantador, Peter Brusilovsky, and Tsvi Kuflik. 2011. 2nd Workshop on Information Heterogeneity and Fusion in Recommender Systems (HetRec 2011). In *Proceedings of the 5th ACM conference on Recommender systems (RecSys 2011)*. ACM, New York, NY, USA.

[2] Arthur da Costa Fortes and Marcelo Garcia Manzato. 2014. Ensemble Learning in Recommender Systems: Combining Multiple User Interactions for Ranking Personalization. In *Proceedings of the 20th Brazilian Symposium on Multimedia and the Web (WebMedia '14)*. ACM, New York, NY, USA, 47–54.

[3] Gayatree Ganu, Yogesh Kakodkar, and AméLie Marian. 2013. Improving the Quality of Predictions Using Textual Information in Online User Reviews. *Inf. Syst.* 38, 1 (2013), 1–15.

[4] Yehuda Koren. 2008. Factorization Meets the Neighborhood: A Multifaceted Collaborative Filtering Model. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '08)*. ACM, New York, NY, USA, 426–434.

[5] Yehuda Koren. 2010. Factor in the Neighbors: Scalable and Accurate Collaborative Filtering. *ACM Transactions on Knowledge Discovery from Data (TKDD)* 4, 1 (Jan. 2010), 1–24.

[6] Steffen Rendle. 2012. Factorization Machines with libFM. *ACM Trans. Intell. Syst. Technol.* 3, 3, Article 57 (May 2012), 22 pages.

[7] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian Personalized Ranking from Implicit Feedback. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence (UAI '09)*. AUAI Press, Arlington, Virginia, United States, 452–461.

[8] F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor. 2011. *Recommender Systems Handbook*. Springer.

2.2 Final Remarks

This chapter proposes a mechanism that is capable of producing users' representations by using data mining, heuristics and sentiment analysis techniques; and further enrich the data by considering different types of users' feedback, addressing research questions 1 and 2: "How can we improve the accuracy of the existing recommender algorithms using pre-processing approaches?" and "How can we integrate different types of user's feedback to improve the quality of my input data?". Such representations are applied to the recommendation itself, aiming to provide better suggestions to users. Results showed interesting outcomes, as we can observe that enriching the representations by inferring sentiment considering their semantic similarity can indeed better describe the items to a recommender system. This gives room to further exploration on the subject, such as other recommendation algorithms, feature extraction techniques and similarity measures can be tested.

However, this pre-processing approach can not deal with other problems inherent in the recommender systems, such as the high dimensionality of the space of items and the recommendation of useless items to a given user. For example, although you have a well defined user profile, the system will have to choose, from thousands of items, only a small portion similar to the one you would like to recommend. Depending on the recommender algorithm and the pre-processing approaches, the computational time and complexity will be high, because it has to generate scores for all items in the database.

In the next chapters, we will continue to improve our pre-processing approaches to deal with the aforementioned problems and to answer the other research questions.

CHAPTER

3

GROUP-BASED CF SUPPORTED BY MULTIPLE USERS' FEEDBACK TO IMPROVE PERSONALIZED RANKING

3.1 Contextualization

Towards tackling the two drawbacks identified in the last chapter, we propose a pre-processing approach using a data clustering technique to reduce search space of the RS and mainly group users with similar behavior were used to generate recommendations. Data clustering is a subarea of machine learning and unsupervised data mining, no labeled data is available, that is, we are not sure what type of clusters we are looking for, or even if they exist. Conventional clustering algorithms can be classified into two main categories: hierarchical and partitioned algorithms (GAN; MA; WU, 2007). Both types are identified as being hard-clustering algorithms, since separate instances belong to only one group. The distinction is that partitioned algorithms generate "flat" partitions, no hierarchical relationship exist between the partitions generated, while hierarchical algorithms generate a hierarchy of groups, in which each group is associated with ascendant and descendant groups (parent-children dependencies) (GAN; MA; WU, 2007).

By creating user groups from different types of user interaction, we can infer relevant information about behavior and trends, as well as reduce the high dimensionality of both items and users (AGGARWAL, 2016), since we will only work with groups that access similar items (e.g. users who enjoy action movies will never receive novel movies, since no one in their group has ever seen that kind of genre). The analysis and extraction of knowledge in RS has been widely used in a number of areas, including social and behavioral sciences, economics, and marketing, in which understanding the behavior of society is strategic (CRNOVRSANIN *et al.*, 2014). This chapter highlights the importance and results obtained using data clustering as a pre-processing approach for recommender systems.

Group-based Collaborative Filtering Supported by Multiple Users' Feedback to Improve Personalized Ranking

Arthur F. da Costa, Marcelo G. Manzato and Ricardo J. G. B. Campello
Institute of Mathematical and Computer Sciences
University of São Paulo, São Carlos, SP, Brazil
{fortes, mmanzato, campello}@icmc.usp.br

ABSTRACT

Recommender systems were created to represent user preferences for the purpose of suggesting items to purchase or examine. However, there are several optimizations to be made in these systems mainly with respect to modeling the user profile and remove the noise information. This paper proposes a collaborative filtering approach based on preferences of groups of users to improve the accuracy of recommendation, where the distance among users is computed using multiple types of users' feedback. The advantage of this approach is that relevant items will be suggested based only on the subjects of interest of each group of users. Using this technique, we use a state-of-art collaborative filtering algorithm to generate a personalized ranking of items according to the preferences of an individual within each cluster. The experimental results show that the proposed technique has a higher precision than the traditional models without clustering.

CCS CONCEPTS

• **Information systems** → **Clustering**; **Personalization**.

PUBLICATION INFORMATION

Published in proceedings of the 22nd Brazilian Symposium on Multimedia and the Web (Webmedia '16). Volume 2. 2016. Pages 279-286.
DOI:10.1145/2976796.2976852

KEYWORDS

recommender systems, collaborative filtering, data clustering, multiple interactions

1 INTRODUCTION

Recommender Systems (RS) represent a technology that uses statistical techniques and machine learning to make recommendations of items to users based on a history of past activities. RS have become an important research area since the mid-90s, because it supports users to deal with the information overload problem existent on the Web [12].

The recommendation task can be seen as a prediction problem: the system tries to predict the relevance of certain items to a user and then sorts them according to the provided relevance values. The importance of an item is usually represented by a numerical value that reflects the degree of user's interest as provided herein. The result of an RS is usually a set of items ordered in descending order by importance scheduled for a given user [1].

Traditionally, recommender systems employ filtering techniques and machine learning information to generate appropriate recommendations to the user's interests from the representation of his profile. However, other techniques, such as Neural Networks, Bayesian Networks and Association Rules are also used in the filtering process [1]. The most used types of filtering are currently: content-based, responsible for selecting information based on descriptions of items which were rated in the past. Email messages filtered out to the trash messages containing unwanted words is an example of content-based filtering. The second approach is collaborative filtering, which is based on the relationship between people and items. A simple example is the selection of electronic messages based on the relationship between sender and recipient of a message. Finally, the hybrid approach which combines the filtering based on content and collaborative filtering [12].

The collaborative filtering approach tends to calculate its recommendations based on all users' interactions that were given to them [12]. However, this task can increase the computational cost and cause problems in the final result. Once the recommendation will be generated based on the preferences of undistinguished users, the system may recommend items for users who has no interest in those selected content. Another problem is that depending on the size of the database, computational cost is very high in order to process all the information. A possible solution to these problems would be the previous construction of groups of users with similar interests, and further recommendation, similarly to when recommending a content to a friend. For the person who receives the recommendation, it works as a filter or a particular view of a universe of possibilities usually inaccessible. It can also take into consideration the preference of those who are looking for suggestions and not just those who make it. It is also possible to make recommendations based on the opinions of others. As for instance, someone who is not an admirer of the jazz genre may be recommended based on what his friends enjoy, except this style which is unpleasant for him. Still, the recommendation may include explanations of how it was generated to allow recipient to assess.

This work proposes a technique to generate more accurate recommendation of items using groups of users with similar preferences. We analyze a variety of users' interaction paradigms in order to obtain richer information about their interests. Such rich information is then used to create groups of similar users. The recommendation list for each user of a particular group is generated using a recommendation algorithm based on collaborative filtering. We evaluated our proposal by comparison with two state-of-the-art collaborative recommenders (User KNN and BPR MF), and we compared different types of recommendations generated from our approach to demonstrate the proposal's generality. The experiments were

executed with two real datasets from different domains: the first is MovieLens, which contains data from a movies reviews system; and the second is Last FM, a music website. Our study shows that the proposed group-based approach is able to provide better accuracy than individual recommenders.

This paper is structured as follows: Section 2 addresses the related work; Section 3 describes techniques which are explored in this work; Section 4 presents the proposal in details; Section 5 reports the evaluation executed in the system; finally, Section 6 as devoted the final remarks and future works.

2 RELATED WORK

In this section, we review some work related to our proposal. First, we depict approaches related to different types of interaction in recommender systems, and then, we provide a review of data mining approaches in recommender systems.

2.1 Multiple Interactions Based Approaches

With the increasing number of interactions between users and content, several studies have emerged in order to work with the integration of these interactions, so that more information about the users preferences are gathered by the systems. The recommendation systems can be extended in various ways in order to improve the understanding of users and items, including, for example, new types of interaction in the recommendation process and making the combination of them [12].

The SVD algorithm proposed in [7] uses explicit (ratings) and implicit (viewing history) information from users in a factorization model, called SVD++. Another factorization model, called Factorization Machines (FM) [13], can consider many types of information regarding users, items and/or their interactions. These techniques have the drawback that they process only certain types of interactions, with little capability of extension to other different types. In recent studies, Costa et al. [4] developed an ensemble recommender technique, called Ensemble BPR Learning, to unify different types of feedback from users, processed in different recommendation techniques. While this model is extensible for any types of user's interaction, its learning phase has high computational cost, since it depends on the execution of several recommendation techniques beforehand.

The present work differs from the aforementioned since it explores an alternative to build user profiles, in a pre-processing step, that summarizes the interactions made by users in the system, producing users groups with similar behavior.

2.2 Data Mining Based Approaches

In this study, we have used data mining techniques to cluster users with similar preferences to generate recommendations based on their group. Several researchers have proposed recommender systems for online personalization through data mining to provide recommendation services. This kind of recommendation system is used to predict the user navigation behavior and their preferences using web log data.

Kim et al. [6] present an improved recommendation algorithm for CF. The algorithm uses the K-Means Clustering method to reduce the search space. It then utilizes a graph approach to the best cluster with respect to a given test customer in selecting the neighbors with higher similarities as well as lower similarities. The graph approach allows us to exploit the transitivity of similarity. The algorithm also considers the attributes of each item. The work developed by Wen and Zhou [16] presents an improved collaborative filtering recommendation algorithm based on dynamic item clustering method. A similarity threshold limits the similarity between clusters. By calculating the similarity between the current item and the cluster center, it chooses the greatest similitude cluster, and then it finds the target items' nearest neighbors. Li et al. [8] propose an improved collaborative filtering method based on user ranking and item clustering, in which the users are classified and ranked in multiple item clusters by computing their rating qualities based on the previous rating records, and items are recommended for target users according to their similar users with high-ranks in different item categories. Experiments on real world data sets have demonstrated the effectiveness of their approach.

Our approach is different than their work in the sense that we analyze various paradigms of users' interactions on a particular item, in order to create groups of users with similar preferences and thus, a more accurate personal profile. The advantages of this approach is the ease of extending the template for insertion of other types of interactions; the reduced time and computational processing, considering that the sets of data would be reduced to a single cluster before computing the recommendation. Our contribution, therefore, can be considered the proposal of a recommender system based on multiple feedback types of similar users' groups.

3 RELATED MODELS OVERVIEW

This study involves the pre-processing of data by means of data mining techniques and the recommendation of items through filtering algorithms. The following sections present the main concepts covered in this paper.

3.1 Notation

We use special indexing letters to distinguish users and items: a user is indicated as u and an item is referred as i, j; and r_{ui} is used to refer to either explicit or implicit feedback from a user u to an item i. In the first case, it is an integer provided by the user indicating how much he liked the content; in the second, it is just a boolean indicating whether the user consumed or visited the content or not. The prediction of the system about the preference of user u to item i is represented by $\hat{r}_{ui}$, which is a floating point value guessed by the recommender algorithm. The set of pairs (u, i) for which r_{ui} is known are represented by the set $K = \{(u, i)|r_{ui} \text{ is known}\}$. Additional sets used in this paper are: $N(u)$ to indicate the set of items for which user u provided an implicit feedback, and $\bar{N}(u)$ to indicate the set of items that are unknown to user u.

3.2 k-medoids Clustering Algorithm

Clustering is the process of grouping a set of objects into clusters so that objects within a cluster are similar to each other but are

dissimilar to objects in other clusters [5]. K-means clustering [9] and partitioning around medoids are well known techniques for performing non-hierarchical clustering. K-means clustering iteratively finds the k centroids and assigns every object to the nearest centroid, where the coordinate of each centroid is the mean of the coordinates of the objects in the cluster. Unfortunately, K-means clustering is known to be sensitive to the outliers although it is quite efficient in terms of the computational time. For this reason, K-medoids clustering are sometimes used, where representative objects called medoids are considered instead of centroids. Because it is based on the most centrally located object in a cluster, it is less sensitive to outliers in comparison with the K-means clustering [11].

k-medoids clustering algorithm is a partition-based clustering algorithm based on k-means. It attempts to minimize the distance between points labeled to be in a cluster and a point designated as the center of that cluster [11]. In contrast to the k-means algorithm, k-medoids chooses datapoints as centers (medoids or exemplars) and works with an arbitrary matrix of distances between datapoints. It is more robust to noise and outliers as compared to k-means because it minimizes a sum of pairwise dissimilarities instead of a sum of squared Euclidean distances [11]. A medoid can be defined as the object of a cluster whose average dissimilarity to all the objects in the cluster is minimal, i.e., it is a most centrally located point in the cluster. The implementation of the K-medoids used in this paper is explained as follows:

K-Medoid Clustering Algorithm

(1) Initialization: randomly select (without replacement) k of the n data points as the medoids.
(2) Associate each data point to the closest medoid. (Using a dissimilarity measure like cosine, Pearson, etc.)
(3) For each medoid m
  - For each non-medoid data point o
    - Swap m and o and compute the total cost of the configuration
(4) Select the configuration with the lowest cost.
(5) Repeat steps 2 to 4 until there is no change in the medoid.

The advantage of this k-medoids implementation is that large datasets can be efficiently classified and its convergence is proved regardless of the dissimilarity measure. Furthermore, it is reliable in theory, simple and fast [11]. In this work, to generate users' groups, we use the k-medoids algorithm to group users with similar preferences. This algorithm was chosen to accept different types of distance and not only metrics, such as Manhattan and Euclidean, which do not work well for recommendation approaches.

3.3 Recommendation Algorithms

We evaluate our proposal with two recommender algorithms: a neighborhood-based [7, 12] and a matrix factorization [14] approaches. The following subsections detail each algorithm.

3.3.1 BPR MF: Recommendation Algorithm. The BPR MF approach [14] consists of providing personalized ranking of items to a user according only to implicit feedback (e.g. navigation, clicks, etc.). An important characteristic of this type of feedback is that we only know the positive observations; the non-observed user-item pairs can be either an actual negative feedback or simply the fact that the user does not know about the item's existence. The authors have proposed a generic method for learning models for personalized ranking, where instead of training the model using only the user-item pairs, they also consider the relative order between a pair of items, according to the user's preferences [14]. It is inferred that if an item i has been viewed by user u and j has not ($i \in N(u)$ and $j \in \bar{N}(u)$), then $i >_u j$, which means that he prefers i over j. Each user u is associated with a user-factors vector $p_u \in \mathbb{R}^f$, and each item i with an item-factors vector $q_i \in \mathbb{R}^f$.

It is important to mention that when i and j are unknown to the user, or equivalently, both are known, then it is impossible to infer any conclusion about their relative importance to the user. To estimate whether a user prefers an item over another, Rendle et al. proposed a Bayesian analysis using the likelihood function for $p(i >_u j|\Theta)$ and the prior probability for the model parameter $p(\Theta)$. The final optimization criterion, BPR-Opt, is defined as:

$$\text{BPR-Opt} := \sum_{(u,i,j)\in D_K} \ln \sigma(\hat{s}_{uij}) - \Lambda_\Theta ||\Theta||^2 , \tag{1}$$

where $\hat{s}_{uij} := \hat{r}_{ui} - \hat{r}_{uj}$ and $D_K = \{(u,i,j)|i \in N(u)\ \&\ j \in \bar{N}(u)\}$. The symbol Θ represents the parameters of the model, Λ_Θ is a regularization constant, and σ is the logistic function, defined as: $\sigma(x) = 1/(1+e^{-x})$.

For learning the model, the authors also proposed a variation of the stochastic gradient descent technique, denominated LearnBPR, which randomly samples from D_K to adjust Θ.

3.3.2 User KNN. This recommendation algorithm is the well-known User KNN, whose details can be found in [7, 12]. We adopted this algorithm because of its well-acceptance, and because it can be intuitively extended to include other information. The main goal of the algorithm is to find similar users and predict the best items for them based on their similar items.

In this way, a score is predicted for a unknown user-item pair $\hat{r}_{ui}$ considering the interaction that other users with similar preferences to u have assigned to item i. To find similar users, a measure of similarity p_{uv} is employed between their vectors. The similarity measure may be based on several similarity measures, such as Pearson correlation coefficient or cosine similarity. The final similarity measure is a retracted coefficient, s_{uv}:

$$s_{uv} = \frac{n_{uv}}{n_{uv} + \lambda_1} p_{uv}, \tag{2}$$

where n_{uv} is the number of items that users u and v have in common, and λ_1 is a regularization constant, set as 100 according to suggestions found in the literature [7].

Using the similarity values obtained, the algorithm identifies the k most similar users of u who evaluated item i, denoted as $S_u^k(i;v)$, and performs a score prediction based on the interactions of the k

similar users weighted by their similarity towards u [7]. Then the final score is predicted using the k most similar users through the Equation (3).

$$\hat{r}_{ui} = \frac{\sum_{v \in S^k(i;u)} s_{uv}}{Number\ of\ neighbors}, \tag{3}$$

4 PROPOSED METHOD

In this paper, we propose a technique capable of generating recommendations based on users groups' preferences. Our approach consists of a pre-processing step, responsible for combining users into groups according different type of interactions in the system with a particular item. In this way, the combination of assigning tags and user history during navigation, for example, can be made to improve the quality of the groups, since we can better represent the behavior of each user. The recommendation list for each user of a particular group is generated using a recommendation algorithm based on collaborative filtering. In this work, we adopted three algorithms: k-medoids, BPR MF and User kNN described in Section 3. The first one is used to generate groups based on the similarity among users, while the second and the third are used to generate recommendations based on interactions of each group.

Figure 3.1 illustrates all steps involved in our technique. Multiple users' interactions are captured and used to generate user vs. item matrices, where each cell in these matrices is a value containing the relevance feedback of each interaction type. These matrices are then used to compute the distance of users using some dissimilarity measure. After this step, we generated a single distance matrix resulting from the combination of the others. Then, we send this matrix to the clustering module to generate users' groups. Each of these clusters corresponds to users who have similar interests on particular subjects. Finally, based on these groups, we compute particular recommendations to each user in the database, using the navigation history (implicit feedback) of each group. The browsing history of each group was built using all kinds of interactions considered by the algorithm, making explicit interactions in binary form and removing duplicate entries.

Particularly in this paper, we adopted different types of implicit feedback, although our approach accepts explicit information too. As implicit feedback, we considered two types: i) whether a user assigned a tag or not to an item; and ii) his navigation history, inferred from other interactions (if the user has some kind of interaction, then he has viewed the item). As shown in Figure 3.1, we used the BPR MF algorithm to generate a personalized ranking for each user using a binary matrix (user per item), where the values will be one if user u made some interaction with an item i and zero otherwise. There are three phases in our technique: data representation; finding the nearest neighbor; and generating the lists of recommendations. The following subsections detail each of them.

4.1 Data representation

The algorithm inputs are represented by user × item interactions matrices, e.g if we consider ratings and tags we will have two input matrices. Each cell in the matrix represents the interactions made by users on items. Different methods can be used to represent interactions. Discrete values (such as 1, 2, 3, 4, 5) can be used to represent degrees of user's preferences towards the items; numerical values can be used to characterize the amount of times a user accessed an item; boolean values (such as 0 and 1) to represent whether a user assigned or not a tag on an item. Each cell of each matrix gets its respective type of interaction. If a user has not interacted with the corresponding item, its value in the matrix is 0, otherwise it will be specific for each type of interaction. For example, if the user explicitly rated an item, this value will be the provided rating; if he has only viewed the item, this value will be 1.

4.2 Finding the nearest neighbors

The dissimilarity between a user and other users is acquired based on their interactions. We use Cosine angle and Pearson correlation to calculate how two users are alike, considering all the interactions made by users on all items in the database. These metrics were chosen because: i) they discards the matrix cells that has no interaction, and ii) they are the metrics most commonly used in the area of recommender systems [1, 12]. This is particularly useful because we can not assume that users are similar based on the fact they have not interacted with certain items. For each interaction, we used these metrics to generate a new matrix of $M \times M$ users, which represents the dissimilarity among users. Cosine similarity between two users u and v is represented by Equation (4):

$$d_{(u,v)}^{cosine} = 1 - \frac{uv}{\|u\|_2 \|v\|_2}, \tag{4}$$

where $d_{(u,v)}$ is the distance of each type of interaction between two users, $\|*\|_2$ is the 2-norm of its argument $*$, and $u \cdot v$ is the dot product of u and v. The Pearson correlation is defined by:

$$d_{(u,v)}^{pearson} = 1 - \frac{\sum(u,v)}{\sigma_u \sigma_v}, \tag{5}$$

where Σ is the covariance between two users u and v, and σ is the standard deviation between them.

To combine the distances of each type of interaction in a single distance matrix, we computed a weighted average of the values, as shown in Equation (6).

$$d_{(u,v)}^{final} = \frac{1}{N_f} \sum_{n=1}^{|N_f|} \frac{1}{\alpha_n} d_n \tag{6}$$

where N_f is the number of interactions' types and α_* are variables used to weight each interaction type. It is defined as:

$$\alpha = \frac{N_{uv}(N_u + N_v)}{(N_u N_v)}. \tag{7}$$

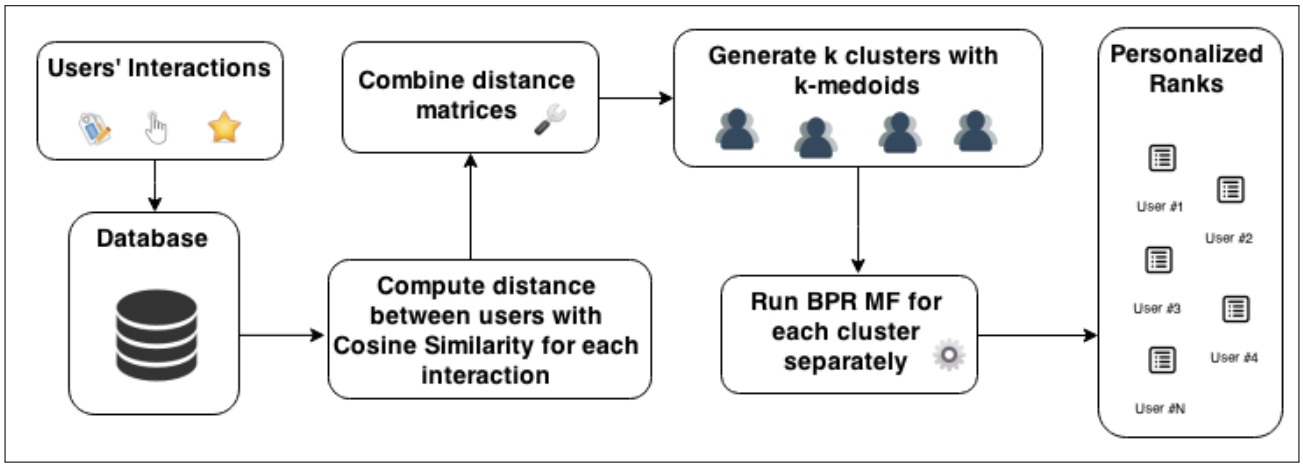


Figure 3. 1. Schematic visualization of the proposed technique.

In the equation above, N_u and N_v denote the number of interactions made by users u and v, respectively, and N_{uv} denotes the number of interactions in common of these users.

After computing the distance matrix, we use k-medoids presented in Subsection 3.2, to generate groups. Here, k data objects are selected randomly as medoids to represent k clusters and all remaining data objects are placed in a cluster having medoid nearest (or most similar) to that data object. After processing all data objects, a new medoid is determined which can represent cluster in a better way and the entire process is repeated. Again all data objects are bound to the clusters based on the new medoids. In each iteration, medoids change their location step by step; in other words, medoids move in each iteration. This process is continued until all medoids stop moving over each iteration. As a result, k clusters are found representing a set of n data objects.

4.3 Generating the lists of recommendations

We have used the state-of-the-art CF-based algorithms to process the interactions of each cluster and generate a list of recommended items for each user in that cluster. At this stage, each user receives personalized recommendations based on his behavior and his neighbors behavior. The algorithm is responsible for assembling a matrix that contains all users and items of a given group k and individual interactions of each user to predict items that he can enjoy, both highlighting the items he visited as the ones he has not. Finally, we concatenate the rankings generated for each user in a single ranking with all users. The proposed technique generates four values for k from training set of samples. In this step, the sample is divided into training and test in order to verify the precision and MAP values generated by this sample; then this procedure is repeated n times, returning the best values for k.

5 EVALUATION

To evaluate our approach we first check which of the dissimilarity metric provided the best accuracy in the proposed model using all the feedback types. Furthermore, we compare the algorithms having the best combination of interactions with CF-based algorithms, namely BPR MF [14] and User KNN [7], presented in Subsection 3.3. In this way, we execute the proposed approach applied to each Group-based BPR MF algorithm (GB-BPR MF) and Group-based User KNN (GB-User KNN), respectively. The algorithms implementation used in our work is available in the MyMediaLite library [3].

5.1 Datasets

The system evaluation was based on two datasets provided by Cantador et al. [3]. Last fm 2k consists of 92,834 user-listened artist relations, 186,479 interaction tags applied by 1,892 users to 17,632 artists. As feedback types, we considered: whether a user tagged an item or not; and the number of times the user has visited a particular item. MovieLens 2k consists of 10 million ratings, 100,000 interactions tags applied to 10,000 users and 72,000 movies. As explicit information, we used the ratings that users assigned to items to calculate the distance matrix, and as implicit information, we considered whether a user tagged an item or not and the navigation history to compute matrices and recommendations.

5.2 Methodology

We adapted the All But One [2] protocol for the construction of the ground truth and 10-fold-cross-validation. Given the data set, we randomly divided it into the same 10 subsets and for each sample we use $n-1$, these subsets of data for training and the rest for testing. The training set t_r was used to test the proposed technique and in the test set T_e we randomly separated an item for each user to create the truth set H. After that, the remaining items form the set of observable O, used to test the algorithm. To assess the outcomes of the systems we use evaluation metric Mean Average Precision (MAP) [15], as follows:

Mean Average Precision computes the precision considering the respective position of items in the ordered list. With this metric, we obtain a single accuracy score value for a set of test users T_e:

$$MAP(T_e) = \frac{1}{|T_e|} \sum_{j=1}^{|T_e|} AveP(R_j, H_j), \quad (8)$$

where R is the set of recommendations that the system computed, given the set of observables O, and the ground truth set H. The average precision (AveP) is given by

$$AveP(R_j, H_j) = \frac{1}{|H_j|} \sum_{r=1}^{|H_j|} [Prec(R_j, r) \times \delta(R_j(r), H_j)], \quad (9)$$

where $Prec(R_j, r)$ is the precision for all recommended items up to rank r and $\delta(R_j(r), H_j) = 1$, iff the predicted item at rank r is a relevant item ($R_j(r) \in H_j$) or zero otherwise.

We executed the clustering algorithm with different k values (between 2 and 30) and used the "knee finding" technique to choice the best number of clusters. For both datasets we deduce experimentally $k = 3$ as best value. For results we use Cosine Similarity, once our results indicate that this similarity is superior than the other measures such as Pearson Correlation, Jaccard measure, Euclidean measure that we tested.

In this work we used MAP@N, where N corresponds to the number of recommendations. We tested for the following values: 1, 3, 5 and 10 in the ranks returned by the system. For each configuration and measure, the 10-fold values are summarized by using mean and standard deviation. In order to compare the results in statistical form, we apply the two-sided paired t-test with a 95% confidence level [10].

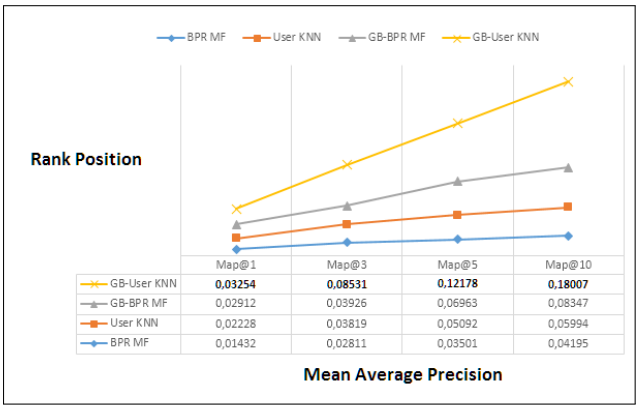
5.3 Results

The results of the experiments in each of the datasets are discussed in the following subsections.

5.3.1 Last fm 2k. Tables 3.1 and 3.2 show the results using only one type of interaction by user (navigation history and tag, respectively) to compute similarity between users. Figure 3.2 shows the results using the combination of both types of interactions in the proposed methods against the best results in the baselines. The best results are highlighted in bold.

We note that MAP has a tendency for higher values as the number of returned items increases. This can be explained because MAP only considers the relevant items and their positions in the ranking. Thus, as more items are returned, the number of relevant items is also increased.

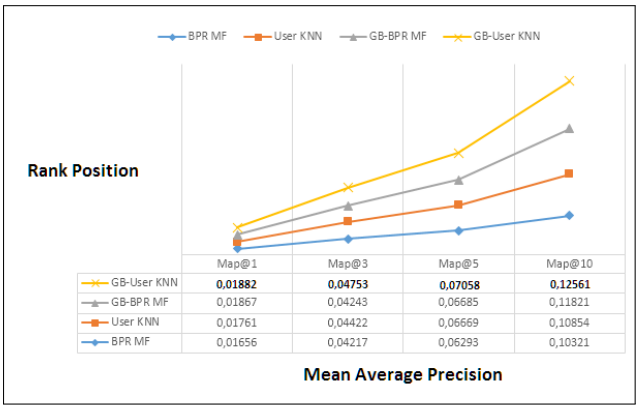
We observed that regardless of the number of groups or types of interactions, the use of clustering algorithms considerably improves the results of recommendation, proven by the t-student analysis ($p < 0.05$). This is because instead of generating recommendations through all interactions in the base, we use only related interactions with users' interactions, discarding items that the user probably would never use. The combination of the two types of feedback (navigation history and tag) resulted in better outcomes for all top N recommendations. This is because the more information a user has, the better we can relate them to other users with similar tastes and we can thus generate more accurate recommendations for him.



	Map@1	Map@3	Map@5	Map@10
GB-User KNN	**0,03254**	**0,08531**	**0,12178**	**0,18007**
GB-BPR MF	0,02912	0,03926	0,06963	0,08347
User KNN	0,02228	0,03819	0,05092	0,05994
BPR MF	0,01432	0,02811	0,03501	0,04195

Figure 3. 2. Graph and table comparing the MAP with two types of interaction (Last fm 2k).

5.3.2 MovieLens 2k. Tables 3.3 and 3.4 show the results of this evaulation in the MovieLens 2k dataset, using tags and ratings as interaction types. Figure 3.3 illustrates the algorithms' performance in Top@N vs. MAP graphs with two types of interactions.



	Map@1	Map@3	Map@5	Map@10
GB-User KNN	**0,01882**	**0,04753**	**0,07058**	**0,12561**
GB-BPR MF	0,01867	0,04243	0,06685	0,11821
User KNN	0,01761	0,04422	0,06669	0,10854
BPR MF	0,01656	0,04217	0,06293	0,10321

Figure 3. 3. Graph and table comparing the MAP with both types of interactions (MovieLens 2k).

The results demonstrate that the proposed technique was able to achieve better accuracy in two different domains (recommendation of artists and movies), proven by the t-student analysis ($p < 0.05$). Most of the results showed improvement in the accuracy of the recommendation, especially when using more than one type of feedback in the process. This emphasizes that we can represent user's behavior better when we have a large number of information about him.

The overall results obtained and described in the paper are small because of the evaluation protocol used in the experiments. The All But One hides one item from each user in the test set and considers it as the ground truth. As we are recommending top N items, the MAP will decrease because the system thinks there are N

Table 3. 1. Comparative MAP table using navigation history (Last fm 2k)

		Top 1	Top 3	Top 5	Top 10
BPR MF	MAP	0,01273	0.03342	0.04456	0.06259
	Standard deviation	0.000043	0.000014	0.000024	0.00011
User KNN	MAP	0.02175	**0.04721**	0.06206	0.07745
	Standard deviation	0.000087	0.000034	0.000217	0.000089
GB-BPR MF	MAP	0.01346	0.03065	0.04509	0.06803
	Standard deviation	0.000132	0.000187	0.000042	0.000131
GB-User KNN	MAP	**0.02374**	0.04573	**0.06532**	**0.07908**
	Standard deviation	0.000017	0.000123	0.000133	0.000054

Table 3. 2. Comparative MAP table using Tags (Last fm 2k)

		Top 1	Top 3	Top 5	Top 10
BPR MF	MAP	0.01538	0.04509	0.06631	0.10291
	Standard deviation	0.000127	0.000051	0.000126	0.000012
User KNN	MAP	0.02864	0.08381	**0.11724**	0.15542
	Standard deviation	0.000152	0.000082	0.000012	0.000215
GB-BPR MF	MAP	0.01754	0.04954	0.07165	0.12658
	Standard deviation	0.000041	0.000504	0.000145	0.000097
GB-User KNN	MAP	**0.03154**	**0.09014**	0.11297	**0.17541**
	Standard deviation	0.000026	0.000071	0.000259	0.000195

Table 3. 3. Comparative MAP table using navigation history (MovieLens 2k)

		Top 1	Top 3	Top 5	Top 10
BPR MF	MAP	0.01594	0.04208	0.06298	0.10226
	Standard deviation	0.000043	0.000014	0.000024	0.00011
User KNN	MAP	0.01809	0.04432	0.06206	0.10882
	Standard deviation	0.000087	0.000034	0.000217	0.000089
GB-BPR MF	MAP	0.01346	**0.04503**	0.06307	0.10903
	Standard deviation	0.000132	0.000187	0.000042	0.000131
GB-User KNN	MAP	**0.01915**	0.04398	**0.06612**	**0.11054**
	Standard deviation	0.000017	0.000123	0.000133	0.000054

Table 3. 4. Comparative MAP table using ratings (MovieLens 2k)

		Top 1	Top 3	Top 5	Top 10
BPR MF	MAP	0.00214	0.00681	0.01071	0.01842
	Standard deviation	0.000043	0.000014	0.000024	0.00011
User KNN	MAP	**0.00233**	0.00775	0.01147	0.02075
	Standard deviation	0.000087	0.000034	0.000217	0.000089
GB-BPR MF	MAP	0.00119	0.00678	**0.01164**	0.01893
	Standard deviation	0.000132	0.000187	0.000042	0.000131
GB-User KNN	MAP	0.00209	**0.00807**	0.01143	**0.02106**
	Standard deviation	0.000017	0.000123	0.000133	0.000054

relevant items, although the protocol has set only the hided item as relevant. In this way, it is important to rely only on the differences among the approaches, and we managed to increase the results of our proposal when compared to the baselines. As shown in the experiments, our approach is flexible and extensible to different algorithms and domains, although some of these configurations result in marginal improvements over the baselines (in particular KNN-based algorithms).

6 FINAL REMARKS

It is very crucial for a recommender system to have the capability of making accurate prediction by analyzing and retrieving customer's

preferences. Although collaborative filtering is widely used in recommender systems, some efforts to overcome its drawbacks have to be made to improve the prediction quality. Selecting the proper neighbors plays an important role to improving the prediction quality.

We have proposed a technique that yields better recommendation accuracy using users' groups, generated from the similarity between them. It considers different type of interactions of each user for customer's preferences and low similarities as well. The experimental results showed that our algorithm provides the better prediction accuracy than baselines in two datasets. The main advantages of our approach are extensibility and flexibility, once it enables developers to use different recommender and clustering algorithms, dissimilarity metrics and different types of feedback.

As future work, we plan to evaluate our approach with additional datasets from other domains in order to check the accuracy with different information types. Furthermore, we plan to consider community detection in graphs to select better users' groups to make the recommendation.

ACKNOWLEDGMENTS

We would like to acknowledge CAPES and CNPq for the financial support.

REFERENCES

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

[2] John S. Breese, David Heckerman, and Carl Kadie. 1998. Empirical Analysis of Predictive Algorithms for Collaborative Filtering. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence (UAI'98)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 43–52. http://dl.acm.org/citation.cfm?id=2074094.2074100

[3] Iván Cantador, Peter Brusilovsky, and Tsvi Kuflik. 2011. 2nd Workshop on Information Heterogeneity and Fusion in Recommender Systems (HetRec 2011). In *Proceedings of the 5th ACM conference on Recommender systems (RecSys 2011)*. ACM, New York, NY, USA.

[4] Arthur da Costa Fortes and Marcelo Garcia Manzato. 2014. Ensemble Learning in Recommender Systems: Combining Multiple User Interactions for Ranking Personalization. In *Proceedings of the 20th Brazilian Symposium on Multimedia and the Web (WebMedia '14)*. ACM, New York, NY, USA, 47–54. https://doi.org/10.1145/2664551.2664556

[5] Jiawei Han, Micheline Kamber, and Anthony K. H. Tung. 2001. Spatial Clustering Methods in Data Mining: A Survey. In *Geographic Data Mining and Knowledge Discovery, Research Monographs in GIS*, Harvey J. Miller and Jiawei Han (Eds.). Taylor and Francis. http://www-faculty.cs.uiuc.edu/~hanj/pdf/gkdbk01.pdf

[6] Taek-Hun Kim, Young-Suk Ryu, Seok-In Park, and Sung-Bong Yang. 2002. An Improved Recommendation Algorithm in Collaborative Filtering. In *Proceedings of the Third International Conference on E-Commerce and Web Technologies (EC-WEB '02)*. Springer-Verlag, London, UK, UK, 254–261. http://dl.acm.org/citation.cfm?id=646162.680356

[7] Yehuda Koren. 2010. Factor in the Neighbors: Scalable and Accurate Collaborative Filtering. *ACM Transactions on Knowledge Discovery from Data (TKDD)* 4, 1 (Jan. 2010), 1–24.

[8] Wenlong Li and Wei He. 2013. An Improved Collaborative Filtering Approach Based on User Ranking and Item Clustering. In *Internet and Distributed Computing Systems*, Mukaddim Pathan, Guiyi Wei, and Giancarlo Fortino (Eds.). Lecture Notes in Computer Science, Vol. 8223. Springer, 134–144.

[9] J. MacQueen. 1967. Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics*. University of California Press, Berkeley, Calif., 281–297.

[10] Thomas M. Mitchell. 1997. *Machine Learning* (1 ed.). McGraw-Hill, Inc., New York, NY, USA.

[11] Jong-Seok Lee Park, Hae-Sang and Chi-Hyuck Jun. 2006. A K-means-like Algorithm for K-medoids Clustering and Its Performance. *Proceedings of ICCIE (2006)* (2006), 102–117.

[12] Francesco R., Lior R., and Bracha S. 2011. Introduction to Recommender Systems Handbook. In *Recommender Systems Handbook*. Springer US, 1–35.

[13] Steffen Rendle. 2012. Factorization Machines with libFM. *ACM Trans. Intell. Syst. Technol.* 3, 3, Article 57 (May 2012), 22 pages.

[14] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2012. BPR: Bayesian Personalized Ranking from Implicit Feedback. *CoRR* abs/1205.2618 (2012).

[15] Ellen M. Voorhees and Donna K. Harman. 2005. *TREC: Experiment and Evaluation in Information Retrieval (Digital Libraries and Electronic Publishing)*. The MIT Press.

[16] Junhao Wen and Wei Zhou. 2012. An Improved Item-based Collaborative Filtering Algorithm Based on Clustering Method. In *Journal of Computational Information Systems*, Mukaddim Pathan, Guiyi Wei, and Giancarlo Fortino (Eds.). Lecture Notes in Computer Science, Vol. 8. Springer Berlin Heidelberg, 571–578.

3.2 Final Remarks

With this study, we started to explore the pre-processing of the data before using the recommender algorithms, using data clustering and heuristics, such as the calculation of the distance presented in the subsection of finding the nearest neighbors. As in previous chapter, we seek to incorporate different types of user feedback, to better understand their behavior in the system, as well as to enrich the input data with explicit information (e.g., ratings and tags).

In this chapter, we partially address the research questions 1, 2 and 3, "How can we improve the accuracy of the existing recommender algorithms using pre-processing approaches?", "How can we integrate different types of user's feedback to improve the quality of my input data?" and "How can we treat the input data to alleviate traditional recommender systems problems, such as cold-start, dimensionality and sparsity?", by proposing an approach that uses only a subset of the data generated by different types of feedback instead of the complete user-item matrix. With this approach, in addition to reducing the problem of dimensionality, indirectly we alleviate the cold-start problem, since we used additional types of users' feedback to compose the input to the recommendations and not only a single type (e.g. rating user-item matrix), such as traditional approaches.

A weak point of this approach is that by using the mean distances of different types of user's feedback, we lose the semantics and part of the representativeness of each type of interaction when making recommendation, since in the end we consider only one matrix generated from this mean distance. Therefore, relevant characteristics, such as rejection or affinity for certain items expressed by a particular type of interaction (mainly explicit), can be smoothed or even disregarded throughout the process, since the calculation of the mean is weighted by the number of interactions in each type of feedback and the number of implicit feedback interactions are more abundant.

In the next chapter, we will extend the current approach to support semantic information of the feedback and more advanced and reliable structured clustering approach and will describe how ensemble clustering techniques can help recommenders to improve its recommendations.

CHAPTER

4

ENSEMBLE CLUSTERING APPROACHES APPLIED IN GROUP-BASED COLLABORATIVE FILTERING SUPPORTED BY MULTIPLE USERS' FEEDBACK

4.1 Contextualization

In this chapter, we propose extended clustering approaches applied to different types of feedback using ensemble techniques. The proposed approaches can be used to solve one limitation of the last chapter, which is the lack of meaning information, by processing the different types of feedback separately, and combining them by well-known ensemble clustering strategies, such as Voting and Consensus Clustering strategies (STREHL; GHOSH, 2003; CHARKHABI; DHOT; MOJARAD, 2014).

As in the Chapter 3, the research questions 1, 2 and 3, "How can we improve the accuracy of the existing recommender algorithms using pre-processing approaches?", "How can we integrate different types of user's feedback to improve the quality of my input data?" and "How can we treat the input data to alleviate traditional recommender systems problems, such as cold-start, dimensionality and sparsity?", are also investigated. We pre-processed more than one type of user's feedback extracting the similarity of users according to each type, and applied these information in a clustering procedure followed by an ensemble approach. Finally, we recommend items for each group of users individually, considering only the items which interacted in that group.

Ensemble Clustering Approaches Applied in Group-based Collaborative Filtering Supported by Multiple Users' Feedback

Arthur F. da Costa[1], Marcelo G. Manzato[1] and Ricardo J. G. B. Campello[2,1]

[1] Institute of Mathematical and Computer Sciences
University of São Paulo, Brazil
{campello, fortes, mmanzato}@icmc.usp.br
[2] James Cook University, Queensland, Australia.
ricardo.campello@jcu.edu.au

Abstract. In this article, we extend our previous work based on group collaborative filtering to improve the quality of groups generated through clustering algorithms with different types of feedback. On the Web, users can interact with content in different ways, such as clicking, commenting or rating and recommender systems should be able to process and use all available information. A pre-processing step using ensemble clustering can be used to combine all this information to create a recommender with better accuracy. In this work, we propose the use of two ensemble clustering approaches to consider different types of feedback and to improve the quality of the recommendations. Experimental results on two different datasets demonstrate that the recommendation accuracy is significantly improved with our approaches when compared to well-known recommender algorithms and with our previous work.

Categories and Subject Descriptors: H.2 [**Database Management**]: Database Applications; H.3 [**Information Storage and Retrieval**]: Information Search and Retrieval

Keywords: Data clustering, Ensemble, Feedback, Recommender systems

1. INTRODUCTION

According to [Gantz and Reinsel 2012], from 2005 to 2020, the information in the digital universe will grow by a factor of 300, from 130 exabytes to 40,000 exabytes, or 40 trillion gigabytes (more than 5,200 gigabytes for every man, woman, and child in 2020). As there is too much information to process and to choose from, it is simply not possible/virtually impossible to grasp even a small percentage of it in a single lifetime. The expression *Information Overload* was introduced to describe the sensation of fatigue and distress that follows the cognitive surplus required to handle the volume of information we have to deal with everyday [Aggarwal 2016].

Recommender Systems (RS) have emerged in response to the information overload problem by learning about users' interests from their past action (ratings, votes, ranked lists, mouse clicks, browse history, product purchases, etc.) and suggesting products that are likely to fit their needs [Aggarwal 2016; Bobadilla et al. 2013]. Collaborative Filtering is one of the most popular and accurate methods used in Recommender Systems. In its simple form it has some limitations such as sparsity, overfitting and cold start [Aggarwal 2016]. Therefore, one solution is the use of hybrid systems, which combine collaborative filtering with additional information describing the contents of the items.

The continuous increase of information, feedback paradigms and content demands some requirements for recommender systems. First is the scalability, that is its ability to generate recommendations

Publication Information: Published in Journal of Information and Data Management (JIDM '17). 2017. Volume 8. Number 3. Pages 180-196. ISSN: 2178-7107.

quickly using the user-item rating matrix [Bobadilla et al. 2013; Aggarwal 2016], which are of huge dimensionality. Second is to find good items and to improve the quality of the recommendation for a user [Aggarwal 2016]. These two properties are in conflict, since the less time an algorithm spends filtering items, the more scalable it will be, but producing a worse item prediction. Furthermore, these systems typically explore only one type of feedback, discarding possible available knowledge found in other types of feedback. The literature reports a shortage of techniques which integrate different types of user feedback into a generic model [Aggarwal 2016; Bobadilla et al. 2013]. In addition, if different types of feedback are considered in these techniques, this task can increase the computational cost and cause problems in the final result.

Another problem is, once the final recommendation is generated based on the preferences of undistinguished users, that is, the recommender algorithm considers the interactions of all users of the dataset, the system may recommend items for users who have no interest in that selected content (e.g. in a music domain, the system could recommend a pop song for a user that likes metal). In addition, depending on the size of the dataset, the computational cost is very high in order to process all the information [Aggarwal 2016]. A possible solution to these problems would be constructing groups of users with similar interests prior to the recommendation, similarly to when recommending a content to a friend. For the person who receives the recommendation, it works as a filter or a particular view of a universe of possibilities usually inaccessible. It is also possible to make recommendations based on the opinions of others. As for instance, someone who is not an admirer of the jazz genre may be recommended based on what her/his friends enjoy, except this style which is unattractive for him.

In this context, we proposed in a previous work a technique to generate more accurate recommendations using groups of users with similar preferences, called Group-based Collaborative Filtering [da Costa et al. 2016]. We combined a variety of users' feedback types to obtain richer information about their interests. For instance, we captured and combined the ratings and the history of the users to understand and build their profiles based on their behavior in the system. Based on their profiles, we divided these users in groups of preferences considering their tastes. The recommendation list for each user of a particular group is generated using a traditional collaborative recommender algorithm. However, this process was done through a combination of all types of feedback, which can result in the loss of semantic information and the actual distribution of data in each type of feedback, since each type of feedback has its own characteristics (e.g. ratings are decimal numbers and explicitly express the taste of users, while the history are boolean numbers and represent whether or not users interacted with an item).

In this article, we extend our previous work to improve the quality of groups generated through ensemble clustering techniques, in order to maintain the characteristics of the data in the clustering process. We propose the use of two ensemble clustering techniques to combine the final results of each clustering algorithm based on each type of feedback. We evaluated our proposal by comparison with two well-known collaborative recommenders (User KNN and BPR MF), and then compared different types of recommendations generated from our previous approach to demonstrate the proposal's generality and efficiency. The experiments were executed with two real datasets from different domains: the first is MovieLens, which contains data from a movies reviews system; and the second is Last FM, a music website. Our study shows that the proposed group-based approach is able to provide better accuracy than individual recommenders and our previous work.

This article is structured as follows: Section 2 addresses the related work; Section 3 describes techniques which are explored in this work; Section 4 discusses our previous work related with this article; Section 5 presents the proposal in details; Section 6 reports the evaluation executed in the system; finally, Section 7 presents the final remarks and future works.

2. RELATED WORK

In this section, we review some work related to our proposal, as well as the main advantages of our work when compared to these related models. First, we depict approaches related to different types of feedback in recommender systems, and then, we provide a review of data clustering approaches in recommender systems.

2.1 Multiple Feedback Based Approaches

With the increasing number of feedback between users and content, several studies have emerged to work with the integration of these interactions, so that more information about users preferences are gathered by the systems. The recommendation systems can be extended in various ways in order to improve the understanding of users and items, including, for example, new types of interaction in the recommendation process and their combination [Aggarwal 2016; Bobadilla et al. 2013].

The SVD++ recommender algorithm proposed in [Koren 2010] uses explicit (ratings) and implicit (viewing history) information from users in a factorization model. Another factorization model, called Factorization Machines (FM) [Rendle 2012], can consider many types of information regarding users, items and/or their interactions. These techniques have the drawback that they process only certain types of interactions, with little capability of extension to other different types or they do not consider the semantics of the data and the context of each type of feedback. Costa and Manzato [da Costa Fortes and Manzato 2014] developed an ensemble recommender technique, called Ensemble BPR Learning, to unify different types of feedback from users, processed by different recommender techniques. While this model is extensible for any types of user's interaction, its learning phase has high computational cost, since it depends on the execution of several recommendation techniques beforehand and a post-processing step for learning weights to combine each feedback type ranking.

In a recent work, Peska [Peska 2016] proposed a recommender framework that aims to bridge the data sparsity problem and the lack of relevant feedback by modeling and utilizing enhanced sources of information, foremost implicit user feedback features. More specifically, his work focuses on the question how to define and collect multiple user feedback in scenarios, where we cannot invasively ask users to provide it. His results were able to improve quality of recommendations over both binary feedback baseline and uncontextualized feedback in terms of the evaluation metrics used in his study.

However, in these works the authors did not investigate the influence and usage of each contextual feature separately or the possibility to combine purchase probabilities coming from different learning methods or recommender algorithms.

2.2 Clustering Based Approaches

In this work, we have used data mining techniques to cluster users with similar preferences to generate recommendations based on their group. Several researchers have proposed recommender systems for on-line personalization through data mining to provide recommendation services. This kind of recommendation system is used to predict the user navigation behavior and their preferences using web log data.

Kim et al. [Kim et al. 2002] developed a recommender algorithm that uses the k-Means clustering to reduce the dimensionality of the data during the recommendation process. In their work, they used a graph approach to choose the best cluster with respect to a given test customer in selecting the neighbors with higher similarities as well as lower similarities. Their approach allows to explore the transitivity of similarity in a pre-processing step and considers the attributes of each item. The work developed by Wen and Zhou [Wen and Zhou 2012] presents an improved collaborative filtering recommendation algorithm based on dynamic item clustering method. A similarity threshold limits the similarity between clusters. By calculating the similarity between the current item and the cluster

center, it chooses the greatest similitude cluster, and then it finds the target items' nearest neighbors. Li et al. [Li and He 2013] propose a collaborative filtering recommender, which uses clustering methods based on users and items. Their approach consists of classifying and ranking the users in multiple item clusters by computing their rating qualities based on the previous rating records. In turn, the items are recommended for target users according to their similar users with high-ranks in different item categories. Experiments on public datasets have demonstrated the effectiveness of their algorithm.

In recent studies, Gupta and Patil [Gupta and Patil 2015] developed an algorithm using a hierarchical clustering algorithm along with a voting scheme to recommend the rating of a particular user with respect to a particular item on movies domain. In their approach, the users with their features are taken and are clustered into different groups using hierarchical clustering. Then, given a pair user and item, the rating prediction is done by mapping a user into a particular group she/he belongs to and then applying voting scheme for all users present in that cluster for a specific item. Katarya and Verma [Katarya and Verma 2016] also present a recommender system based on movies domain through data clustering and computational intelligence. In their research article, a novel recommender system makes use of k-means clustering by adopting cuckoo search optimization algorithm to find the best results based on the most suitable weight among all possible ones applied on the Movielens dataset. Their approach delivers better results than the baselines using mean absolute error, standard deviation, and root mean square error.

Our work is different from related work because we analyze various types of users' feedback on a particular item, in order to create groups of users with similar preferences and thus, a more accurate user's profile. In our previous work [da Costa et al. 2016], we proposed a pre-processing step, which was responsible for generating a single distance matrix weighted from different types of feedback. The advantages of this approach are the ease of extending the template for insertion of other types of feedback and the reduced time and computational processing, considering that the sets of data would be reduced to a single cluster before computing the recommendation. However, in this process we may end up missing important features, such as semantics and data distribution, in the clustering process. In the present work, our model is extended to support these characteristics, processing each type of feedback individually in the clustering algorithm and making only the combination of the final clustering result.

3. RELATED MODELS OVERVIEW

This study involves the pre-processing of data by means of data clustering techniques and the recommendation of items through collaborative filtering algorithms. The following sections present the main concepts covered in this article.

3.1 Notation

In this article, we use a consistent mathematical notation for referencing elements of the recommender system works. The feedback matrix is denoted by $\mathbf{R}$, with r_{ui} being the explicit or implicit interaction that user u provided for item i. In the first case, it is an integer provided by the user indicating how much she/he liked the content; in the second, it is just a boolean indicating whether the user consumed or visited the content or not. The set of pairs (u, i) for which known interaction in $\mathbf{R}$ are represented by the set $K = \{(u, i)|r_{ui} \text{ is known}\}$. $N(u)$ is the set of items for which user u provided a feedback, $\bar{N}(u)$ to indicate the set of items that are unknown to user u and K. Finally, the prediction of the recommender algorithm for the pair (u, i) is represented by $\hat{r}_{ui}$, which is a floating point value guessed by the recommender algorithm.

3.2 Clustering Approaches

Clustering is the assignment of objects into clusters, so that objects from the same cluster are more similar to each other than objects from different clusters [Estivill-Castro 2002]. Often similarity is assessed according to a distance measure, e.g. Euclidean, Cosine, Correlation, Jaccard, etc. Clustering is a common technique for statistical data analysis, which is used in many fields, including recommender systems. In the following subsections we present the main clustering concepts related to this article.

3.2.1 *K-Medoids.* This algorithm is a partition-based clustering algorithm based on k-means. It attempts to minimize the distance between points labeled to be in a cluster and a point designated as the center of that cluster [Park and Jun 2006]. The k-medoids algorithm differs from k-means because it chooses datapoints as centers (medoids or exemplars) and works with an arbitrary matrix of distances between these datapoints, besides minimizing a sum of pairwise dissimilarities instead of a sum of squared Euclidean distances [Park and Jun 2006]. A medoid can be defined as the object of a cluster whose average dissimilarity to all the objects in the cluster is minimal, i.e., it is the most centrally located point in the cluster. The implementation of the k-medoids used in this article is explained as follows:

K-Medoid Clustering Algorithm

(1) Initialization: randomly select (without replacement) k of the n data points as the medoids.
(2) Associate each data point to the closest medoid.
(Using a dissimilarity measure like cosine, Pearson, etc.)
(3) For each medoid m
—For each non-medoid data point o
—Swap m and o and compute the total cost of the configuration
(4) Select the configuration with the lowest cost.
(5) Repeat steps 2 to 4 until there is no change in the medoid.

The advantage of this k-medoids implementation is that large datasets can be efficiently classified and its convergence is proved regardless of the dissimilarity measure. Furthermore, it is reliable in theory, simpler and faster than k-means, since it does not calculate the distance between data points. [Park and Jun 2006]. In this study, we will use K-medoids to generate the most similar groups of users, in order to reduce the dimensionality and sparsity of the data, since this process will cause the recommendation to be computed based only on the group's preferences of each user.

3.2.2 *Ensemble Clustering.* Strehl and Ghosh [Strehl and Ghosh 2003] introduce the problem of ensemble multiple clusters of a set of objects into a single consolidated group without accessing the features or algorithms that determined these partitionings. This approach, also called consensus clustering or aggregation of clustering, refers to the situation in which a number of different clusterings have been obtained for a particular dataset and it is desired to find a single clustering which is a better fit in some sense than the existing clusterings [Punera and Ghosh 2008].

In Strehl and Ghosh work [Strehl and Ghosh 2003], they discuss three approaches towards solving this problem to obtain high quality consensus functions, which have low computational costs. This characteristic makes them feasible to evaluate each of the techniques discussed below and find the best results by comparing the solution against the objective function. Their proposed hard ensemble clustering methods are Cluster-based similarity partitioning algorithm (CSPA), Hyper-graph partitioning algorithm (HGPA) and Meta-clustering algorithm (MCLA). In CSPA the similarity between two datapoints is defined to be directly proportional to the number of constituent clusterings of the ensemble in which they are clustered together [Punera and Ghosh 2008]. The main idea is that the more related

two data-points are the higher is the chance of belonging to the same cluster. This similarity graph between data-points is partitioned using METIS* to obtain the desired number of clusters. CSPA is the simplest heuristic, but its computational and storage complexity are both quadratic in n [Punera and Ghosh 2008].

In turn, according to Punera and Gosh [Punera and Ghosh 2008], the HGPA algorithm takes a very different approach to find the consensus clustering than the previous approach,in which partitioning the hypergraph by cutting a minimal number of hyperedges was formulated. Finally, the MCLA algorithm is based on clustering clusters, where each cluster is also represented as a hyperedge. The algorithm groups and collapses related hyperedges into k clusters; and then assigns each data point to the collapsed hyperedge in which it participates most strongly [Punera and Ghosh 2008].

In this article we evaluate the performance of some of these ensembles techniques on more complex real-life datasets in the recommender systems' context. One advantage of these approaches is that they enable us to conveniently model final clusters of different sizes via priors in the mixture model. Graph partitioning methods tend to yield roughly balanced clusters. However, this is a disadvantage in situations where the data distribution is not uniform.

3.3 Recommendation Algorithms

We evaluate our proposal with two recommender algorithms: a neighborhood-based [Koren 2010; Aggarwal 2016] and a matrix factorization [Rendle et al. 2012] approach. The following subsections detail each algorithm.

3.3.1 *BPRMF.* This approach [Rendle et al. 2012] consists of providing personalized ranking of items to a user according only to implicit feedback (e.g. navigation, clicks, etc.). An important characteristic of this type of feedback is that we only know the positive observations; the non-observed user-item pairs can be either an actual negative feedback or simply the fact that the user does not know about the item's existence. The authors have proposed a generic method for learning models for personalized ranking, where instead of training the model using only the user-item pairs, they also consider the relative order between a pair of items, according to the user's preferences [Rendle et al. 2012]. It is inferred that if an item i has been viewed by user u and j has not ($i \in N(u)$ and $j \in \bar{N}(u)$), then $i >_u j$, which means that she/he prefers i over j. Each user u is associated with a user-factors vector $p_u \in \mathbb{R}^f$, and each item i with an item-factors vector $q_i \in \mathbb{R}^f$, where f is the number of factors in each vector.

It is important to mention that when i and j are unknown to the user, or equivalently, both are known, then it is impossible to infer any conclusion about their relative importance to the user. To estimate whether a user prefers an item over another, Rendle et al. proposed a Bayesian analysis using the likelihood function for $p(i >_u j|\Theta)$ and the prior probability for the model parameter $p(\Theta)$. The final optimization criterion, BPR-Opt, is defined as:

$$\text{BPR-Opt} := \sum_{(u,i,j)\in D_K} \ln \sigma(\hat{s}_{uij}) - \Lambda_\Theta ||\Theta||^2 \quad , \tag{1}$$

where $\hat{s}_{uij} := \hat{r}_{ui} - \hat{r}_{uj}$ and $D_K = \{(u,i,j)|i \in N(u)\ \&\ j \in \bar{N}(u)\}$. The symbol Θ represents the parameters of the model, Λ_Θ is a regularization constant, and σ is the logistic function, defined as: $\sigma(x) = 1/(1+e^{-x})$.

For learning the model, the authors also proposed a variation of the stochastic gradient descent technique, denominated LearnBPR, which randomly samples from D_K to adjust Θ.

*http://glaros.dtc.umn.edu/gkhome/metis/hmetis/overview

3.3.2 *User KNN.* This recommendation algorithm is the well-known User KNN, whose details can be found in [Koren 2010; Aggarwal 2016]. We adopted this algorithm because of its well-acceptance, and because it can be intuitively extended to include other information. The main goal of the algorithm is to find similar users and predict the best items for them based on their similar items.

In this way, a score is predicted for a unknown user-item pair $\hat{r}_{ui}$ considering the interaction that other users with similar preferences to u have assigned to item i. To find similar users, a measure of similarity p_{uv} is employed between their vectors. The similarity measure may be based on several similarity measures, such as Pearson correlation coefficient or cosine similarity. The final similarity measure is a retracted coefficient, s_{uv}:

$$s_{uv} = \frac{n_{uv}}{n_{uv} + \lambda_1} p_{uv}, \tag{2}$$

where n_{uv} is the number of items that users u and v have in common, and λ_1 is a regularization constant, set as 100 according to suggestions found in the literature [Koren 2010].

Using the similarity values obtained, the algorithm identifies the k most similar users of u who evaluated item i, denoted as $S_u^k(i; v)$, and performs a score prediction based on the interactions of the k similar users weighted by their similarity towards u [Koren 2010]. Then the final score is predicted through the Equation (3).

$$\hat{r}_{ui} = \frac{\sum_{v \in S^k(i;u)} s_{uv}}{k} \tag{3}$$

4. GROUP-BASED COLLABORATIVE FILTERING APPROACH

In our previous work [da Costa et al. 2016], we proposed an approach capable of generating recommendations based on users groups' preferences. This approach consisted of a pre-processing module, responsible for combining users into groups according to different types of feedback. In this way, the combination of tags assignments and user history during navigation, for example, could be made to improve the quality of the groups, since they can better represent the behavior of each user. The recommendation list for each user of a particular group is generated using a well-known recommender algorithm based on collaborative filtering.

In a first step, multiple users' feedbacks are captured and used to generate user vs. item matrices, where each cell in these matrices is a value containing the relevance of each feedback type. These matrices are then used to compute the distance of users using some dissimilarity measure. After this step, the algorithm generates a single distance matrix resulting from the combination of the others. Then, this matrix is used by the clustering module to generate users' groups. Each of these clusters corresponds to users who have similar interests on particular subjects. Finally, based on these groups, particular recommendations are computed to each user in the dataset, through well-known recommender algorithms based on collaborative filtering, using the navigation history (implicit feedback) of each group. The browsing history of each group is built using all kinds of interactions considered by the algorithm, making explicit interactions in binary form and removing duplicate entries. There are three phases in our approach: data representation; data clustering; and recommendation. The following subsections detail each of them.

4.1 Data representation

The algorithm inputs are represented by user × item interactions matrices, so if the system's data are compared by two types of feedback (e.g. ratings and tags assignments), we will have two input

matrices. Each cell in the matrix represents the interactions made by users on items. Different methods can be used to represent interactions. Discrete values (such as 1, 2, 3, 4, 5) can be used to represent degrees of user's preferences towards the items; numerical values can be used to characterize the amount of times a user accessed an item; boolean values (such as 0 and 1) to represent whether a user assigned or not a tag on an item. In this way, each cell of each matrix gets its respective type of interaction (explicit or implicit). If a user has not interacted with the corresponding item, its value in the matrix is 0, otherwise it will be specific for each type of interaction. For example, if the user explicitly rated an item, this value will be the provided rating in an explicit feedback matrix; if she/he has only viewed the item, this value will be 1 in an implicit feedback matrix.

4.2 Data Clustering

Data Clustering is the process of grouping a set of objects into clusters so that objects within a cluster are similar to each other but are dissimilar to objects in other clusters [Han et al. 2001]. The similarity between a user and other users is acquired based on their interactions. In this work were used Cosine angle and Pearson correlation to calculate how two users are alike, considering all the interactions made by users on all items in the database. These metrics were chosen because: i) they discard the matrix cells that have no interaction, and ii) they are the most commonly used metrics in the area of recommender systems [Bobadilla et al. 2013; Aggarwal 2016]. This is particularly useful because we can not assume that users are similar based on the fact they have not interacted with certain items. For each feedback, these metrics were used to generate a new matrix of $M \times M$ users, which represents the dissimilarity among users. To combine the distances of each type of interaction in a single distance matrix, a weighted average of the values was computed, as shown in Equation (4).

$$d_{(u,v)}^{final} = \frac{1}{N_f} \sum_{n=1}^{|N_f|} \frac{1}{\alpha_n} d_n \tag{4}$$

where N_f is the number of types of feedback, d_n is the distance calculated based on each type of interaction and α_* are variables used to weight each interaction type, which are defined as:

$$\alpha = \frac{N_{uv}(N_u + N_v)}{(N_u N_v)}. \tag{5}$$

In the equation above, N_u and N_v denote the number of interactions made by users u and v, respectively, and N_{uv} denotes the number of interactions in common of these users.

After computing the distance matrix, the approach used k-medoids, presented in Section 3.2.1, to generate groups. In this way, in this work k data objects are selected randomly as medoids to represent k clusters and all remaining data objects are placed in a cluster having a medoid nearest or most similar to that data object. In the next step, a new medoid is determined which can represent the cluster in a better way and the entire process is repeated. Again all data objects are bound to the clusters based on the new medoids. In each iteration, medoids change their location step by step; in other words, medoids move in each iteration. This process is continued until all medoids stop moving over each iteration. As a result, k clusters are found representing a set of n data objects.

4.3 Recommendation

In this step, well-known CF-based algorithms were used to process the interactions of each cluster and generate a list of recommended items for each user in that cluster. At this stage, each user

receives personalized recommendations based on her/his behavior and her/his neighbors behavior. The algorithm is responsible for assembling a matrix that contains all users and items of a given group k and individual interactions of each user to predict items that she/he can enjoy, both highlighting the items she/he visited as the ones she/he has not. Finally, the individual recommendations are concatenated in a single ranking with all users, which contains pairs (u, i) sorted by scores generated by the recommenders. The proposed technique generates four values of k from the training set of samples. In this step, the sample is divided into training and test in order to verify the precision and MAP values generated by this sample; then this procedure is repeated n times, returning the best values for k.

5. PROPOSED APPROACHES

In our previous work, the process was done through a combination of all types of feedback before a clustering step in a merge of distance matrices, which can result in the loss of semantic information and the actual distribution of data in each type of feedback. Each type of feedback has its own characteristics and can be expressed in different ways, for example ratings are decimal numbers and explicitly express the taste of users, while the history are boolean numbers and represent whether or not users interacted with an item. The number of interactions and semantic value attributed by the user in each type of feedback can directly influence the representation of her/ his behavior during the clustering and the recommendation processes and the combination of these interactions prior to the clustering algorithm can generate noise and data distortion, since the distance matrix is made only with heuristic weighting.

In this article we extend the work presented in Section 4, in order to conserve the considered characteristics of each type of feedback during the clustering process. In this way, we propose a generic ensemble clustering to combine multiple feedback types extracted from datasets and we also use ensemble clustering algorithms well-known in the literature [Strehl and Ghosh 2003]. The ensembles are accomplished in a pre-processing module based on clustering approach, which combine the outputs generated by a clustering algorithm using individual users' feedback types. Our proposed approach allows the use of different types of clustering and recommendation algorithms, which makes it generic and extensible. Figure 4.1 shows a representation of the proposed approach.

In this article, we process each feedback type in a separate way, combining only the final result of the clusterings. In this way, we are able to preserve the characteristics and distribution of the data of each type of feedback in the clustering process. In Step (1), we build a user vs. item representation in the same way as in previous work, where each value in these representations is a score representing each feedback type. These matrices are then used to compute the dissimilarity of users using some distance measures. Then, each distance matrix is processed by a clustering algorithm, which is responsible for generating groups of users based on their preferences.

Then, in Step (2), the results generated by the clustering algorithm for each type of interaction are combined in an ensemble clustering process, in order to improve the quality of the generated groups. In this article, we propose the use of two strategies of ensemble clustering. The first ensemble clustering technique is based on heuristics, which combines individual results of each feedback generated by clustering algorithm using vote strategy, where the most common group label for each user in the results of each approach is considered the real label group of that user. For a more reliable and robust tool, the second technique was based on CSPA, HGPA and MCLA [Strehl and Ghosh 2003], called Consensus Clustering Based Strategy, in which a user's label is defined by well-known clustering ensembles strategy.

Finally, based on the groups defined by ensemble clustering strategies, we compute particular recommendations to each user in the dataset, using all feedback of each group int the Step (3). The main difference of this new approach is the implementation of an ensemble process after the clustering

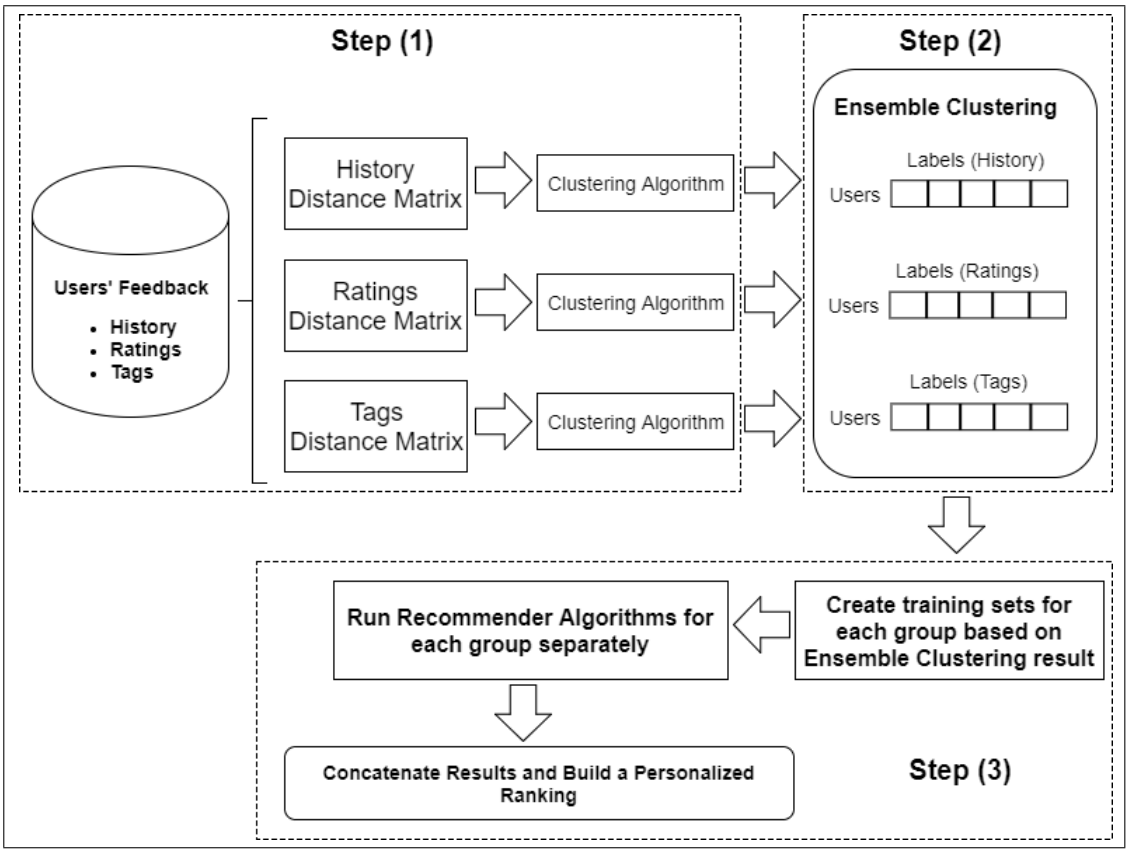


Figure 4.1. Schematic visualization of the proposed approach.

step, instead of a weighted combination of the distance matrices of each type of feedback. Thus, the other steps of the approach are the same as those presented in Section 4 and the proposed ensemble strategies are detailed in the following subsections.

5.1 Voting Strategy

In this ensemble technique, the target is to combine multiple clustering solutions or partitions of a set into a single consolidated clustering that maximizes the information shared among all available clustering solutions. We are looking for a final partition $C*$ of a given dataset $\{user_1, ..., user_N\}$ into k labels which optimally represent a given set of M partitions of the different feedback sets. Each of these M partitions is generated by a clustering algorithm for a given type of feedback and is represented by a vector with N positions, where each cell of this vector contains the label on which a particular user u belongs to. In this way, we create an auxiliary matrix $A_{N\times k}$ to count the number of votes of each label for each user taking into account all types of interaction. The element a_{ij} is the degree of membership of $user_u$ to the j-th label of the m-th partition. For every type of feedback M, a element a_{ij} in matrix A is defined by:

$$a_{uj} = a_{uj} + \begin{cases} 1 & \text{if } user \text{ is assigned in cluster } j, \\ 0 & \text{otherwise.} \end{cases} \tag{6}$$

The final partition $C*$ is encoded as a vector with elements c_u, where each cell is the ensemble clustering result of the combination of the M feedback types, defined by:

$$c_{user_u} = \arg\max a_u \tag{7}$$

The main idea of this approach is to keep the most common label assigned by the clustering algorithm to a given user, since the repetition of this assignment in more than one type of feedback provides a greater chance of a given user to be part of a group k. Figure 4.2 illustrates an example for a given user u, which interact with the system with ratings, tags and history.

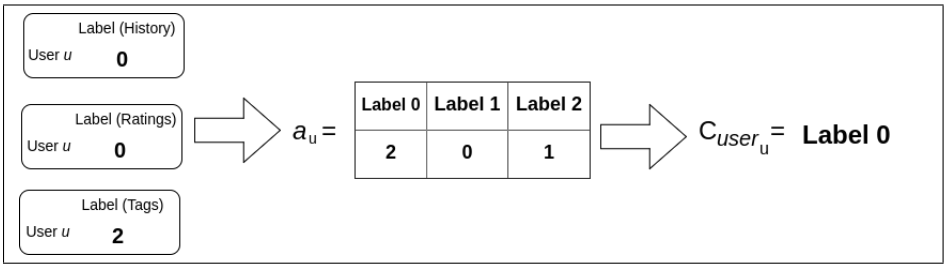


Figure 4.2. Example of Ensemble based on Voting Strategy.

Once the clustering algorithm is applied to each type of feedback, our approach evaluates the groups that were assigned to the user i and then assigns him/her to the group in which he/she had the most votes based on all considered feedback (in the case of the example in Figure 4.2, user u is assigned to the group 0).

5.2 Consensus Clustering Strategy

In this ensemble approach, we aggregate the clustering information, searching for a consensus clustering that is on average the most consistent with the different M partitions in the ensemble, using a strategy proposed by Strehl and Ghosh [Strehl and Ghosh 2003]. For a population of n cells, the similarity between a pair of clusterings, λ^a and λ^b , which contains k^a and k^b clusters respectively, is quantified by the normalized mutual information (NMI), defined as:

$$\phi^{(NMI)}(\lambda^a, \lambda^b) = \frac{\sum_{h=1}^{k^a}\sum_{l=1}^{k^b} n_{h,l} log(\frac{n \cdot n_{h,l}}{n_h^a n_l^b})}{\sqrt{(\sum_{h=1}^{k^a} n_h^a log(\frac{n_h^a}{n}))(\sum_{l=1}^{k^b} n_l^b log(\frac{n_l^b}{n}))}} \quad , \tag{8}$$

where n_h^a and n_l^b denote the numbers of cells in the corresponding clusters, and $n_{h,l}$ stands for the number of cells in their intersection.

For an ensemble of M partitions, λ^1, ... , λ^M, the consensus clustering $\lambda^* = \{C_1^*, ..., C_K^*\}$ is defined as the one that maximizes the average NMI with the M partitions in the process. The final result is computed by combining three approximation algorithms, CSPA, HGPA and MCLA, and selecting the one that performs the best [Strehl and Ghosh 2003].Thus, each algorithm is executed n times, and generates an approximate solution for each approach. Finally, one chooses the best result among all approaches.

6. EVALUATION

To evaluate our proposal, we first compare our ensemble techniques against each individual collaborative recommender. This comparison was carried out regarding each feedback type. Then, we compare the different types of ensembles against our previous work [da Costa et al. 2016], in order to

demonstrate which one provides the best results. This evaluation was executed for both considered collaborative recommenders, User KNN and BPRMF, the clustering algorithm, K-Medoids, and two datasets, as follows.

6.1 Datasets

The system evaluation was based on two datasets provided by Cantador et al. [Cantador et al. 2011]. Last fm 2k consists of 92,834 user-listened artist relations, 186,479 interaction tags applied by 1,892 users to 17,632 artists. As feedback types, we considered: whether a user tagged an item or not (e.g. a user annotate an item with freely chosen keywords based on her/his taste); and the number of times the user has visited a particular item. MovieLens 2k consists of 10 million ratings, 100,000 interactions tags applied to 10,000 users and 72,000 movies. As explicit information, we used the ratings that users assigned to items to calculate the distance matrix, and as implicit information, we considered whether a user tagged an item or not and the navigation history to compute matrices and recommendations.

6.2 Used Resources

The recommender approach proposed in this article was developed in Python[**] version 2.7, with NumPy[***] and SciPy[†] libraries, responsible for data optimization and matrix structures. Its source-code is freely available on Github[‡].

The recommender algorithms integrated into the tool belong to MyMediaLite library [Gantner et al. 2011] and Case Recommender [da Costa Fortes and Manzato 2016], which are open-source tools, developed in $C\#$ and Python, with numerous features and algorithms for recommender systems. Among the algorithms implemented are User KNN and BPR MF, both used in this study. For Consensus Clustering, we used an ensemble clustering Python library, called Cluster Ensembles version 1.16 [§].

6.3 Experimental Setup and Evaluation Metric

We adopted the All But One [Breese et al. 1998] protocol for the construction of the ground truth and 10-fold cross-validation. Given the data set, we randomly divided it into the same 10 subsets and for each sample we use $n-1$, these subsets of data for training and the rest for testing. The training set t_r was used to test the proposed technique and in the test set T_e we randomly separated one item for each user to create the truth set H. In this way, the truth set H will have only one item for each user of the original test set and the evaluation of the generated rankings will be made using this set. To assess the outcomes of the systems we use the evaluation metric Mean Average Precision (MAP) [Voorhees and Harman 2005], as follows:

Mean Average Precision computes the precision considering the respective position of items in the ordered list. With this metric, we obtain a single accuracy score value for a set of test users T_e:

$$MAP(T_e) = \frac{1}{|T_e|} \sum_{j=1}^{|T_e|} AveP(R_j, H_j), \tag{9}$$

[**]http://www.python.org/
[***]http://www.numpy.org/
[†]http://www.scipy.org/
[‡]http://www.github.com/ArthurFortes/CaseRecommender/
[§]https://pypi.python.org/pypi/Cluster_Ensembles/1.16

where R is the set of recommendations that the system computed, given the set of observables O, and the ground truth set H. The average precision (AveP) is given by

$$AveP(R_j, H_j) = \frac{1}{|H_j|} \sum_{r=1}^{|H_j|} [Prec(R_j, r) \times \delta(R_j(r), H_j)], \tag{10}$$

where $Prec(R_j, r)$ is the precision for all recommended items up to rank r and $\delta(R_j(r), H_j) = 1$, if the predicted item at rank r is a relevant item ($R_j(r) \in H_j$) or zero otherwise.

In this work we used MAP@N, where N corresponds to the number of recommendations. We tested for the following values: $1, 3, 5$ and 10 in the ranks returned by the system. For each configuration and measure, the 10-fold values are summarized by using mean and standard deviation. In order to compare the results in statistical form, we apply the two-sided paired t-test with a 95% confidence level [Mitchell 1997].

Regarding the parameters of the algorithms, we defined a set of values which performed better for both datasets, HetRec MovieLens 2k and LastFm. Such definitions were made by cross-validation in the training set and some techniques like "Knee Finding" to choose the best number of clusters [Estivill-Castro 2002]. For the k-Medoids algorithm, we ran experiments for k equals to 3, 5, 7, 10, 30, 50 and 100 neighbors, and chose k equals to 3 as it provided the best results. To execute Consensus Clustering Strategy, we ran experiments for n equals to 2, 5, 10 and 15 times, and chose $n = 5$ as it provided the best results. For the BPRMF algorithm, we ran experiments for the latent factors 10, 40, 70 and 100, and chose the number of 40 latent factors as it presented the best results for this algorithm. For User KNN, we ran experiments for k equals to 30, 50, 80 and 100 neighbors, and chose k equals to 50 as it provided the best results. For User KNN and K-Medoids, we used Cosine Similarity to generate the distance and similarity matrices, respectively, once the results indicate that this similarity is superior than the other measures such as Pearson Correlation and Jaccard measure that we tested. For other parameters, we use default values of each recommender tool and library chosen.

For the final results, once the medoids are chosen randomly in the clustering algorithm, we run ten times each experiment and choose the best value generated. In addition, we restricted the clusters to have no less than 10 users, since we assume that this is the minimum value to generate a good neighborhood, as well as a good recommendation for a given user.

6.4 Results

In our previous work, we compared the Group-based approach using each type of feedback with well-known recommender algorithms, which were described in Section 3.3. Then, we compared our approach using all types of feedback with each recommender algorithm using a training set, which was built based on the concatenation of the training sets of each feedback, demonstrating the accuracy improvement of the proposed approach in relation to baselines.

In this article, the results of the experiments in each of the datasets are discussed in the following subsections, in which we first compare our proposed approach using all interactions to each recommender algorithm executed with each type of feedback and then compare our approach to our previous work using all types of feedback.

6.4.1 *Last Fm 2k.* In this experiment, we considered two different types of implicit feedback available in the LastMF 2k dataset, history and tags. Table 4.1 shows the results using the combination of both types of feedback considered in this dataset (history and tags) against the best results in the baselines trained with each feedback separately. Each ensemble clustering and Group-based approach

result is applied to the chosen recommender algorithms (User KNN and BPRMF). The best results are highlighted in bold.

Table 4.1. Comparison among Ensemble Clustering approaches and recommender algorithms in terms of Map@N (Last Fm 2k).

Algorithms	Feedback	Map@1	Map@3	Map@5	Map@10
User KNN					
Traditional Recommender	History	0.101986	0.208803	0.271068	0.358561
	Tags	0.040257	0.096081	0.137412	0.205045
Group-Based	History and Tags	0.117552	0.223832	0.278582	0.365002
Voting Strategy	History and Tags	0.124530	0.243156	**0.301127***	0.387546
Consensus Clustering	History and Tags	**0.127750***	**0.244766**	0.298443	**0.390164***
BPRMF					
Traditional Recommender	History	0.052603	0.104133	0.1438539	0.213097
	Tags	0.030595	0.059044	0.088031	0.132581
Group-based	History and Tags	0.071853	0.134971	0.180891	0.256575
Voting Strategy	History and Tags	0.082662	0.151905	**0.205045***	0.273752
Consensus Clustering	History and Tags	**0.085346***	**0.155126***	0.201825	**0.276435***

Bold typeset indicates the best performance. * indicates statistical significance at $p < 0.01$ pairwise compared to the other results.

As it can be seen, the ensemble approaches based on the different types of feedback provided the best results in the vast majority of cases. In both datasets and algorithms, we could achieve a gain by enriching the representations with different types of feedback. The most robust model, Consensus Clustering, provided the best results, giving substantial indication that, for the clustering scenarios addressed, it can find suitable similarities between groups.

6.4.2 *MovieLens 2k.* This dataset contains several implicit and explicit feedback. For this experiment, we used as explicit feedback the ratings and as implicit feedback the history and tags assignments. The same experiments were performed in the MovieLens dataset and the results are presented on Table 4.2.

Table 4.2. Comparison among Ensemble Clustering approaches and recommender algorithms in terms of Map@N (MovieLens 2k).

Algorithms	Feedback	Map@1	Map@3	Map@5	Map@10
User KNN					
Traditional Recommender	History	0.023300	0.067047	0.100808	0.161184
	Ratings	0.029481	0.070375	0.105087	0.163086
	Tags	0.003804	0.013789	0.021398	0.038040
Group-Based	History, Ratings and Tags	0.026628	0.072753	0.108416	0.172610
Voting Strategy	History, Ratings and Tags	0.029481	**0.075130***	0.109843	0.166428
Consensus Clustering	History, Ratings and Tags	**0.030153***	0.071326	**0.113185***	**0.179365***
BPRMF					
Traditional Recommender	History	0.014741	0.050404	0.092249	0.147883
	Ratings	0.021398	0.061816	0.092724	0.154065
	Tags	0.002377	0.007608	0.013789	0.029481
Group-Based	History, Ratings and Tags	0.026153	0.068473	0.097479	0.150016
Voting Strategy	History, Ratings and Tags	0.026153	**0.069900***	0.097479	0.156916
Consensus Clustering	History, Ratings and Tags	**0.031383***	0.069473	**0.109161***	**0.164051***

Bold typeset indicates the best performance. * indicates statistical significance at $p < 0.01$ pairwise compared to the other results.

Again, one can see that the representations that used ensemble clustering strategies provided the best results when compared to traditional approaches. Considering the results in the MovieLens

2k dataset, the same effect happens when we used ensemble clustering approaches compared with traditional recommender and with our previous work. This happens because the descriptive power of the enhanced representations of each type of feedback applied separately in a clustering algorithm is superior to the normal representation, giving better close-related neighbors in each feedback and the data semantics.

6.5 Analysis and Discussion

As detailed in the previous sections, the use of ensemble clustering approaches provides better results than combining distance matrices and processing individual feedback for most cases, which indicates that our proposal has potential to improve results provided by recommender systems. In Tables 1 and 2, we see that by using the history and ratings, we obtain better values of MAP than using tags in each recommender algorithm separately. But, if we combine all feedback types using our proposed clustering ensemble approaches, the results are even better than the previous combination. We can also see in the experiments that the proposed approaches using different types of feedback provide better results than a collaborative filtering based on matrix factorization (BPRMF) and neighborhood (User KNN), demonstrating that by using different types of feedback we can provide better recommendations.

We also notice that the ensemble clustering approaches proposed in this article provided the best results for both recommender algorithms than our previous work. Consensus clustering approach provided the best results for the BPRMF and User KNN algorithms for the two datasets, while Group-based approach provided the worst results in most of the cases when compared to ensemble approaches. However, as it can be seen in our previous work [da Costa et al. 2016], the group-based approach can also overcome all the baselines run separately with each type of feedback.

The Voting Strategy provided results close to the Consensus Clustering and was proposed to have a low computational cost and be easy to implement. In this way, this approach may be an alternative to scenarios with a larger number of data. The improvement in the results of the clustering-based approaches is due to the fact that we can keep the semantics of the data and the actual configuration of each type of feedback at the time of clustering. Thus, unlike the Group-based approach, in which we combine distance matrices, in the proposed approaches we combine the groups of users generated by clustering, through data clustering techniques that allow us to combine groups of users from different information.

Finally, most of the results showed improvement in the accuracy of the recommendation, especially when using more than one type of feedback in the recommender process. This emphasizes that we can better represent a user's behavior when we have a large number of metadata about him. As shown in the experiments, the approach is flexible and extensible to different combinations of feedback types, clustering and recommender algorithms and datasets, although some of these configurations result in marginal improvements over the baselines (in particular collaborative filtering algorithms).

7. FINAL REMARKS

It is very important for a recommender algorithm to have the capability of making recommendations with high quality by analyzing and retrieving user's preferences. Although collaborative filtering is largely used in recommender systems, some efforts to overcome its drawbacks have to be made to improve the prediction accuracy. Selecting the similar users plays an important role on improving the prediction quality. In this article, we extend our previous work, called Group-based Collaborative Filtering [da Costa et al. 2016], to improve the quality of groups generated through two clustering ensemble approaches. The first ensemble clustering technique is based on heuristics, which combines individual results of each feedback generated by clustering algorithm using vote strategy, where the most common label in the result is attributed to the final result. For a more reliable and robust tool, the second technique was used based on CSPA, HGPA and MCLA, called Consensus Clustering Based

Strategy, in which user's label is defined by well-known clustering ensembles strategy.

We conducted experiments in two datasets and the results show that our ensemble strategies improve the overall system's performance and make progress in the data clustering in recommender systems. The main advantage of our approach is the possibility of providing more accurate recommendations, consequently reducing the effects of sparsity, since the approach enriches the recommendation process with different types of feedback based on ensemble clustering approaches.

As future work, we plan to evaluate our approach with additional datasets from other domains in order to check the accuracy with different information types. Furthermore, we plan to consider community detection in graphs to select better users' groups to make the recommendation.

REFERENCES

Aggarwal, C. C. *Recommender Systems: The Textbook*. Springer Publishing Company, Incorporated, 2016.

Bobadilla, J., Ortega, F., Hernando, A., and GutiéRrez, A. Recommender systems survey. *Knowledge-Based Systems. http://dx.doi.org/10.1016/j.knosys.2013.03.012* vol. 46, pp. 109–132, July, 2013.

Breese, J. S., Heckerman, D., and Kadie, C. Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence. http://dl.acm.org/citation.cfm?id=2074094.2074100*. UAI'98. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, pp. 43–52, 1998.

Cantador, I., Brusilovsky, P., and Kuflik, T. 2nd workshop on information heterogeneity and fusion in recommender systems (hetrec 2011). In *Proceedings of the 5th ACM conference on Recommender systems*. RecSys 2011. ACM, New York, NY, USA, 2011.

da Costa, A. F., Manzato, M. G., and Campello, R. J. Group-based collaborative filtering supported by multiple users' feedback to improve personalized ranking. In *Proceedings of the 22Nd Brazilian Symposium on Multimedia and the Web. http://doi.acm.org/10.1145/2976796.2976852*. Webmedia '16. ACM, New York, NY, USA, pp. 279–286, 2016.

da Costa Fortes, A. and Manzato, M. G. Ensemble learning in recommender systems: Combining multiple user interactions for ranking personalization. In *Proceedings of the 20th Brazilian Symposium on Multimedia and the Web*. WebMedia '14. ACM, New York, NY, USA, pp. 47–54, 2014.

da Costa Fortes, A. and Manzato, M. G. Case recommender: A recommender framework. In *Proceedings of the 22Nd Brazilian Symposium on Multimedia and the Web*. Webmedia '16. SBC, pp. 99–102, 2016.

Estivill-Castro, V. Why so many clustering algorithms: A position paper. *SIGKDD Explor. Newsl.* 4 (1): 65–75, June, 2002.

Gantner, Z., Rendle, S., Freudenthaler, C., and Schmidt-Thieme, L. MyMediaLite: A free recommender system library. In *Proceedings of the Fifth ACM Conference on Recommender Systems*. RecSys '11. ACM, New York, NY, USA, pp. 305–308, 2011.

Gantz, J. and Reinsel, D. The digital universe in 2020: Big data, bigger digital shadows, and biggest growth in the far east, 2012.

Gupta, U. and Patil, N. Recommender system based on hierarchical clustering algorithm chameleon. In *2015 IEEE International Advance Computing Conference (IACC)*. pp. 1006–1010, 2015.

Han, J., Kamber, M., and Tung, A. K. H. Spatial clustering methods in data mining: A survey. In *Geographic Data Mining and Knowledge Discovery, Research Monographs in GIS*, H. J. Miller and J. Han (Eds.). Taylor and Francis, 2001.

Katarya, R. and Verma, O. P. An effective collaborative movie recommender system with cuckoo search. *Egyptian Informatics Journal*, 2016.

Kim, T.-H., Ryu, Y.-S., Park, S.-I., and Yang, S.-B. An improved recommendation algorithm in collaborative filtering. In *Proceedings of the Third International Conference on E-Commerce and Web Technologies*. EC-WEB '02. Springer-Verlag, London, UK, UK, pp. 254–261, 2002.

Koren, Y. Factor in the neighbors: Scalable and accurate collaborative filtering. *ACM Transactions on Knowledge Discovery from Data (TKDD)* 4 (1): 1–24, Jan., 2010.

Li, W. and He, W. An improved collaborative filtering approach based on user ranking and item clustering. In *Internet and Distributed Computing Systems*, M. Pathan, G. Wei, and G. Fortino (Eds.). Lecture Notes in Computer Science, vol. 8223. Springer, pp. 134–144, 2013.

Mitchell, T. M. *Machine Learning*. McGraw-Hill, Inc., New York, NY, USA, 1997.

Park, Hae-Sang, J.-S. L. and Jun, C.-H. A k-means-like algorithm for k-medoids clustering and its performance. *Proceedings of ICCIE (2006)*, 2006.

Peska, L. Using the context of user feedback in recommender systems. In *Proceedings 11th Doctoral Workshop on Mathematical and Engineering Methods in Computer Science, MEMICS 2016, Telč, Czech Republic, 21st-23rd October 2016*. pp. 1–12, 2016.

Punera, K. and Ghosh, J. Consensus-based ensembles of soft clusterings. *Applied Artificial Intelligence* 22 (7-8): 780–810, 2008.

Rendle, S. Factorization machines with libFM. *ACM Trans. Intell. Syst. Technol.* 3 (3): 57:1–57:22, May, 2012.

Rendle, S., Freudenthaler, C., Gantner, Z., and Schmidt-Thieme, L. Bpr: Bayesian personalized ranking from implicit feedback. *CoRR* vol. abs/1205.2618, 2012.

Strehl, A. and Ghosh, J. Cluster ensembles — a knowledge reuse framework for combining multiple partitions. *J. Mach. Learn. Res.* vol. 3, pp. 583–617, Mar., 2003.

Voorhees, E. M. and Harman, D. K. *TREC: Experiment and Evaluation in Information Retrieval (Digital Libraries and Electronic Publishing)*. The MIT Press, 2005.

Wen, J. and Zhou, W. An improved item-based collaborative filtering algorithm based on clustering method. In *Journal of Computational Information Systems*, M. Pathan, G. Wei, and G. Fortino (Eds.). Lecture Notes in Computer Science, vol. 8. Springer Berlin Heidelberg, pp. 571–578, 2012.

4.2 Final Remarks

In this chapter, we presented an extension of Group-based approach, presented in the Chapter 3, for keeping the meaning of different types of feedback used in the clustering task. The proposed ensembles add the option of processing independently each users' interactions and combining them in a more robust and reliable process than heuristic mean, used in the approach described in Chapter 3. Two ensemble clustering strategies were implemented: Voting and Consensus Clustering. These two approaches are well-known in the literature and they were developed according to the need to maintain the semantics of the data already processed by the traditional clustering approach.

The main goal of this work is to reduce the search space for any recommender using relevant information extracted from different types of users' feedback, and simultaneously aid its predictive power by offering a more guided recommendation. By doing this, the reduced, personalized set of known pairs of each group can be plugged into any recommender. We conducted experiments in two datasets from different domains (movies and music) and the results show that our strategy improves the overall system's performance in a variety of experimental setups. The main advantage of our approaches is to provide a more guided recommendation for each user, which helps to mitigate the high dimensionality problem in RS.

A potential shortcoming of this and the last chapters is that data clustering approaches take into account only information from users to relate possible recommendations. In these approaches, each user only receives recommendations whose items have been visited by users with similar preferences. Consequently, these users will never receive recommendations of different items (even the same category) from those observed in his group. This problem is commonly known in the recommender systems area as *diversity* (RICCI; ROKACH; SHAPIRA, 2015; AGGARWAL, 2016).

In the next chapter, we will present a pre-processing approach to consider not only the users but also the items in the clustering approach.

CHAPTER

5

AN EXPLORATION OF IMPROVING COLLABORATIVE RECOMMENDER SYSTEMS VIA TWO-WAY CLUSTERING

5.1 Contextualization

Collaborative Filtering (CF) is the most popular method for recommender systems (BOBADILLA *et al.*, 2013; AGGARWAL, 2016). Its main motivation is that users might be interested in items that are viewed by similar users, and most of the existing CF methods measure users' preferences by their behaviors over all the items. However, users might have different interests over different topics (diversity), thus might share similar preferences with different groups of users over different sets of items. The work presented so far, as well as the most of CF approaches, consider only the users representations to infer information about the data, discarding the information and relationships about the items.

In this chapter, we continue investigating the research questions of this thesis with a two-way clustering approach applied on recommender systems, which generate user-item submatrices (bi-groups) based on the combination of individual clustering of users and items. The purpose of this approach is to incorporate users and items relationships into a same subgroup that can be used by recommender algorithms and so alleviate the problem mentioned above.

5.2 Introduction

Recommender systems emerged from the need for personalized content filtering on the large amount of information available on the Web. Such systems perform this filtering by (RICCI;

ROKACH; SHAPIRA, 2015; AGGARWAL, 2016; SAHU; DWIVEDI, 2019): i) comparing a user interest profile with the description of an item (content-based filtering); ii) analyzing user interactions and comparing them to users with similar interactions (collaborative filtering); and iii) combining aspects of both approaches (hybrid filtering). Among these types of filtering, collaborative filtering is one of the most widely adopted and successful recommendation approaches (XU *et al.*, 2012; WU *et al.*, 2016; AGGARWAL, 2016). Unlike many algorithms based on content-based filtering which utilize the representations of users and items, CF approaches make recommendation based only on the user-item feedback matrix. The advantage of this approach is that it can capture the hidden connections between users and items and have the ability to provide novelty items, which are helpful to improve the diversity of recommendation.

Traditionally CF-based algorithms and some clustering based techniques applied on RS associate a user with a group of like-minded users based on their preferences over all the items, and then recommend to the user those items enjoyed by others in the group (BOBADILLA *et al.*, 2013; COSTA; MANZATO; CAMPELLO, 2017; RICCI; ROKACH; SHAPIRA, 2015). The basic assumption is that users with similar behaviors (e.g., web history, ratings, tags, etc) will have similar tastes on all the items. However, this assumption is not always plausible, since two users having similar tastes on one item subset may have totally different tastes on another set (e.g two users like action movies, but only one of them likes romance). In addition, some user's interests are usually focused on some topics, but not dispersive over all items which he/she interacted with. So it is natural to say a group of users are like-minded on a subset of items.

We can model users and items using subgroups, where each subgroup includes a set of like-minded users and the subset of items that they are interested in, and each user/item can be assigned to multiple subgroups so that users can share their interests with different subsets of users on different subsets of items (XU *et al.*, 2012; BATULE; ITKAR, 2016; HOVALE; GHULI, 2016). Partitioning users and items into subgroups for CF has been studied in several previous works, where user clustering (RAMEZANI; MORADI; TAB, 2013; COSTA; MANZATO; CAMPELLO, 2016; NETO; MANZATO; CAMPELLO, 2018), item clustering (BATULE; ITKAR, 2016) and co-clustering (XU *et al.*, 2012; VLACHOS *et al.*, 2014; WU *et al.*, 2016) methods have been proposed to boost the performance of collaborative filtering. However, in these methods each user/item is only allowed to be assigned to a single subgroup, so that they are not able to model the case where users have multiple interests. Moreover, some algorithms are highly susceptible to parametrization, and/or require enriched information that may not be available for every application domain. Finally, many of these methods do not take into consideration that an RS usually demands a minimum amount of information in order to work properly, especially CF approaches.

In order to fill these gaps, we propose a pre-processing step based on two-way clustering for recommender systems, called TWCRS, which can be applied prior to any recommender algorithm, categorizing the users and items into bi-groups (allowing overlapping). Two-way

clustering techniques are particularly valuable when important information is entangled in a large body of experimental data (CHANDRA; SHANKER; MISHRA, 2006). These techniques are helpful in clustering the users that have similar behavior and also in clustering the items based on their users' interactions. Thus, a dual purpose can be achieved simultaneously, i.e., finding the important co-regulated users that make the main contribution to efficient clustering of items (CHANDRA; SHANKER; MISHRA, 2006; WU *et al.*, 2012). Therefore, the result of this approach are bi-groups that contain users and items correlated by both dimensions.

In this way, we developed a two-way clustering approach to RS, which partitions users and items from the original feedback matrix, independently, and combine them using an enhancement process based on the sparsity of the generated submatrix of each dimension. Specifically, a hierarchy clustering algorithm with a cut optimization criterion is applied twice (one for clustering users and the other for partitioning items) and an enrichment step is made to combine the groups generated by the two dimensions using sparsity as criterion for the optimization. Finally, we used the generated bi-groups as input for a recommender algorithm to generate rankings for each user. We evaluated our proposal by employing it with a well-known recommender algorithm for top-*N* recommendation, called BPR-MF. We also compared our results against other clustering-based techniques as competitors. The main goal here is to demonstrate that TWCRS helps not only to alleviate traditional RS problems, such as high dimensionality, diversity and sparsity, but also the quality of the generated bi-groups and effectively increases the recommendation accuracy. The experiments were performed with ten real datasets from different domains: music, movies, e-commerce, etc., and the results were analyzed and discussed.

This chapter is structured as follows: in Section 5.3 we overview research results related to clustering approaches applied on recommender systems. We then present our approach in Section 5.5 and report the experimental results in Sections 5.6 and 5.7. In Section 5.8 we present our analysis and conclusions.

5.3 Related Work

Recommender systems often have to deal with high dimensionality, noisy and data sparsity problems. Several approaches have been proposed in the literature to tackle these issues. Ramezani, Moradi and Tab (2013) proposed to condense the user-item matrix by removing unrepresentative users or items. This technique has the problem of not generating recommendations for some users and items, since they have been removed in the pre-processing step. In addition, such method does not significantly improve the performance of RS. An alternative is to divide the user-item matrix in subsets using clustering techniques, which have also been employed directly as solutions for the presented problems.

Costa, Manzato and Campello (2017) proposed Group-Based (GB), which is a recommender approach based on users' clustering, where the distance among users is computed using

multiple types of feedback. The authors applied *k*-medoids with a weighted distance matrix built using users' feedback, in order to remove noise and reduce the search space. Zhang *et al.* (2016b) developed a novel effective collaborative filtering algorithm based on user preference clustering, called UPUC-CF, to reduce the impact of data sparsity and high search space. In their approach, user groups are introduced to distinguish users with different preferences. Then, considering the preference of the active user, they obtain the nearest neighbor set from the corresponding user group/groups. Besides, a new similarity measure method is proposed to calculate the similarity between users, which considers user preferences in both local and global perspectives. Xiaojun (2017) proposed an improved clustering-based collaborative filtering recommendation algorithm (ICCFRA), that contains a time decay function for pre-processing the user's rating, and uses item attribute vectors and user interest vectors applied on clustering algorithms to group them individually. To this end, improved similarity measures are used to both users and items clustering, and also to find the best match between those groups. The main problem of these works is that it requires a large number of explicit feedback, which may not be available for every application domain or are applied only on the rating prediction scenario. Furthermore, this method does not take into consideration that a recommender usually demands a minimum amount of information in order to work properly.

In order to fill these gaps, Neto, Manzato and Campello (2018) proposed a pre-processing approach for recommender systems that can be applied prior to any recommender algorithm, categorizing the users into disjoint groups. This method automatically provides an optimal partition from a hierarchy of clusters based on optimization criteria (e.g. life time and sparsity), leading to a partition with more stable clusters. For each generated cluster, the authors used three recommender algorithms to generate the final ranking. However, these clustering approaches take into account only one dimension of the data to generate the groups: users or items. Thus, independent relationships of the unused dimension are ignored, for example, when grouping users, the similarity between items of the same category is not considered.

The usage of clustering approaches, such as co-clustering and bi-clustering, has been studied in the context of RS by some previous works in order to alleviate this problem. Xu *et al.* (2012) proposed an overlapping user-item co-clustering approach based on a weighted graph-cut optimization problem for Top-*N* recommendation scenario. The objective function of their problem is NP-Hard, and they propose to optimize a relaxed problem. However, the learning involves solving the top eigenvectors of a large squared matrix so it cannot scale to large datasets. Vlachos *et al.* (2014) proposed PaCo, a co-clustering strategy to generate subsets of users and items to alleviate this problem and to reduce the high dimensionality during the recommendation process. This algorithm was inspired by *k*-Means and agglomerative hierarchical clustering approaches and, as such, it can discover co-clusters of better quality and relevance than the current co-clustering techniques to group correlated users and items to generate recommendation. Wu *et al.* (2016) proposed the scalable co-clustering technique to improve the performance of recommender algorithms to generate rankings. Their approach first clusters users and items into

several subgroups, where each subgroup includes a set of like-minded users and a set of items in which these users share their interests. Then, traditional recommenders can be applied to each subgroup, and the recommendation results from all the subgroups can be easily aggregated.

In this work, we report a two-way clustering pre-processing solution using the work of Neto, Manzato and Campello (2018), which considers relationships of users and items. Our approach, like the approaches by Vlachos *et al.* (2014) and Wu *et al.* (2016), has several benefits. The first is that it divides the matrix into several overlapping sub-groups, which are denser than the original matrix, thus making CF methods more feasible. Second, any recommender algorithm can be used in each generated subgroup. Since the performance of different methods varies under different scenarios, this allows users to select their favorite CF base methods for the subgroups. Last, but not least, the training of the recommender algorithms in subgroups can be trivially parallelized. Unlike the related work that also follows this type of approach, our solution is based on a two-way clustering approach with a optimization process to join group of users and items in bi-groups, requiring very little information to operate.

5.4 Cluster Extraction applied on RS

In this section, we present the work developed by Neto, Manzato and Campello (2018), which we used as clustering approach to extract from original user-item matrix the groups of users and items. In their work, the authors introduce three novel variants of the standard FOSC formulation (CAMPELLO *et al.*, 2013), which are specifically designed to directly tackle recommender systems problems (e.g. data sparsity and high dimensionality) and the need to ensure that there is a minimum amount of user information in each group in order to successfully compute recommendations. Campello *et al.* (2013) proposed a "Framework for Optimal Selection of Clusters" (FOSC) to make local (non-horizontal) cuts to any cluster tree hierarchy. Given a local unsupervised clustering quality measure (one that can be computed for each cluster individually), and a set of constraints, FOSC determines a local cut of a given hierarchy that is optimal with respect to the quality measure and the constraint set. In the same work, these authors reported a cluster quality measure, named *Stability*, based on the notion of lifetime of an object belonging to a cluster in the clustering hierarchy, which is used in their formulation.

Since the standard formulation has not been designed to directly tackle RS problems (e.g. ensure that there is a minimum number of users in a group), Neto, Manzato and Campello (2018) propose some variants of this technique varying parameters and optimization criteria to adopt this framework for the recommendation scenario. In our work, we used only one of these variants, called LifetimeMCS, which uses the same optimization criterion as in the original FOSC framework (CAMPELLO *et al.*, 2013). The difference is that the authors always check the minimum number of objects within a group during (rather than prior to) the optimization process. This process is exemplified in Figure 5.1, in which the lifetime of a cluster is the size of

the edge in the dendrogram and the LifetimeMCS maximizes the weighted sum of the edges in the final partition (respecting the minimum of objects per cluster).

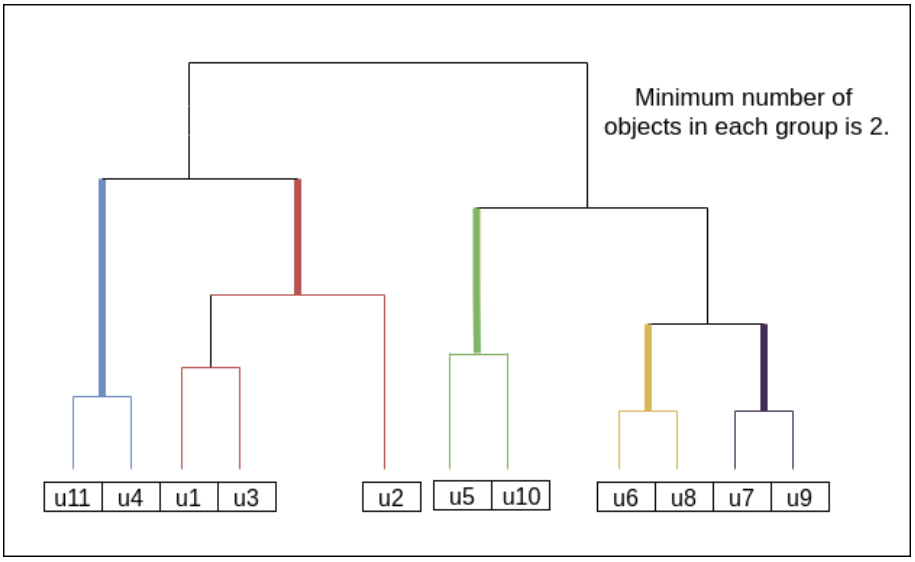


Figure 5.1 – Representation of a clustering hierarchy filtered with minimum number of objects equals 2. The optimal FOSC solution according to the Stability criterion is highlighted in colors.

In summary, to run all proposed complete pipeline, Neto, Manzato and Campello (2018) first applied a hierarchical clustering algorithm to the rows/columns of the user-item interaction matrix to produce a clustering hierarchy of users according to their interactions with the items/users, then they extracted a partition of disjoint clusters of users from the hierarchy, using one of their new variants of FOSC suitable for the RS domain. Finally, for each extracted cluster, the authors applied a recommender algorithm, considering only information regarding the users/items in that cluster.

In this work, we will follow the first two steps proposed by the authors and then combine the groups of users and items into bi-groups using a refinement process that will be discussed in the next section.

5.5 Two-Way Clustering Approach

In this section, we present a two-way clustering approach for recommender systems (TWCRS), in which our goal is to find relevant subgroups (bi-groups), where each of them includes a set of like-minded users and the subset of items they are interested in. In this way, our first task is to find potential user and item interest groups, independently, flooded in the large user-item feedback matrix, and then use them to generate subgroups of interests. For this purpose, we find the groups using the hierarchical clustering solution optimized by FOSC (NETO; MANZATO; CAMPELLO, 2018), and then apply an unified strategy to combine the generated groups in bi-groups with a two-way clustering approach. Each generated bi-group is

composed by users/items membership vectors. These memberships vectors reveal users' interests on different types of items and items that would be consumed by users, respectively.

In order to improve the quality of the generated bi-groups, since there is no optimization during the combination of clusters of users and items in bi-groups, we propose an enhancement process to combine potential bi-groups and discard those with unrelated information. In this process, we use the sparsity metrics of the bi-groups to measure its potential information. Finally, for each generated bi-group, we apply a given recommender systems algorithm to generate the list of predictions. The following subsections explain these tasks in more details.

5.5.1 Clustering Extraction and Generating Bi-Groups

In this step, we aim to group users and items of the database through a data hierarchical clustering technique, in order to obtain different groups containing users and items with similar representations. In our work, we generate the groups using the LifetimeMCS approach, presented in the Section 5.4, which uses the distance between the representation of users/items to cluster the objects.

The distance between a user/item and other users/items is computed/measured based on their representation. In recommender systems, usually the cosine angle, correlation and co-occurrence metrics are used to calculate how two items are alike, considering their representation in the dataset. These metrics are commonly used because: i) they disregard or soften the matrix cells that have no interaction, and ii) they are the most commonly used metrics in the area of information retrieval (RICCI; ROKACH; SHAPIRA, 2015; AGGARWAL, 2016). This is particularly useful because we can not assume that items are dissimilar based on the fact they do not have certain feature, since this attribute may not have been filled due to lack of information. We employ these metrics in the user and item representations, generating a new matrix of $M \times N$ users/items, which represents the entry of the clustering approach proposed by (NETO; MANZATO; CAMPELLO, 2018). After generating the clusters of users and items, we then combine these partitions into bi-groups according to their dispositions, setting up a two-way clustering approach, where:

- A user cluster (C_{u1}, C_{u2}, C_{uk}) of rows (U)
- An item cluster (C_{i1}, C_{i2}, C_{il}) of columns (I)
- Bi-groups of $U \times I$ are obtained by fully crossing the row and column partitioning.

Figure 5.2 illustrates this process, identifying with different colors the groups and bi-groups generated.

This two-way clustering approach is responsible for aggregating the existing clustering in rows and columns arranged in the user-item interaction matrix. Although there is an optimization

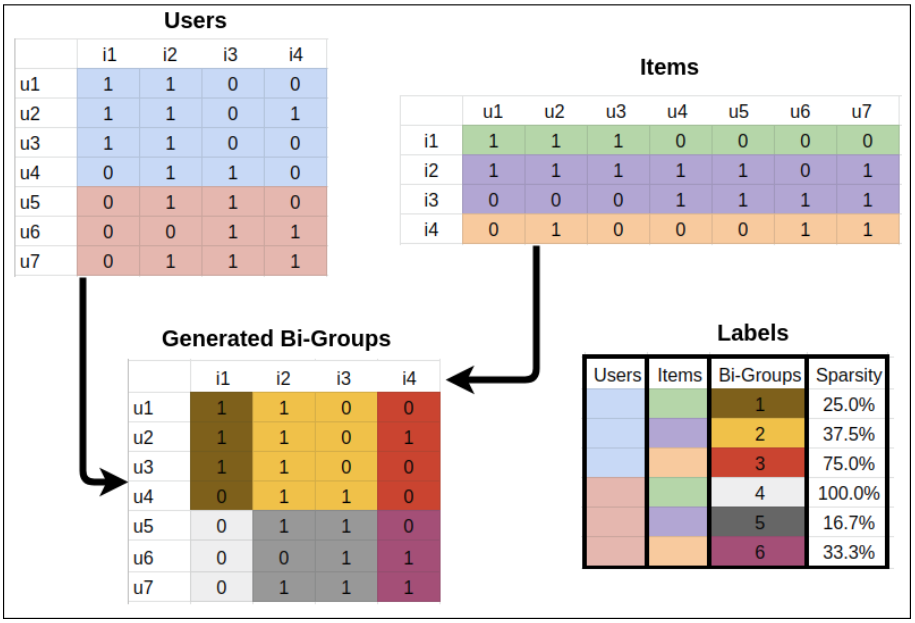


Figure 5.2 – Bi-groups generated. On the top we have the generated groups of users and items. On the bottom left we have the bi-groups generated and on the bottom right the combinations that generated the groups along with their sparsity of interactions.

criterion to find the groups of users and items, we need to ensure that the generated bi-groups also offered a good quality, since in this approach there is no optimization criterion for the combination of user groups and items. For example, in Figure 5.2 bi-group 4 was generated without any relevant information (there is no interaction within the bi-group), which will not generate any recommendations. In order to investigate and extend this combination process, we also develop a enhancement process, which provides a more robust solution for the problem and additionally removes irrelevant information for the recommendation task.

5.5.2 Enhancement Process

In this task, we endeavor to optimize the quality of the bi-groups using a criterion based on sparsity, since this metric measures how much information each group can offer, in order to improve the input data of recommender algorithms. The sparsity of a interaction matrix is defined as (SOMBOONVIWAT; AOYAMA, 2016):

$$Sparsity = \left(1 - \frac{Number\ of\ interactions}{Number\ of\ users \times Number\ of\ items}\right) 100\% \tag{5.1}$$

Based on this metric our process verifies the possibility of union between one bi-group with another considering each of the bi-groups generated. We first order and filter bi-groups based on their sparsity, where only bi-groups with sufficiently low sparsity and with some missing information are candidates, excluding a large amount of bi-groups with sparsity near 100% and bi-groups with sparsity equals 0%. In the second case, since the bi-group is fully populated, that is, all users see all items, no recommendation will be made. Then, for each bi-group, it is verified

its current sparsity and its sparsity when combined with the other bi-groups. If in any of these combinations the sparsity is smaller than that of the bi-group alone, the combination is made (in the case of a tie the combination is made with the bi-group with the least number of interactions).

For example, bi-group A has a sparsity of 70%, its combination with bi-group B generates a sparsity of 65% and its combination with bi-group C generates a sparsity of 83%. In this case, there will be a combination of bi-groups A and B. It is important to remember that this relationship is not symmetrical, so the combination of bi-group B with A may not occur. The only case it will be symmetrical is if the sparsity of the two groups is exactly the same. The amount of interactions in each bi-group is used in case of a tie during this process.

However, the number of bi-groups generated can be very large, which increases the computational complexity of the algorithm. In addition, the combination of bi-groups randomly causes the semantics used in user and item hierarchies to be lost. In order to avoid excessive comparisons, only adjacent bi-groups are compared. In our problem, bi-group A is adjacent to B if they share the same set of users or items. Consider a valid and active bi-group A, with sparsity sA, the adjacent bi-group B having the least sparsity will join the bi-group A provided that the sparsity of B (sB) is less than sA. A set of interactions is generated, and used later in the recommendation step.

Another constraint we implemented in this process is to consider only joins between two bi-groups whose respective groups in the user hierarchy or in the item hierarchy are children of the same parent cluster. This process is repeated considering each of the valid bi-groups as active. It is important to notice that in this process there only will be union of the bi-groups if the resulting bi-group has a sparsity smaller than the active bi-group and that the relationships are not symmetric.

5.5.3 Recommendation Task

After getting the pre-processed bi-groups generated by the last step in TWCRS, any recommender algorithm can be applied to process the interactions of each generated bi-group and predict a Top-N recommendation list of items for each user in the dataset. The algorithm is responsible for assembling a matrix that contains all users and items of a given bi-group k and individual interactions of each user to predict items that she/he can enjoy, both highlighting the items she/he visited as the ones she/he has not. At this stage, each user receives personalized recommendations based on his/her bi-group information, which considers relationships of similar users as well as relevant items.

In order to obtain more consistent recommendations, we order the bi-groups in ascending order according to their sparsity. Therefore, the recommendations for each user should be generated first in bi-groups with more information and if they can not get the number K of recommendations, they resort to the next bi-group. Finally, the individual recommendations are

concatenated in a single ranking with all users, which contains pairs (u, i) sorted by scores which were generated by the recommender.

It is important to notice that in our approach not all groups generated in the grouping and enhancement process will be applied to recommenders, so there is no need to compute a score for all items not seen by a user. This is a reduction in computational cost as compared to traditional approaches.

5.6 Experimental Settings

We conduct many experiments to evaluate the effectiveness of our proposed approach. In this section, we describe the experimental settings in detail.

5.6.1 Databases

The experiments were performed using nine datasets from different domains: music, movies, games, e-commerce and books, varying in size as well as in their configurations of users, items and interactions.

- **Yahoo Movies:** this database contains a sample of the Yahoo! Movies community's preferences for various movies, rated on a scale from A+ to F. Developed by the Yahoo! Research Alliance Webscope program[1], this database consists of 169,767 interactions given by 4,385 users to 4,339 movies.

- **FilmTrust:** consists of 35,497 interactions given by 1,508 users to 2,071 movies developed by Guo, Zhang and Yorke-Smith (2013).

- **CiaoDVD:** is a DVD movie database proposed by Guo *et al.* (2014), and consists of 280,391 ratings given by 7,375 users to 99,746 items.

- **MovieLens** 2k**:** consists of 800,000 ratings and 10,000 interaction tags applied to 2,113 users and 10,197 movies. This database was developed by Cantador, Brusilovsky and Kuflik (2011) and we only used the ratings information for the experiments.

- **Steam:** consists of 129,511 purchases made by 12,393 users to 5,155 games in the Steam platform. this database was made available through a Kaggle challenge[2].

- **Booking Crossing:** this database was developed in the work of Ziegler *et al.* (2005) and contains 278,858 users (anonymized, but with demographic information) providing 1,149,780 ratings (explicit / implicit) about 271,379 books.

[1] https://webscope.sandbox.yahoo.com/
[2] https://www.kaggle.com/tamber/steam-video-games/data

- **Amazon Digital Music:** consists of 206,282 ratings applied by 16,396 users to 101,708 music albums. For our experiments we used only a sample of the original data set, proposed by McAuley, Pandey and Leskovec (2015), containing 59,793 interactions made by 6,337 users to 4,744 items.

- **Anime:** consists of 520,610 interactions given by 5,000 users to 7,390 anime provided by Kaggle[3].

- **Retail Rocket:** also provided by Kaggle[4], this is a e-commerce database, which contains 92,490 interactions from 3,431 users on 8,885 items.

- **LastFM:** this database was proposed by Cantador, Brusilovsky and Kuflik (2011) and consists of 92,834 user listened-to-artists relations involving a set of 1892 users and 17,632 artists.

We follow the pre-processing steps that have been used in recent works (ZHANG; WANG, 2015; WU *et al.*, 2016; YU; QIN, 2019). We first remove users and items with less than 5 interactions from each database, since we will not work with the cold-start problem. Then, for the databases with explicit ratings, we convert the remaining ratings to 1. The statistics of the resulting databases are shown in Table 5.1.

Table 5.1 – Databases statistics.

Database	Users	Items	Interactions	Sparsity (%)
Yahoo!Movies	4,385	4,339	169,767	99.11
FilmTrust	1,508	2,071	35,497	98.86
CiaoDVD	2,046	2,533	29,921	99.48
MovieLens 2*k*	2,113	10,197	800,000	96.37
Steam	12,393	5,155	129,511	99.80
BookCrossing	4,035	4,781	71,791	99.67
Amazon Digital Music	10,081	17,261	108,237	99.94
Anime	4,983	7,291	468,548	98.71
Retail Rocket	1,504	4,182	9,028	99.87
LastFM	1,892	17,632	92,834	99.72

5.6.2 Methodology

To evaluate the accuracy of the proposed approach against its baselines, we adopted a 10-fold cross-validation methodology (LANGFORD, 2005), evaluating the results for each fold and take the average value. We measure the quality of recommendations by the Normalized Discounted Cumulative Gain at N ($NDCG@N$) (MANNING; RAGHAVAN; SCHüTZE, 2008), using $N = \{1, 5, 10\}$. In order to check statistical relevance between the results we applied Wilcoxon test with 95% of confidence (WILCOXON, 1945).

[3] https://www.kaggle.com/CooperUnion/anime-recommendations-database
[4] https://www.kaggle.com/retailrocket/ecommerce-dataset

To evaluate the impact of our proposed approach in the recommender systems scenario, we performed our experiments using a well-known recommender algorithms called Bayesian personalized ranking with matrix factorization (BPR-MF) (RENDLE *et al.*, 2009), with its default parameters suggested by the original authors.

We developed our pre-processing approach in Python version 3.6[5]. The baseline competitors (recommender algorithms only) belong to the Case Recommender Framework version 1.0.9[6], an open source tool developed in *Python*, with numerous features and algorithms for RS. We used the default framework settings in our evaluations. The experiments were performed using a personal computer with an Intel Core i7 4930K processor running at 3.40GHz and with 32 GB of DDR3 RAM, with Ubuntu 17.04 OS.

5.7 Experimental Results and Analysis

In this section, we run two well-known recommender algorithms and check whether their top-N recommendations are improved after using our two-way clustering approach. After that, we compare the rankings generated by our approach against Neto, Manzato and Campello (2018) applied on users and items dimensions. Then, we run our approach against two co-clustering approaches presented in the related work. Finally, we make an empirical analysis to the results, as well as the complexity and the limitations of out approach.

5.7.1 TWCRS vs. Traditional Recommender

We first analyze the accuracy of our pre-processing approach comparing with BPR-MF, which is directly applied on the training data without two-way clustering approach. The results are shown in Table 5.2. The overall observation is that TWCRS improves the pure recommender algorithm on most of the cases – the best scores on all the data sets are achieved by TWCRS, except on the MovieLens 2k database. For most of the cases, our pre-processing approach improves the corresponding base method by at least 11%, especially, in databases where the sparsity is greater.

We notice that the performance of BPR-MF varies a lot on different data sets. This means that no single CF method can perform well at all scenarios. The advantage of the proposed approach is that we do not rely on the underlying base recommender algorithm, and TWCRS users are free to configure their favorite recommenders in order to adapt to different scenarios.

[5] https://www.python.org/
[6] https://pypi.python.org/pypi/CaseRecommender

Table 5.2 – TWCRS vs. traditional recommender compairson in terms of NDCG.

Database	Method	NDCG@1	NDCG@5	NDCG@10
Yahoo Movies	TWCRS	**0.2384***	**0.4055***	**0.4204***
	BPR-MF	0.1436	0.2699	0.2921
FilmTrust	TWCRS	**0.4804***	**0.63402***	**0.6427***
	BPR-MF	0.4354	0.6008	0.6149
CiaoDVD	TWCRS	**0.0371***	**0.0883***	**0.1040***
	BPR-MF	0.0152	0.0421	0.0523
MovieLens 2*k*	TWCRS	0.3088	0.5191	0.5158
	BPR-MF	**0.4614***	**0.6129***	**0.6029***
Steam	TWCRS	**0.2364***	**0.3747***	**0.3926***
	BPR-MF	0.0757	0.1644	0.1880
Booking Crossing	TWCRS	**0.0453***	**0.0895***	**0.1010***
	BPR-MF	0.0070	0.0204	0.0278
Amazon Music	TWCRS	**0.0567***	**0.1148***	**0.1285***
	BPR-MF	0.0065	0.0163	0.0205
Anime	TWCRS	**0.3123***	**0.5128***	**0.5122***
	BPR-MF	0.2146	0.3917	0.4064
Retail Rocket	TWCRS	**0.0275***	**0.0498***	**0.0538***
	BPR-MF	0.0063	0.0178	0.0219
LastFM	TWCRS	**0.2091***	**0.3517***	**0.3675***
	BPR-MF	0.1028	0.1944	0.2130

Bold typeset indicates the best performance. * indicates statistical significance at p-value <0.05 pairwise compared to baseline.

5.7.2 TWCRS vs. LifetimeMCS

As discussed in Section 5.4, our approach is based on Neto, Manzato and Campello (2018) work, a clustering extraction approach applied on recommender systems. In this subsection, we compare TWCRS with this approach applying the clustering optimization in terms of users and items independently. Table 5.3 shows the results obtained by all regarded configurations of our approach, against the results obtained by the LifetimeMCS approach applied on users and items.

The overall observation is that LifetimeMCS applied on items dimension has worse performance than the users dimension in all databases. One hypothesis is that because the items are represented by the interactions of users with them, their representations are highly sparse and little descriptive, which negatively influences the clustering of these objects. Based on these results, it can also be noticed that our approach can not statistically improve the LifetimeMCS approach applied to the users' dimension in most cases. Since item partitioning has not been effective, we believe that this may negatively influence our two-way clustering approach, incorporating items with features that are not relevant to bi-group users.

5.7.3 TWCRS vs. Co-Clustering Approaches

Finally, we compare TWCRS against two co-clustering based methods PaCo (VLACHOS *et al.*, 2014) and CCCF (WU *et al.*, 2016), which are recent co-clustering methods for recommender systems. For both baselines, we carefully tuned the parameters using a cross

Table 5.3 – TWCRS vs. LifetimeMCS approach in terms of NDCG.

Database	Method	NDCG@1	NDCG@5	NDCG@10
Yahoo Movies	TWCRS	**0.2384***	**0.4055***	**0.4204***
	LifetimeMCS User	0.2173	0.3697	0.3875
	LifetimeMCS Item	0.0990	0.1810	0.1932
FilmTrust	TWCRS	**0.4804**	**0.6340**	**0.6427**
	LifetimeMCS User	0.4803	0.6303	0.6412
	LifetimeMCS Item	0.3744	0.5028	0.5149
CiaoDVD	TWCRS	0.0371	0.0883	0.1040
	LifetimeMCS User	**0.0439***	**0.0916***	**0.1063**
	LifetimeMCS Item	0.0091	0.0263	0.0335
MovieLens 2*k*	TWCRS	0.3088	0.5191	0.5158
	LifetimeMCS User	**0.4451***	**0.6166***	**0.6106***
	LifetimeMCS Item	0.1806	0.2992	0.3070
Steam	TWCRS	0.2364	**0.3747**	**0.3926***
	LifetimeMCS User	**0.2823***	0.3616	0.3700
	LifetimeMCS Item	0.0739	0.1486	0.1596
Booking Crossing	TWCRS	0.0453	0.0895	0.1010
	LifetimeMCS User	**0.0592***	**0.1066***	**0.1179***
	LifetimeMCS Item	0.0034	0.0110	0.0143
Amazon Music	TWCRS	0.0567	**0.1148**	**0.1285**
	LifetimeMCS User	**0.0573**	0.1112	0.1245
	LifetimeMCS Item	0.0064	0.0183	0.0233
Anime	TWCRS	**0.3123***	**0.5128***	**0.5122***
	LifetimeMCS User	0.2862	0.4566	0.4622
	LifetimeMCS Item	0.0662	0.1208	0.1307
Retail Rocket	TWCRS	0.0275	0.0498	0.0538
	LifetimeMCS User	**0.0372***	**0.0684***	**0.0783***
	LifetimeMCS Item	0.0048	0.0140	0.0174
LastFM	TWCRS	0.2091	**0.3517**	0.3675
	LifetimeMCS User	**0.2122**	0.3509	**0.3681**
	LifetimeMCS Item	0.1010	0.1950	0.2142

Bold typeset indicates the best performance. * indicates statistical significance at p-value <0.05 pairwise compared to baseline.

validation approach to ensure fair comparisons. Table 5.4 shows the results for NDCG@N on the ten databases. The general observation is that our two-way clustering approach outperforms the other two co-clustering methods in most cases. The only exceptions are on the MovieLens 2*k* and Amazon Music datasets, where both PaCo and CCCF outperformed TWCRS, and PaCo obtained the best results on both databases. This is likely due to the fact that the sparsity of these datasets are smaller than the others, which has a direct impact on the criterion we have defined to the enhancement process of bi-groups. The performance of CCCF is not as good as expected on all data sets. One reason may be that it assumes the user-item matrix follows some specific structure (users should have many interactions), and not all the data sets have this property, even using the filter in the interactions described in Section 5.6.1. It is worth mentioning that unlike our competitors, our approach automatically determines the number of clusters, which is a byproduct of the cluster extraction procedure proposed by Neto, Manzato and Campello (2018). In contrast, our competitors take the number of clusters as a parameter, and the best choice was not universal across different databases, or even across different algorithms in the same database.

Table 5.4 – TWCRS vs. co-Clustering Methods in terms of NDCG.

Database	Method	NDCG@1	NDCG@5	NDCG@10
Yahoo Movies	TWCRS	**0.2384***	**0.4055***	**0.4204***
	PaCo	0.1618	0.2713	0.2946
	CCCF	0.1414	0.2560	0.2760
FilmTrust	TWCRS	**0.4804***	**0.63402***	**0.6427***
	PaCo	0.4214	0.5484	0.5559
	CCCF	0.4520	0.6116	0.6245
CiaoDVD	TWCRS	**0.0371***	**0.0883***	**0.1040***
	PaCo	0.0033	0.0089	0.0091
	CCCF	0.0193	0.0440	0.0551
MovieLens 2*k*	TWCRS	0.3088	0.5191	0.5158
	PaCo	**0.4194***	**0.5627**	**0.5605***
	CCCF	0.3944	0.5540	0.5454
Steam	TWCRS	**0.2364***	**0.3747***	**0.3926***
	PaCo	0.1561	0.2099	0.2513
	CCCF	0.0774	0.1722	0.1943
Booking Crossing	TWCRS	**0.0453***	**0.0895***	**0.1010***
	PaCo	0.0149	0.0235	0.0246
	CCCF	0.0120	0.0290	0.0373
Amazon Music	TWCRS	0.0567	0.1148	0.1285
	PaCo	**0.1373***	**0.2224***	**0.2308***
	CCCF	0.0059	0.0140	0.0193
Anime	TWCRS	**0.3123***	**0.5128***	**0.5122***
	PaCo	0.1455	0.2620	0.2723
	CCCF	0.1388	0.2722	0.2970
Retail Rocket	TWCRS	**0.0275***	**0.0498***	**0.0538***
	PaCo	0.0058	0.0183	0.0209
	CCCF	0.0080	0.0190	0.0234
LastFM	TWCRS	**0.2091***	**0.3517***	**0.3675***
	PaCo	0.1453	0.2438	0.2479
	CCCF	0.0378	0.1236	0.1420

Bold typeset indicates the best performance. * indicates statistical significance at p-value <0.05 pairwise compared to baseline.

5.7.4 Discussion and Limitations

The goal of our proposed approach is to investigate and enhance bi-groups, where the items in the same bi-group share some common latent properties that attract a group of like-minded users. The results evidently show that our approach has positive effect when compared with pure recommender algorithm and recent co-clustering works applied on recommender systems. However, much of this improvement is due to the use of the LifetimeMCS approach, especially in the user dimension. As reported in Table 5.3, our approach in most cases fails to improve the results of the isolated LifetimeMCS approach. One of the reasons is that the optimization made by this algorithm in each of the isolated dimensions (users and items) should already contain the best possible partition of the data and when combining them, we can be injecting data not relevant to the recommendation.

Another reason may be the use of the sparsity criterion for the union of bi-groups without any weight, since in spite of the implemented restrictions (adjacency, bi-groups of the same

parent, etc.), this criterion may end up adding bi-groups with useless information of users or items, resulting in worse performance in the clustering approach, and directly impacting the quality of recommendation. Nevertheless, we realize that the results in some databases achieved some improvements using our approach, which demonstrates a possibility of better investigation in the enhancement process, through strategies of weighting during the combination, change of metric, etc., since when bi-groups become denser with more reliable information, the better will be the recommendations (XU *et al.*, 2012; WU *et al.*, 2016).

5.7.5 Computational Complexity

Let us assume the worst case, where no bi-groups were removed by insufficient sparsity (i.e. sparsity near 100%) or there is no recommendation available (sparsity equal to 0%). There is no need to visit all adjacent bi-groups except the one with smaller sparsity. As our approach visits each bi-group trying to find their best neighbor, we have the cost of visiting times the cost of merging, which is a list concatenation. Then, the total cost is $O(k * l)$ where k is the number of bi-groups and l is the largest amount of interactions in any bi-group. Therefore, the refinement cost is linear.

It is important to mention that for the total complexity we must also account for the cluster extraction step, which is part of pre-processing, and also the recommendation phase.

5.8 Final remarks

In this chapter, we proposed a two-way clustering approach for recommender systems (TWCRS) – utilizing a clustering extraction approach and a enhancement process in order to improve the quality of recommendations. The intuition of our model is that users may have different preferences over different subsets of items, and we explore the enhancement process of theses bi-groups through sparsity. Experimental results show that using groups of users and bi-groups (users/items) are a promising way to further improve the top-N recommendation performance for recommender algorithms. However, more studies need to be done to improve our proposed approach.

Despite of the accuracy of the obtained results in these works, in many real-life applications, both the number of items and users are large. In such cases, even when many events have been recorded, the user-item feedback matrix can still be extremely sparse, that is, there are very few elements in the matrix whose value is known. This problem, commonly referred to as the sparsity problem, has a major negative impact on the effectiveness of collaborative filtering approaches (AGGARWAL, 2016). Because of sparsity, it is very likely that the similarity (or correlation) between two given users cannot be properly measured or may not be reliable, rendering poor collaborative filtering-based recommendations (AGGARWAL, 2016; KUMAR; , 2018). Besides that, very often different types of feedback are limited and expensive to obtain,

since labeling typically requires users' effort (BOBADILLA *et al.*, 2013; AGGARWAL, 2016) (e.g. apply a rating/ tag, make a review). It can be even more critical if one wants to train personalized models based on different types of feedback, as we did, since it requires large amounts of labeled data of each type of feedback from each individual.

In the next chapter, we will present an approach to increase the number of interactions in a sparse database with reliable data, through an active learning technique[7].

[7] "Active learning is a special case of machine learning in which a learning algorithm is able to interactively query the user (or some other information source) to obtain the desired outputs at new data points (SETTLES, 2010)."

CHAPTER

6

COREC: A CO-TRAINING APPROACH FOR RECOMMENDER SYSTEMS

6.1 Contextualization

As mentioned in the previous chapter, sparsity issue, another occurring problem in recommender systems, especially with collaborative filtering algorithms, is the cold-start problem (BOBADILLA *et al.*, 2013; AGGARWAL, 2016; KUMAR; , 2018). Items or users are considered cold if they have few or none interactions, leading to the system not being capable to recommend a cold item or to provide proper suggestions to a cold user. The item cold-start problem, also called new item problem, can be smoothed by aggregating item information into its calculation, thus performing a hybrid recommendation solution. To solve these problems, we describe in this chapter an architecture to enrich the user-item feedback matrix with reliable information generated from a pre-processing semi-supervised strategy, capable of labeled relevant unknown pairs user-item to improve the quality of the recommendation using a semi-supervised learning approach strategy, called Co-training.

CoRec: A Co-Training Approach for Recommender Systems

Arthur F. da Costa[1], Marcelo G. Manzato[1] and Ricardo J. G. B. Campello[2,1]
[1]Institute of Mathematics and Computer Science
University of São Paulo, São Carlos, SP, Brazil
{fortes, mmanzato, campello}@icmc.usp.br
[2]College of Science and Engineering
James Cook University, Townsville, QLD, Australia
ricardo.campello@jcu.edu.au

ABSTRACT

In Recommender Systems, a large amount of labeled data must be available beforehand to obtain good predictions. However, labeled data are often limited and expensive to obtain, since labeling typically requires human expertise, time, and labor. This paper proposes a framework, named CoRec, which is based on a co-training approach that drives two recommenders to agree with each other's predictions to generate their own. We used three publicly available datasets from movies, jokes and books domains, as well as two well-known recommender algorithms, to demonstrate the efficiency of the approach under different configurations. The experiments show that better accuracy can be obtained when recommender algorithms are simultaneously co-trained from multiple views to make predictions.

CCS CONCEPTS

• **Information systems** → **Recommender systems**; • **Computing methodologies** → *Machine learning approaches.*

PUBLICATION INFORMATION

Published in proceedings of the 33rd Annual ACM Symposium onApplied Computing (SAC '18). 2018. Pages 696-703.
DOI:10.1145/3167132.3167209

KEYWORDS

Co-Training, Recommender Systems, Semi-supervised Learning

1 INTRODUCTION

In the last decades, the widespread access to the Web has increased the diversity of available new content, causing an information overload, which has created a potential problem for many Internet users [7]. The concept of information overload was proposed by Bertram Gross in 1964 as the attempt of users to handle more information than they are able to process in order to make sensible decisions [7]. Although computational resources are constantly improving, the human mind can not follow and process the amount of information at the same pace.

In this context, recommender systems (RS) have emerged using machine learning and information retrieval techniques to deal with and improve the experience of users on the Web [1, 14]. These systems are responsible for learning from users about their tastes on past interactions (ratings, tags, browsing history, etc.) and recommending content that is similar to their preferences [1, 3]. Collaborative filtering has been the most popular and efficient approach used in RS. It works by aggregating the interactions of similar users to generate personalized recommendations. However, this approach has limitations, such as sparsity and cold start [1, 3].

The sparsity problem is the phenomenon of not observing enough information from users and items in the dataset to train the model accurately [1, 3, 12, 14]. The cold start problem refers to the situation in which a user or item is new in the system, and the lack of sufficient previous feedback makes the system to not generate accurate recommendations. These problems are often tackled by using hybrid approaches or external sources of information (e.g. demographics of users, characteristics of items, etc.) [1, 3]. However, they narrow down the palette of applicable algorithms to the few ones able to use them and exclude many users who do not have the required data.

To overcome the lack of labeled data during training and still obtain high quality recommendations, semi-supervised learning (SSL) can be adopted [19]. One of the goals of the study of semi-supervised learning is to understand how the combination of labeled and unlabeled data can change the learning behavior, designing algorithms to take advantage of both types of data simultaneously. Basically, it exploits relationships between labeled and unlabeled data to make more accurate predictions, as opposed to using solely labeled data for training, as in the traditional supervised learning setting [19]. An important research field of SSL is Co-training, a multi-view learning approach where the feature space can be partitioned into disjoint subsets (views) that would suffice to learn the target concept independently, if there were sufficient labeled data to do so [2]. Co-training algorithms can significantly reduce the amount of labeled data required for learning, relying on the assumption that the views are compatible and uncorrelated (i.e., every learning example is identically labeled by the target concept in each view; and, given the label of any example, its descriptions in each view are independent) [2, 16]. We argue that co-training approaches can be used to tackle traditional recommendation problems such as sparsity and cold start by enriching the datasets as a pre-processing step for the recommender algorithms. This enrichment, in turn, is accomplished in a natural way, since we are able to define different and independent views to describe users and items. For example, information about users' ratings can be treated as one view and items' descriptions can be treated as another view.

We propose in this paper a co-training approach named CoRec to predict unlabeled examples and increase accuracy of results. To that end, we choose the rating prediction scenario, in which recommender algorithms try to predict users' ratings for all unseen items in the dataset, similarly to regression and classification tasks. Our approach is able to construct different recommenders by incorporating different views of the data, which helps to reduce sparsity

and cold start by making use of cheap and abundant unlabeled data in addition to the original labeled data available for training.

Our proposal was compared against different well-known collaborative recommenders, and different types of predictions generated by CoRec were evaluated to demonstrate its generality. The experiments were performed with three real datasets from different domains and show that the proposed co-training approach outperforms standalone predictions, by coupling the learning of multiple recommenders from independent views during the training step.

This paper is structured as follows: in Section 2 we overview recommender models related to multi-view learning and cold-start. We then present our approach in Section 4, and show evaluation results in Section 5. In Section 6 we present our conclusions as well as discuss current limitations and future work.

2 RELATED WORK

One of the main problems in recommender systems is dealing with extreme sparsity of data and cold-start [1, 3, 14]. The most common solution to these problems is to fill the missing feedback in the user-item matrix with default values (e.g. most popular item, or user averages). However, this approach lacks personalization, is based on ad-hoc heuristics, and may include noisy data into the process [1, 3], besides being time and memory consuming. Another alternative is to use active learning techniques, where an algorithm chooses which feedback to request from a user in order to improve a predefined gain criterion [10]. Some studies have suggested the use of explicit feedback agents (e.g. ratings), known as filterbots, to increase the feedback of items based on their features [10, 13], while some others have combined a user model with trust and distrust networks to recognize trustworthy users [6]. In Sun's work [16], a recommender approach was proposed that conducts an interview process as guided by a decision tree with multiple questions at each split, to learn users' behavior. The main problem with the aforementioned approaches is that they use side information, such as items' categories and users' location, which may not be available. Furthermore, the algorithms usually have high computational cost, which may make them impractical for many real scenarios.

As an alternative, algorithms based on multi-view learning, such as co-training, can be applied. Rather than side information, which may not be available, these algorithms can use instead, for instance, recommender algorithms following completely different paradigms, as different views of the same data. So far, very little work has been done in this direction though. The authors in [19] proposed a co-training method for stationary, batch-based recommender systems, using items' metadata to enrich the traditional training model. In a more recent work [20], the authors proposed a multi-view learning framework with the ability of knowledge transferring for recommendation. Their framework was developed for item recommendation and can learn multi-view representations automatically from data, without the need for multi-view data representation to be available in advance. However, both techniques achieve improvements by using extra content/information; in fact, two or more different and independent types of *metadata* are required to build different training sets; for example, items' genre and ratings. In real scenarios, it is rare to find a variety of such information for recommendation tasks.

Our proposed approach lifts those limitations by using only a training set and two recommenders for multi-view learning, thus we use as viewpoints the recommenders and not the input data. In addition, we propose and use a more robust reliability (confidence) function, which weights the predictions made by each recommender.

3 BACKGROUND

In this section, we review concepts related to our approach. Specifically, prior to describing our proposal, in the following we revise the recommender algorithms and the co-training concept that will be used later on.

3.1 Notation

We use special indexing letters to distinguish users and items: a user is indicated as u, a known item is referred to as i; and r_{ui} is used to refer to a rating from a user u to an item i. The final prediction of the system about the preference of user u to item i is represented by $\hat{r}_{ui}$, which is a floating point value guessed by the recommender algorithm. The set of pairs (u, i), for which r_{ui} is known, is represented by the set $D = \{(u, i) : r_{ui} \text{ is known}\}$.

3.2 User k-Nearest Neighbors (User-KNN)

One of the recommendation algorithms that we will use for experiments with our proposed CoRec approach is the well-known User-KNN [1, 14, 17], which produces recommendations based on similar users. The basic idea of the algorithm is to predict a user's rating based on the ratings of similar users.

In detail, a rating $\hat{r}_{ui}$ is predicted for an unknown user-item pair, (u, i), considering the ratings that other users with similar preferences to u have assigned to item i. To find similar users, a measure of proximity p_{uv} is employed between the feature vectors describing the users (i.e., the respective rows of the user-item matrix). The proximity measure can be based, e.g., on the Pearson correlation coefficient, or the cosine similarity, among others. The final similarity measure is a retracted coefficient, defined as:

$$s_{uv} = \frac{n_{uv}}{n_{uv} + \lambda_1} p_{uv} \tag{1}$$

where n_{uv} is the number of items that users u and v have in common, and λ_1 is a regularization constant, set to 100 according to the literature [11]. The idea is to assign higher (lower) weights to proximity values associated with pairs of users sharing larger (smaller) numbers of rated items.

In order to predict a rating $\hat{r}_{ui}$ for an unknown user-item pair, (u, i), the algorithm identifies the k users that are the most similar to u using the similarity measure between users described above. The algorithm then selects the subset of those users that have evaluated item i. This subset is denoted as $S^k(i; u)$. The final rating is then predicted using Equation (2):

$$\hat{r}_{ui} = b_{ui} + \frac{\sum_{v \in S^k(i;u)} s_{uv}(r_{vi} - b_{vi})}{\sum_{v \in S^k(i;u)} s_{uv}} \tag{2}$$

where b_{ui} and b_{vi} are baseline estimates based on the ratings provided by users, proposed by [11] for neighborhood models, and are defined as:

$$b_{ui} = \mu + b_u + b_i \tag{3}$$

where μ is the global average rating and b_u and b_i are rating estimates for user u and item i, respectively, which are based on their interactions. Such estimates can be calculated by iterating n times the following equations:

$$b_i = \frac{\sum_{u:(u,i)\in D}(r_{ui} + \mu - b_u)}{\lambda_2 + |\{u : (u,i) \in D\}|} \tag{4}$$

$$b_u = \frac{\sum_{u:(u,i)\in D}(r_{ui} + \mu - b_i)}{\lambda_3 + |\{u : (u,i) \in D\}|} \tag{5}$$

where λ_2 and λ_3 are constants set as 10 and 15, respectively, according to [11]. The main purpose of b_{ui} is to capture systematic tendencies of certain users to give higher/lower ratings than others, and certain items to receive higher/lower ratings than others.

3.3 Item k-Nearest Neighbors (Item-KNN)

The second recommender algorithm used in our experiments is Item-KNN [1, 11, 14, 15], which also uses the concept of nearest neighbors. The main difference between User-KNN and Item-KNN is that the latter predicts the unknown rating r_{ui} of an item i by a user u based on u's ratings of the k items most similar to i. In order to find similar items, a similarity measure is employed between items, as shown in Equation (6):

$$s_{ij} = \frac{n_{ij}}{n_{ij} + \lambda_1} p_{ij} \tag{6}$$

where p_{ij} is a measure of proximity (e.g. Pearson or cosine) employed between the feature vectors describing the items (i.e., the respective columns of the user-item matrix) and n_{ij} is the number of shared features that describe items i and j (i.e., number of users that rated both items). The value of λ_1 is the same used in Equation (1).

Using the similarity measure in Equation (6), we identify the set of k items that are most similar to i, the so-called k-nearest neighbors, and select the subset of those items that have been rated by u. Using this subset, denoted as $Si^k(i;u)$, the final predicted rating is an average of such similar items' ratings, adjusted to their baseline estimate:

$$\hat{r}_{ui} = b_{ui} + \frac{\sum_{j\in Si^k(i;u)} s_{ij}(r_{uj} - b_{uj})}{\sum_{j\in Si^k(i;u)} s_{ij}} \tag{7}$$

where b_{ui} and b_{uj} are computed by Equation (3).

3.4 Co-Training

Co-training relates to the problem of simultaneously learning from data represented by multiple distinct views [2]. The emergence of this learning scheme is largely motivated by the fact that data in a variety of real applications are (or can be) described by different feature sets or different views of the same feature set [16, 18]. Generally, co-training is used when there are only small amounts of labeled data yet large amounts of unlabeled data described by two or more views that are ideally conditionally independent and sufficient to represent the target concept [2]. Co-training first learns a separate prediction model (e.g. a classifier) for each view using any labeled examples. The most confident predictions of each model on the unlabeled data are then used to iteratively obtain additional (augmented) labeled training sets for further refinement of the other model(s).

This method assumes that each example is described using at least two different views that provide complementary information. Typically, different views can be obtained from disjoint feature subsets of different natures. For example, in multimedia content understanding, multimedia segments can be simultaneously described by their video and audio signals. In web-page classification, a web page can be described by the document text itself and also by the anchor text attached to hyperlinks pointing to this page [2].

However, co-training can also be performed when there is only one representation of the data, if such a single representation is processed by independent prediction models, such as two different classifiers [18]. In this case, the method exploits two views of the same, single data representation by co-training two classifiers using the available training examples. Iteratively, each classifier labels the unlabeled examples and the most confident predictions are provided as additional training examples to improve the accuracy of the other classifier.

Here we extend this idea to the ratings prediction problem in recommender systems in order to tackle sparsity and cold start issues by simultaneously considering multiple views of the same data. Our method couples the training of two different, independent recommender systems in order to mutually and iteratively improve their accuracy by making each recommender provide the other with its most confident predictions, in a co-training fashion. The method is discussed in more detail in the next section.

4 PROPOSED WORK

We propose a co-training approach (CoRec) to improve the accuracy in recommender systems. Our approach aims at building a semi-supervised learning using two recommender algorithms as viewpoints, in order to provide more accurate predictions and to significantly reduce the sparsity problem, since the proposed method enriches the original ratings matrix. CoRec uses only a single training set as input data and two distinct recommender algorithms as different views.

In CoRec, a recommender is trained aiming to reach an agreement on the predictions of the other recommender, such that both recommenders learn not only from the original training set of labeled data, but also from each other's most confident predictions of the unlabeled data. In the context of rating predictions, a "label"

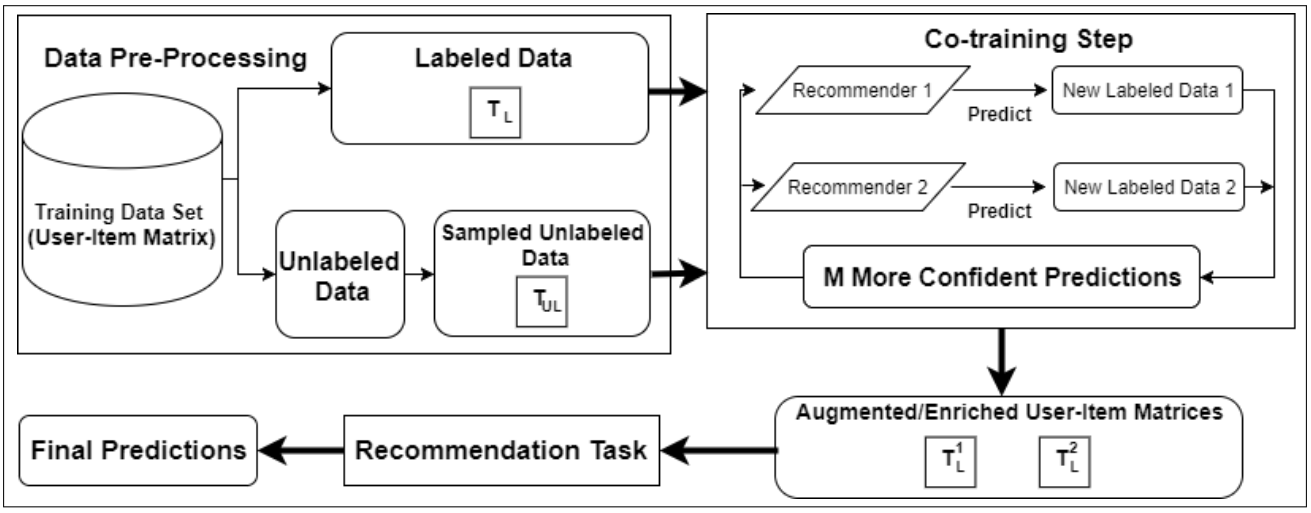


Figure 6. 1. Schematic visualization of the proposed system.

means a rating r_{ui} of an item i by a user u, and "labeling" unlabeled data means assigning predicted ratings $\hat{r}_{ui}$ for unobserved user-item pairs (u, i).

In CoRec, at each iteration, the recommender representing one view labels the unlabeled data, which are then added to the training pool of the other recommender, and vice-versa. Thus, the information underlying two different views can be exchanged. Specifically, each recommender predicts ratings for all unobserved user-item pairs (unlabeled set), and the most confident predictions are used to populate the training set (i.e., to augment/enrich the labeled set) of the other recommender, in an iterative fashion. Figure 6.1 illustrates the overall process. It can be seen in the figure that, in contrast to single recommenders, our approach simultaneously learns two models interactively, in order to exploit different viewpoints of the same input data and boost the learning performance.

There are three phases in our technique: data pre-processing, co-training, and recommendation. In the following subsections, we describe all procedures in more details.

4.1 Data Pre-Processing

CoRec first splits the original data set into a labeled set (T_L^η) and an unlabeled set (T_{UL}^η) for co-training its two recommender algorithms ($\eta = 1, 2$). Initially, these sets are the same (as input) for both recommenders, i.e., $T_L^1 = T_L^2$ and $T_{UL}^1 = T_{UL}^2$. The initial labeled set, T_L^η, is constituted by all observed user-item pairs, i.e., all entries of the user-item matrix for which ratings are available. Since the user-item matrix is sparse, the amount of unobserved entries can be enormous. Instead of including all the unobserved user-item pairs into the initial unlabeled set T_{UL}^η that is provided to the recommenders, which would impose an unnecessary additional computational burden and slow down convergence of the co-training process, we instead randomly select X unlabeled pairs (u, i) per user. For example, if $X = 10$, for each user in the original user-item matrix, 10 items unknown to that user will be randomly chosen to be included into the unlabeled set (T_{UL}^η). The task of the recommenders during co-training is to jointly learn how to predict the ratings that users would give to the pairs in this unlabeled set.

4.2 Co-Training Approach

With the recommenders and the training sets in hand, the next step is to apply the co-training on these recommender algorithms. Algorithm 6.1 shows how the method of semi-supervised co-training works in CoRec.

As mentioned before, our co-training process consists of evaluating the same data on two different recommenders, driving them to agree to each other's most confident predictions. To this end, any two different rating prediction algorithms could in principle be used, provided that they represent two distinct views of the data. In this paper, we use User-KNN and Item-KNN (reviewed in Sections 3.2 and 3.3, respectively) as different views of the same data, where the first favors similarity between users, whereas the second favors similarity between items.

For every iteration of the CoRec algorithm, each recommender η ($\eta = 1, 2$) is responsible for predicting ratings for its unlabeled set, T_{UL}^η, based on its labeled set, T_L^η (line 10). Then, CoRec calculates the confidence of each predicted rating, choosing the M most confident ones and removing them from the set of unlabeled data for the corresponding recommender (line 18). Once this procedure has been performed for both recommenders, the algorithm updates the labeled set of each recommender by adding the M more confident examples to the other recommender's labeled set (lines 20 and 21), since we expect each model to learn only from the newly labeled examples from the other view (besides the original examples). The intuition behind our approach is that one recommender adds confident new examples to the labeled set that the other recommender will then use for learning, but from another perspective (view). The algorithm only stops when all unlabeled data of both views have been labeled.

Input: User-item matrix, X and M (positive integers)
Step 1:
$T_L \leftarrow$ non-null entries of the user-item matrix
$T_{UL} \leftarrow X$ randomly selected null entries per user
Step 2: Co-training
for η=*1 to* 2 **do**
  $T_L^\eta, T_{UL}^\eta \leftarrow T_L, T_{UL}$
end
while $T_{UL}^1 \neq \emptyset, T_{UL}^2 \neq \emptyset$ **do**
  for η=*1 to* 2 **do**
    Train the Recommender η with T_L^η to predict T_{UL}^η
    foreach $r_{ui} \in T_{UL}^\eta$ **do**
      Obtain the confidence C_{ui}^η by Equation (8)
    end
    $T_\eta \leftarrow$ Select M more confident examples
  end
  $T_F \leftarrow T_1 \cup T_2$
  foreach η=*1 to* 2 **do**
    $T_{UL}^\eta \leftarrow T_{UL}^\eta - T_F$
  end
  $T_L^1 \leftarrow T_L^1 \cup T_2$
  $T_L^2 \leftarrow T_L^2 \cup T_1$
end
Step 3: Recommendation Task
for η=*1 to* 2 **do**
  Train the Recommender η with T_L^1 and T_L^2 to predict new ratings for unknown items for each user
end
Output: Two sets of unknown pairs (u, i) with their respective predicted ratings ($\hat{r}_{u.i}$).

Algorithm 6. 1: CoRec Algorithm.

Notice that the (positive integer) parameter M controls the learning rate of the co-training process. As usual in statistical and machine learning algorithms, slow learning (corresponding to small values of M) tends to produce more accurate models, at a higher computational price though (larger number of iterations required for convergence).

Confidence Measure: In line 14 of Algorithm 6.1, we need to determine the examples that have been predicted more confidently by each recommender, in the sense that the prediction tends to be more accurate according to some criterion. In traditional methods, the user may benefit from observing confidence scores [9, 19]; for example, when the algorithm reports a low confidence in a recommended item, the user may tend to further inspect the item before making a decision. However, not every recommender algorithm can generate a confidence score by itself, or there may be a high cost to compute one. The confidence criterion we propose and used in this work is based on the fact that collaborative filtering recommenders tend to improve their accuracy as the amount of data over items/users grows. Inspired by Zhang et al. [19], we then propose our confidence measure for a prediction $\hat{r}_{ui}^\eta$ made by recommender algorithm η for the rating of pair (u, i) as follows:

$$C_{ui}^\eta = \sigma_{ui}^\eta N_u N_i \tag{8}$$

where N_u and N_i are the popularity of user u and item i, respectively, given by the number of ratings applied/received by user u and item i in the labeled set associated with recommender η, and σ_{ui}^η represents a measure of trustworthiness of the prediction made by recommender η, defined as:

$$\sigma_{ui}^\eta = \frac{1}{|b_{ui} - \hat{r}_{ui}^\eta|} \tag{9}$$

where b_{ui} is a baseline estimate of the rating for the pair (u, i) in question, calculated based on the global mean and the mean of interactions of users and items (in each augmented labeled set generated by CoRec, T_L^1 and T_L^2), using Equation (3).

The rationale behind our proposed confidence measure is that the more popular users and items are (i.e. the more ratings are associated with them), the more accurate the predictions calculated based on these users and items tend to be. In addition, the confidence on a prediction deviating largely from the baseline rating b_{ui}, which takes into account the overall ratings given/received by the user u and the item i in question, tends to be penalized.

4.3 Recommendation Task

The output of CoRec are two augmented labeled sets, T_L^1 and T_L^2, which can then be used to enrich the original user-item matrix, each of which gives rise to a new, more populated matrix. The resulting enriched matrices can be used by a new recommender to make rating predictions for the user-item pairs still missing in these matrices (unlabeled), or for new users and items yet to appear.

In other words, the rating prediction for an unknown user-item pair can be computed from a given recommender that will be trained using the user-item matrix enriched with either T_L^1 or T_L^2. For the experiments reported in this paper, both User-KNN and Item-KNN recommenders as well as both augmented labeled sets T_L^1 and T_L^2 will be evaluated.

5 EVALUATION

In our experiments, we applied the co-training process using Item-KNN and User-KNN as recommenders (see Section 3), with Pearson correlation as proximity measure. These algorithms were chosen because they make recommendations based on different views of the data: User-KNN uses interrelations between users, while Item-KNN uses interrelations between items to make predictions. These algorithms are very efficient and well-known in the literature [1, 3, 14]. In order to evaluate our proposed method, CoRec, we compare it against the standalone results of User-KNN and Item-KNN, as baselines. In addition, we also compare CoRec against the semi-supervised learning method CSEL [19], which is a recommender framework for processing different types of information, including item's metadata. We adapted their algorithm to accept ratings only, so a fair comparison between the results could be achieved.

5.1 Datasets

We evaluate our approach on three publicly available datasets from different domains, which contain explicit user's feedback.

- **MovieLens** $2k$**:** consists of 855,599 ratings (between 0.5 to 5) applied to 2,113 users and 10,197 movies. This dataset was developed by [5] and has not been modified for the experiments.
- **BookCrossing:** was developed in [21], and contains 278,858 users (anonymized but with demographic information) providing 1,149,780 ratings (explicit / implicit) about 271,379 books. Following the original authors, we also used a subsample of the data, filtering out all items with less than 5 interactions in the dataset. We also rescaled the original ratings (0 to 10) into a new range (0.5 to 5), resulting in 102,963 interactions made by 6,628 users to 7,164 books.
- **Jester:** consists of 1,810,455 ratings applied by 23,500 users to 100 jokes. For our experiments, we used only a sampling of the original dataset picking random users as proposed by [8], resulting in 360,917 interactions made by 5,000 users to 100 items. Original ratings (−10 to 10) have also been rescaled into 0.5 to 5.

5.2 Experimental Setup and Methodology

Recall from Section 4 that CoRec generates two augmented labeled sets, T_L^1 and T_L^2, which are then used to populate the original user-item matrix with predictions made during the co-training process. The enriched matrices resulting from T_L^1 and T_L^2 are both used in order to evaluate the results. In particular, we apply all four recommenders (CoRec and the three baseline competitors, namely, User-KNN, Item-KNN, and CSEL) to both enriched matrices.

To measure the predictive ability of the different methods, we used the root mean square error measure (RMSE) [1, 3, 14], with 10-fold cross-validation. We computed the mean across the 10-folds and compared our method against the competitors using a two-sided paired t-test with a 99% confidence level [1, 4].

The parameters of the algorithms were determined experimentally. We ran experiments for X (number of unlabeled items per user in co-training) equal to 5, 10, 30, 50, 70 and 100, and chose X equal to 10, 30 and 50 as these values provided the best results. As for the parameter M, which is used in both CoRec and CSEL, we experimented with different values, and chose M as 10% of the size of the unlabeled set, T_{UL}, which leads to a good trade-off between convergence rate (computational cost) and accuracy of the final results across all datasets. For User-KNN and Item-KNN, we use Pearson correlation to compute similarity, as our results indicate that this similarity is superior to the other measures that we tested.

We developed CoRec in Python version 3.6, and the source-code is freely available in Github. The baseline competitors belong to the Case Recommender Framework version 0.0.20, an open source tool developed in *Python*, with numerous features and algorithms for recommender systems. We used the default framework settings in our evaluations.

[1] http://www.python.org/
[2] https://github.com/ArthurFortes/CaseRecommender
[3] https://pypi.python.org/pypi/CaseRecommender

5.3 Results

Table 6.1 compares the results of the User-KNN and Item-KNN recommenders (a) when applied to the original user-item ratings matrix (column "Original") as well as (b) when applied to the enriched matrices obtained from the augmented labeled sets T_L^1 and T_L^2 produced by CoRec, for different values of the parameter X (remaining columns).

Table 6.2 is analogous, however, it is designed specifically to assess the cold-start scenario. In particular, it reports the RMSE computed only for those users with less than 20 ratings in the dataset.

In both tables, it can be seen that using the enriched user-item matrices produced by CoRec leads to improved results, and such an improvement is more noticeable for larger values of X. To a certain extent, increasing X increases the chances that more examples are predicted with high confidence during co-training, to suitably fill the enriched matrix.

Table 6.3 shows the results comparing CSEL and CoRec, using the best enriched matrix and recommender for each dataset from Table 6.1 in the case of CoRec, and the best among all the results obtained from CSEL. In all three datasets, CoRec achieved better results in overall RMSE than CSEL.

Table 6. 3. Comparison between KNN-based CoRec and CSEL in terms of RMSE. Bold typeset indicates the best performance. * indicates statistical significance at p-value <0.01.

Dataset / Approach	CoRec	CSEL
MovieLens $2k$	**0.7410***	0.7580
BookCrossing	**0.7762***	0.7864
Jester	**0.9534***	0.9895

5.4 Discussion

The experiments show that, by using CoRec to enrich the user-item matrix, we obtain better results than by using standalone recommenders based on the original dataset. The reason is that the enrichment reduces sparseness by filling unobserved entries with highly confident predictions obtained from multiple views of the data, provided by different recommender algorithms. Table 6.4 shows the number of ratings in each dataset with and without the use of CoRec.

Table 6. 4. Number of ratings in each dataset.

		X		
Dataset	**Original**	**10**	**30**	**50**
BookCrossing	82,370	148,650	281,210	413,770
MovieLens 2k	684,447	876,729	918,989	961,249
Jester	288,734	338,734	438,734	538,734

Our co-training approach was able to successfully increase the number of ratings, leading to lower RMSE in KNN-based algorithms. The improvement in RMSE suggests that the augmented ratings are accurate, which is due to two main factors. The first factor is the use of a confidence measure that gives less importance to outliers

Table 6. 1. Comparison between CoRec and standalone KNN-based recommenders in terms of RMSE.

Unlabeled items per user (X)		**10**		**30**		**50**	
Algorithms / User-Item Matrix	Original	T_L^1	T_L^2	T_L^1	T_L^2	T_L^1	T_L^2
MovieLens $2k$							
User-KNN	0.7995	0.7897	0.7904	0.7894	0.7906	**0.7814***	0.7893
Item-KNN	0.7460	0.7458	0.7456	0.7452	0.7448	**0.7410***	0.7429
BookCrossing							
User-KNN	0.9936	0.7854	0.7874	0.7861	0.7859	**0.7853***	0.7855
Item-KNN	0.8662	0.8180	0.8189	0.7955	0.7885	0.7832	**0.7762***
Jester							
User-KNN	1.0208	0.9851	0.9966	0.9723	0.9893	**0.9664***	0.9855
Item-KNN	0.9879	0.9645	0.9631	0.9606	0.9608	0.9588	**0.9534***

Bold typeset indicates the best performance. * indicates statistically significant difference at p <0.01.

Table 6. 2. KNN-based recommenders with and without CoRec data enrichment in terms of RMSE: cold-start scenario.

Unlabeled items per user (X)		**10**		**30**		**50**	
Algorithms / User-Item Matrix	Original	T_L^1	T_L^2	T_L^1	T_L^2	T_L^1	T_L^2
MovieLens $2k$ **(113 new users)**							
User-KNN	1.0055	0.9995	0.9986	0.9855	0.9889	**0.9631***	0.9686
Item-KNN	0.9655	0.9652	0.9643	0.9647	0.9645	0.9607	**0.9597***
BookCrossing (2,675 new users)							
User-KNN	1.2438	1.1086	1.1164	0.8689	0.8677	**0.7971***	0.7997
Item-KNN	1.0844	0.9884	0.9865	0.8194	0.8208	0.8180	**0.8159***
Jester (43 new users)							
User-KNN	1.0027	1.0015	1.0016	1.0021	1.0023	**0.9937***	0.9946
Item-KNN	1.0028	0.9598	0.9595	0.9593	**0.9588***	0.9666	0.9659

Bold typeset indicates the best performance. * indicates statistically significant difference at p <0.01.

and users/items with few interactions (ratings). The other factor is the co-learning of prediction models from independent views of the data.

The results presented in Tables 6.1, 6.2 and 6.3 show that CoRec was able to achieve better accuracy in three different domains (recommendation of books, movies and jokes) with statistical significance (p-value <0.01). Although some recommenders may exhibit worse predictions than others individually, during the co-training process they complement each other and generate better results than the baselines. Most of the results showed improvement in the accuracy of the recommendation, especially when using more than ten new ratings per user in the co-training process. This emphasizes that we can better represent a user's behavior when we have more information about him/her.

In CoRec, convergence is reached when the unlabeled set (T_{UL}) is empty, i.e., when all its original pairs (u, i) have been labeled. Hence, the computational cost is approximately t times the cost of the individual recommender algorithms, where t is the number of iterations required for convergence. In practice, this additional burden can be mitigated by performing the co-training process off-line as well as by training the two recommenders in a parallel fashion.

6 CONCLUSIONS AND FUTURE WORK

This paper proposed a semi-supervised co-training approach, named CoRec, which processes data from different views in order to circumvent the lack of ratings and provide better recommendations. We conducted experiments in three data sets from different domains and the results show that our strategy improves the overall system's performance and makes progress in the semi-supervised problem in recommender systems. The main advantage of our approach is the possibility to reduce the effects of sparsity and cold start by enriching the original feedback matrix with confident ratings obtained through a co-training process. The proposed approach, CoRec, can be seen as a framework that enables the use of different recommender algorithms as well as alternative confidence measures for co-training.

In our experiments, we used two neighborhood models (User-KNN and Item-KNN) as multiple views of the same data. As future work, we aim at evaluating the system with additional datasets from

other domains and checking their accuracy with different recommender types, including a hybrid approach which uses both neighborhood models and factorization matrix methods. Meanwhile, this semi-supervised strategy can be easily extended to include more than two recommenders. For example, three recommenders can be constructed and a tri-training can be performed. Experiments in this direction will also be conducted in future work. Finally, we also intend to investigate how to combine the enriched matrices generated by CoRec, and/or the corresponding recommendation results, through ensembles.

ACKNOWLEDGMENTS

We would like to acknowledge CAPES and FAPESP (2016/20280-6) for the financial support.

REFERENCES

[1] Charu C. Aggarwal. 2016. *Recommender Systems: The Textbook* (1st ed.). Springer Publishing Company, Incorporated. 1221.31212.10.2/5423

[2] Avrim Blum and Tom Mitchell. 1998. Combining Labeled and Unlabeled Data with Co-training. In *Proceedings of the Eleventh Annual Conference on Computational Learning Theory (COLT' 98)*. ACM, New York, NY, USA, 92–100. https://doi.org/10.1145/279943.279962

[3] J. Bobadilla, F. Ortega, A. Hernando, and A. GutiéRrez. 2013. Recommender Systems Survey. *Know.-Based Syst.* 46 (July 2013), 109–132. https://doi.org/10.1016/j.knosys.2013.03.012

[4] Joan Fisher Box. 1987. Guinness, Gosset, Fisher, and Small Samples. *Statist. Sci.* 2, 1 (02 1987), 45–52. https://doi.org/10.1214/ss/1177013437

[5] Brusilovsky Peter Cantador, Ivan and Tsvi Kuflik. 2011. 2nd Workshop on Information Heterogeneity and Fusion in Recommender Systems (HetRec 2011) *(RecSys 2011)*. ACM, New York, NY, USA.

[6] Chien Chin Chen, Yu-Hao Wan, Meng-Chieh Chung, and Yu-Chun Sun. 2013. An effective recommendation method for cold start new users using trust and distrust networks. *Information Sciences* 224 (2013), 19 – 36. https://doi.org/10.1016/j.ins.2012.10.037

[7] John Gantz and David Reinsel. 2013. The digital universe in 2020: Big data, bigger digital shadows, and biggest growth in the far east. *IDC Analize the Future* (2013).

[8] Ken Goldberg, Theresa Roeder, Dhruv Gupta, and Chris Perkins. 2001. Eigentaste: A Constant Time Collaborative Filtering Algorithm. *Inf. Retr.* 4, 2 (July 2001), 133–151. https://doi.org/10.1023/A:1011419012209

[9] Guibing Guo. 2013. Improving the Performance of Recommender Systems by Alleviating the Data Sparsity and Cold Start Problems. In *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence (IJCAI '13)*. AAAI Press, 3217–3218.

[10] R. Karimi, C. Freudenthaler, A. Nanopoulos, and L. Schmidt-Thieme. 2011. Towards Optimal Active Learning for Matrix Factorization in Recommender Systems. In *2011 IEEE 23rd International Conference on Tools with Artificial Intelligence*. 1069–1076.

[11] Yehuda Koren. 2008. Factorization meets the neighborhood. *Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD 08* (2008), 426–434. https://doi.org/10.1145/1401890.1401944

[12] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix Factorization Techniques for Recommender Systems. *Computer* 42, 8 (2009), 30–37. https://doi.org/10.1109/MC.2009.263

[13] Seung-Taek Park, David Pennock, Omid Madani, Nathan Good, and Dennis DeCoste. 2006. NaïVe Filterbots for Robust Cold-start Recommendations. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '06)*. ACM, New York, NY, USA, 699–705. https://doi.org/10.1145/1150402.1150490

[14] Francesco Ricci, Lior Rokach, Bracha Shapira, and Paul B. Kantor. 2011. *Recommender Systems Handbook* (1st ed.). Springer-Verlag New York, Inc., New York, NY, USA.

[15] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2001. Item-based Collaborative Filtering Recommendation Algorithms. In *Proceedings of the 10th International Conference on World Wide Web (WWW '01)*. ACM, New York, NY, USA, 285–295. https://doi.org/10.1145/371920.372071

[16] Mingxuan Sun, Fuxin Li, Joonseok Lee, Ke Zhou, Guy Lebanon, and Hongyuan Zha. 2013. Learning Multiple-question Decision Trees for Cold-start Recommendation. In *Proceedings of the Sixth ACM International Conference on Web Search and Data Mining (WSDM '13)*. ACM, New York, NY, USA, 445–454. https://doi.org/10.1145/2433396.2433451

[17] Jun Wang, Arjen P. de Vries, and Marcel J. T. Reinders. 2006. Unifying User-based and Item-based Collaborative Filtering Approaches by Similarity Fusion. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '06)*. ACM, New York, NY, USA, 501–508. https://doi.org/10.1145/1148170.1148257

[18] Chang Xu, Dacheng Tao, and Chao Xu. 2013. A Survey on Multi-view Learning. *CoRR* abs/1304.5634 (2013). http://arxiv.org/abs/1304.5634

[19] Mi Zhang, Jie Tang, Xuchen Zhang, and Xiangyang Xue. 2014. Addressing Cold Start in Recommender Systems: A Semi-supervised Co-training Algorithm. In *Proceedings of the 37th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '14)*. ACM, New York, NY, USA, 73–82. https://doi.org/10.1145/2600428.2609599

[20] Qing Zhang and Houfeng Wang. 2015. *Collaborative Multi-view Learning with Active Discriminative Prior for Recommendation.* Springer International Publishing, Cham, 355–368. https://doi.org/10.1007/978-3-319-18038-0_28

[21] Cai-Nicolas Ziegler, Sean M. McNee, Joseph A. Konstan, and Georg Lausen. 2005. Improving Recommendation Lists Through Topic Diversification. In *Proceedings of the 14th International Conference on World Wide Web (WWW '05)*. ACM, New York, NY, USA, 22–32. https://doi.org/10.1145/1060745.1060754

6.2 Final remarks

In this research we investigated how to solve the sparsity and cold-start problems of both items and users, through a semi-supervised learning approach, where we make two recommenders to concur with each other to predict scores of unknown items of each user and select those who have the greatest confidence to enrich the original input matrix. We apply the generated enriched representations into two well-known collaborative filtering algorithms to help minimizing the cold-start (for users and items) and sparsity problems. We compared the representations against each other and against the original recommender systems. Our study shows that their use can help reducing the prediction error of recommender systems in the normal and cold-start scenarios, besides alleviating the sparsity problem.

In this chapter, we investigated the research questions 1 and 3, "How can we improve the accuracy of the existing recommender algorithms using pre-processing approaches?" and "How can we treat the input data to alleviate traditional recommender systems problems, such as cold-start, dimensionality and sparsity?", using a co-training technique to enrich the traditional user-item matrix with new labeled user-item pairs in order to improve the accuracy of the recommendations. We built a framework, called CoRec, to facilitate the integration of the inputs and outputs of the recommendations and to calculate the confidence of the samples in the end of each iteration of co-training task. The final result is enriched data matrices (users-items), with a greater amount and representativeness of the data than the original matrix.

In the next chapter, we will present a more robust extension of the proposed approach presented in this chapter, named ECoRec, which combines the outputs of each of the recommender algorithms in the co-training strategy.

CHAPTER

7

BOOSTING COLLABORATIVE FILTERING WITH AN ENSEMBLE OF CO-TRAINED RECOMMENDERS

7.1 Contextualization

In this chapter, we present an ensemble approach for the combination of the outputs generated by CoRec algorithm, described in Chapter 6, in order to generate only one enriched set as input for recommender systems. In the last years, several recent studies (JAHRER; TöSCHER; LEGENSTEIN, 2010; RISTOSKI; MENCíA; PAULHEIM, 2014; VINAGRE; JORGE; GAMA, 2018; ZHANG *et al.*, 2014; ZHANG; WANG, 2015) demonstrate the effectiveness of ensembles and confident metrics of several individual and simpler techniques, and show that ensemble-based methods outperform any single, more complex algorithm and confident metrics could improve the accuracy and reliability of the recommendation. Following these works, we developed an ensemble module, since it is able to learn the weights of contribution of each view of the co-training process and combining then in a single enriched data set.

Instead of designing an model composed of two different enriched outputs as we performed in our previous work, we developed a pre-processing module based on ensemble technique, which combines a number of user-item ratings generated by collaborative filtering recommenders using a confidence metric based on user's feedback. In addition, we make possible the use of more than two recommenders in the co-training approach and intensify the experiments to ensure the efficiency of our approach. The evaluation experiments show the effectiveness of our proposal.

Boosting Collaborative Filtering with an Ensemble of Co-trained Recommenders

Arthur F. da Costa[a], Marcelo G. Manzato[a], Ricardo J. G. B. Campello[b,a]

[a]*Institute of Mathematical and Computer Science, University of São Paulo, Brazil*
{*fortes, mmanzato*}*@icmc.usp.br*
[b]*School of Mathematical & Physical Sciences, University of Newcastle, Australia*
ricardo.campello@newcastle.edu.au

Abstract

Collaborative Filtering (CF) is one of the best performing and most widely used approaches for recommender systems. Although significant progress has been made in this area, current CF methods still suffer from cold-start and sparsity problems. A primary issue is that the fraction of users willing to rate items tends to be very small in most applications, which causes the number of users and/or items with few or no interactions in recommendation databases to be large. As a direct consequence of sparsity, recommender algorithms may provide poor recommendations (reducing accuracy) or decline recommendations (reducing coverage). This paper proposes an ensemble scheme based on a co-training approach, named ECoRec, that drives two or more recommenders to agree with each others' predictions to generate their own. The experiments on eight real-life databases show that better accuracy can be obtained when recommender algorithms are simultaneously trained from multiple views and combined into an ensemble to make predictions.

Keywords:
Co-Training, Ensembles, Recommender Systems, Semi-supervised Learning

1. Introduction

In recent years, the interest in recommender systems has drastically increased, from both researchers and practitioners, with the explosive growth

Publication information: Published in Expert Systems with Applications Journal. 2019. Volume 115. Pages 427-441. DOI: https://doi.org/10.1016/j.eswa.2018.08.020

and variety of information available on the Web (Aggarwal, 2016; Ricci et al., 2015; Bobadilla et al., 2013). In this field, Collaborative Filtering (CF) approaches, especially matrix factorization and neighborhood models, have achieved significant improvements (Aggarwal, 2016; Bobadilla et al., 2013). These approaches operate based on users' previous interests encoded by the ratings matrix reflecting the similarities of users or items. However, they underperform when little collaborative information is available, which is related to known traditional recommender problems, such as cold-start and sparsity.

In many real-life applications, both the number of items and the number of consumers are large. In such cases, even when many events have been recorded, the consumer-product interaction matrix can still be extremely sparse, that is, there are very few elements in the matrix whose value is known. This problem, commonly referred to as the sparsity problem, has a major negative impact on the effectiveness of collaborative filtering approaches (Bobadilla et al., 2013). Because of sparsity, it is very likely that the similarity (or correlation) between two given users cannot be properly measured or may not be reliable, rendering poor collaborative filtering-based recommendations (Aggarwal, 2016; Bobadilla et al., 2013).

Another problem, known as cold-start, regards recommendations for new (or rarely rated) items or new (or inactive) users (Aggarwal, 2016; Bobadilla et al., 2013). For instance, as opposed to popular items, it is difficult for standard recommendation approaches to provide high-quality recommendations involving newly released items or items that are not so popular, for which very few ratings are available.

From the machine learning perspective, the cold-start and sparsity problems have a common root in the shortage of labeled data to train recommender algorithms in a traditional, fully supervised (i.e., informed/guided) way. In the realm of collaborative filtering, "labeled" data means known user/item interactions, such as ratings. However, very often labeled data are limited and expensive to obtain, since labeling typically requires users' feedback or human expertise (e.g. conducting user study). It can be even more critical if one wants to train personalized models, which requires large amounts of labeled data from each individual.

In order to minimize these problems, semi-supervised learning (SSL) has been adopted in recommender systems to boost learning performance by simultaneously using labeled and unlabeled data (Karimi et al., 2011; Zhang et al., 2014). The goal of semi-supervised learning is to take advantage of the combination of labeled and unlabeled data to improve on learning behav-

ior. SSL is of particular interest in big data scenarios as it can make use of abundant unlabeled data to enhance supervised learning tasks when labeled data are difficult or expensive to obtain (Zhu et al., 2010). Basically it takes advantage of the relationship among labeled and unlabeled data to make more accurate predictions than purely using labeled data for training (Zhu et al., 2010; Zhang et al., 2014). However, many SSL approaches achieve improvements by using extra content/information that may not be available in practical applications, such as independent types of *metadata* required to build different training sets; e.g., items' genres and ratings. Methods that follow this scheme typically enrich the original matrix with extra information that is difficult to obtain or requires an expert, or using over-simplistic heuristics instead (e.g. the average of ratings or the most common rating of the user) (Zhang et al., 2014; Karimi et al., 2011; Ristoski et al., 2014).

An important research field of SSL is *co-training*, a multi-view learning approach originally proposed by Blum and Mitchell (1998), where the features of the domain can be partitioned into disjoint subsets (views) that would presumably suffice to learn the target concept if enough labeled data was available (Xu et al., 2013). In a broader sense, co-training algorithms can use different views of the data other than disjoint subsets of features. The only assumption is that the different views are expected to be uncorrelated and compatible, i.e., a problem has uncorrelated views if, given the label of any example, its descriptions in each view are independent and two views are compatible if all examples are labeled identically by the target concepts in each view (Xu et al., 2013; Sun et al., 2013). In this context, the unlabeled data can be used to filter non compatible functions, hence yielding information that can help to learn the target function (Zhang and Lee, 2005). Beside that, the uncorrelated assumption ensures that each model will have different views from the same problem, removing bias on the data.

In a recent, very preliminary conference publication (Da Costa et al., 2018), we showed that co-training can be used to tackle traditional recommendation problems such as sparsity and cold-start by enriching the databases as a pre-processing step for the recommender algorithms, thus reducing the amount of labeled data required for learning. The proposed co-training technique for recommendation, called CoRec, was shown to be able to simultaneously construct different recommenders as different views of the data, making use of cheap and abundant unlabeled data in addition to the original labeled data available for training. However, CoRec produces two distinct models and predictions of the same data, each of which from a new,

enriched user-item matrix generated from one of its two independent views. This imposes the additional overhead of assessing and choosing between different models for the final recommendation. In addition, CoRec was designed and tested to operate with two distinct recommender algorithms (views) only.

In this paper, we propose a thorough extension of our preliminary work (Da Costa et al., 2018). Here, CoRec is not only described and tested much more comprehensively, but it is also conceptually extended. The proposed extensions are twofold. First, CoRec is generalized to operate with any number of recommender algorithms as multiple views, rather than only two. Second, and more importantly, we incorporated into the method an aggregation scheme that combines the multiple models generated during co-training into an ensemble. The resulting, extended algorithm is named ECoRec. Like CoRec, ECoRec is able to construct multiple recommenders by incorporating different views of the data, which ideally helps capture their characteristics from different and complementary viewpoints. In order to combine and improve the individual models generated by CoRec as they are simultaneously co-trained, ECoRec incorporates an in-built ensemble module based on a confidence criterion, to generate only one enriched and confident user-item matrix, which can be used by any recommender algorithm at the end of the process. In addition, our approach allows different types of explicit feedback and recommender algorithms to be integrated into the model, as opposed to other methods in the literature that are hooked to particular types of feedback and recommender algorithms. As a result, ECoRec is able to significantly reduce well-known problems such as sparsity and cold-start, allowing one to build better training models than using standalone recommenders.

Unlike our previous work (Da Costa et al., 2018), in this paper we evaluate our approach with more than two different collaborative recommenders, and we compare different types of predictions generated by ECoRec to demonstrate the generality of the proposal. We have also run experiments with a larger collection of datasets. In fact, the experiments were run with eight real databases (in contrast to only three in the preliminary, conference paper) from different domains, namely, movies, books, jokes and songs. Our study shows that the proposed ensemble co-training approach is able to provide better accuracy than individual predictions by combining results from multiple co-trained recommenders.

This article is structured as follows: in Section 2 we overview research results related to multi-view learning and ensemble approaches for recommender systems. In Section 3 we review the recommender algorithms that

we use in our study and the co-training process in a broad sense. We then present our approach in Section 4 and discuss the experimental results in Section 5. In Section 6 we present our conclusions and future work.

2. Related Work

One of the most straightforward techniques to tackle the sparsity and cold-start problems in recommender systems is filling the missing ratings in the user-item matrix with default values (e.g. averages). This method, however, lacks personalization, is based on ad-hoc heuristics, and may include noisy data into the process, besides being time and memory consuming (Ricci et al., 2015; Bobadilla et al., 2013). For this reason, a variety of other, more sophisticated techniques have been developed.

In the following we discuss semi-supervised algorithms and ensemble methods applied to recommender systems, since these are the two topics mostly related to our proposal.

2.1. Semi-Supervised Learning in Recommender Systems

The recommender system domain is very challenging due to its idiosyncrasies, such as, for instance, dealing with large matrices that are typically up to 99% empty. In face of these challenges, only little work has been done on semi-supervised learning for recommender systems in order to enrich the original user-item matrix.

One approach involves active learning techniques, where an algorithm chooses which label (rating) to request from a user in order to maximize the performance gain from a given labeling budget (Karimi et al., 2011). For example, in Sun's work (Sun et al., 2013), a recommender approach was proposed that conducts an interview process as guided by a decision tree with multiple questions at each split, to learn users' behavior in order to improve the power of prediction of the recommender algorithm. Active learning techniques, however, are based on the assumption that a user is able and willing to provide any requested labels, which is often not the case in real applications of recommender systems.

Another alternative is co-training. The enrichment of user-item matrix using co-training technique is proposed by some works fully be integrated ratings of collaborative filtering, user profiles, item profiles, relationships between user profiles and item profiles of content-based filtering. Zhang et al. (2014) proposed a collaborative filtering co-training method for stationary,

batch-based recommender systems that makes use of items' metadata in order to enrich the traditional training model. Zhang and Wang (2015) proposed a multi-view learning framework with the ability of knowledge transfer for recommendation. Their framework was developed for item recommendation and can learn multi-view representations automatically from data. However, both techniques achieve improvements by using extra content/information. In fact, two or more different and independent types of metadata are required to build different training sets — e.g., items' genres and ratings.

In more recent works, Quang et al. (2015) proposed a collaborative filtering method based on co-training that iteratively expands the training set by switching between two different feature sets. In the collaborative filtering settings, their co-training based method used users and items as two different feature sets. Their approach leads to improved prediction accuracy than isolated recommenders and reduces the sparsity problem. Matuszyk and Spiliopoulou (2017) proposed a semi-supervised framework for stream-based recommendations based on co-training, which uses abundant unlabelled information to improve the quality of recommendations and one recommender algorithm based on matrix factorization. In their approach a single learner provides reliable predictions to itself and uses them as labels.

Despite the presented collaborative filtering methods based on memory (Karimi et al., 2011; Zhang et al., 2014; Matuszyk and Spiliopoulou, 2017) or based on model (Zhang and Wang, 2015; Quang et al., 2015) to enrich the original user-item matrix using extra information, in real scenarios, a variety of such information may not be available for recommendation or they require accurate input from a domain expert. Moreover, the computational cost to extract, treat and process the extra information within the recommender algorithms is high, which can make these algorithms extremely slow.

Our proposed approach lifts the aforementioned limitations by using only a training set composed by only one type of information and a number of recommenders for multi-view learning; thus we use different recommenders as multiple viewpoints, rather than varied input data that may not be available. Moreover, ECoRec allows different recommender algorithms to be used during the co-training process, as opposed to other approaches, which are dependent on both recommendation algorithms and external metadata.

2.2. Ensembles in Recommender Systems

Ensemble is a machine learning approach that uses a combination of various models in order to improve on the results obtained by each model indi-

vidually. Several recent studies, e.g. (Jahrer et al., 2010), have demonstrated the effectiveness of an ensemble of simpler techniques, showing that they can outperform more complex algorithms operating standalone.

Su et al. (2008) alleviated the data sparsity problem in their work by inputting the matrix with artificial ratings, prior to using the collaborative filtering algorithm. The authors used ten different machine learning models to be evaluated for the data imputing task including an ensemble classifier. Conceptually, their proposed ensemble is a sort of hybridization approach. In (Lee and Olafsson, 2009), the recommendations of several k-NN models were combined into an ensemble. The suggested model was a fusion between the User-KNN and Item-KNN algorithms. In addition, the authors suggested the lazy bagging learning approach for computing the user-user or item-item similarities. Their experiments improved the accuracy of predictions in the neighborhood models. Schclar et al. (2009) proposed a modified version of the AdaBoost ensemble regressor in order to improve the RMSE measure of a hybrid approach (matrix factorization and a neighborhood techniques). The authors demonstrated that adding more recommender algorithms to the ensemble reduces the RMSE.

Lastly, (Bar et al., 2013) proposed a systematic framework for applying ensembles to recommender systems. They employ automatic methods for generating an ensemble of collaborative filtering models based on a single CF algorithm (homogeneous ensemble). The authors experimentally showed the usefulness of this framework by applying several ensemble methods to various base CF algorithms. In (Ristoski et al., 2014), the authors discussed the development of a hybrid multi-strategy book recommendation system using Linked Open Data. Their approach builds on training individual base recommenders and using global popularity scores as generic recommenders. The results of the individual recommenders are combined using an ensemble method and rank aggregation techniques. The authors showed that their approach delivers very good results in different recommendation settings and also allows incorporating diversity of recommendations. However, their work requires several types of user interactions, which is not commonly available in real databases.

Our proposal is based on a heterogeneous (rather than homogeneous) ensemble, as it combines multiple predictions generated by different recommenders. It also differs from the above related work on ensembles in that we perform exchange of information between recommenders in a co-training fashion, in order to generate more accurate predictions. Our contribution, thus,

can be considered a pre-processing approach to generate augmented/enriched user-item matrices based on multiple view types and a confidence measure, but it also uses an ensemble technique to combine and enhance the results of individual views in a more robust user-item matrix.

3. Background

In this section, we review concepts related to our approach. Specifically, prior to describing our proposal, in the following we revise the recommender algorithms, confidence measures and the co-training method that will be used later on. It is worth remarking that there are different variants in the literature of some of the concepts reviewed in this section, particularly the recommender algorithms, so we will restrict ourselves to the specific implementations used in our experiments.

3.1. Notation

We use special indexing letters to distinguish users and items: a user is indicated as u, a known item is referred to as i; and r_{ui} is used to refer to a rating from a user u to an item i. The final prediction and the confidence estimate of the system about the preference of user u to item i are represented by $\hat{r}_{ui}$ and $\mathcal{C}_{ui}$, respectively, which are a floating point values produced by the recommender algorithm. The set of pairs (u, i), for which r_{ui} is known, is represented by the set $D = \{(u, i)|r_{ui} \text{ is known}\}$.

3.2. User k-Nearest Neighbors (User-KNN)

The first recommendation algorithm is the well-known User-KNN (Koren, 2008; Ricci et al., 2015), which produces recommendations based on similar users. The main goal of the algorithm is to find similar users and predict a rating based on their rating assignments.

In detail, a rating $\hat{r}_{ui}$ is predicted for an unknown user-item pair, (u, i), considering the ratings that other users with similar preferences to u have assigned to item i. To find similar users, a measure of proximity p_{uv} is employed between the feature vectors describing the users (i.e., the respective rows of the user-item matrix). The proximity measure can be based, e.g., on the Pearson correlation coefficient, or the cosine similarity, among others. The final similarity measure is a retracted coefficient, defined as:

$$s_{uv} = \frac{n_{uv}}{n_{uv} + \lambda_1} p_{uv} \tag{1}$$

where n_{uv} is the number of items that users u and v have in common, and λ_1 is a regularization constant, set to 100 according to the literature (Koren, 2008). The idea is to assign higher (lower) weights to proximity values associated with pairs of users sharing larger (smaller) numbers of rated items.

In order to predict a rating $\hat{r}_{ui}$ for an unknown user-item pair, (u, i), the algorithm identifies the k users that are the most similar to u using the similarity measure between users described above. The algorithm then selects the subset of those users that have evaluated item i. This subset is denoted as $S^k(i; u)$. The final rating is then predicted using Equation (2):

$$\hat{r}_{ui} = b_{ui} + \frac{\sum_{v \in S^k(i;u)} s_{uv}(r_{vi} - b_{vi})}{\sum_{v \in S^k(i;u)} s_{uv}} \tag{2}$$

where b_{ui} and b_{vi} are baseline estimates based on the ratings provided by users, proposed by Koren (2008) for neighborhood models, and defined as:

$$b_{ui} = \mu + b_u + b_i \tag{3}$$

where μ is the global average rating and b_u and b_i indicate the observed deviations of user u and item i, respectively, from the average, which are based on their interactions. The main purpose of b_{ui} is to capture systematic tendencies of certain users to give higher/lower ratings than others, and certain items to receive higher/lower ratings than others. For example, suppose that we need to calculate a baseline predictor for the rating of the song Yellow Submarine by user Paul, in a music database where the overall average (μ) of all ratings is 3.5 stars. However, Yellow Submarine is deemed better than an average song, so it tends to be rated 1 star above the average ($b_i = +1$), whereas Paul is a critical user, who tends to rate 0.5 stars lower than the average ($b_u = -0.5$). Hence, the baseline predictor (b_{ui}) for Yellow Submarine's rating by Paul would be 4 stars ($3.5 - 0.5 + 1$). The estimates b_u and b_i can be calculated by iterating n times the following equations:

$$b_i = \frac{\sum_{u:(u,i) \in D}(r_{ui} + \mu - b_u)}{\lambda_2 + |\{u : (u, i) \in D\}|} \tag{4}$$

$$b_u = \frac{\sum_{u:(u,i) \in D}(r_{ui} + \mu - b_i)}{\lambda_3 + |\{u : (u, i) \in D\}|} \tag{5}$$

where λ_2 and λ_3 are constants set as 10 and 15, respectively, according to Koren (2008).

3.3. Item k-Nearest Neighbors (Item-KNN)

The second recommender algorithm used in our experiments is Item-KNN (Ricci et al., 2015; Koren, 2008; Aggarwal, 2016), which also uses the concept of nearest neighbors. The main difference between User-KNN and Item-KNN is that the latter predicts the unknown rating r_{ui} of an item i by a user u based on u's ratings of the k items most similar to i. In order to find similar items, a similarity measure is employed between items, as shown in Equation (6):

$$s_{ij} = \frac{n_{ij}}{n_{ij} + \lambda_1} p_{ij} \tag{6}$$

where p_{ij} is a measure of proximity (e.g. Pearson or cosine) employed between the feature vectors describing the items (i.e., the respective columns of the user-item matrix) and n_{ij} is the number of shared features that describe items i and j (i.e., number of users that rated both items). The value of λ_1 is the same used in Equation (1).

Using the similarity measure in Equation (6), we identify the set of k items that are most similar to i, the so-called k-nearest neighbors, and select the subset of those items that have been rated by u. Using this subset, denoted as $Si^k(i;u)$, the final predicted rating is an average of such similar items' ratings, adjusted to their baseline estimate:

$$\hat{r}_{ui} = b_{ui} + \frac{\sum_{j \in Si^k(i;u)} s_{ij}(r_{uj} - b_{uj})}{\sum_{j \in Si^k(i;u)} s_{ij}} \tag{7}$$

where b_{ui} and b_{uj} are computed by Equation (3).

3.4. Singular Value Decomposition (SVD)

A matrix factorization technique, called Singular Value Decomposition for recommender systems (SVD) (Koren et al., 2009), stands out from other recommender systems for allowing the discovery of underlying latent characteristics to the interactions between users and items. In order to apply matrix factorization, initially we have a set U of users and a set I of items, as well as a sparse matrix $\mathbf{R}$ ($|U| \times |I|$) that corresponds to the ratings given by the users to the items. To discover the latent features of dimensionality F,

we have to find two matrices $\mathbf{P}$ and $\mathbf{Q}$ of dimensions $|U| \times |F|$ and $|I| \times |F|$, respectively, such that their product approximates $\mathbf{R}$:

$$\mathbf{R} \approx \mathbf{P} \times \mathbf{Q}^T = \hat{\mathbf{R}} \tag{8}$$

Each row of $\mathbf{P}$ carries the strength of the associations between a user and the resources, whereas each row of $\mathbf{Q}$ carries the strength of the associations between an item and resources (Koren et al., 2009). For predicting a rating of an item i by u, we can calculate the dot product of two vectors corresponding to u and i:

$$\hat{r}_{ui} = b_{ui} + p_u^T q_i = b_{ui} + \sum_{f=1}^{F} p_{uf} q_{if}. \tag{9}$$

where the baseline b_{ui} is defined as $b_{ui} = \mu + b_u + b_i$ and accounts for differences of users and items in comparison to the overall rating average μ. All parameters are estimated by minimizing the associated squared error function:

$$\min_{p_*, q_*, b_*} \sum_{(u,i) \in K} (r_{ui} - \mu - b_u - b_i - p_u^T q_i)^2 + \lambda(b_u^2 + b_i^2 + ||p_u||^2 + ||q_i||^2) \quad . \tag{10}$$

After estimating $\mathbf{P}$ and $\mathbf{Q}$ by minimizing the regularized squared error, one can predict the unknown ratings by taking the dot product of the latent features for users and items. To minimize the loss function, it is possible to use stochastic gradient descent (SGD) or alternating least square (ALS), which can also be used for on-line learning, i.e., updating the model incrementally each time a new rating comes in (Koren et al., 2009; Bobadilla et al., 2013).

3.5. Confidence Measures

Collaborative filtering approaches should provide confidence values associated with accurate values of prediction, in order to distinguish individual predictions that can be trusted from those that cannot. Research about confidence measures in RS requires the existence of simple, plausible and universally reliable quality measures, usually focused on accuracy (Bobadilla et al., 2018). Novelty, serendipity and diversity have been studied; nevertheless, there is a noticeable lack of systematic and experimental analysis in confidence quality measures (Mazurowski, 2013; Hernando et al., 2013).

Along with the well-known and traditional collaborative filtering algorithms (e.g. neighborhood and matrix factorization models), the literature reports some confidence measures, such as:

- **Support for user (SU)**: the confidence of an individual prediction is simply the number of ratings that the user assigned previously (i.e., if a user assigned 30 ratings in the database, the confidence estimate $\mathcal{C}_{ui}$ will be 30, for any rated item i) (Mazurowski, 2013).

- **Support for item (SI)**: the confidence $\mathcal{C}_{ui}$ of prediction $\hat{r}_{ui}$ is simply the number of ratings that were assigned to the item i within the available database (Mazurowski, 2013).

- **Variability for item (VI)**: the confidence $\mathcal{C}_{ui}$ of an individual prediction is the standard deviation of ratings that were assigned to item i for all users in the dataset (Adomavicius et al., 2007).

- **Resample**: repeats the entire prediction process of the CF algorithm using different subsets of the entire set of ratings available for system development (Mazurowski, 2013). The variability is measured by the standard deviation of the predictions. Since a higher standard deviation is expected to correspond to lower confidence, the inverse of the standard deviation can be taken as the confidence prediction $\mathcal{C}_{ui}$. This measure also contains two extensions: *Fast*, which reduces the computational time and uses only the prediction step to calculate confidence; and *Noise*, that instead of re-sampling the subset, it replaces the ratings with a value randomly sampled from a Gaussian distribution.

The first three measures have low computational cost and complexity since they involve simple calculations (e.g standard deviation and mean), however they only consider information about items or users in a separated way. On the other hand, Resample measures are more robust and accurate in estimating confidence of individual predictions, but they have high computational cost, since they require the execution of the recommender algorithm many times as well as the re-calculation of the similarity matrix.

3.6. Co-Training

Co-training is concerned with the problem of learning from data represented by multiple distinct views (Blum and Mitchell, 1998). The emergence

of this learning scheme is largely motivated by the fact that data in a variety of real applications are (or can be) described by different feature sets or different views of the same feature set (Sun et al., 2013; Xu et al., 2013). Generally, co-training is used when there are only small amounts of labeled data yet large amounts of unlabeled data described by two or more views that are ideally conditionally independent and sufficient to represent the target concept (Blum and Mitchell, 1998). Co-training first learns a separate prediction model (e.g. a classifier) for each view using any labeled examples. The most confident predictions of each model on the unlabeled data are then used to iteratively obtain additional (augmented) labeled training sets for further refinement of the other model(s).

This method assumes that each example is described using at least two different views that provide complementary information. Typically, different views can be obtained from disjoint feature subsets of different natures. For example, in multimedia content understanding, multimedia segments can be simultaneously described by their video and audio signals. In webpage classification, a webpage can be described by the document text itself and also by the anchor text attached to hyperlinks pointing to this page (Blum and Mitchell, 1998).

However, co-training can also be performed when there is only one representation of the data, if such a single representation is processed by independent prediction models, such as two different classifiers (Xu et al., 2013). In this case, the method exploits two views of the same, single data representation by co-training two classifiers using the available training examples. Iteratively, each classifier labels the unlabeled examples and the most confident predictions are provided as additional training examples to improve the accuracy of the other classifier.

In this paper we extend this idea to the ratings prediction problem in recommender systems in order to tackle sparsity and cold start issues by simultaneously considering multiple views of the same data. Our method couples the training of different, independent recommender systems in order to mutually and iteratively improve their accuracy by making each recommender provide the other(s) with its most confident predictions, in a co-training fashion, which is further enhanced by an ensemble of the resulting recommenders. The method is discussed in more detail in the next section.

4. Proposed Approach

We propose an Ensemble Co-training Recommender system (ECoRec) that uses two or more recommender algorithms as viewpoints, such as those presented in Section 3, in order to provide more accurate predictions and to significantly reduce the sparsity problem by enriching the original user-item ratings matrix. ECoRec uses only a single training set as input data and two or more distinct recommender algorithms as different views, and combine the output of each recommender into an ensemble.

In the co-training stage, each recommender is trained aiming to agree with other recommenders' predictions, such that they learn not only from the original training set of labeled data, but also from each other's most confident predictions of the unlabeled data. In a rating prediction scenario, a "label" means a rating r_{ui} of an item i by a user u, and "labeling" unlabeled data means assigning predicted ratings $\hat{r}_{ui}$ for unseen user-item pairs (u, i).

At each iteration of the co-training process, the recommender representing one view labels the unlabeled data, the most confident predicted ratings are then added to the training pool of the other recommender(s), and vice-versa. Thus, the information underlying two or more different views can be exchanged. Specifically, each recommender predicts ratings for all unobserved user-item pairs (unlabeled set), and the most confident predictions are used to populate the training set (i.e., to augment/enrich the labeled set) of the other recommender(s), in an iterative fashion. In addition, at each iteration the predictions of each recommender are combined into an ensemble, weighted by their confidence. The overall process is shown in Figure 7.1, where it can be seen that, in contrast to single recommenders, our approach simultaneously learns two or more models iteratively, in order to exploit different viewpoints of the same input data and boost the learning performance.

There are four phases in our technique, namely, data pre-processing, co-training, ensemble, and recommendation. In the following subsections, we describe these procedures in more detail.

4.1. Data Pre-Processing

In this stage, ECoRec splits the original data set into a labeled set (T_L^η), which contains all triples (u, i, r_{ui}) of the user-item matrix, and an unlabeled set (T_{UL}^η), which contains randomly selected unobserved pairs (u, i) per user.

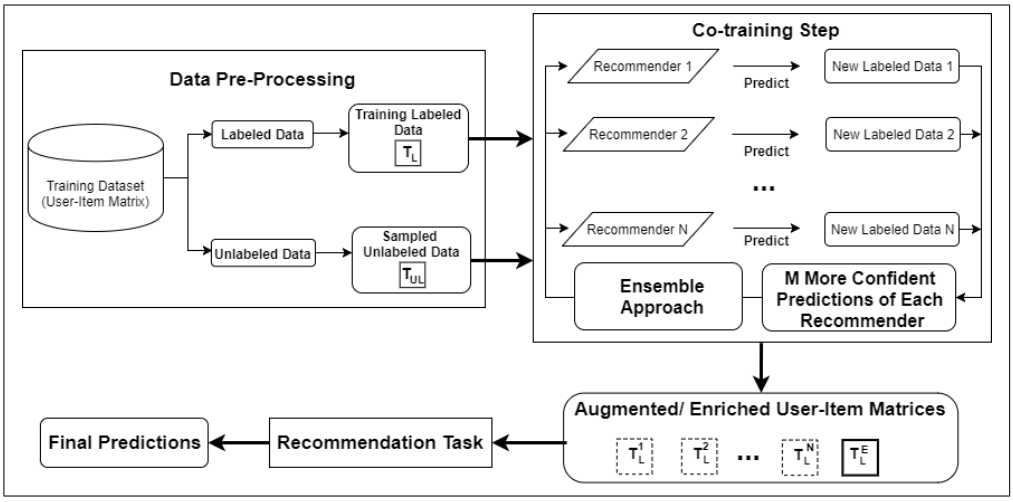


Figure 7. 1. Schematic view of the proposed system.

These are the sets used as input to co-training the pool of recommender algorithms ($\eta = 1, 2, \ldots, N$). The initial labeled and unlabeled sets are common to all recommenders, i.e., $T_L^1 = T_L^2 = \ldots = T_L^N$ and $T_{UL}^1 = T_{UL}^2 = \ldots = T_{UL}^N$. The initial labeled set, T_L^η, is constituted by all observed user-item-ratings triples, i.e., all entries of the user-item matrix for which ratings are available (non-empty entries). Since the user-item matrix is sparse, the amount of unobserved (empty) entries can be enormous. Instead of including all the unrated user-item pairs into the initial unlabeled set T_{UL}^η that is provided to the recommenders, which would impose an unnecessary additional computational burden and slow down convergence of the co-training process, we instead randomly select X unlabeled pairs (u, i) per user. For example, if $X = 40$, for each user in the original user-item matrix, 40 items unknown to that user will be randomly chosen to be included into the unlabeled set, T_{UL}^η. We used random subsampling to incorporate novelty and aggregated diversity into the original datasets and thus soften the problems of new users and items (Castells et al., 2015). Since all pairs that are randomly chosen are unlabeled, all the X new items chosen to be labeled for each user will be novel to that user. As these items are selected randomly, we have a chance to pick up both good and bad items for users based on their behavior. The task of the recommenders during co-training is to jointly learn how to predict the ratings that users would give to the pairs in this unlabeled set.

4.2. Co-training Step

Having the recommenders as well as the labeled and unlabeled sets in hand, the next step is to apply the co-training on these recommender algorithms. Algorithm 7.1 shows how the method of co-training works in ECoRec.

Input: User-item matrix, positive integers X and M
Task 1: Data Pre-Processing
$T_L \leftarrow$ all labeled triples (u, i, r_{ui}) of the user-item matrix
$T_{UL} \leftarrow X$ randomly selected unlabeled pairs (u, i) per user
for η=*1 to* N **do**
    T_L^{η}, $T_{UL}^{\eta} \leftarrow T_L$, T_{UL}
end
$T_L^E \leftarrow T_L$
Task 2 : Co-training and Ensemble Technique
while $T_{UL}^1 \neq \emptyset$, $T_{UL}^2 \neq \emptyset$, ..., $T_{UL}^N \neq \emptyset$ **do**
    for η=*1 to* N **do**
        foreach *user-item pair* $(u, i) \in T_{UL}^{\eta}$ **do**
            $\hat{r}_{ui}^{\eta} \leftarrow$ prediction made by recommender η
            $\mathcal{C}_{ui}^{\eta} \leftarrow$ confidence of prediction $\hat{r}_{ui}^{\eta}$ by Eq. (11)
        end
        $R_{\eta} \leftarrow M$ most confident triples $(u, i, \hat{r}_{ui}^{\eta})$ according to $\mathcal{C}_{ui}^{\eta}$
        $T_{\eta} \leftarrow M$ pairs (u, i) corresponding to the triples in R_{η}
    end
    $T_F \leftarrow T_1 \cup T_2 \cup ... \cup T_N$
    for η=*1 to* N **do**
        $T_{UL}^{\eta} \leftarrow T_{UL}^{\eta} - T_F$
    end
    for η=*1 to* N **do**
        $L_{\eta} \leftarrow \emptyset$
        for ω=*1 to* N **do**
            if η != ω **then**
                foreach *user-item-rating triple* $(u, i, \hat{r}_{ui}^{\omega}) \in R_{\omega}$ **do**
                    $L_{\eta} \leftarrow L_{\eta} \cup (u, i, \hat{r}_{ui}^{\omega})$
                end
            end
        end
        Combine triples $(u, i, \hat{r}_{ui}^{(\cdot)}) \in L_{\eta}$ sharing a common (u, i) into a single entry, by Eq. (13)
        $T_L^{\eta} \leftarrow T_L^{\eta} \cup L_{\eta}$
    end
    $R_F \leftarrow R_1 \cup R_2 \cup ... \cup R_N$
    Combine triples $(u, i, \hat{r}_{ui}^{(\cdot)}) \in R_F$ sharing a common (u, i) into a single entry, by Eq. (13)
    $T_L^E \leftarrow T_L^E \cup R_F$
end
Task 3: Recommendation Task
for η=*1 to* N **do**
    Train the Recommender η with T_L^E to predict new ratings for unknown items for each user
end
Output: Sets of unknown pairs (u, i) with their respective predicted ratings $(\hat{r}_{ui})$.

Algorithm 7. 1: ECoRec Algorithm

Our co-training process consists of evaluating the same training set in

different views, driving them to agree and complement the predictions made by each other. To this end, different rating prediction algorithms can be used, provided that they represent different views of the data. For example, we can use Item-KNN and User-KNN as views, where the first favors similarity between users, whereas the second favors similarity between items.

For every iteration of the ECoRec algorithm, each recommender η ($\eta = 1, 2, ..., N$) is responsible for predicting ratings for its unlabeled set, T_{UL}^{η}, based on its labeled set, T_L^{η} (lines 11 to 14). Then, ECoRec calculates the confidence of each predicted rating, choosing the M most confident ones (lines 15-16) and removing them from the set of unlabeled data for all recommenders (lines 18 to 21). Notice that the (positive integer) parameter M controls the learning rate of the co-training process. As usual in statistical and machine learning algorithms, slow learning (corresponding to small values of M) tends to produce more accurate models, at a higher computational cost though (larger number of iterations required for convergence).

Once this procedure is completed for all the recommenders, the algorithm updates the labeled set of each recommender by adding its M most confident examples to the other recommenders' labeled sets (lines 22 to 33), since we expect each model to learn only from the newly labeled examples provided by the other views, besides the original labeled examples. If an unlabeled pair (u, i) belongs to the confident sets (T_η) of two or more views, then the weighted average of the corresponding predicted ratings ($\hat{r}_{ui}^{\eta}$) is taken as the final rating $\hat{r}_{ui}^{\eta}$, and the triple $(u, i, \hat{r}_{ui}^{\eta})$ is added to the designated labeled sets. The average is weighted by the confidence of the predicted ratings in question. The intuition behind our approach is that one recommender adds confident new examples to the labeled sets that the other recommender(s) will then use for learning, but from another perspective (view). In our approach the predictions made by each recommender never feed their own labeled set, as usual in co-training. For example, using two recommenders in ECoRec, the predictions made by recommender 1 will only feed the labeled set T_2^L, but never its own (T_1^L) (line 25). The algorithm only stops when all unlabeled data of all views have been labeled, which is guaranteed by construction.

The outputs of the co-training process are new augmented/enriched matrices for each recommender used (T_L^1, T_L^2, ..., T_L^N), besides a T_L^E set that is generated by an ensemble using the confident sets of all individual recommenders (T_F) (lines 34 to 36), as detailed in Section 4.3.

From the recommender systems perspective, the recommender algorithms tend to improve their prediction power as the data increase (Aggarwal, 2016;

Mazurowski, 2013; Hernando et al., 2013). In fact, how well the recommenders work depends directly on the number of labeled examples and their associated confidence measure. Both factors contribute to have an aggregated impact on the final results and convergence during the co-training approach. The idea behind the confidence measure is that examples labeled with high confidence are expected to decrease the most the error of the recommender on the labeled example set, if they are included. Our proposed confidence criterion is discussed next.

Confidence Criterion: One challenge in our method, which is inherent to co-training approaches, is how to select a subset of predicted examples from each recommender's unlabeled set (line 15) so as to enhance the labeled set of the other recommenders. The selection criterion we used in this work is the recommender algorithm's *confidence*. Every candidate prediction must have a confidence value associated with it.

Confidence in recommender systems can be defined as the system's trustworthiness of its recommendations or predictions (Ricci et al., 2015). When a confidence measure is available, the user can benefit from observing the confidence scores along with the recommendations (Zhou and Li, 2005; Guo, 2013). For instance, when the system reports a low confidence in a recommended item, the user may tend to further inspect the item before making a decision. However, not every recommender algorithm generates a confidence score, or there may be a high cost associated to computing one.

The confidence criterion we propose and use in this work is based on the fact that collaborative filtering recommenders tend to improve their accuracy as the amount of data over items/users grows. Based on our previous work (Da Costa et al., 2018), our confidence measure for a prediction $\hat{r}_{ui}^{\eta}$ made by recommender algorithm η for the rating of pair (u, i) is as follows:

$$\mathcal{C}_{ui}^{\eta} = \sigma_{ui}^{\eta} N_u N_i \tag{11}$$

where N_u and N_i are the popularity of user u and item i, respectively, given by the number of ratings applied/received by user u and item i in the labeled set associated with recommender η, T_L^{η}, and σ_{ui}^{η} represents a measure of trustworthiness of the prediction made by recommender η, defined as:

$$\sigma_{ui}^{\eta} = \frac{1}{|b_{ui} - \hat{r}_{ui}^{\eta}|} \tag{12}$$

where b_{ui} is a baseline estimate of the rating for the pair (u, i) in question, calculated based on the global mean and the mean of interactions of users and items in T_L^η, using Equation (3).The purpose of using σ_{ui}^η is to weight the users' and items' popularities by the inverse of the error between the baseline estimate and the rating predicted by the recommender, since a small error means that the predicted rating is close to the behavior of that user and item in the dataset. For example, a user has a $b_{ui} = 4.5$ and the recommender η has generated $\hat{r}_{ui}^\eta = 4$, so the difference between them will be 0.5. Since we want to weight popularities so that the smaller the difference the greater the weighting, we take the inverse of the difference, i.e., $\sigma_{ui}^\eta = 2$.

The main rationale behind our proposed confidence measure is that the more popular users and items are (i.e. the more ratings are associated with them), the more accurate the predictions calculated based on these users and items tend to be. In addition, the confidence on a prediction deviating largely from the baseline rating b_{ui}, which takes into account the overall ratings given/received by the user u and the item i in question, tends to be penalized. Unlike several approaches presented in Section 3.5, our confidence measure considers both user and item interactions, as well as the quality of the prediction generated by each recommender in each epoch of the co-training approach.

4.3. Ensemble

As a result of each iteration of the co-training process, we get multiple recommenders, which may provide different recommendations from their different views. In order to get a single, more robust reconciled prediction, we can ensemble the individual predictions of each recommender during the co-training process. A straightforward alternative is to take the average of the ratings over all recommenders, which results in a *bagging-like* strategy that is known to be able to simultaneously reduce bias and variance of prediction models. However, this method does not consider the confidence associated with the predictions from different recommenders. To take confidence into account, we ensemble the results by a weighted average of the predictions generated by individual recommenders at each iteration of the co-training process. The confidence measure in Equation (11) is used for weighting:

$$\hat{r}_{ui} = \frac{\sum_{\eta \in S} \mathcal{C}_{ui}^\eta \hat{r}_{ui}^\eta}{\sum_{\eta \in S} \mathcal{C}_{ui}^\eta} \tag{13}$$

In Equation (13), $\hat{r}^{\eta}_{u,i}$ and $\mathcal{C}^{\eta}_{ui}$ stand for the predicted rating and its confidence, respectively, provided by recommender η for user-item (u, i). Set $S \subseteq \{1, \cdots, N\}$ is the subset of recommenders for which the predicted rating $\hat{r}^{\eta}_{u,i}$ should be averaged. In the case of the ensemble performed at the end of each iteration of the ECoRec algorithm (line 35 of Algorithm 7.1), S is the subset of recommenders η for which $\hat{r}^{\eta}_{u,i}$ is one of the M most confident predictions of that recommender at that iteration, i.e., $(u, i, \hat{r}^{\eta}_{u,i}) \in R_{\eta}$. Formally, $S = \{\eta : (u, i, \hat{r}^{\eta}_{u,i}) \in R_{\eta}\}$. This way, multiple confident predictions for the same entry (u, i) can be combined into a single prediction, $\hat{r}_{ui}$.

From Equation (13) we obtain, at each iteration of the co-training process, a single rating prediction that takes into consideration the confidence of the individual constituent predictions. For example, let us consider 3 recommenders in ECoRec ($N = 3$). Let's suppose that, at a given iteration of the algorithm, Recommenders 1 and 3 both have entry (user k, item j) within their most confident predictions ($r^1_{kj} = 5$ with $\mathcal{C}^1_{kj} = 0.5$ and $r^3_{kj} = 3.5$ with $\mathcal{C}^3_{kj} = 0.8$). Assuming that entry (user k, item j) is not within the most confident set of Recommender 2, the computation of the final prediction for this entry according to the ensemble is $\hat{r}_{kj} = (0.5 \cdot 5 + 0.8 \cdot 3.5)/(0.5 + 0.8) = 4.07$.

Notice that we perform the ensemble within the co-training loop (lines 34 to 36 of Algorithm 7.1), because we need the confidence values generated at each iteration of the co-training process to capture the confidence information as it evolves throughout iterations.

4.4. Recommendation Task

Algorithm 7.1 outputs a number of augmented/enriched user-item matrices, T^1_L, T^2_L, ..., T^N_L, besides T^E_L generated by the ensemble. These matrices can now be used by different recommenders to calculate predictions for new ratings. In our experiments, we applied only the user-item matrix generated by our ensemble approach (T^E_L) in all individual recommender algorithms to predict new ratings, such that we obtain recommendations from multiple recommender algorithms but using only a single augmented/enriched user-item matrix. Optionally, the resulting ratings from the different recommenders can also be combined into a single rating using Equation (13), as long as the confidence values associated with each pair (u, i) are stored during the co-training process.

5. Evaluation

To evaluate our proposal, we compare ECoRec against each individual collaborative recommender presented in Section 3. In our experiments, we applied our algorithm in two different settings: using two and three recommenders as different views. In the first case, we used Item-KNN and User-KNN as recommenders (with Pearson Correlation similarity). These algorithms were chosen because they make recommendations based on different views of the data: User-KNN uses the relations between users, while Item-KNN uses the similarity between the items to make the prediction. In the second case, we used the SVD recommender (Koren, 2008) in addition to the two previous neighborhood models. The latter generates predictions using a matrix factorization approach, which differs largely from the neighborhood models in that it uses latent factors to increase its predictive power, thus representing an alternative, complementary view of the data. These algorithms are very effective and widely used both in the literature as well as in practical applications of recommender systems (Ricci et al., 2015; Bobadilla et al., 2013).

We also compare ECoRec against the semi-supervised learning method CSEL (Zhang et al., 2014), which is a co-training framework for processing different types of information, including item's metadata. We adapted their algorithm to accept ratings only, so a fair comparison between the results could be achieved. Similar to ECoRec, this algorithm was proposed to incorporate unlabeled examples into the original user-item matrix using a co-training process. Its output also generates N enriched matrices, where N is the number of recommenders used in the process. We report the results corresponding to the best enriched matrix generated by CSEL. In addition, we compare our approach with one additional co-training approach, called Co-CF (Quang et al., 2015), and also with AdaBoost.RT (AB.RT), a traditional ensemble approach presented in the work of Bar et al. (2013).

5.1. Databases

We evaluate our approach on eight publicly available databases from different domains, which contain explicit user's interactions.

- **Yahoo Movies:** this database contains a small sample of the Yahoo! Movies community's preferences for various movies, rated on a scale from A+ to F. Developed by the Yahoo! Research Alliance Webscope

program[1], this database consists of 169,767 ratings given by 4,385 users to 4,339 movies. For our experiments we normalized the original ratings in the range from 1 to 5.

- **FilmTrust:** consists of 35,497 ratings given by 1,508 users to 2,071 movies developed by Guo et al. (2013). The ratings used as explicit feedback in our study are in the range between 0.5 and 5.

- **CiaoDVD:** is a DVD movie database proposed by Guo et al. (2014), and consists of 280,391 ratings given by 7,375 users to 99,746 items in a range from 1 to 5.

- **MovieLens** $2k$**:** consists of 800,000 ratings (from 0.5 to 5) and 10,000 interaction tags applied to 2,113 users and 10,197 movies. As explicit information, we used the ratings users assigned to items. This database was developed by Cantador et al. (2011) and has not been modified for the experiments.

- **Jester:** consists of 1,810,455 ratings applied by 23,500 users to 100 jokes. As explicit information, we used the normalized ratings users assigned to items. For our experiments we used only a subsample of the original database proposed by Goldberg et al. (2001), resulting in 360,917 interactions made by 5,000 users to 100 items. Original ratings (-10 to 10) were rescaled into the range from 0.5 to 5.

- **Booking Crossing:** this database was developed in the work of Ziegler et al. (2005) and contains 278,858 users (anonymized, but with demographic information) providing 1,149,780 ratings (explicit / implicit) about 271,379 books. For our experiments, we used only users' ratings, filtering out all items with less than 5 interactions in the database, resulting in 102,963 interactions made by 6,628 users to 7,164 books. We also rescaled the original ratings (0 to 10) into a new range (0.5 to 5).

- **Amazon Digital Music:** consists of 206,282 ratings applied by 16,396 users to 101,708 music albums. For our experiments we used only a sample of the original data set, proposed by McAuley et al. (2015), containing 59,793 interactions made by 6,337 users to 4,744 items.

[1]https://webscope.sandbox.yahoo.com/

- **Yahoo Music:** this database was also developed by Yahoo! Research Alliance Webscope program[1] and contains ratings for songs collected from two different sources. In this work we use a source that consists of ratings supplied by users during normal interaction with Yahoo! Music services. The rating data includes 196,150 interactions made by 15,400 users to 1,000 songs.

These databases have been made available in Github[1] with their respective terms of use and licenses.

5.2. Experimental Setup and Evaluation Metric

Recall from Section 4 that ECoRec generates augmented labeled sets, T_L^1, T_L^2, ..., T_L^N, T_L^E, which are then used to populate the original user-item matrix (T_L) with predictions made during the co-training process. The enriched user-item matrices resulting from T_L^1, T_L^2, ..., T_L^N, T_L^E are used in order to evaluate the accuracy of the results. In particular, for the two different settings considered in our experiments (User-KNN/Item-KNN and User-KNN/Item-KNN/SVD), the algorithm then produces three and four new enriched data sets ($T_L^1/T_L^2/T_L^E$ and $T_L^1/T_L^2/T_L^3/T_L^E$), respectively. In order to evaluate the results, we apply all base recommenders to all resulting enriched sets. The same recommender algorithms used to generate enriched matrices in ECoRec (User-KNN, Item-KNN, SVD) are also used in our closest competitor, CSEL.

To measure the predictive ability of the different methods, we used root mean square error measure (RMSE) and F-measure (Aggarwal, 2016; Bobadilla et al., 2013; Ravi and Vairavasundaram, 2016), with 10-fold cross-validation. We computed the mean across the 10-folds and compared our method against the competitors using a two-sided paired t-test with a 99% confidence level (Box, 1987; Aggarwal, 2016). We compute the root mean square error (RMSE) of every predicted rating $\hat{r}_{ui}$ in relation to its real rating r_{ui} found in the test set:

$$\text{RMSE} = \frac{1}{|U|} \sum_{u \in U} \sqrt{\frac{1}{|O_u|} \sum_{i \in O_u} (\hat{r}_{ui} - r_{ui})^2}, \tag{14}$$

where O_u is the set of items evaluated by user u.

[1]https://github.com/ArthurFortes/Datasets-for-Recommneder-Systems

Due to the high sparsity of the datasets used, some recommender algorithms may not be capable of predicting the ratings (Ravi and Vairavasundaram, 2016), so the coverage measure is also adopted. If the recommender algorithm is not capable of predicting a rating for a (u, i) pair, then it is said that the algorithm did not cover the particular pair of user and item:

$$coverage = \frac{CS}{N_{ratings}} \quad , \tag{15}$$

where CS denotes the number of ratings predicted and $N_{ratings}$ denotes the number of ratings tested. RMSE and coverage can be combined into a single measure as F-measure (Ravi and Vairavasundaram, 2016). To calculate F-measure, it is necessary to determine precision values. Precision can be obtained from RMSE as:

$$precision = 1 - \frac{RMSE}{Maximum\ possible\ error} \quad , \tag{16}$$

where *Maximum possible error* is the difference between the highest and the lowest rating value in each database. In this way, F-measure is determined by using coverage and precision as follows:

$$F\text{-}measure = \frac{2 \cdot precision \cdot coverage}{precision + coverage} \quad , \tag{17}$$

which reaches its best value at 1 (perfect precision and coverage) and worst at 0.

Regarding the parameters of the algorithms, we determined a collection of values which performed well for all databases. In particular, we ran experiments for X (new ratings per user) equal to 10, 20, 30, 40, 50 and 100 ratings, and chose 20, 30 and 40 as they provided the best results. Besides, we found that the confident set size M, as expected, has an impact on the learning rate for ECoRec and CSEL. We experimented with several values of M from 1,000 up to 20,000 and found that 10% of the unlabeled set size ($|T_{UL}|$) favors both CSEL and ECoRec, leading to a good trade-off between convergence rate (computational cost) and accuracy across all datasets. For User-KNN and Item-KNN, we used Pearson Correlation to generated the similarity matrices, once the results indicate that this similarity is superior to other relevant measures that we tested, such as Cosine and Jaccard. For the SVD algorithm, we ran experiments for factors equal to 10, 30, 50, and

chose 10 as it provided the best results in all databases. For other parameters, we used the default values of each recommender tool and library chosen.

Our approach (ECoRec) was developed in Python version 3.6[1], and the source-code is freely available in Github[2]. The baseline competitors belong to the Case Recommender Framework version 0.0.20[3], an open source tool developed in *Python*. We used the default framework settings for recommender algorithms.

5.3. Results

In the following we discuss the results in different experimental setups.

5.3.1. Performance of the Confidence Measure

This section shows a comparative evaluation of the confidence measures presented in Section 3.5 against our proposed measure (PC) using ECoRec with User-KNN and Item-KNN algorithms. The Resample measure is not considered in this evaluation since it requires running each algorithm in each iteration of co-training many times to converge, which considerably increases the computational cost of our approach. In this experiment, we used the best enriched matrix and recommender for each database using 40 new ratings per user based on a preliminary analysis. The results are summarized in Table 7.1.

It is clear that the proposed confidence measure provides superior performance when compared against the support for user, support for item and variability for item measures. Our confidence measure emerges as the best for all databases with a considerable accuracy gain.

5.3.2. ECoRec with Two Recommenders

The overall performances of the algorithms on the eight databases are given in Table 7.2. By using the enhanced training sets generated by ECoRec (T_L^1, T_L^2 and T_L^E), the User-KNN and Item-KNN results were improved over the traditional training set (T_L) in terms of overall RMSE in all experiments.

We note that adding more users' ratings using the proposed ECoRec algorithm with neighborhood-based models yielded better results than the baselines using the original database. This is because the enriched data

[1]http://www.python.org/

[2]https://github.com/caserec/CaseRecommender

[3]https://pypi.python.org/pypi/CaseRecommender

Table 7. 1. Comparison between our proposed confidence measure against three traditional measures in terms of RMSE and F-measure. Bold typeset indicates the best performance.

Datasets	Measures/ Confidence	SI	SU	VI	PC
Yahoo Movies	RMSE	0.9766	0.9753	0.9747	**0.9489**
	F-Measure	0.8311	0.8151	0.8322	**0.8461**
FilmTrust	RMSE	0.8673	0.8657	0.8645	**0.8011**
	F-Measure	0.6111	0.6251	0.6372	**0.6578**
Ciao DVD	RMSE	1.1628	1.1697	1.1644	**1.0288**
	F-Measure	0.7799	0.7987	0.8096	**0.8210**
MovieLens 2k	RMSE	0.7911	0.79085	0.79097	**0.7401**
	F-Measure	0.7757	0.7736	0.7836	**0.8135**
Jester	RMSE	0.9955	0.9987	0.9986	**0.9568**
	F-Measure	0.5566	0.5551	0.5624	**0.5882**
Booking Crossing	RMSE	0.8526	0.8518	0.8526	**0.7693**
	F-Measure	0.8912	0.8843	0.8942	**0.9165**
Amazon Digital Music	RMSE	0.9376	0.9318	0.9345	**0.8798**
	F-Measure	0.8597	0.8606	0.8602	**0.8901**
Yahoo Music	RMSE	1.1574	1.1564	1.1559	**1.1229**
	F-Measure	0.4308	0.4311	0.4397	**0.5261**

generated by ECoRec are populated with examples confidently predicted by different recommender algorithms as different views of the users' preferences. In addition, we note that the enriched set generated by the ensemble (T_L^E) outperforms both baselines and their results from the individual enriched sets (T_L^1 and T_L^2) in almost all cases, showing the effectiveness of the ensemble process. The results obtained by Item-KNN using T_L^E were the best overall for all experiments.

Table 7.4 shows the results comparing the baseline with our approach, using the best enriched matrix and recommender for each database from Table 7.2. For the baselines, we used the best final result generated by each approach. In all eight databases, ECoRec achieved better results than the other baselines.

5.3.3. ECoRec with Three Recommenders

Table 7.5 shows the results of the generated enriched sets (T_L^1, T_L^2, T_L^3 and T_L^E) using both neighborhood models (User-KNN and Item-KNN) as well as SVD. The results show that even though individual recommenders may perform worse than others, they still contribute positively to the ensemble, which considerably outperforms the baselines.

The comparison between our approach and CSEL in this experiment is depicted in Table 7.7, once again using the best enriched matrix and recom-

Table 7. 2. Comparison between ECoRec with two recommenders and standalone KNN-based recommenders in terms of RMSE.

Recommenders	**User-KNN**				**Item-KNN**			
User-Item Matrix	T_L	T_L^1	T_L^2	T_L^E	T_L	T_L^1	T_L^2	T_L^E
Yahoo Movies								
20 new ratings	0.9834	0.9786	0.9821	0.9778	0.9637	0.9621	0.9581	0.9530
30 new ratings		0.9771	0.9789	0.9767		0.9520	0.9495	0.9498
40 new ratings		0.9737	0.9780	**0.9716**		0.9508	0.9499	**0.9489***
FilmTrust								
20 new ratings	0.8584	0.8369	0.8465	0.8410	0.8336	0.8297	0.8325	0.8274
30 new ratings		0.8278	0.8417	0.8207		0.8204	0.8198	0.8118
40 new ratings		0.8287	0.8384	**0.8198**		0.8160	0.8172	**0.8011***
CiaoDVD								
20 new ratings	1.0761	1.0815	1.0810	1.0769	1.0664	1.0634	1.0616	1.0585
30 new ratings		**1.0711**	1.0750	1.0716		1.0569	1.0611	1.0502
40 new ratings		1.0897	1.0886	1.0876		1.0472	1.0409	**1.0288***
MovieLens 2k								
20 new ratings	0.7908	0.7897	0.7901	0.7834	0.7460	0.7458	0.7456	0.7437
30 new ratings		0.7894	0.7906	0.7811		0.7452	0.7448	0.7413
40 new ratings		0.7814	0.7893	**0.7803**		0.7414	0.7429	**0.7401***
Jester								
20 new ratings	1.0208	1.0045	1.0098	1.0011	0.9679	0.9653	0.9644	0.9639
30 new ratings		0.9955	0.9985	0.9904		0.9627	0.9625	0.9615
40 new ratings		0.9761	0.9854	**0.9702**		0.9592	0.9579	**0.9568***
Booking Crossing								
20 new ratings	0.9935	1.0677	1.0824	1.0524	0.8659	0.9146	0.9151	0.9054
30 new ratings		0.9928	0.9916	0.9910		0.8637	0.8781	0.8386
40 new ratings		0.8078	0.8308	**0.8074**		0.7738	0.7820	**0.7693***
Amazon Digital Music								
20 new ratings	0.9715	1.0415	1.0505	1.0148	0.9057	0.9952	1.0074	0.9889
30 new ratings		0.9700	0.9684	0.9695		0.9001	0.9121	0.8989
40 new ratings		0.8973	0.9088	**0.8918**		0.8890	0.8905	**0.8798***
Yahoo Music								
20 new ratings	1.1476	1.1466	1.1463	1.1460	1.1451	1.1297	1.1280	1.1276
30 new ratings		1.1458	1.1453	**1.1441**		1.1318	1.1278	**1.1229***
40 new ratings		1.1539	1.1628	1.1572		1.1356	1.1327	1.1328

Bold typeset indicates the best performance in each recommender algorithm. * indicates statistical significance with p-value <0.01.

mender for each database from Table 7.5 in the case of ECoRec, and the best among all the results obtained from CSEL. Overall, ECoRec achieved better results than CSEL also in this scenario.

5.3.4. Cold-Start problem

In order to explore the effectiveness of our approach to mitigate the cold-start problem, we report the recommendation performance for a subset of

Table 7. 3. Comparison between ECoRec with two recommenders and standalone KNN-based recommenders in terms of F-Measure.

Recommenders	**User-KNN**				**Item-KNN**			
User-Item Matrix	T_L	T_L^1	T_L^2	T_L^E	T_L	T_L^1	T_L^2	T_L^E
Yahoo Movies								
20 new ratings	0.8356	0.8402	0.8410	0.8415	0.8284	0.8325	0.8323	0.8365
30 new ratings		0.8412	0.8420	0.8427		0.8325	0.8323	0.8367
40 new ratings		0.8417	0.8424	**0.8461***		0.8334	0.8334	**0.8374***
FilmTrust								
20 new ratings	0.6084	0.6307	0.6276	0.6429	0.6275	0.6312	0.6307	0.6432
30 new ratings		0.6364	0.6303	0.6487		0.6320	0.6297	0.6482
40 new ratings		0.6428	0.6323	**0.6578***		0.6345	0.6327	**0.6570***
CiaoDVD								
20 new ratings	0.7972	0.8042	0.8056	0.8092	0.7961	0.8022	0.8013	0.8058
30 new ratings		0.8067	0.8092	0.8102		0.8029	0.8025	0.8070
40 new ratings		0.8110	0.8129	**0.8210***		0.8033	0.8035	**0.8110***
MovieLens 2k								
20 new ratings	0.8040	0.8042	0.8048	0.8052	0.7988	0.8095	0.8102	0.8112
30 new ratings		0.8039	0.8049	0.8047		0.8102	0.8109	0.8115
40 new ratings		0.8045	0.8053	**0.8103***		0.8108	0.8115	**0.8135***
Jester								
20 new ratings	0.5269	0.5853	0.5861	**0.5882***	0.5666	0.5638	0.5673	0.5692
30 new ratings		0.5853	0.5845	0.5860		0.5670	0.5672	**0.5701**
40 new ratings		0.5831	0.5868	0.5877		0.5659	0.5662	0.5687
Booking Crossing								
20 new ratings	0.8770	0.8946	0.8952	0.8995	0.8725	0.8754	0.8754	0.8802
30 new ratings		0.8967	0.8973	0.9017		0.8781	0.8774	0.8844
40 new ratings		0.8973	0.8977	**0.9165***		0.8798	0.8796	**0.8954***
Amazon Digital Music								
20 new ratings	0.8434	0.8683	0.8684	0.8741	0.8177	0.8440	0.8438	0.8572
30 new ratings		0.8693	0.8702	0.8809		0.8458	0.8453	0.8589
40 new ratings		0.8705	0.8710	**0.8901***		0.8464	0.8461	**0.8610***
Yahoo Music								
20 new ratings	0.4571	0.4756	0.4780	**0.4866***	0.5028	0.5187	0.5200	**0.5261***
30 new ratings		0.4742	0.4773	0.4817		0.5153	0.5181	0.5209
40 new ratings		0.4718	0.4756	0.4805		0.5101	0.5129	0.5193

Bold typeset indicates the best performance in each recommender algorithm. * indicates statistical significance with p-value <0.01.

users according to their popularity. We estimate the popularity of each user based on the number of their ratings, and select a subset of users with less than 20 interactions in the training set. We then average the RMSE scores obtained for those users in our first experiment in Subsection 5.3.2. The results are displayed in Table 7.8 for the KNN-based recommenders on the augmented T_L^E dataset generated by ECoRec. We evaluate the gains when we gradually added 20, 30 and 40 new ratings per user.

Table 7. 4. Comparison between KNN-based ECoRec and baselines in terms of RMSE and F-Measure. Bold typeset indicates the best performance. * indicates statistical significance with p-value <0.01.

Database	Measure / Approach	ECoRec	CSEL	Co-CF	AB.RT
Yahoo Movies	*RMSE*	**0.9489**	0.9501	0.9394	0.9477
	F-Measure	**0.8461***	0.8114	0.8323	0.8371
FilmTrust	*RMSE*	**0.8011***	0.8432	0.8385	0.8184
	F-Measure	**0.6578***	0.5987	0.6356	0.6296
CiaoDVD	*RMSE*	**1.0288***	1.0487	1.0452	1.0357
	F-Measure	**0.8210***	0.8012	0.7798	0.8049
MovieLens 2k	*RMSE*	**0.7401***	0.7498	0.7492	0.7531
	F-Measure	**0.8103***	0.7921	0.7852	0.7772
Jester	*RMSE*	0.9568	0.9598	**0.9520**	0.9622
	F-Measure	**0.5882***	0.5791	0.5722	0.5741
Booking Crossing	*RMSE*	**0.7693***	0.7864	0.7739	0.7761
	F-Measure	**0.9165***	0.8972	0.8931	0.8987
Amazon Digital Music	*RMSE*	**0.8798**	0.8906	0.8845	0.8807
	F-Measure	**0.8901***	0.8721	0.8778	0.8732
Yahoo Music	*RMSE*	**1.1229**	1.1443	1.1289	1.1333
	F-Measure	**0.5261***	0.5073	0.5177	0.5170

Table 7.8 shows how the cold-start problem can be addressed by the proposed ECoRec approach. Compared to the baseline algorithms, our ensemble approach successfully tackles the cold-start problem by largely improving the performance of the unpopular users in most of the experiments.

5.3.5. Sparsity problem

The sparsity of a rating matrix is defined as (Somboonviwat and Aoyama, 2016):

$$Sparsity = \left(1 - \frac{Number\ of\ Ratings}{Number\ of\ users \times Number\ of\ items}\right) 100\% \qquad (18)$$

ECoRec alleviates sparsity by labeling new examples and thereby expanding the original dataset used to train recommender systems. In order to quantify the achieved label expansion, Table 7.9 displays the databases before and after the co-training process with two and three views when using 20, 30 and 40 new unlabeled items per user.

We note that the reduction of sparsity is directly related to the improvement of the accuracy of the results, since the more labeled examples the better the ability of the recommender algorithms to learn the users' behavior. In some databases, such as CiaoDVD, Booking Crossing and Amazon

Table 7. 5. Comparison between ECoRec with three recommenders and standalone KNN-based and Matrix Factorization-based recommenders in terms of RMSE.

Recommenders	**User-KNN**					**Item-KNN**					**SVD**				
User-Item Matrix	T_L	T_L^1	T_L^2	T_L^3	T_L^E	T_L	T_L^1	T_L^2	T_L^3	T_L^E	T_L	T_L^1	T_L^2	T_L^3	T_L^E
Yahoo Movies															
20 new ratings	0.983	0.972	0.973	0.975	0.972	0.963	0.957	0.955	0.953	0.952	0.984	0.974	0.974	0.975	0.972
30 new ratings		0.973	0.974	0.972	0.971		0.951	0.951	0.950	0.949		0.976	0.976	0.976	0.973
40 new ratings		0.971	0.972	0.973	**0.970**		0.949	0.951	0.950	**0.946***		0.975	0.973	0.970	**0.968**
FilmTrust															
20 new ratings	0.858	0.838	0.836	0.838	0.835	0.833	0.827	0.821	0.821	0.820	0.819	0.819	0.817	0.818	0.817
30 new ratings		0.843	0.828	0.838	0.828		0.818	0.817	0.825	0.817		0.819	0.818	0.818	0.816
40 new ratings		0.831	0.825	0.834	**0.824**		0.814	0.818	0.816	**0.813**		0.816	0.811	0.814	**0.807***
CiaoDVD															
20 new ratings	1.076	1.071	1.081	1.072	**1.070**	1.066	1.085	1.064	1.085	1.064	0.981	0.973	0.971	0.975	0.970
30 new ratings		1.108	1.113	1.108	1.106		1.070	1.063	1.076	1.061		0.967	0.970	0.970	0.969
40 new ratings		1.1072	1.1046	1.1332	1.0902		1.040	1.052	1.048	**1.036**		0.965	0.968	0.970	**0.962***
MovieLens $2k$															
20 new ratings	0.790	0.790	0.789	0.789	**0.787**	0.746	0.746	0.746	0.746	0.745	0.787	0.781	0.781	0.781	**0.778**
30 new ratings		0.789	0.789	0.789	0.789		0.746	0.746	0.746	**0.744***		0.785	0.786	0.785	0.785
40 new ratings		0.790	0.789	0.789	0.789		0.746	0.747	0.747	0.747		0.785	0.786	0.785	0.784
Jester															
20 new ratings	1.020	1.002	0.996	0.994	0.986	0.967	0.962	0.961	0.963	0.962	0.957	0.952	0.951	0.952	0.951
30 new ratings		0.996	0.990	0.998	**0.979**		0.962	0.960	0.961	0.959		0.952	0.951	0.956	**0.949***
40 new ratings		1.000	0.986	0.995	0.980		0.961	0.957	0.960	**0.955**		0.955	0.954	0.959	0.954
Booking Crossing															
20 new ratings	0.993	1.002	0.994	1.023	0.978	0.865	0.842	0.861	0.872	0.839	0.769	0.767	0.763	0.766	**0.761**
30 new ratings		0.941	0.991	0.994	0.890		0.812	0.855	0.853	0.789		0.765	0.764	0.765	0.763
40 new ratings		0.853	0.872	0.888	**0.814**		0.779	0.771	0.792	**0.757***		0.766	0.764	0.766	0.764
Amazon Digital Music															
20 new ratings	0.971	0.987	0.988	1.00	0.963	0.905	0.943	0.947	0.957	0.942	0.871	0.874	0.873	0.877	0.870
30 new ratings		0.951	0.976	0.989	0.915		0.931	0.973	0.967	0.911		0.870	0.871	0.872	**0.867**
40 new ratings		0.902	0.929	0.934	**0.883**		0.889	0.902	0.933	**0.866***		0.870	0.872	0.871	0.870
Yahoo Music															
20 new ratings	1.147	1.141	1.149	1.142	**1.140**	1.145	1.124	1.125	1.126	**1.118***	1.197	1.171	1.170	1.189	**1.169**
30 new ratings		1.147	1.143	1.144	1.142		1.128	1.131	1.129	1.127		1.189	1.192	1.192	1.188
40 new ratings		1.149	1.148	1.150	1.147		1.134	1.134	1.130	1.129		1.194	1.191	1.193	1.190

Bold typeset indicates the best performance in each recommender algorithm. * indicates statistical significance with p-value <0.01.

Table 7. 6. Comparison between ECoRec with three recommenders and standalone KNN-based and Matrix Factorization-based recommenders in terms of F-Measure.

Recommenders	**User-KNN**					**Item-KNN**					**SVD**				
User-Item Matrix	T_L	T_L^1	T_L^2	T_L^3	T_L^E	T_L	T_L^1	T_L^2	T_L^3	T_L^E	T_L	T_L^1	T_L^2	T_L^3	T_L^E
Yahoo Movies															
20 new ratings	0.834	0.837	0.840	0.839	0.842	0.827	0.831	0.832	0.831	0.838	0.825	0.831	0.832	0.832	0.837
30 new ratings		0.838	0.841	0.840	0.843		0.830	0.831	0.831	0.838		0.830	0.832	0.831	0.839
40 new ratings		0.838	0.844	0.839	**0.850***		0.833	0.837	0.836	**0.841**		0.831	0.838	0.839	**0.845***
FilmTrust															
20 new ratings	0.608	0.623	0.615	0.628	0.630	0.617	0.633	0.632	0.631	0.638	0.630	0.643	0.642	0.642	0.649
30 new ratings		0.622	0.625	0.637	0.641		0.633	0.633	0.632	0.637		0.643	0.647	0.6422	0.647
40 new ratings		0.638	0.627	0.637	**0.651***		0.639	0.635	0.632	**0.645***		0.644	0.648	0.645	**0.652***
CiaoDVD															
20 new ratings	0.797	0.805	0.807	0.802	0.815	0.796	0.801	0.802	0.800	0.808	0.828	0.831	0.837	0.841	0.847
30 new ratings		0.806	0.809	0.807	0.819		0.802	0.803	0.804	0.811		0.832	0.833	0.833	0.839
40 new ratings		0.806	0.814	0.819	**0.825**		0.812	0.813	0.814	**0.827***		0.835	0.837	0.839	**0.848***
MovieLens $2k$															
20 new ratings	0.783	0.804	0.812	0.809	**0.819***	0.803	0.809	0.816	0.809	**0.821***	0.805	0.809	0.810	0.810	**0.819**
30 new ratings		0.804	0.804	0.803	0.808		0.810	0.810	0.809	0.812		0.810	0.810	0.810	0.816
40 new ratings		0.804	0.805	0.804	0.810		0.810	0.811	0.810	0.818		0.810	0.811	0.811	0.817
Jester															
20 new ratings	0.526	0.580	0.586	0.581	0.598	0.556	0.563	0.567	0.560	0.570	0.544	0.543	0.555	0.556	0.562
30 new ratings		0.576	0.585	0.572	0.581		0.564	0.566	0.557	0.568		0.564	0.566	0.566	0.572
40 new ratings		0.589	0.595	0.596	**0.611***		0.576	0.577	0.573	**0.586***		0.558	0.567	0.567	**0.579**
Booking Crossing															
20 new ratings	0.877	0.893	0.894	0.891	0.901	0.852	0.874	0.875	0.873	0.882	0.887	0.884	0.891	0.895	0.899
30 new ratings		0.892	0.897	0.895	0.915		0.873	0.877	0.875	0.889		0.893	0.897	0.897	0.900
40 new ratings		0.904	0.899	0.907	**0.927***		0.886	0.888	0.886	**0.893**		0.910	0.901	0.915	**0.932***
Amazon Digital Music															
20 new ratings	0.833	0.856	0.858	0.856	0.866	0.827	0.841	0.843	0.842	0.848	0.849	0.851	0.856	0.854	0.862
30 new ratings		0.857	0.863	0.860	0.869		0.844	0.844	0.844	0.849		0.858	0.857	0.857	0.860
40 new ratings		0.860	0.868	0.864	**0.878***		0.849	0.857	0.854	**0.859***		0.859	0.860	0.866	**0.874***
Yahoo Music															
20 new ratings	0.457	0.471	0.475	0.473	0.479	0.502	0.515	0.517	0.516	0.520	0.465	0.517	0.518	0.517	0.522
30 new ratings		0.470	0.474	0.471	0.477		0.511	0.514	0.512	0.517		0.516	0.514	0.514	0.520
40 new ratings		0.487	0.488	0.491	**0.498***		0.526	0.527	0.528	**0.533***		0.526	0.527	0.527	**0.531***

Bold typeset indicates the best performance in each recommender algorithm. * indicates statistical significance with p-value <0.01.

Table 7. 7. Comparison between ECoRec and CSEL with three recommenders in terms of RMSE. Bold typeset indicates the best performance. * indicates statistical significance with p-value <0.01.

Database	Measure / Approach	ECoRec	CSEL	Co-CF	AB.RT
Yahoo Movies	*RMSE*	**0.9467***	0.9892	0.9578	0.9704
	F-Measure	**0.8507***	0.8254	0.8323	0.8260
FilmTrust	*RMSE*	**0.8079***	0.8436	0.8481	0.8327
	F-Measure	**0.6529***	0.6174	0.6081	0.6148
CiaoDVD	*RMSE*	**0.9626***	0.9867	0.9988	0.9761
	F-Measure	**0.8486**	0.8254	0.8019	0.8332
MovieLens 2k	*RMSE*	**0.7446**	0.7689	0.7772	0.7507
	F-Measure	**0.8218***	0.7982	0.7873	0.8023
Jester	*RMSE*	**0.9494***	0.9695	0.9623	0.9573
	F-Measure	**0.6118***	0.5782	0.5691	0.5820
Booking Crossing	*RMSE*	**0.7572***	0.8416	0.7920	0.8045
	F-Measure	**0.9325***	0.8927	0.8843	0.8906
Amazon Digital Music	*RMSE*	0.8661	0.8857	**0.8575***	0.8738
	F-Measure	**0.8784***	0.8675	0.8476	0.8532
Yahoo Music	*RMSE*	**1.1186***	1.2132	1.3645	1.2472
	F-Measure	**0.5339***	0.5124	0.4950	0.5173

Digital Music, despite the small percentage reduction in sparsity, the results exhibited considerable accuracy improvements when compared to the original training set.

5.4. Discussion

The experiments show that, by using ECoRec, we obtain better results than by using recommenders trained with the original data in all databases. This is due to the fact that we can infer new users' feedback, reducing the sparseness and cold-start of the original database with a certain confidence based on the multiple views of different recommender algorithms.

Our co-training approach was able to successfully increase the number of ratings in the training sets, leading to lower RMSE and higher F-Measure using two or more recommenders, as shown in Tables 7.2, 7.3, 7.5 and 7.6. This improvement can be explained by the following two reasons. First, ECoRec ranks all ratings using the confidence measure, giving less importance to unreliable ratings, such as pairs (u, i) with few interactions and largely deviating from the average scores. The other reason is that it combines the individual results generated by each recommender in the final recommendation, driving them to agree with each other and improving the accuracy of predictions throughout iterations and covering as many pairs as possible.

Table 7. 8. RMSE for KNN-based recommenders before (T_L) and after ECoRec (T_L^E) with 20, 30 and 40 new ratings per user. The results displayed are only for those users with less than 20 interactions in the training set.

Measures	User-KNN				Item-KNN			
	T_L	ECoRec (T_L^E)			T_L	ECoRec (T_L^E)		
		Unlabeled item per user (X)				Unlabeled item per user (X)		
		20	**30**	**40**		**20**	**30**	**40**
Yahoo Movies (*484 new users*)								
RMSE	1.1929	1.1419	1.1389	**1.1361***	1.1635	1.1570	1.1570	1.1434
F-Measure	0.8461	0.8589	0.8590	**0.8667***	0.8405	0.8542	0.8535	0.8587
FilmTrust (*889 new users*)								
RMSE	0.8030	0.8072	0.7924	0.7710	0.8100	0.7942	0.7789	**0.7612***
F-Measure	0.8161	0.8539	0.8552	0.8501	0.8064	0.8459	0.8461	**0.8622***
CiaoDVD (*1,658 new users*)								
RMSE	1.0933	**0.9575***	1.1668	1.2148	1.1350	1.1308	1.0954	0.9988
F-Measure	0.8018	0.8331	0.8338	**0.8453***	0.7939	0.8253	0.8258	0.8313
MovieLens 2k (*113 new users*)								
RMSE	1.0055	0.9975	0.9827	**0.9554**	0.9655	0.9640	0.9641	0.9585
F-Measure	0.8848	0.8952	0.8957	0.9097	0.8960	0.9076	0.9076	**0.9249***
Jester (*43 new users*)								
RMSE	1.0027	1.0001	1.0019	0.9935	1.0027	0.9590	**0.9584**	0.9650
F-Measure	0.8043	0.8489	0.8464	**0.8654***	0.8167	0.8268	0.8231	0.8354
Booking Crossing (*2,675 new users*)								
RMSE	1.2438	1.1069	0.8655	**0.7968***	1.0844	0.9851	0.8183	0.8143
F-Measure	0.8924	0.9145	0.9149	**0.9286***	0.8981	0.9006	0.9007	0.8986
Amazon Digital Music (*5,426 new users*)								
RMSE	1.0817	1.0758	0.9439	0.8974	1.0637	1.0566	0.9371	**0.8326***
F-Measure	0.8470	0.8678	0.8681	0.8728	0.8562	0.8619	0.8618	**0.8748**
Yahoo Music (*664 new users*)								
RMSE	1.2313	1.1926	1.1962	1.2084	1.2313	**1.1731***	1.1769	1.1797
F-Measure	0.7062	0.7378	0.7349	**0.7469***	0.6899	0.7086	0.7040	0.7369

Bold typeset indicates the best performance. * indicates statistical significance with p-value <0.01.

Table 7. 9. Sparsity of databases before and after ECoRec with 20, 30 and 40 new ratings per user using two and three recommenders (views).

		Unlabeled items per user (X)		
Database / Approach	**Original (T_L)**	**20**	**30**	**40**
Yahoo Movies	99.85 %	98.46 %	98.06 %	97.66 %
FilmTrust	98.93 %	97.18 %	96.28 %	95.40 %
CiaoDVD	99.84 %	99.57 %	99.43 %	99.29 %
MovieLens 2K	96.37 %	96.27 %	96.07 %	95.87 %
Jester	33.65 %	23.48 %	16.97 %	12.17 %
Booking Crossing	99.66 %	98.93 %	98.55 %	98.17 %
Amazon Digital Music	99.81 %	99.06 %	98.67 %	98.29 %
Yahoo Music	96.28 %	93.00 %	91.16 %	89.31 %

The results presented in Section 5.3 show that ECoRec, especially with the ensemble approach, was able to achieve better accuracy in databases from four different domains (books, movies, music and jokes), with statistical

significance. Although some recommenders provided worse predictions than others individually, during the co-training process they complement each other and generate better results than the baselines. Most of the results showed improvement in the accuracy of recommendation, especially when using more than ten new ratings per user in the ensemble co-training process.

The number of new ratings per user (X) affects performance in a reasonably predictable way. Typically, increasing X will tend to increase performance, but only to a certain data-dependent extent, beyond which performance may plateau or start degrading. The reason for this behavior is that, as X increases, the proportion of estimated ratings increases as compared to the proportion of real ratings in the augmented user-item matrix. Since the estimated ratings are subject to estimation errors, there is a point beyond which the gains from reducing sparsity no longer overweigh the increasing uncertainty in the data. In our experiments, performance dropped in most databases when $X > 50$, due to the fact that samples become less reliable (ratings tend to be more global rather than reflect user's behavior) and the method thus becomes more sensitive to noise.

In CSEL (Zhang et al., 2014) and in Co-CF (Quang et al., 2015), the authors chose the best sample of the unlabeled set based on their confidence measure, whereas we randomly select X new items for each user from the database. We can see in Tables 7.4 and 7.7 that ECoRec outperforms CSEL and Co-CF with two or three recommenders in the most cases. Our approach, in which the distribution of new data is the same for all users, has thus been shown to be more effective to mitigate cold-start and sparsity. Moreover, our approach also outperforms AdaBoost.RT (Bar et al., 2013).

The computational cost of the algorithm is approximately t times the cost of the recommender algorithms used, since the individual models are re-trained t times during the co-training process. This additional burden can be lessened by using task parallel libraries and computer clusters, as the multiple recommenders can be trained independently in each iteration of the algorithm. In our experiments, we parallelized the recommenders, confidence calculation and ensemble steps, using a native Python library[1]. It is worth noticing that the entire co-training processes can be performed off-line, so it is by no means a bottleneck for real-world applications.

[1]https://docs.python.org/3/library/multiprocessing.html

6. Final Remarks

This paper proposed an ensemble-based co-training approach, named ECoRec, to process data from two or more different views in order to create a more precise and robust model for recommendation. In our experiments, these views were accomplished by different recommender algorithms, namely, User-KNN, Item-KNN, and SVD.

We conducted experiments in eight databases from different domains (movies, books, jokes and music) and the results show that our strategy improves the overall system's performance in a variety of experimental setups. The main advantage of our approach is to provide an enriched user-item matrix that helps mitigate the sparsity and cold-start problems. The proposed ECoRec approach is extensible and flexible, as it enables developers to use different recommender algorithms, confidence measures, and ensemble strategies.

As future work, we aim to assess some of these possible extensions. We also intend to investigate how to incorporate different types of feedback, for example ratings and sentiment review, so they can improve the accuracy of the predictions when they are available. Finally, we intend to investigate an active learning technique for linear regression as part of the ensemble step, in order to weight each recommenders' estimated accuracy as evaluated during the co-training process.

Acknowledgements

This work was supported by the Coordination for the Improvement of Higher Education Personnel (CAPES), the São Paulo Research Foundation, FAPESP [2016/20280-6], and the Brazilian National Research Council, CNPq [304137/2013-8].

References

Adomavicius, G., Kamireddy, S., Kwon, Y., 1 2007. Towards more confident recommendations: Improving recommender systems using filtering approach based on rating variance. In: WITS 2007 - Proceedings, 17th Annual Workshop on Information Technologies and Systems. Social Science Research Network, pp. 152–157.

Aggarwal, C. C., 2016. Recommender Systems: The Textbook, 1st Edition. Springer Publishing Company, Incorporated.
URL http://www.springer.com/us/book/9783319296579

Bar, A., Rokach, L., Shani, G., Shapira, B., Schclar, A., 2013. Improving simple collaborative filtering models using ensemble methods. In: Multiple Classifier Systems: 11th International Workshop. Vol. 7872 of MCS '13. Springer Berlin Heidelberg, pp. 1–12.
URL https://doi.org/10.1007/978-3-642-38067-9_1

Blum, A., Mitchell, T., 1998. Combining labeled and unlabeled data with co-training. In: Proceedings of the Eleventh Annual Conference on Computational Learning Theory. COLT' 98. ACM, New York, NY, USA, pp. 92–100.
URL http://doi.acm.org/10.1145/279943.279962

Bobadilla, J., Gutirrez, A., Ortega, F., Zhu, B., 2018. Reliability quality measures for recommender systems. Information Sciences 442-443, 145 – 157.
URL http://www.sciencedirect.com/science/article/pii/S0020025516310052

Bobadilla, J., Ortega, F., Hernando, A., Gutiérrez, A., 2013. Recommender systems survey. Knowledge-Based Systems 46, 109–132.
URL https://doi.org/10.1016/j.knosys.2013.03.012

Box, J. F., 02 1987. Guinness, gosset, fisher, and small samples. Statistical Science 2 (1), 45–52.
URL http://dx.doi.org/10.1214/ss/1177013437

Cantador, I., Brusilovsky, P., Kuflik, T., 2011. 2nd workshop on information heterogeneity and fusion in recommender systems (hetrec 2011). RecSys 2011. ACM, New York, NY, USA, pp. 387–388.
URL http://doi.acm.org/10.1145/2043932.2044016

Castells, P., Hurley, N. J., Vargas, S., 2015. Novelty and Diversity in Recommender Systems. Springer US, Boston, MA, pp. 881–918.

Da Costa, A. F., Manzato, M. G., Campello, R. J. G. B., 2018. Corec: A co-training approach for recommender systems. SAC '18. ACM, New York, NY, USA.

Goldberg, K., Roeder, T., Gupta, D., Perkins, C., Jul. 2001. Eigentaste: A constant time collaborative filtering algorithm. Information Retrieval 4 (2), 133–151.
URL https://doi.org/10.1023/A:1011419012209

Guo, G., 2013. Improving the performance of recommender systems by alleviating the data sparsity and cold start problems. In: Proceedings of the 23rd International Joint Conference on Artificial Intelligence. IJCAI '13. AAAI Press, pp. 3217–3218.
URL http://dl.acm.org/citation.cfm?id=2540128.2540617

Guo, G., Zhang, J., Thalmann, D., Yorke-Smith, N., 2014. Etaf: An extended trust antecedents framework for trust prediction. In: Proceedings of the 2014 International Conference on Advances in Social Networks Analysis and Mining. ASONAM '14. pp. 540–547.
URL https://doi.org/10.1109/ASONAM.2014.6921639

Guo, G., Zhang, J., Yorke-Smith, N., 2013. A novel bayesian similarity measure for recommender systems. In: Proceedings of the 23rd International Joint Conference on Artificial Intelligence. IJCAI '13. pp. 2619–2625.
URL http://dl.acm.org/citation.cfm?id=2540128.2540506

Hernando, A., Bobadilla, J., Ortega, F., Tejedor, J., Jan. 2013. Incorporating reliability measurements into the predictions of a recommender system. Inf. Sci. 218, 1–16.
URL https://doi.org/10.1016/j.ins.2012.06.027

Jahrer, M., Töscher, A., Legenstein, R., 2010. Combining predictions for accurate recommender systems. In: Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. KDD '10. ACM, New York, NY, USA, pp. 693–702.
URL http://doi.acm.org/10.1145/1835804.1835893

Karimi, R., Freudenthaler, C., Nanopoulos, A., Schmidt-Thieme, L., Nov 2011. Towards optimal active learning for matrix factorization in recommender systems. In: 2011 IEEE 23rd International Conference on Tools with Artificial Intelligence. IEEE Computer Society, Washington, DC, USA, pp. 1069–1076.
URL http://dx.doi.org/10.1109/ICTAI.2011.182

Koren, Y., 2008. Factorization meets the neighborhood. KDD '08. ACM, New York, NY, USA, pp. 426–434.
URL http://doi.acm.org/10.1145/1401890.1401944

Koren, Y., Bell, R., Volinsky, C., 2009. Matrix factorization techniques for recommender systems. Computer 42 (8), 30–37.
URL http://dx.doi.org/10.1109/MC.2009.263

Lee, J.-S., Olafsson, S., 2009. Two-way cooperative prediction for collaborative filtering recommendations. Expert Systems with Applications 36 (3, Part 1), 5353 – 5361.
URL http://www.sciencedirect.com/science/article/pii/S0957417408003746

Matuszyk, P., Spiliopoulou, M., Jun 2017. Stream-based semi-supervised learning for recommender systems. Machine Learning 106 (6), 771–798.
URL https://doi.org/10.1007/s10994-016-5614-4

Mazurowski, M. A., 2013. Estimating confidence of individual rating predictions in collaborative filtering recommender systems. Expert Systems with Applications 40 (10), 3847 – 3857.
URL http://www.sciencedirect.com/science/article/pii/S0957417413000031

McAuley, J., Pandey, R., Leskovec, J., 2015. Inferring networks of substitutable and complementary products. In: Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. KDD '15. ACM, New York, NY, USA, pp. 785–794.
URL http://doi.acm.org/10.1145/2783258.2783381

Quang, T. N., Lien, D. T., Phuong, N. D., 2015. Collaborative filtering by co-training method. In: Nguyen, V.-H., Le, A.-C., Huynh, V.-N. (Eds.), Knowledge and Systems Engineering. Springer International Publishing, Cham, pp. 273–285.

Ravi, L., Vairavasundaram, S., Mar. 2016. A collaborative location based travel recommendation system through enhanced rating prediction for the group of users. Intell. Neuroscience 2016, 7–.
URL http://dx.doi.org/10.1155/2016/1291358

Ricci, F., Rokach, L., Shapira, B., 2015. Recommender Systems Handbook, 2nd Edition. Springer Publishing Company, Incorporated, New York, NY,

USA.
URL http://www.springer.com/la/book/9781489976369

Ristoski, P., Mencia, E. L., Paulheim1, H., 2014. A hybrid multi-strategy recommender system using linked open data. In: Semantic Web Evaluation Challenge: SemWebEval 2014. ESWC '14. Cham, pp. 150–156.
URL https://doi.org/10.1007/978-3-319-12024-9_19

Schclar, A., Tsikinovsky, A., Rokach, L., Meisels, A., Antwarg, L., 2009. Ensemble methods for improving the performance of neighborhood-based collaborative filtering. In: Proceedings of the Third ACM Conference on Recommender Systems. RecSys '09. ACM, New York, NY, USA, pp. 261–264.
URL http://doi.acm.org/10.1145/1639714.1639763

Somboonviwat, K., Aoyama, H., 2016. Empirical analysis of the relationship between trust and ratings in recommender systems. In: Nguyen, N. T., Trawiński, B., Fujita, H., Hong, T.-P. (Eds.), Intelligent Information and Database Systems: 8th Asian Conference. ACIIDS '16. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 116–126.
URL https://doi.org/10.1007/978-3-662-49381-6_12

Su, X., Khoshgoftaar, T. M., Zhu, X., Greiner, R., 2008. Imputation-boosted collaborative filtering using machine learning classifiers. In: Proceedings of the 2008 ACM Symposium on Applied Computing. SAC '08. ACM, New York, NY, USA, pp. 949–950.
URL http://doi.acm.org/10.1145/1363686.1363903

Sun, M., Li, F., Lee, J., Zhou, K., Lebanon, G., Zha, H., 2013. Learning multiple-question decision trees for cold-start recommendation. In: Proceedings of the Sixth ACM International Conference on Web Search and Data Mining. WSDM '13. ACM, New York, NY, USA, pp. 445–454.
URL http://doi.acm.org/10.1145/2433396.2433451

Xu, C., Tao, D., Xu, C., 2013. A survey on multi-view learning. Computing Research Repository - CoRR abs/1304.5634.
URL http://arxiv.org/abs/1304.5634

Zhang, D., Lee, W. S., 2005. Validating co-training models for web image classification. Computer Science (CS); Singapore-MIT Alliance (SMA).

Zhang, M., Tang, J., Zhang, X., Xue, X., 2014. Addressing cold start in recommender systems: A semi-supervised co-training algorithm. In: Proceedings of the 37th International ACM SIGIR Conference on Research and Development in Information Retrieval. SIGIR '14. ACM, New York, NY, USA, pp. 73–82.
URL http://doi.acm.org/10.1145/2600428.2609599

Zhang, Q., Wang, H., 2015. Collaborative multi-view learning with active discriminative prior for recommendation. In: Advances in Knowledge Discovery and Data Mining: 19th Pacific-Asia Conference, PAKDD 2015, Ho Chi Minh City, Vietnam, May 19-22, 2015, Proceedings, Part I. Springer International Publishing, Cham, pp. 355–368.
URL https://doi.org/10.1007/978-3-319-18038-0_28

Zhou, Z.-H., Li, M., 2005. Semi-supervised regression with co-training. In: Proceedings of the 19th International Joint Conference on Artificial Intelligence. IJCAI'05. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, pp. 908–913.
URL http://dl.acm.org/citation.cfm?id=1642293.1642439

Zhu, T., Hu, B., Yan, J., Li, X., 2010. Semi-supervised learning for personalized web recommender system. Computing and Informatics 29, 617–627.

Ziegler, C.-N., McNee, S. M., Konstan, J. A., Lausen, G., 2005. Improving recommendation lists through topic diversification. In: Proceedings of the 14th International Conference on World Wide Web. WWW '05. ACM, New York, NY, USA, pp. 22–32.
URL http://doi.acm.org/10.1145/1060745.1060754

7.2 Remarks about the chapter

Aiming to improve the quality of recommendations and reduce the number of outputs generated by CoRec algorithm, in this chapter we proposed an exetension of CoRec algorithm with an ensemble approach to combine the generated enriched sets in a single robust user-item feedback matrix using an heuristic approach based on a confidence metric. In this work, we also extended the co-training task to support more than two recommenders, in order to improve the power of prediction of our pre-processing approach.

In spite of the variety of ways to analyze the content which are specific to each technique and purposes, the combination of different views of the content may create a rich, reliable and meaningful set of descriptions that, when used together, will be able to improve the efficiency of recommender systems and reduce the amount of unnecessary training (e.g. using 3 recommenders, we would have 3 outputs to be tested). Indeed, the combination of the output of different recommenders explores the best of each one, capturing the semantics of the content in a richer and more detailed way.

Our proposal differs from others reviewed previously, e.g. (VLACHOS *et al.*, 2014; ZHANG *et al.*, 2014; COSTA *et al.*, 2016), because it does not require enriched information or any specific type of feedback, while being able to use enriched information when it is available. Furthermore, in ECoRec we deal directly with the cold-start and sparsity problems, labeling several unknown user-item pairs in the original dataset. The presented results show that the use of our proposed ensemble approach improves the overall system's performance in a variety of experimental setups.

In the next chapter, we present our recommender systems framework, named Case Recommender, that was developed to integrate and make available all algorithms and evaluation strategies used in our research.

CHAPTER

8

CASE RECOMMENDER: A RECOMMENDER FRAMEWORK

8.1 Contextualization

This chapter's article,entitled "CaseRecommender: a recommender framework", presents an extensible and robust recommender systems framework developed during this PhD project, which contains well-known and state-of-the-art recommender algorithms, as well as evaluation and validation metrics. To implement the approaches proposed in our research, we do not use existing tools for three main reasons:

- Wide Range of Applications: Many systems available focus on few applications of RS.
- Difficult Implementation: Many frameworks exhibit design flaws which hinders the integration with another systems.
- Low Scalability: There is a lack of flexibility in features, and it is difficult to implement new features or adapt existing ones.

Case Recommender is a Python implementation of a number of popular recommendation algorithms for both implicit and explicit feedback developed during the my master and doctoral degree. Our framework has different types of top-N recommendation and rating prediction approaches, in order to cover as many scenarios as possible in RS. The framework aims to integrate and to provide a rich set of components from which researchers and developers can construct a customized recommender system from a set of algorithms, and mainly pre-process input data in order to alleviate the problems presented in this dissertation and to improve the quality of the recommendations of the implemented and proposed algorithms.

Case Recommender: A Recommender Framework

Arthur F. da Costa and Marcelo G. Manzato
Institute of Mathematics and Computer Science
University of São Paulo, São Carlos, SP, Brazil
{fortes, mmanzato}@icmc.usp.br

ABSTRACT

Case Recommender is a Python implementation of a number of popular content-based and collaborative recommendation algorithms which use implicit and explicit feedback. The framework aims to provide a rich set of components from which developers can construct a customized recommender system. One differential of the framework is the possibility to combine multiple recommenders in a post-processing ensemble approach to produce more accurate results. Case Recommender can be used for rating prediction and item recommendation tasks, and it includes different metrics and procedures for validation and evaluation.

CCS CONCEPTS

• **Information systems** → **Recommender systems**; • **Computing methodologies** → *Machine learning approaches.*

PUBLICATION INFORMATION

Published in proceedings of the 22nd Brazilian Symposium on Multimedia and the Web (Webmedia '16). Workshop de Ferramentas e Aplicações (WFA). 2016. Volume 2. Pages 99-102.

KEYWORDS

Recommender Systems; Framework

1 INTRODUCTION

The growth of on-line content available to users has made finding and consuming relevant items a challenge that users have to deal with everyday. In response to this problem, recommender systems have been created, which are an information filtering technology that can be used to predict preference ratings of items, not currently rated by the user, and/or to output a personalized ranking of items/recommendations that are likely to be of interest to the user [7].

In order to obtain such interests, profiling mechanisms have been developed, which consist of acquiring, representing and maintaining pieces of information relevant (and/or irrelevant) to the user. In the particular case of obtaining user's preferences, the three most known techniques are based on explicit feedback, implicit feedback and hybrid approaches. Implicit information is collected indirectly during user navigation with the system while visiting a page, mouse movement and clicks on various links of interest [4]. Regarding explicit feedback, the data is intentionally provided, e.g., the users express themselves in some direct way (e.g. filling in forms or rating a content). This type of information is considered more reliable, since the user is the one who provides the topics of interest, but the cost of this procedure is the effort of the individual, who is not always willing to cooperate with the system [7]. Finally, the hybrid approach consists of applying the implicit and explicit feedback together, in order to obtain a greater number of user information [7].

Traditionally, recommendation systems employ filtering techniques and machine learning information to generate appropriate recommendations to the user's interests from the representation of his profile. However, other techniques, such as Neural Networks, Bayesian Networks and Association Rules, are also used in the filtering process [2]. The most used types of filtering are currently: Content-Based (CBF), responsible for selecting information based on content filtering of elements, e.g., e-mail messages filtered out as trash for containing unwanted words; Collaborative Filtering (CF), based on the relationship between people and their subjective regarding the information to be filtered. The selection of e-mails based on the relationship between sender and recipient is an example of that. Besides, there is a hybrid approach that combines the content-based and the collaborative filtering methods. [7].

Although there are useful libraries which can be used to implement, evaluate and extend classical recommenders, such as Apache Mahout[1], MyMediaLite[2], LensKit[3], among others, the viability of a framework which is able to contain multiple recommender algorithms available using a variety of data sources is unknown. Such framework, indeed, should be able to deal with the effects of using large volumes of data considering hardware configuration, and the impact of model updates in the recommender performance.

In this paper we describe a new framework called Case Recommender, which contains a variety of content-based and collaborative recommender algorithms, as well as ensembling approaches for combining multiple algorithms and data sources. In addition, it provides a set of popular evaluation

[1] https://mahout.apache.org/
[2] http://www.mymedialite.net/
[3] http://lenskit.org/

methods and metrics for rating prediction and item recommendation. As a result, it can be used to conduct fair and comprehensive comparisons between different recommendation algorithms. The framework is published in PyPi[4], a international repository of software for the Python programming language, as a GNU GPLv3 License, making it easy for third parties to contribute with additional implementations and features.

This paper is structured as follows: Section 2 addresses the related work; Section 3 describes the framework proposed in this work; Section 4 reports Case Recommender architecture; Section 5 presents details of the license and distribution of the code; finally, Section 6 is devoted to final remarks and future works.

2 RELATED WORK

A number of frameworks for recommender systems have been proposed by the scientific community, involving different programming languages, such as C/C++, Java, C#, Python, among others. One of the most popular and complete frameworks is the open-source recommendation library MyMediaLite, which has been developed since 2011 and contains a variety of recommender algorithms for rating prediction and item recommendation. However, it lacks content-based filtering approaches, besides its design which embarrasses new developers to extend it with new features.

Other libraries, such as EasyRec[5], Mathout, LensKit and RiVal[6] have also been proposed, but most of them contain only a small set of recommendation approaches. Similarly to MyMediaLite, most of them do not provide content-based recommenders. In addition, none of them provides ensemble approaches that allow developers to combine multiple recommenders. Table 8.2 summarizes the features available in each considered recommender library.

In this sense, Case Recommender was developed to provide flexibility and extensibility in research environments, while maintaining high performance. Its primary concern is to maximize usefulness for research and education, instead of large-scale commercial operations. The framework is also designed to support a wide variety of recommendation approaches, including content-based and collaborative filtering.

3 CASE RECOMMENDER

Case Recommender addresses two common scenarios in content-based and collaborative filtering: rating prediction and item recommendation, using eighter explicit and/or implicit feedback. It offers state-of-the-art algorithms for those two tasks and validation and evaluation metrics to measure the recommender algorithms accuracy. Our framework is implemented in Python with scientific environments for numerical applications such as Scipy[7] and NumPy[8]. Using these free and open-source libraries, it can be used on the most popular operating systems. To install Case Recommender on Mac OS X or Linux, chances are that one of the following two commands will work for developers and users (with admin privileges):

easy_install CaseRecommender

or alternatively:

pip install CaseRecommender

Windows users who do not have the easy_install command must first install it before installing the library. For this, they need to check the pip and setuptools on Windows tutorial[9] for more information about how to do that. Once they have it installed, it is possible to run the same commands above.

An important feature in the design of our framework was to enable the computation of recommendations in large-scale. Case Recommender is currently being rewritten to support optimized calculations using known Python scientific libraries. Another feature is to support sparse and large datasets in a way that there is as little as possible overhead for storing data and intermediate results. Moreover, our framework aims to support scaling in recommender systems in order to build high-scale, dynamic and fast recommendations over simple calls.

Another feature of the framework is its extensibility and flexibility, as developers can implement new recommendation algorithms while using the available data structures and routines. According to the application scenario, developers can choose between using one of the available recommender algorithms, or combining multiple recommendations using one of the available ensemble techniques [2, 3]. Table 3 shows the recommender algorithms in the framework.

Case Recommender contains a generic interface for recommender systems implementations, among them the collaborative and content-based filtering approaches such as neighborhood-based (NB) and matrix factorization (MF) models, which are already available for use. The recommender interfaces can be easily combined with more than 15 different pairwise metrics already implemented, like the cosine, tanimoto, pearson, euclidean using Scipy basic optimized

[4]https://pypi.python.org/pypi/CaseRecommender/
[5]http://www.easyrec.org/
[6]http://rival.recommenders.net/
[7]https://www.scipy.org/
[8]http://www.numpy.org/
[9]http://docs.python-guide.org/en/latest/starting/install/win/

Table 8. 1. Case Recommender Algorithms.

Approach	Item Recommender	Rating Prediction
Neighborhood Based	UserKNN User Attr KNN ItemKNN Item Attr KNN	UserKNN User Attr KNN ItemKNN Item Attr KNN
Matrix Factorization	BPR MF	MF SVD ItemNSVD1 UserNSVD1
Ensemble	Tag-Based Average-Based BPR Learning	-

functions[10]. Moreover, it offers support for using similarity functions such as user-to-user or item-to-item and allows easy integration with different input domains like databases, text files or python libraries.

4 ARCHITECTURE

The proposed framework consists of a set of classes and methods for generating and evaluating recommendations in rating prediction and item recommendation scenarios. Figure 8.1 illustrates all components involved.

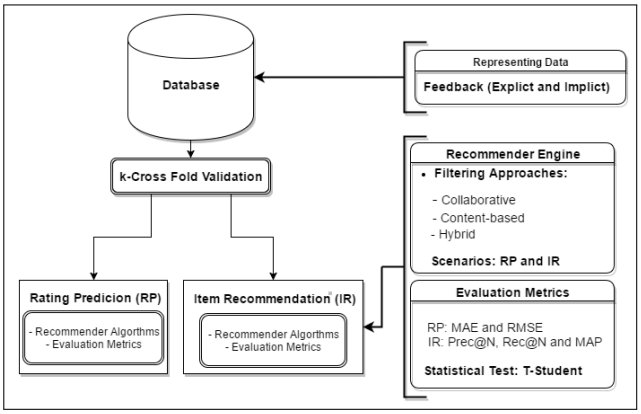


Figure 8. 1. Case Recommender Architecture.

There are three main tasks in Case Recommender: data representation, recommendation and evaluation. The following subsections detail each of them.

[10]http://docs.scipy.org/doc/scipy-0.14.0/reference/generated/scipy.spatial.distance.pdist.html

4.1 Representing Data

Case Recommender is mainly a framework, designed to be used by other applications. We provide command-line interfaces, classes and functions that offer most of framework's functionality. The framework allows developers to deal with different datasets and not having to develop their own programs to execute recommender functions.

The input of rating prediction algorithms expects the data to be in a simple text format:

$$u\_id\ i\_id\ rating$$

where u_id and i_id are integers referring to users and items IDs, respectively, and $rating$ is a floating-point number expressing how much the user likes an item. The separator between the values can be either spaces, tabs, or commas. If there are more than three columns, all additional columns are ignored.

The item recommendation approaches behave similarly to the rating prediction approaches. The main difference is that in most of the item recommendation algorithms the user can omit or ignore the third column, which corresponds to the feedback value. However, there are some algorithms that may or not use the feedback value in both scenarios.

4.2 Recommender Engines

The proposed framework contains a recommender engine composed of several algorithms described in the literature, such as user and item-based kNN and matrix factorization. It provides an assortment of components that may be plugged together and customized to create an ideal recommender for a particular domain.

4.2.1 Rating Prediction. Ratings are a popular kind of explicit feedback. Users assess how much they like a given item (e.g. a movie or a news article) on a predefined scale, e.g. 1 to 5, where 5 could mean that the user likes the item very much, whereas 1 means the user strongly dislikes the item [4]. Rating prediction algorithms estimate unknown ratings from a given set of known ratings and possibly additional data like user or item attributes. The predicted ratings can then indicate to users how much they will like an item, or the system can suggest items with high predicted ratings. Our available prediction algorithms are presented in Table 8.1.

4.2.2 Item Recommendation. In the item recommendation task, the system constructs a list of items which are more likely to be preferred by a given user. Usually in the item recommendation scenario the user's feedback is implicit, which means that his preferences on items are unobservable [5]. For example, in the movie recommendation task, the number of

Table 8. 2. Comparison of free/open source recommender system toolkits.

Toolkits	Case Recommender	EasyRec	Mathout	LensKit	RiVal	MyMediaLite	Crab
Version	0.0.6	0.99	0.12.2	2.2.1	0.2	3.11	0.1
Last update	Jul 2016	Nov 2013	Jun 2016	Nov 2015	Aug 2015	Dec 2015	Jan 2012
Actively Developed	✓	-	✓	✓	-	✓	-
License	GPL 3	GPL 3	Apache 2.0	LGPL 2	Apache 2.0	GPL 3	BSD
Language	Python	Java	Java	Java	Java	C#	Python
Scalable	✓	-	✓	✓	-	✓	✓
Rating Prediction	✓	✓	✓	✓	✓	✓	✓
Item Recommender	✓	-	-	✓	-	✓	✓
CF Techniques	✓	✓	✓	✓	✓	✓	✓
CBF Techniques	✓	-	-	-	-	-	-
Ensemble Techniques	✓	-	-	-	-	-	-

times a user watches a movie is observable (i.e., implicit feedback) but not his explicit preference rating about the movie (i.e., explicit feedback). Recommendation models are then built based on some pre-defined preference assumptions, e.g., the more times a user watches a movie, the more he/ she likes it. Such assumptions may not accurately describe usersâĂŹ preferences. Moreover, the observed user-item interaction data are generally sparse, which makes the preference modeling even more challenging. As a result, existing solutions often deliver unsatisfactory recommendation accuracies.

4.2.3 Ensemble Approaches. Case Recommender provides an extensible platform for ensemble approaches which allow developers to combine different recommender algorithm using different input data. Its flexibility allows developers to build complex architectures for better accuracy depending on the application scenario. Such ensemble approaches, as they are integrated in the framework, allow their evaulation with fair comparisons against individual filtering approaches. Our implementation provide a clear interface, where the ensemble strategies receive the results of previous probe runs (whether required), and the set of results from previous trained recommenders.

4.3 Validation and Evaluation Metrics

Case Recommender contains routines for recommendation evaluation, including measures such as root mean square error (RMSE) and mean average error (MAE) for the rating prediction task. For the item recommendation task, the available metrics are precision-at-N (prec@N), recall-at-N (recall@N) and mean average precision (MAP). For both scenarios, internal K-fold cross-validation is supported to validate the experiments.

The framework also contains the All But One [1] protocol for the construction of the ground truth using K-fold-cross-validation. Given the data set, the system randomly divides it into the same K subsets and for each sample it uses $n-1$, these subsets of data for training and the rest for testing. The training set t_r is used to test the recommendation technique and in the test set T_e the system randomly separates an item for each user to create the truth set H. After that, the remaining items form the set of observable O, which is used to test the algorithm.

In order to compare the results with statistical significance, the two-sided paired t-test [6] is available in Case Recommender.

4.4 Usage

For each of the two recommendation scenarios, Case Recommender comes with a command-line instructions that allow users to train and evaluate all available recommenders on data provided in text files. When a new recommender is added into the framework, it is automatically detected, so developers do not have to manually add new features into the programs, only use the command *import* in Python.

5 LICENSE AND DISTRIBUTION

Case Recommender is freely available under GNU GPLv3 license. The GPL grants the recipients of a computer program the rights of the Free Software Definition[11] and uses copyleft to ensure the freedoms are preserved whenever the work is distributed, even when the work is changed or added to. The GPL is a copyleft license, which means that derived works can only be distributed under the same license terms.

We chose this license because it allows programs to be distributed and reused, keeping, however, the author's rights

[11] https://www.gnu.org/licenses/

in order to not allow this information to be used in a way that limits the original freedoms.

Our project is hosted at Github repository and it is available for the scientific community to use, test and contribute. Future releases are planned which will include more features and an evaluation tool with several plots and graphs to help developers to better understand the behavior of their recommender algorithms. The source code is freely available at *https://github.com/caserec/CaseRecommender*.

6 FINAL REMARKS

In this paper, we presented a framework to recommender systems, called Case Recommender. The goal of this framework is to integrate and facilitate the experiments and development of new recommender techniques for different domains. Case Recommender contains a recommender engine, that contains content-based and collaborative approaches based on neighborhood and matrix factorization models for rating prediction and item recommendation scenarios. In addition, the framework contains ensemble algorithms and validation and evaluation metrics to improve and measure the quality of the recommendation.

The main advantages of our framework are extensibility and flexibility, once it enables developers to use and develop different recommender algorithms in both scenarios of recommendation and also allows different ways of usage of methods and classes during the recommendation process. Moreover, the framework stands out for containing recent content-based filtering algorithms and ensemble approaches, differently from the other recommender toolkits cited in this paper.

We will continue Case Recommender's development in several directions: implementing new recommendation algorithms, evaluation and validation metrics and porting the framework for other programming languages. We also plan to add additional recommendation tasks and types of input, e.g. item prediction from other kinds of implicit feedback like viewing times or click counts, or tag recommendation.

ACKNOWLEDGMENTS

We would like to acknowledge CAPES, CNPq and FAPESP (2016/20280-6) for the financial support.

REFERENCES

[1] John S. Breese, David Heckerman, and Carl Kadie. 1998. Empirical Analysis of Predictive Algorithms for Collaborative Filtering. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence (UAI'98)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 43–52. http://dl.acm.org/citation.cfm?id=2074094.2074100

[2] Arthur F. da Costa and Marcelo G. Manzato. 2016. Exploiting multimodal interactions in recommender systems with ensemble algorithms. *Information Systems* 56 (2016), 120 – 132. https://doi.org/10.1016/j.is.2015.09.007

[3] Arthur da Costa Fortes and Marcelo Garcia Manzato. 2014. Ensemble Learning in Recommender Systems: Combining Multiple User Interactions for Ranking Personalization. In *Proceedings of the 20th Brazilian Symposium on Multimedia and the Web (WebMedia '14)*. ACM, New York, NY, USA, 47–54. https://doi.org/10.1145/2664551.2664556

[4] Zeno Gantner, Steffen Rendle, Christoph F., and Lars Schmidt-Thieme. 2011. MyMediaLite: A Free Recommender System Library. In *Proceedings of the 5th ACM Recommender Systems Conference*.

[5] Nathan N. Liu, Min Zhao, and Qiang Yang. 2009. Probabilistic Latent Preference Analysis for Collaborative Filtering. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management (CIKM '09)*. ACM, New York, NY, USA, 759–766.

[6] Thomas M. Mitchell. 1997. *Machine Learning* (1 ed.). McGraw-Hill, Inc., New York, NY, USA.

[7] Francesco R., Lior R., and Bracha S. 2011. Introduction to Recommender Systems Handbook. In *Recommender Systems Handbook*. Springer US, 1–35.

8.2 Final remarks

Motivated by making available all the approaches and recommender algorithms developed during the masters and the beginning of the doctorate, we developed a recommender systems framework, named Case Recommender, presented in this chapter. In this framework, we implemented well-known and state-of-the-art recommendation algorithms in rating prediction and top-N recommendation scenarios, with several evaluation and validations methodologies. In this way, our framework aims to improve and facilitate the development and use of the recommender algorithms by developers and researchers, making possible to process large-scale data and construct new algorithms without worrying about trivial details of implementation, such as class building or data allocation structures.

In the next chapter, we will report on the current state of our framework and what new features have been added to it during the doctoral research.

CHAPTER

9

CASE RECOMMENDER: A FLEXIBLE AND EXTENSIBLE PYTHON FRAMEWORK FOR RS

9.1 Contextualization

In this chapter, we report the last modifications made in our recommender framework, based on the proposed pre-processing approaches presented in this research and new features of the Python language. This update was done to provide more flexibility and extensibility in research environments, while maintaining high performance, providing a new variety of recommendation and clustering algorithms, as well as functions for data manipulation. Its primary concern is to maximize usefulness for research and education, instead of large-scale commercial operations. The important features of this last version are:

- **Completitude:** Addresses algorithms from both rating prediction and ranking. Beside evaluation metrics and auxiliary tools, such as database splitting. One of few that brings clustering and ensemble techniques already implemented.
- **Simplicity:** Easy creation and extension of algorithms for different types of application, by using many already implemented features for every step of recommendation, from data handling to evaluation.
- **Scalability:** Developers can choose between using one of the available recommender algorithms, combining multiple recommendations using one of the available ensemble techniques, or develop their own algorithm using the *BaseRatingPrediction* or *BaseItemRecommendation* classes.
- **Efficiency:** Enables the computation of sparse and large datasets in a way that there is as little as possible overhead for storing data and intermediate results.

Case Recommender: A Flexible and Extensible Python Framework for Recommender Systems

Arthur da Costa[1], Eduardo Fressato[1], Fernando Neto[1], Marcelo Manzato[1] and Ricardo Campello[2,1]
[1]Institute of Mathematical and Computer Sciences - University of São Paulo, São Carlos, SP, Brazil
[2]School of Mathematical Physical Sciences - University of Newcastle, Newcastle, Australia
fortes@icmc.usp.br, eduardofressato@usp.br, fernando.soares.aguiar@usp.br,
mmanzato@icmc.usp.br and ricardo.campello@newcastle.edu.au

ABSTRACT

This paper presents a polished open-source Python-based recommender framework named Case Recommender, which provides a rich set of components from which developers can construct and evaluate customized recommender systems. It implements well-known and state-of-the-art algorithms in rating prediction and item recommendation scenarios. The main advantage of the Case Recommender is the possibility to integrate clustering and ensemble algorithms with recommendation engines, easing the development of more accurate and efficient approaches.

CCS CONCEPTS

• **Information systems → Recommender systems**.

PUBLICATION INFORMATION

Published in proceedings of the 12th ACM Conference on Recommender Systems (RecSys '18). 2018. Pages 494-495.
DOI:10.1145/3240323.3241611

KEYWORDS

Recommender Systems, Framework, Python

1 INTRODUCTION

A number of frameworks for Recommender Systems (RS) have been proposed by the scientific community, involving different programming languages, such as Java, C#, Python, among others[2]. However, most of them lack an integrated environment containing clustering and ensemble approaches which are capable to improve recommendation accuracy. On the one hand, clustering may support developers to pre-process their data to optimize or extend recommender algorithms. On the other hand, ensemble approaches are efficient tools to combine different types of data in a personalized way. The features available in each considered recommender library are summarized in Table 9.1.

Case Recommender was developed to provide flexibility and extensibility in research environments, while maintaining high performance, providing a variety of recommendation and clustering algorithms, as well as functions for data manipulation [1]. Its primary concern is to maximize usefulness for research and education, instead of large-scale commercial operations. The framework is also designed to support a wide variety of recommendation approaches, including content-based, collaborative filtering and hybrid approaches. Case Recommender is published in PyPi[1], a international repository of software for the Python programming language, as an MIT License, making it easy for third parties to contribute with additional implementations and features.

2 THE FRAMEWORK

Case Recommender is currently being rewritten to support optimized calculations using known Python scientific libraries. We also developed and improved classes that allow developers to manipulate files, predict and evaluate models, compute (dis)similarities between users or items, beside use metadata and external information. The framework is now implemented in *Python 3* and it addresses two common scenarios in Recommender Systems: rating prediction and item recommendation, using explicit, implicit or both types of feedback in several recommender strategies.

The important features while designing our framework were to enable the computation of recommendations in large-scale, easy creation and extension of algorithms for different types of filtering and scenarios. Another feature is to support sparse and large datasets in a way that there is as little as possible overhead for storing data and intermediate results. Developers can choose between using one of the available recommender algorithms, combining multiple recommendations using one of the available ensemble techniques, or develop their own algorithm using the *BaseRatingPrediction* or *BaseItemRecommendation* classes. Up to the current version, the algorithms available in the framework are shown at Table 9.2 and their references can be found in the header of the respective code.

Case Recommender contains a generic interface for recommender systems implementations, among them the collaborative and content-based filtering approaches such as neighborhood-based (NB) and matrix factorization (MF) models, which are already available for use. Additionally, two clustering algorithms are available: PaCo and KMedoids, in order to support recommendation engines. Also, more than 15 different already implemented (dis)similarity metrics can be easily applied to the recommender and clustering interfaces, like cosine and Pearson, using Scipy basic optimized functions[2]. Moreover, it allows easy integration with different input domains like databases, text files or python libraries.

2.1 Recommender Engines

Case Recommender contains a recommender engine composed of several algorithms described in the literature, such as clustering, neighborhood and matrix factorization models. The framework provides an assortment of classes and methods that may be plugged together and customized to create and evaluate an ideal recommender for a particular purpose. Our framework also provides an

[1]https://pypi.org/project/CaseRecommender/

[2]http://docs.scipy.org

Table 9. 1. Comparison of free/open source recommender system toolkits.

Toolkits	Case Recommender	EasyRec	Mahout	LensKit	RiVal	MyMediaLite	Crab
Version	1.0.19	1.0.4	0.13.0	2.2.1	0.2	3.11	0.1
Last update	Jun 2018	May 2016	May 2018	Nov 2015	Aug 2015	Dec 2015	Jan 2012
Actively Developed	✓	-	✓	✓	-	✓	-
License	MIT	GPL 3	Apache 2.0	LGPL 2	Apache 2.0	GPL 3	BSD
Language	Python	Java	Java	Java	Java	C#	Python
Scalable	✓	-	✓	✓	-	✓	✓
Rating Prediction	✓	✓	✓	✓	✓	✓	✓
Item Recommender	✓	-	-	✓	-	✓	✓
CF Techniques	✓	✓	✓	✓	✓	✓	✓
CBF Techniques	✓	-	-	-	-	✓	-
Ensemble Techniques	✓	-	-	-	-	-	-
Clustering Techniques	✓	-	-	-	-	-	-

Table 9. 2. Case Recommender Implemented Algorithms.

Approach	**Item Recommender**	**Rating Prediction**
Neighborhood-Based	User KNN, User Attr KNN, Item KNN, Item Attr KNN, Content Based	User KNN, User Attr KNN, Item KNN, Item Attr KNN
Matrix Factorization	BPR MF	MF, Item-MFMS, SVD, SVD++, GSVD++, Item NSVD1, User NSVD1
Non-Personalized	Most Popular, Random	Most Popular, Random
Ensemble	BPR Learning, Tag Based, Average Based	-
Clustering-based	PaCo, Group-based	-

extensible platform for ensemble and clustering approaches which allow developers to combine different recommender algorithm using different input data. Such ensemble approaches, as they are integrated in the framework, allowing their evaluation to be fair comparisons against individual filtering approaches. Its flexibility allows developers to build architectures for better accuracy depending on the application scenario. Our implementation provides a clear interface, where the ensemble and clustering strategies may receive the results of previous probe runs, and the set of results from previous trained recommenders or other pre-processing techniques.

2.2 Validation and Evaluation Metrics

Case Recommender contains routines for recommendation evaluation, including measures such as RMSE and MAE for the rating prediction task. For the item recommendation task, the available metrics are prec@N, recall@N, MAP@N and NDCG@N. For both scenarios, internal K-fold cross-validation and Shuffle Sample is supported to validate the experiments. The framework also contains the All But One protocol and statistical tests, such as Wilcoxon signed-rank test and the two-sided paired t-test, in order to compare and evaluate the results.

3 INSTRUCTIONS AND DEMONSTRATION

Case Recommender is hosted at Github and PyPi repositories and it is freely available for developers and the scientific community to use, test and contribute under MIT license. To install Case Recommender on Mac OS X, Windows or Linux, the following command will work: *pip install caserecommender* .

Case Recommender comes with instructions that allow users to train and evaluate all available recommenders on data provided in text files. When a new recommender is added into the framework, it is automatically detected, so developers do not have to manually add new features into the programs, only use the command *import* in Python. The main instructions and demonstration of use are available at **Github**[3] and **YouTube**[4], respectively.

4 FINAL REMARKS

The goal of Case Recommender is to integrate and facilitate the experiments and development of new recommender techniques for different domains. Our framework contains a recommender engine, that contains content-based, collaborative and hybrid filtering approaches for rating prediction and item recommendation scenarios. In addition, the framework contains ensemble and clustering algorithms, validation and evaluation metrics to improve and measure the quality of the recommendation.

Future releases are planned which will include more features (parallel processing techniques and new algorithms) and an evaluation tool with several plots and graphs to help developers to better understand the behavior of their recommender algorithms.

ACKNOWLEDGMENTS

We would like to acknowledge CAPES and FAPESP (2016/20280-6) for the financial support.

REFERENCES

[1] Arthur F da Costa and Marcelo G Manzato. 2016. Case Recommender: A Recommender Framework. *In: Workshop of Tools and Applications (WFA), 16, 2016. XXII WebMedia. , 2016. v. 2.*.

[2] Zeno Gantner, Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2011. MyMediaLite: A Free Recommender System Library. In *Proceedings of the Fifth ACM Conference on Recommender Systems (RecSys '11)*. ACM, New York, NY, USA, 305–308. https://doi.org/10.1145/2043932.2043989

[3] https://github.com/caserec2018/caserecommender

[4] www.youtube.com/watch?v=NHCQQD3ZSNA

9.2 Final remarks

In this chapter, we report the latest version of our framework, published in ACM Recommender Systems Conference[1], the main conference in our research area. Towards integrating the pre-processing techniques developed during my doctoral research, we changed many features in the original framework, such as classes structures, Input/Output (I/O) strategies, Python version, distribution license and others. In addition, we implemented new recommender algorithms, evaluation and statistical metrics and last but not least the clustering and data mining approaches to pre-processing input information.

This work had the participation of the recommender systems research group at ICMC/USP, coordinated by professor Marcelo Manzato. The participants added their recent published algorithms in the framework and helped in the documentation and writing. Meanwhile, other contributions were made by the community in the Github[2] and during presentations in which the framework was exposed (e.g. Webmedia[3], Python Sudeste[4], SciPyLA[5] and RecSys).

Finally, in the next chapter we present the final remark of results, highlighting the main results obtained and how we reached the proposed objectives.

[1] https://recsys.acm.org/
[2] https://github.com/caserec/CaseRecommender
[3] https://www6.ifpi.edu.br/webmedia/
[4] https://2018.pythonsudeste.org/
[5] http://scipyla.org/es/

CHAPTER

10

CONCLUSIONS AND FINAL REMARKS

In this thesis, we proposed offline pre-processing approaches that aim to deal with traditional recommender systems problems in order to improve the quality of input data for the algorithms, leading to faster and more precise suggestions in the rating prediction and top-*k* recommendation scenarios. In order to accomplish these objectives, differently from other related works (VLACHOS *et al.*, 2014; ZHANG *et al.*, 2014; GUPTA; PATIL, 2015; ZHANG *et al.*, 2016a; XIAOJUN, 2017), we apply different pre-processing and data filtering techniques to obtain concise and semantic user and item descriptions before the recommendation step, improving only the input data instead of the recommender algorithms. We used the available information to perform clustering, statistical, data mining and matching solutions that are responsible to reduce the number of unknown items a recommender must process for a user.

The main research question in this thesis was: "How can we improve the accuracy of the existing recommender algorithms using pre-processing approaches?". Our hypothesis was that enriching the input data by adding extra information and removing useless information will improve the recommenders, since they will work with more data. Throughout the presented works, we were able to prove our hypothesis, presenting superior results to the baselines, as well as showing the efficiency of the proposed approaches to solve problems such as high dimensionality, sparsity and cold-start.

We also investigated two other related research questions. First we investigated: "How can we integrate different types of user's feedback to improve the quality of my input data?". The hypothesis was that representing the user's profile as an enriched vector generated by machine learning techniques combining the different feedback types, it would allow the recommender to improve its accuracy, since the input data are more representative. In Chapters 2, 3 and 4, we demonstrated that once the input data are based on the combination of users' feedback, they are responsible for the increase of the accuracy of the recommendations, independently of the technique adopted (e.g. clustering, statistical or heuristic). The results obtained in these works showed that when using different types of users' feedback, when available, it considerably

improves the results of the predictors of the recommenders when comparing the same process with the non-enriched data (e.g. using the original user-item feedback matrix).

Second, we investigated the question "How can we treat the input data to alleviate traditional recommender systems problems, such as cold-start, dimensionality and sparsity?". Our hypothesis was that increasing and enriching the number of input data for recommender systems, we could alleviate the cold-start and sparsity problems. Indeed, as we reported in Chapters 6 and 7, this is possible. In this work, we proposed a co-training algorithm to enrich the original user-item feedback matrix with more precise and reliable data. In addition, we reported in Chapter 5 a two way clustering approach for recommender systems to deal with dimensionality problem, since we used only similar users and items to generate recommendation. Our results showed that the use of the proposed pre-processing approaches performed better than using traditional algorithms and baselines, as well as these approaches have been able to deal and/or smooth recommendation problems efficiently.

Finally, we presented a recommender systems framework developed during this research, that contains the pre-processing steps and recommendation algorithms and allows both to be integrated in a simple and fast way. This may facilitate the development and use of these techniques by other developers and researchers. Case Recommender was presented in the Chapters 8 and 9, highlighting the several types of recommender algorithms, pre-process techniques and validation and evaluation metrics.

The adoption of pre-processing techniques in recommender systems not only improves data representation but also the modeling of the knowledge contained in it, as well as alleviates traditional problems in the recommender systems area.

10.1 Limitations

Although our proposed approaches performs better than the well-known and state-of-the-art baselines in most cases, our approaches have some limitations related to data, performance and evaluation, which will be listed below.

- Most available datasets have a limited type of feedback. Usually datasets available only contain ratings or history of navigation;
- Clustering approaches may compromise the diversity of recommendations, since it limits the type of items to be recommended for similarity. For example, users who like action movies and romance, but are only in a group of users who only have action films, never received romance movies;
- We were not able to perform an online evaluation of our model (e.g. A/B testing). An online evaluation would be necessary and highly recommended to measure the real impact

of this recommender system on the number of products sold/ interacted (AGGARWAL, 2016; SAHU; DWIVEDI, 2019).

10.2 Future Work

There are a number of interesting avenues of future work that can emerge from our research, as described in the following:

First, this thesis has been mainly focused on pre-processing input data for recommender algorithms using heuristics, data mining and data clustering approaches. We proposed enriched user profiles based on statistical metrics and with clustering approaches in order to reduce the dimensionality of the original representation of the users. In this context, we aim to use features extraction and dimensionality reduction tools, such as Latent semantic indexing (LSI), Principal Component Analysis (PCA) and Singular Value Decomposition (SVD) to automatically generate these profiles keeping the characteristics of the original representation.

Furthermore, we believe that the dimensionality reduction presented in the clustering based approaches could be optimized including (i) the option to aid the new item problem, since the proposed approaches do not consider items that had no interaction. This could help to optimize and increase the diversity and the quality of recommendations; (ii) the option to generate soft clusters in order to allow a user to appear in more than one group or that there may be overlaps among the generated groups; (iii) use graph-based algorithms to view and explore the relationship between the users, using the different types of feedback as weights for edges; (iv) investigate the clustering approaches in the items dimension in order to filter and further reduce the search space during the recommendation process.

Finally, new approaches have been proposed in the last years to generate recommendations using deep learning (WEI *et al.*, 2017; KARATZOGLOU; HIDASI, 2017; LEE; LEE, 2018; Mu, 2018), dealing in a simple way with high dimensionality and sparsity problems (ZHANG; YAO; SUN, 2018). We also intend to follow this research line using deep learning for pre-processing steps and also to generate recommendations, since one of our main challenges is to train accurate machine learning models on very large and sparse datasets in a reasonable amount of time. In turn, deep learning approaches require a lot of training data to work well, because of the huge number of parameters needed to be tuned by a learning algorithm.

10.3 Publications

The research described in this thesis has lead to a total of 7 publications consisting of 2 journal articles and 5 conference papers. At the start of each chapter in this thesis, each article was listed. Meanwhile, I also have contributed as an author and a co-author to 6 publications

related with my research theme, including 2 journal articles and 4 conference publications. These works are listed in chronological order below:

- Introducing the concept of always-welcome recommendations. Dos Santos, Edson B.; Da Costa, Arthur F.; D'addio, Rafael M.; Manzato, Marcelo G.; Goularte, Rudinei - In: IEEE International Conference on Computer and Information Science (ICIS), 2015. *(Conference paper)*

- Mining unstructured content for recommender systems: an ensemble approach. Manzato, Marcelo G.; Domingues, Marcos A.; Da Costa, Arthur F.; Sundermann, Camila V.; D'Addio, Rafael M.; Conrado, Merley S.; Rezende, Solange O.; Pimentel, Maria G. C. - Information Retrieval Journal, Springer, 2016. *(Journal article)*

- Incorporating Semantic Item Representations to Soften the Cold Start Problem. D'Addio, R. M.; Fressato, E. P.; Da Costa, Arthur F.; Manzato, Marcelo G. - In: ACM Brazilian Symposium on Multimedia and the Web (WebMedia), 2018. (***Best Paper***) *(Conference paper)*

- Similarity-based Matrix Factorization for Item Cold-Start in Recommender Systems. Fressato, E. P.; Da Costa, Arthur F.; Manzato, Marcelo G. - In: IEEE Brazilian Conference on Intelligent Systems (BRACIS), 2018. *(Conference paper)*

- CoBaR: Confidence-Based Recommender. Aguiar Neto, F. S.; Da Costa, Arthur F.; Manzato, Marcelo G. - In: ACM Recommender Systems Conference (RecSys), 2018. *(Conference paper)*

- Enhancing Spatial Keyword Preference Query with Linked Open Data. Almeida, J. P. D. ; Durao, F. A. ; Da Costa, Arthur F. - Journal Of Universal Computer Science (JUCS), 2019. *(Journal article)*

- A personalized clustering-based approach using open linked data for search space reduction in recommender systems. Da Costa, Arthur F.; D'Addio, R. M.; Fressato, E. P.; Manzato, Marcelo G. - In: ACM Brazilian Symposium on Multimedia and the Web (WebMedia), 2019. *(Conference paper)*

BIBLIOGRAPHY

AGGARWAL, C. C. **Recommender Systems: The Textbook**. 1st. ed. Springer Publishing Company, Incorporated, 2016. ISBN 3319296574, 9783319296579. Available: <http://www.springer.com/us/book/9783319296579>. Citations on pages 27, 28, 29, 33, 39, 67, 69, 70, 75, 84, 85, 87, and 153.

BATULE, R. M.; ITKAR, S. A. A survey paper on different clustering techniques for collaborative filtering for services recommendation. In: . [S.l.: s.n.], 2016. Citation on page 70.

BOBADILLA, J.; ORTEGA, F.; HERNANDO, A.; GUTIÉRREZ, A. Recommender systems survey. **Knowledge-Based Systems. http://dx.doi.org/10.1016/j.knosys.2013.03.012**, Elsevier Science Publishers B. V., Amsterdam, The Netherlands, The Netherlands, v. 46, p. 109–132, Jul. 2013. ISSN 0950-7051. Available: <http://dx.doi.org/10.1016/j.knosys.2013.03.012>. Citations on pages 28, 29, 69, 70, 85, and 87.

CAMPELLO, R. J. G. B.; MOULAVI, D.; ZIMEK, A.; SANDER, J. A framework for semi-supervised and unsupervised optimal extraction of clusters from hierarchies. **Data Mining and Knowledge Discovery**, v. 27, n. 3, p. 344–371, Nov 2013. ISSN 1573-756X. Available: <https://doi.org/10.1007/s10618-013-0311-4>. Citation on page 73.

CANTADOR, I.; BRUSILOVSKY, P.; KUFLIK, T. 2nd workshop on information heterogeneity and fusion in recommender systems (hetrec 2011). In: . New York, NY, USA: ACM, 2011. (RecSys 2011), p. 387–388. ISBN 978-1-4503-1027-7. Available: <http://doi.acm.org/10.1145/2043932.2044016>. Citations on pages 78 and 79.

CHANDRA, B.; SHANKER, S.; MISHRA, S. A new approach: Interrelated two-way clustering of gene expression data. **Statistical Methodology**, v. 3, n. 1, p. 93 – 102, 2006. ISSN 1572-3127. Bioinformatics. Available: <http://www.sciencedirect.com/science/article/pii/S1572312705000699>. Citation on page 71.

CHARKHABI, M.; DHOT, T.; MOJARAD, S. A. Cluster ensembles, majority vote, voter eligibility and privileged voters. **International Journal of Machine Learning and Computing**, v. 4, p. 275–278, 06 2014. Citation on page 49.

COSTA, A. da; FRESSATO, E.; NETO, F.; MANZATO, M.; CAMPELLO, R. Case recommender: A flexible and extensible python framework for recommender systems. In: **Proceedings of the 12th ACM Conference on Recommender Systems**. New York, NY, USA: ACM, 2018. (RecSys '18), p. 494–495. ISBN 978-1-4503-5901-6. Available: <http://doi.acm.org/10.1145/3240323.3241611>. Citation on page 32.

COSTA, A. F. da; MANZATO, M. G.; CAMPELLO, R. J. Group-based collaborative filtering supported by multiple users' feedback to improve personalized ranking. In: **Proceedings of the 22Nd Brazilian Symposium on Multimedia and the Web**. New York, NY, USA: ACM, 2016. (Webmedia '16), p. 279–286. ISBN 978-1-4503-4512-5. Available: <http://doi.acm.org/10.1145/2976796.2976852>. Citations on pages 32 and 70.

____. Boosting collaborative filtering with an ensemble of co-trained recommenders. **Journal of Information and Data Management**, v. 8, p. 180 – 196, 2017. ISSN 2178-7107. Available: <https://seer.ufmg.br/index.php/jidm/article/view/4560>. Citations on pages 32, 70, and 71.

____. Boosting collaborative filtering with an ensemble of co-trained recommenders. **Expert Systems with Applications**, v. 115, p. 427 – 441, 2019. ISSN 0957-4174. Available: <http://www.sciencedirect.com/science/article/pii/S0957417418305281>. Citation on page 32.

COSTA, A. F. da; MANZATO, M. G.; CAMPELLO, R. J. G. B. Corec: A co-training approach for recommender systems. In: **Proceedings of the 33rd Annual ACM Symposium on Applied Computing**. New York, NY, USA: ACM, 2018. (SAC '18), p. 696–703. ISBN 978-1-4503-5191-1. Available: <http://doi.acm.org/10.1145/3167132.3167209>. Citation on page 32.

COSTA, A. F. da; MARTINS, R. D.; MANZATO, M. G.; CAMPELLO, R. J. G. B. Exploiting different users' interactions for profiles enrichment in recommender systems. In: **Proceedings of the 31st Annual ACM Symposium on Applied Computing**. New York, NY, USA: ACM, 2016. (SAC '16), p. 1080–1082. ISBN 978-1-4503-3739-7. Available: <http://doi.acm.org/10.1145/2851613.2851923>. Citations on pages 32 and 138.

CRNOVRSANIN, T.; MUELDER, C. W.; FARIS, R.; FELMLEE, D.; MA, K.-L. Visualization techniques for categorical analysis of social networks with multiple edge sets. **Social Networks**, v. 37, p. 56–64, 2014. Citation on page 39.

GAN, G.; MA, C.; WU, J. **Data clustering - theory, algorithms, and applications**. [S.l.]: SIAM, 2007. I-XXII, 1-466 p. Citation on page 39.

GUO, G.; ZHANG, J.; THALMANN, D.; YORKE-SMITH, N. Etaf: An extended trust antecedents framework for trust prediction. In: **Proceedings of the 2014 International Conference on Advances in Social Networks Analysis and Mining**. [s.n.], 2014. (ASONAM '14), p. 540–547. Available: <https://doi.org/10.1109/ASONAM.2014.6921639>. Citation on page 78.

GUO, G.; ZHANG, J.; YORKE-SMITH, N. A novel bayesian similarity measure for recommender systems. In: **Proceedings of the 23rd International Joint Conference on Artificial Intelligence**. [s.n.], 2013. (IJCAI '13), p. 2619–2625. Available: <http://dl.acm.org/citation.cfm?id=2540128.2540506>. Citation on page 78.

GUPTA, U.; PATIL, N. Recommender system based on hierarchical clustering algorithm chameleon. In: **2015 IEEE International Advance Computing Conference (IACC)**. [s.n.], 2015. p. 1006–1010. Available: <http://dx.doi.org/10.1109/IADCC.2015.7154856>. Citations on pages 28 and 151.

HOVALE, S. V.; GHULI, P. Survey paper on recommendation system using data mining techniques. **International Journal Of Engineering And Computer Science**, 05 2016. Citation on page 70.

JAHRER, M.; TöSCHER, A.; LEGENSTEIN, R. Combining predictions for accurate recommender systems. In: **Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining**. New York, NY, USA: ACM, 2010. (KDD '10), p. 693–702. Citation on page 97.

KARATZOGLOU, A.; HIDASI, B. Deep learning for recommender systems. In: **Proceedings of the Eleventh ACM Conference on Recommender Systems**. New York, NY, USA: ACM,

2017. (RecSys '17), p. 396–397. ISBN 978-1-4503-4652-8. Available: <http://doi.acm.org/10.1145/3109859.3109933>. Citation on page 153.

KATARYA, R.; VERMA, O. P. An effective collaborative movie recommender system with cuckoo search. **Egyptian Informatics Journal**, v. 18, n. 2, p. 105 – 112, 2017. ISSN 1110-8665. Available: <http://www.sciencedirect.com/science/article/pii/S1110866516300470>. Citation on page 28.

KUMAR, S.; , V. . Survey on personalized web recommender system. **International Journal of Information Engineering and Electronic Business**, v. 10, p. 33–40, 07 2018. Citations on pages 27, 29, 84, and 87.

LANGFORD, J. The cross validation problem. In: AUER, P.; MEIR, R. (Ed.). **Learning Theory**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005. p. 687–688. ISBN 978-3-540-31892-7. Citation on page 79.

LEE, H.; LEE, J. Scalable deep learning-based recommendation systems. **ICT Express**, 2018. ISSN 2405-9595. Available: <http://www.sciencedirect.com/science/article/pii/S2405959518302029>. Citation on page 153.

MANNING, C. D.; RAGHAVAN, P.; SCHüTZE, H. Evaluation in information retrieval. In: ____. **Introduction to Information Retrieval**. Cambridge University Press, 2008. p. 139–161. Available: <http://dx.doi.org/10.1017/CBO9780511809071.009>. Citation on page 79.

MCAULEY, J.; PANDEY, R.; LESKOVEC, J. Inferring networks of substitutable and complementary products. In: **Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining**. New York, NY, USA: ACM, 2015. (KDD '15), p. 785–794. ISBN 978-1-4503-3664-2. Available: <http://doi.acm.org/10.1145/2783258.2783381>. Citation on page 79.

Mu, R. A survey of recommender systems based on deep learning. **IEEE Access**, v. 6, p. 69009–69022, 2018. ISSN 2169-3536. Citation on page 153.

NADEE, W. **Modelling user profiles for recommender systems**. Phd Thesis (PhD Thesis) — Queensland University of Technology, 2016. Available: <https://eprints.qut.edu.au/93723/>. Citation on page 33.

NETO, F. S. de A.; MANZATO, M. G.; CAMPELLO, R. J. G. B. **Pre-processing approaches for collaborative filtering based on hierarchical clustering**. 2018. Citations on pages 70, 72, 73, 74, 75, 80, 81, and 82.

RAMEZANI, M.; MORADI, P.; TAB, F. A. Improve performance of collaborative filtering systems using backward feature selection. In: **The 5th Conference on Information and Knowledge Technology**. [s.n.], 2013. p. 225–230. Available: <https://doi.org/10.1109/IKT.2013.6620069>. Citations on pages 70 and 71.

RENDLE, S.; FREUDENTHALER, C.; GANTNER, Z.; SCHMIDT-THIEME, L. Bpr: Bayesian personalized ranking from implicit feedback. In: **Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence**. Arlington, Virginia, United States: AUAI Press, 2009. (UAI '09), p. 452–461. ISBN 978-0-9749039-5-8. Available: <http://dl.acm.org/citation.cfm?id=1795114.1795167>. Citations on pages 27 and 80.

RICCI, F.; ROKACH, L.; SHAPIRA, B. Recommender systems: Introduction and challenges. In: ____. **Recommender Systems Handbook**. Boston, MA: Springer US, 2015. p. 1–34. Citations on pages 27, 28, 67, 70, and 75.

RISTOSKI, P.; MENCíA, E. L.; PAULHEIM, H. A hybrid multi-strategy recommender system using linked open data. In: PRESUTTI, V.; STANKOVIC, M.; CAMBRIA, E.; CANTADOR, I.; IORIO, A. D.; NOIA, T. D.; LANGE, C.; RECUPERO, D. R.; TORDAI, A. (Ed.). **Semantic Web Evaluation Challenge**. [S.l.]: Springer International Publishing, 2014, (Communications in Computer and Information Science, v. 475). p. 150–156. Citation on page 97.

SAHU, A.; DWIVEDI, P. User profile as a bridge in cross-domain recommender systems for sparsity reduction. **Applied Intelligence**, p. 1–21, 01 2019. Citations on pages 33, 70, and 153.

SETTLES, B. **Active learning literature survey**. [S.l.], 2010. Citation on page 85.

SOMBOONVIWAT, K.; AOYAMA, H. Empirical analysis of the relationship between trust and ratings in recommender systems. In: NGUYEN, N. T.; TRAWIŃSKI, B.; FUJITA, H.; HONG, T.-P. (Ed.). **Intelligent Information and Database Systems: 8th Asian Conference**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2016. (ACIIDS '16), p. 116–126. ISBN 978-3-662-49381-6. Available: <https://doi.org/10.1007/978-3-662-49381-6_12>. Citation on page 76.

STREHL, A.; GHOSH, J. Cluster ensembles — a knowledge reuse framework for combining multiple partitions. **J. Mach. Learn. Res.**, JMLR.org, v. 3, p. 583–617, Mar. 2003. ISSN 1532-4435. Citation on page 49.

VINAGRE, J.; JORGE, A.; GAMA, J. Online bagging for recommender systems. **Expert Systems**, p. 1–13, 07 2018. Citation on page 97.

VLACHOS, M.; FUSCO, F.; MAVROFORAKIS, C.; KYRILLIDIS, A.; VASSILIADIS, V. G. Improving co-cluster quality with application to product recommendations. In: **Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management**. New York, NY, USA: ACM, 2014. (CIKM '14), p. 679–688. ISBN 978-1-4503-2598-1. Available: <http://doi.acm.org/10.1145/2661829.2661980>. Citations on pages 28, 70, 72, 73, 81, 138, and 151.

WEI, J.; HE, J.; CHEN, K.; ZHOU, Y.; TANG, Z. Collaborative filtering and deep learning based recommendation system for cold start items. **Expert Systems with Applications**, v. 69, p. 29 – 39, 2017. ISSN 0957-4174. Available: <http://www.sciencedirect.com/science/article/pii/S0957417416305309>. Citation on page 153.

WILCOXON, F. Individual comparisons by ranking methods. **Biometrics bulletin**, JSTOR, v. 1, n. 6, p. 80–83, 1945. Citation on page 79.

WU, M.-L.; CHANG, C.-H.; LIU, R.-Z.; FAN, T.-K. Aggregate two-way co-clustering of ads and user data for online advertisements. **J. Inf. Sci. Eng.**, v. 28, p. 83–97, 01 2012. Citation on page 71.

WU, Y.; LIU, X.; XIE, M.; ESTER, M.; YANG, Q. Cccf: Improving collaborative filtering via scalable user-item co-clustering. In: **Proceedings of the Ninth ACM International Conference on Web Search and Data Mining**. New York, NY, USA: ACM, 2016. (WSDM '16), p. 73–82. ISBN 978-1-4503-3716-8. Available: <http://doi.acm.org/10.1145/2835776.2835836>. Citations on pages 70, 72, 73, 79, 81, and 84.

XIAOJUN, L. An improved clustering-based collaborative filtering recommendation algorithm. **Cluster Computing**, v. 20, n. 2, p. 1281–1288, Jun 2017. ISSN 1573-7543. Available: <https://doi.org/10.1007/s10586-017-0807-6>. Citations on pages 28, 72, and 151.

XU, B.; BU, J.; CHEN, C.; CAI, D. An exploration of improving collaborative recommender systems via user-item subgroups. In: **Proceedings of the 21st International Conference on World Wide Web**. New York, NY, USA: ACM, 2012. (WWW '12), p. 21–30. ISBN 978-1-4503-1229-5. Available: <http://doi.acm.org/10.1145/2187836.2187840>. Citations on pages 70, 72, and 84.

YU, W.; QIN, Z. Spectrum-enhanced pairwise learning to rank. In: **The World Wide Web Conference**. New York, NY, USA: ACM, 2019. (WWW '19), p. 2247–2257. ISBN 978-1-4503-6674-8. Available: <http://doi.acm.org/10.1145/3308558.3313478>. Citation on page 79.

ZHANG, J.; LIN, Y.; LIN, M.; LIU, J. An effective collaborative filtering algorithm based on user preference clustering. **Applied Intelligence**, v. 45, n. 2, p. 230–240, Sep 2016. ISSN 1573-7497. Available: <https://doi.org/10.1007/s10489-015-0756-9>. Citations on pages 28 and 151.

____. An effective collaborative filtering algorithm based on user preference clustering. **Applied Intelligence**, v. 45, n. 2, p. 230–240, Sep 2016. ISSN 1573-7497. Available: <https://doi.org/10.1007/s10489-015-0756-9>. Citation on page 72.

ZHANG, M.; TANG, J.; ZHANG, X.; XUE, X. Addressing cold start in recommender systems: A semi-supervised co-training algorithm. In: **Proceedings of the 37th International ACM SIGIR Conference on Research and Development in Information Retrieval**. New York, NY, USA: ACM, 2014. (SIGIR '14), p. 73–82. ISBN 978-1-4503-2257-7. Available: <http://doi.acm.org/10.1145/2600428.2609599>. Citations on pages 28, 97, 138, and 151.

ZHANG, Q.; WANG, H. Collaborative multi-view learning with active discriminative prior for recommendation. In: **Advances in Knowledge Discovery and Data Mining: 19th Pacific-Asia Conference, PAKDD 2015, Ho Chi Minh City, Vietnam, May 19-22, 2015, Proceedings, Part I**. Cham: Springer International Publishing, 2015. p. 355–368. ISBN 978-3-319-18038-0. Available: <https://doi.org/10.1007/978-3-319-18038-0_28>. Citations on pages 28 and 97.

ZHANG, S.; YAO, L.; SUN, A. Deep learning based recommender system: A survey and new perspectives. **CoRR**, abs/1707.07435, 2018. Available: <http://arxiv.org/abs/1707.07435>. Citation on page 153.

ZHANG, W.; WANG, J. Prior-based dual additive latent dirichlet allocation for user-item connected documents. In: **Proceedings of the 24th International Conference on Artificial Intelligence**. AAAI Press, 2015. (IJCAI'15), p. 1405–1411. ISBN 978-1-57735-738-4. Available: <http://dl.acm.org/citation.cfm?id=2832415.2832445>. Citation on page 79.

ZIEGLER, C.-N.; MCNEE, S. M.; KONSTAN, J. A.; LAUSEN, G. Improving recommendation lists through topic diversification. In: **Proceedings of the 14th International Conference on World Wide Web**. New York, NY, USA: ACM, 2005. (WWW '05), p. 22–32. ISBN 1-59593-046-9. Available: <http://doi.acm.org/10.1145/1060745.1060754>. Citation on page 78.


ICMC
USP
SÃO CARLOS