UNIVERSIDADE DE LISBOA

INSTITUTO SUPERIOR TÉCNICO

Leveraging Existing Technologies to Improve Large-Scale Recommender Systems

André Filipe Caldaça da Silva Carvalho

Supervisor: Doctor Pável Pereira Calado
Co-Supervisor: Doctor João Paulo Carvalho

Thesis approved in public session to obtain the PhD Degree in
Information Systems and Computer Engineering

Jury final classification: Pass with Distinction

2019

UNIVERSIDADE DE LISBOA

INSTITUTO SUPERIOR TÉCNICO

Leveraging Existing Technologies to Improve Large-Scale Recommender Systems

André Filipe Caldaça da Silva Carvalho

Supervisor: Doctor Pável Pereira Calado
Co-Supervisor: Doctor João Paulo Carvalho

Thesis approved in public session to obtain the PhD Degree in
Information Systems and Computer Engineering

Jury final classification: Pass with Distinction

Jury

Chairperson: Doctor Mário Jorge Costa Gaspar da Silva, Instituto Superior Técnico, Universidade de Lisboa
Members of the Committee:
Doctor Alípio Mário Guedes Jorge, Faculdade de Ciências, Universidade do Porto
Doctor João Miguel da Costa Magalhães, Faculdade de Ciências e Tecnologia, Universidade Nova de Lisboa
Doctor Pável Pereira Calado, Instituto Superior Técnico, Universidade de Lisboa
Doctor David Manuel Martins de Matos, Instituto Superior Técnico, Universidade de Lisboa

Funding Institutions:

INESC-ID
Universidade de Lisboa

2019

Title Leveraging Existing Technologies to Improve Large-Scale Recommender Systems

Abstract

Recommender Systems have as goal providing valuable recommendations to their users. Most research on Recommender Systems aims to improve the recommendation quality exclusively, while overlooking the computational efficiency of such solutions. Although Recommender Systems, based on collaborative filtering, do not have many ratings available, in this work by strategically removing redundant ratings it is possible to offer a similarity metric, that improves the computational efficiency of Recommender Systems.

This work focus on improving the computational efficiency of similarity metrics and enhance quality, using two different approaches. The first relies on Collaborative Filtering, i.e., exclusively on ratings, to produce a computationally efficient similarity metric for Recommender Systems. The second approach uses contextual information regarding users and items to further improve recommendation quality, while still maintaining the same computational efficiency. The solutions here proposed can be readily deployed on real Recommender Systems.

The first approach methodology intends to improve similarity metrics for Memory-based Collaborative Filtering using Fuzzy Models. Memory-based Collaborative Filtering relies heavily on similarity metrics to produce recommendations. Fuzzy Fingerprints are used as a framework to create a novel similarity metric, providing a fast and effective solution. The second approach also uses Fuzzy Fingerprints, and it combines contextual information with ratings into a single Fuzzy Fingerprint, or create a multi-context Fuzzy Fingerprint where each contextual information has its own Fuzzy Fingerprint. Each contextual Fuzzy Fingerprint allows a ranking fusion algorithm to produce similarity values. This work is validated using four well-known datasets which are ML-1M, HetRec 2011, Netflix and Jester.

The application of the Fuzzy Fingerprint similarity improves recommendation quality but, more importantly, requires four times less computational resources than current solutions on large datasets. When using contextual information, the recommendation quality improves either by combining contextual information and ratings into a single Fuzzy Fingerprint or by using multi-context Fuzzy Fingerprints. The solutions using contextual information achieve further recommendation quality improvements while maintaining a comparable computational efficiency in comparison with well-known similarity metrics.

Keywords: Recommender Systems, Memory-based Collaborative Filtering, Similarity metric, Fuzzy Fingerprint, Computational complexity.

Título Técnicas adaptadas para Sistemas de Recomendação de larga-escala
Nome André Filipe Caldaça da Silva Carvalho
Doutoramento em Engenharia Informática e de Computadores
Orientador Doutor Pável Pereira Calado
Co-orientador Doutor João Paulo Carvalho

Resumo

Os Sistemas de Recomendação têm como principal objetivo gerar recomendações com interesse para o utilizador. A maioria da investigação em torno de Sistemas de Recomendação tenta exclusivamente melhorar a qualidade das recomendações, ignorando a sua complexidade computacional. Embora os Sistemas de Recomendação, usando filtragem colaborativa baseada em memória, não tenham *ratings* em abundância, devido a esparsidade, é possível ainda estrategicamente remover alguns *ratings* redundantes conseguindo-se melhorar a eficiência computacional das métricas de similaridade.

Este trabalho foca-se em melhorar a eficiência computacional e ao mesmo tempo melhorar a qualidade de recomendação. Duas abordagens são apresentadas: a primeira utilizando filtragem colaborativa, i.e. exclusivamente ratings, para produzir uma métrica de similaridade mais eficiente computacionalmente; a segunda abordagem, usa informação contextual relativamente aos utilizadores e itens, conseguindo melhor qualidade de recomendação e mantendo ainda a mesma complexidade computacional que as métricas de similaridade tradicionais. A eficiência computacional é um requisito deste trabalho, pois o objetivo é desenvolver um Sistema de Recomendação capaz de integrar a aplicação em um cenário real e não ser apenas aplicável a nível acadêmico.

As abordagens desenvolvidas procuram melhorar a métrica de similaridade para Sistemas de Recomendação de filtragem colaborativa baseada em memória, através de modelos fuzificados. A sua qualidade depende muito da métrica de similaridade usada. Propõe-se o uso de *Fuzzy Fingerprints* para criar uma métrica de similaridade capaz de produzir uma solução de rápida integração e eficiente para Sistemas de Recomendação. A segunda abordagem também utiliza *Fuzzy Fingerprint similarity*, quer seja por combinar informação contextual e ratings numa só *Fuzzy Fingerprint* quer seja por criar uma multi-contextual *Fuzzy Fingerprint similarity* utilizando algoritmos de *ranking fusion*. Este trabalho é validado usando quatro populares *datasets* designados por ML-1M, HetRec 2011, Netflix e Jester.

Os resultados mostram que as *Fuzzy Fingerprints* conseguem melhorar ligeiramente a qualidade de recomendação mas mostram principalmente a sua capacidade de melhorar a complexidade computacional, que é quarto vezes menor face as outras métricas de similaridade testadas. O uso de informação contextual permite melhorar a qualidade de recomendação quer seja combinando informação contextual e *ratings* em uma só *Fuzzy Fingerprint*, ou usando multi-contextual *Fuzzy Fingerprints* criando uma *Fuzzy Fingerprint* por cada fonte de informação contextual. Estas duas soluções, usando informação contextual, conseguem melhorias na qualidade de recomendação e a sua eficiência computacional é ainda assim comparável com outras métricas usadas pela comunidade científica.

Palavras-Chave: Sistemas de Recomendação, Filtragem colaborativa baseada em memória, Métrica de Similaridade, *Fuzzy Fingerprint*, Complexidade computacional.

Titulo Técnicas adaptadas para Sistemas de Recomendação de larga-escala

Resumo Alargado em Português

Os Sistemas de Recomendação têm como principal objetivo gerar recomendações com interesse para o utilizador. A maioria da investigação em torno de Sistemas de Recomendação tenta exclusivamente melhorar a qualidade das recomendações, ignorando a complexidade computacional das soluções desenvolvidas. Embora os Sistemas de Recomendação, usando filtragem colaborativa baseada em memória não tenham *ratings* em abundância, devido a esparsidade, é possível ainda estrategicamente remover alguns *ratings* redundantes conseguindo-se melhorar a eficiência computacional das métricas de similaridade.

Este trabalho foca-se em melhorar a eficiência computacional e, ao mesmo tempo melhorar, a qualidade de recomendação. Duas abordagens são apresentadas para este efeito: a primeira utilizando exclusivamente filtragem colaborativa, i.e. exclusivamente ratings, para produzir uma métrica de similaridade mais eficiente em termos computacionais; a segunda abordagem utilizando informação contextual relativamente aos utilizadores e itens, conseguindo mais qualidade de recomendação e mantendo ainda a mesma complexidade computacional que as métricas de similaridade tradicionais. A eficiência computacional é um requisito deste trabalho, pois o objetivo é desenvolver um Sistema de Recomendação capaz de integrar um cenário real e não apenas ser aplicável a nível acadêmico.

As metodologias desenvolvidas procuram melhorar a métrica de similaridade para Sistemas de Recomendação de filtragem colaborativa baseada em memória, através de modelos fuzificados. Propõe-se o uso de *Fuzzy Fingerprints* para criar uma nova métrica de similaridade capaz de produzir uma solução de rápida integração e eficiente para Sistemas de Recomendação. A segunda abordagem explorada também utiliza a *Fuzzy Fingerprint similarity* quer seja por combinar informação contextual e ratings numa só *Fuzzy Fingerprint* quer seja por criar uma multi-contextual *Fuzzy Fingerprint similarity* utilizando algoritmos de *ranking fusion*. Este trabalho é validado usando quatro populares *datasets* designados por *ML-1M*, *HetRec 2011*, *Netflix* e *Jester*.

A *Fuzzy Fingerprint similarity* é capaz, em outras áreas, de reduzir a complexidade computacional e garantir qualidade na determinação de similaridades. Este trabalho é o primeiro a aplicar a *Fuzzy Fingerprint similarity* no âmbito de Sistemas de Recomendação. Para criar uma *Fuzzy Fingerprint* é necessário primeiro criar uma *Fingerprint*. Em um Sistema de Recomendação baseado em filtragem colaborativa, as *Fingerprints* são compostas por *ratings* ordenados segundo um dado critério de ordenação. Em este trabalho testaram-se vários critérios de ordenação pois devido às características dos ratings, geralmente em uma escala discreta compreendida entre 1 e 5, não é direta a sua ordenação. Seguindo um dos critérios de ordenação é, então, criada uma lista ordenada de *ratings* para cada um dos itens. A dimensão

desta lista e reduzida a *k ratings* criando-se assim a *Fingerprint*. A cada uma das *Fingerprints* é aplicada uma função de fuzificação criando-se, deste modo, uma *Fuzzy Fingerprint*. As *Fuzzy Fingerprints* são capazes de reduzir a complexidade computacional pois o número de *ratings* que as representam está limitado a um número máximo. Cada *Fuzzy Fingerprint* poderá ter menos elementos que esse número máximo de elementos e, desta forma, itens que tenham poucos *ratings* a descrevê-los não perdem a pouca informação que têm disponível. Com as *Fuzzy Fingerprints* construídas é possível calcular a *Fuzzy Fingerprint similarity* entre dois itens a qual utiliza a *Gödel t-norm* entre duas *Fuzzy Fingerprints*.

Os resultados desta primeira abordagem permitiram garantir uma melhoria na complexidade computacional e em dois *datasets* melhorar ainda a qualidade de recomendação. No *dataset ML-1M* obtém-se um RMSE de 0.8565 e uma redução de 7.8% na complexidade computacional face a métrica de similaridade *Jaccard mean square distance* que tem um RMSE de 0.8670. No *dataset Netflix* obtém-se um RMSE de 0.9490 e uma redução de 64.2% na complexidade computacional face a métrica de similaridade *Pearson correlation* que tem um RMSE de 0.9517. No *dataset Jester* obtém-se um RMSE de 4.0664 e uma redução de 98.4% na complexidade computacional face a métrica de similaridade *Pearson correlation* que tem um RMSE de 4.0419. Estes resultados mostraram que em *datasets* ricos em *ratings* as melhorias na eficiência computacional são mais fáceis de obter permitindo, ainda, uma melhoria na qualidade de recomendação.

A segunda abordagem envolve, para além da filtragem colaborativa, a inclusão também de informação contextual. Esta inclui a descrição dos utilizadores, a descrição dos itens, as keywords associadas aos itens entre outras possíveis fontes de informação. A segunda abordagem foi alvo de várias versões e melhoramentos ao longo desta Tese.

Inicialmente, os ratings foram combinados com a descrição textual dos itens (TF-IDF) durante criação das *Fingerprints* dos itens. Os resultados permitiram verificar que existem possíveis melhorias na qualidade das recomendações produzidas ao introduzir informação contextual, mas a eficiência computacional é sacrificada. Os testes conduzidos no *dataset Hetrec-2011* mostram que o sistema proposto tem um RMSE de 0.757 e a métrica de similaridade *Jaccard mean square distance* tem um RMSE de 0.787.

A segunda versão, desta segunda abordagem, propõe a criação de um Sistemas de Recomendação semelhante a tradicional filtragem colaborativa baseada em memória mas sem se realizar previsão dos *ratings* que um utilizador atribuirá aos itens. Este Sistema de Recomendação cria uma lista de itens mais apropriados para recomendar ao utilizador com base nos itens que os vizinhos mais gostaram. Estes são identificados usando a métrica de similaridade *Fuzzy Fingerprint similarity* a qual tira partido da informação contextual dos itens e dos *ratings*. Este Sistema de Recomendação é capaz de gerar melhores recomendações

que as soluções anteriormente desenvolvidas usando *Fuzzy Fingerprints*. Os testes foram realizados no *dataset ML-1M* onde este Sistema de Recomendação consegue obter um F1-score de 0.769 enquanto o *Jaccard mean square distance* apenas obtém 0.766 . Ainda assim, a eficiência computacional do sistema proposto, não é comparável com a solução proposta na primeira abordagem desenvolvida.

A terceira e última versão, da segunda abordagem, propõe a criação de uma métrica de similaridade multi-contextual usando *Fuzzy Fingerprints*. Métrica essa que é integrada no Sistema de Recomendação que não realiza previsão de *ratings*, da versão anterior. A solução combina diferentes *Fuzzy Fingerprints* com diferentes fontes de informação contextual usando um algoritmo de *ranking fusion*. Esta versão obteve a mais alta precisão de todas as soluções apresentadas nesta Tese. A sua precisão é de 0.637 enquanto o *Jaccard mean square distance* apenas consegue uma precisão de 0.621. A eficiência computacional da métrica de similaridade multi-contextual usando *Fuzzy Fingerprints* é comparável à eficiência computacional da similaridade de coseno ou *Pearson correlation* que não utilizam informação contextual. Esta terceira versão apresenta resultados promissores quer para melhorias significativas em termos de qualidade de recomendação quer para melhoria da eficiência computacional. Tal como se observou na primeira abordagem no *dataset* ML-1M as melhorias na eficiência computacional estavam limitadas pelo número de *ratings*, utilizadores e itens. Será interessante a aplicação das ideias da segunda abordagem a *datasets* de maiores dimensões pois será esperado uma maior margem de melhoria da eficiência computacional, tal como se verifica no *datasets Netflix* e *Jester*.

Em conclusão, os resultados mostram que a aplicação das *Fuzzy Fingerprints* no âmbito de Sistemas de Recomendação permite melhorar ligeiramente a qualidade de recomendação mas, principalmente, a capacidade de ter uma maior eficiência computacional, face a outras métricas de similaridade testadas. O uso de informação contextual permite melhorar a qualidade de recomendação quer seja combinando informação contextual e *ratings* numa só *Fuzzy Fingerprint*, ou usando multi-contextual *Fuzzy Fingerprints*. Porém, a utilização de informação contextual permite melhorias na qualidade de recomendação em detrimento de alguma eficiência computacional segundo as experiências até agora realizadas nos *datasets ML-1M* e *Hetrec-2011*. Ainda assim, é totalmente comparável com outras métricas usadas pela comunidade científica esperando-se que, com a utilização de *datasets* maiores como o *Netflix*, seja possível melhorias da eficiência computacional similares às obtidas com a primeira abordagem deste trabalho.

Palavras-Chave: Sistemas de Recomendação, Filtragem colaborativa baseada em memória, Métrica de Similaridade, *Fuzzy Fingerprint*, Complexidade computacional.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The field of Recommender Systems (RS) has significantly been affected by the continuous growth of the Internet, which boomed with the appearance of search engines and is now part of our society. Most Recommender Systems rely heavily on similarity metrics to generate quality recommendations. Many fields besides Recommender Systems also take advantage of similarity metrics, for example, search engines, image classification, data mining, among others. In the context of Recommender Systems, similarity metrics are optimized to compare items or users. This work concerns the use of similarity metrics that exploit information such as ratings, item descriptions, and keywords, to provide recommendations using techniques similar to current Recommender Systems but requiring less computational resources. This Chapter discusses the relevance of Recommender Systems and Fuzzy Fingerprints, the goals and contributions of this Thesis, and concludes with a brief overview of the contents of each Chapter.

1.1 Recommender Systems

Users of the digital world are overloaded with information [1]. Nowadays, Recommender Systems provide a solution to those that, regularly, rely on

services such as video and audio streaming, online shops, hotel booking, restaurant search, job offers, among others. Recommender Systems allow us to cope with this large amount of information, by cataloging a vast list of items, that later can be recommended. These recommendations can be determined by a large number of techniques, as highlighted by the scientific community [1, 2]. In fact, due to their success, Recommender Systems can be found in several services [3].

In general, a Recommender System (RS) works by collecting information from its users on a given set of items [4]. This information can be explicit (e.g. ratings) or implicit (e.g. frequency of a mouse clicks on an item). Recommender Systems (RSs) can also rely on *contextual information* sources, such as item descriptions, temporal information, demographic information, social information, among others [4]. With such information, the system should be able to provide valuable recommendations, filtering through the overload of information by bridging users preferences profiles and comparing them to others.

Having these techniques allows the RSs to discover new items that might please the user, among the diversity of items available. Some companies, such as Amazon, eBay, Netflix, and LinkedIn, are current examples of services that benefit from providing recommendations to their clients [3, 5, 6]. As a result, much research has been done in the area, contributing to the development of new techniques to improve recommendations, helping to increase the RS providers' income. Note that these companies have requirements such as provide valuable recommendations; and being profitable, which sometimes can be conflicting. In this Thesis, only considers the first requirement.

1.2 Collaborative filtering based Recommender Systems

Despite the efforts to improve RSs, there are still challenges to be addressed. For example, turning state-of-art solutions into real-world scenarios is still difficult, mainly due to a large amount of available data and the scalability issues that ensue [7, 8]. For this reason, more traditional approaches, such as *item-based* Collaborative Filtering (CF) are still the most widely used [9, 10, 11]. Regardless of its simplicity, CF can provide quite accurate results, thus yielding an advantageous trade-off between engineering effort and user satisfaction.

CF exploits how humans make decisions. As users provide ratings to a plethora of items, such information is stored in a rating matrix. These stored ratings create patterns that allow computing similarity values between items, via a similarity metric. In general, CF systems then compute rating predictions based on the similarities between items. The rating predictions are used to generate recommendations to users. Provided that there is enough collected information, the RS can find items similar to those a user like [4].

In CF systems, the issue of scalability is related to the need to compute similarities between a high number of items in the database. The used similarity metrics can introduce a high load, as the number of users and items grow, thus reducing their applicability for real-world services [12]. To solve this, two complementary types of solution are often proposed. One is to provide scalability by distributing the storage and computational cost over several machines [13, 14]. The other is to devise computationally efficient similarity metrics [15, 16, 17], in which this Thesis focuses. Different approaches have also been proposed to improve computational efficiency. In [15], for example, the goal is to speed up the k-nearest neighbors algorithm using a hardware-based similarity function. Its use allows for

a computation approximately twice as fast than a traditional metric. Authors in [16] represent items and their biases in a Euclidean space that preserves the item consumption patterns. This embedding representation approach can retrieve similar items faster from a Euclidean space, as shown by experiment results. However, in this Thesis a Fuzzy Fingerprint similarity function is used to improve the computational efficency of RSs.

1.3 Fuzzy Fingerprints

The concept of Fuzzy Fingerprints has its roots in *Fuzzy systems*. Although there are RSs using Fuzzy systems, this Thesis is the first where Fuzzy Fingerprints are applied in the context of RSs.

A Fuzzy Fingerprint (FFP) is a top-k list to which is applied a fuzzifying function. The key information in an FFP is the ranking of the list and not the actual weights of each element [18]. Fuzzy Fingerprints can be compared using a Fuzzy Fingerprint similarity function. The proper formation of the top list of features in the FFP, and the application of a proper membership function enables to improve the information used by the similarity metric, consequently reducing the number of features used [19] and thus speeding up the similarity computation.

Fuzzy Fingerprints have been used on other domains, such as text authorship identification [20], identifying mobile users based on call logs [19], topic detection for micro-blogging (e.g. Twitter posts) [21], text classification into categories [18], and event determination using sparse textual information [22]. FFPs were used in contexts where its features result from textual information or logs, and the data is on a continuous domain. For example, in [19] and [22] FFP features are composed of statistical information derived from words in a text (TF-IDF values), whereas in [19] FFP features are derived from call logs from each user, these call logs are ordered starting with the most frequently called number to the least frequent

called number. Due to the proprieties of RSs, and more specifically ratings on CF, the existing information is on a discrete scale. This raises different challenges than previous solutions using FFPs on other domains. As stated above the ranking of the top-k list of features is crucial for the success of FFPs. This Thesis addresses these challenges and exploits the capabilities of FFPs and FFP similarity, to create a better performing similarity metric, in both computational efficiency but also recommendation quality.

1.4 Objectives and Contributions

This Thesis addresses the following research question: *Is it possible to improve traditional CF methods using contextual information sources without increasing computational complexity?* This question can be answered by testing the following hypothesis:

- Develop efficient similarity functions to reduce the computational complexity of current CF solutions without impacting the recommendation quality.

- Use the developed similarity functions to combine information from other sources and improve the quality of recommendations while still maintaining low computational complexity.

To address these issues, a novel similarity metric is proposed for Recommender Systems, using the concept of Fuzzy Fingerprint (FFP) [20], more specifically, representing items by their low-dimensional Fingerprints. These can then be directly used to determine similarities between items, instead of the higher dimensional raw set of item ratings commonly used in RSs. Although FFPs are a compact representation, i.e. they require removing some information from the full feature vector, FFPs present several characteristics that are fundamental for RSs:

- minimal number of features to represent an item (or user), in comparison to the original feature vector;
- easy to create and update, as new data enters the RS;
- there is a maximum size for each FFP, thus as new data enters the RS, the increase in computational complexity is limited;
- faster than traditional similarity metrics;
- the FFP of each item (or user) is independent, which allows for easy integration of new users to the system;
- it requires a minimal implementation effort.

These characteristics make FFPs an appropriate technique for achieving the goals of this Thesis, i.e. the development of a similarity metric able to require less computational resources than current solutions, and incorporate different sources of features while outperforming existent similarity metrics.

To demonstrate these claims, experiments were performed on four datasets. Results show that FFPs are a promising route to be applied for recommendations, requiring from 23% through 95% fewer iterations to compute the similarities for a rating prediction, depending on the density of the dataset. This improvement is achieved while maintaining a comparable quality of results, varying from a loss of 0.0241 through a gain of 0.0105 in Root Mean Square Error (RMSE), which combined with an improvement of computational efficiency is a worthy trade-off.

1.5 Organization of this work

The first part of this work, Chapters 1, 2, 3 provide background on RSs and Fuzzy Systems. Chapter 2 contains an overview description of RSs,

the existing filtering techniques, focusing more on the explanation of traditional memory-based CF and its key components, followed by a broad overview of Fuzzy Systems, with an explanation of their key characteristics and their applications. Chapter 3 provides a literature review for RSs with the particular focus on computational efficiency improvement, similarity metrics, and how Fuzzy systems have been used in RSs. It finishes with the literature review of Fuzzy systems and Fuzzy Fingerprints.

In the second part of this work, Chapters 4 and 5 are composed by application of Fuzzy Fingerprints on RSs detailing its challenges, development, and solutions produced. Chapter 4 details the application of Fuzzy Fingerprint to RSs. A newer approach that reformulates previous Fuzzy Fingerprint solutions applied to other domains. Chapter 5 starts by providing extensive details regarding the evaluation process, which includes quality metrics, datasets, and framework used. Afterwards, it presents a series of results discussing the quality of recommendations and the computational complexity of the proposed solutions.

Finally, Chapter 6 draws conclusions and a final analysis of the results of this Thesis. This research left some questions opened which is how this document ends, with a proposal of future work.

Chapter 2

Basic Concepts

On this Chapter, an overview of Recommender System (RS) and Fuzzy Systems is given. RSs have used different filtering techniques over the years. Section 2.2 provides an overview of these. The most broadly used filtering technique is called Collaborative Filtering (CF), Section 2.3 contains an extensive and detailed explanation of different CF approaches. One of these approaches is Memory-based CF, more specifically item-based CF, Section 2.4 explains traditional item-based CF, how to produce recommendations, compute rating predictions and determine similarity metrics. Another technique is Content-based filtering also widely popular on RSs, it uses descriptive information about the item and users. Such information is exploited by the RS, either as an alternative to CF, but more often combined with CF. Section 2.5 explains how Content-based filtering works, and its most common elements. Section 2.6 examines different approaches to create a hybrid filtering RS and how filtering techniques discussed in Section 2.2 can be complementary to provide better recommendations to users. This Chapter concludes with Section 2.7 about Fuzzy Systems. Its basic concepts are shown, starting with an overview of the domains in which it has been applied, followed by an explanation of Fuzzy sets and T-norms. It then ends with how Fuzzy Systems can be applied for

RSs.

2.1 Principles of Recommendation Systems

A Recommender System (RS) helps users to find items that correspond to their preferences. This provides an interesting tool for both users and e-commerce providers, as users discover new items and e-commerce providers can increase their revenues. For this reason, the demand for RSs has only increased, allowing users to deal with large amounts of data and providing them with a selection of personalized recommendations, services, and contents [4]. As a result, various techniques have been developed and studied by both the scientific community and service providers [4, 23, 24, 25]. The scientific community has mostly focused on increasing the quality of recommendations [16, 26]. The problem is that some of these solutions are computationally demanding and are engineering challenging for real-world applications. Nowadays, cloud computing alleviates the computational complexity but does not solve it completely.

Recommendation Systems started by helping users distinguish between relevant and irrelevant emails, in a work that coined the term *Collaborative Filtering (CF)* [27]. Users would provide feedback (relevant/non-relevant) for the emails and, after a few interactions, the system was able to classify emails as relevant or non-relevant for the remaining users.

A Recommendation System traditionally relies on two entities: the users and the items, these share a relation between each other, described as an interaction. Items can be many things, such as movies, music, restaurants, job offers, products from a website, among other possibilities. The interactions between users and items generate a large set of tuples that can be represented as *(user, item, interaction) triplets*. Such a set can be alternatively represented as a sparse matrix where each user corresponds to a row, an item to a column, and each entry is the interaction between the

two. These interactions can be *explicit*, such as quality ratings, or *implicit* such as mouse clicks.

A RS can store both implicit and explicit feedback. Youtube is a good example having the like and dislike button while at the same time tracking which videos (and for how long) have been watched, by each user. Some systems exploit implicit feedback, for example, Amazon or eBay use the browsing pattern from users (via searches and clicked items), Spotify exploits if a user keeps skipping specific type of songs and listens to other types until the end[1]. Implicit feedback is considered as a mean to provide a sentiment toward an item without user intervention. Nowadays, research is increasing to improve implicit RSs. In 2017, Recsys released a challenge to improve job RSs in collaboration with XING[2]. In this challenge, participants are tasked with predicting which jobs (items) have been clicked, jobs that users applied to, and should take into account if it was a premium job listing, as these premium job listings are a source of income for XING. This challenge was more focused on short-lived items since job listings are short-lived in a RS context. This problem is focused on cold-start problems making this challenge more suited for Content-based filtering.

In 2019, the challenge is co-organized with Trivago[3]. Participants are tasked with predicting which accommodations (items) have been clicked in the search result during the last part of a user session, in an offline evaluation setup.

Explicit feedback requires the user intervention, by actively providing a measurable sentiment toward an item, e.g. rating a movie in IMDB or Netflix. The matrix that stores these ratings as interactions between users and items is known as Rating Matrix in which Collaborative filtering approaches rely on. This Thesis focuses only on explicit interactions.

[1]Information obtained from http://www.cp.jku.at/tutorials/mrs_recsys_2017/ on 21 December 2018

[2]XING: http://www.recsyschallenge.com/2017/ Accessed in 15-01-2019.

[3]Trivago: http://www.recsyschallenge.com/2019/ Accessed in 15-01-2019.

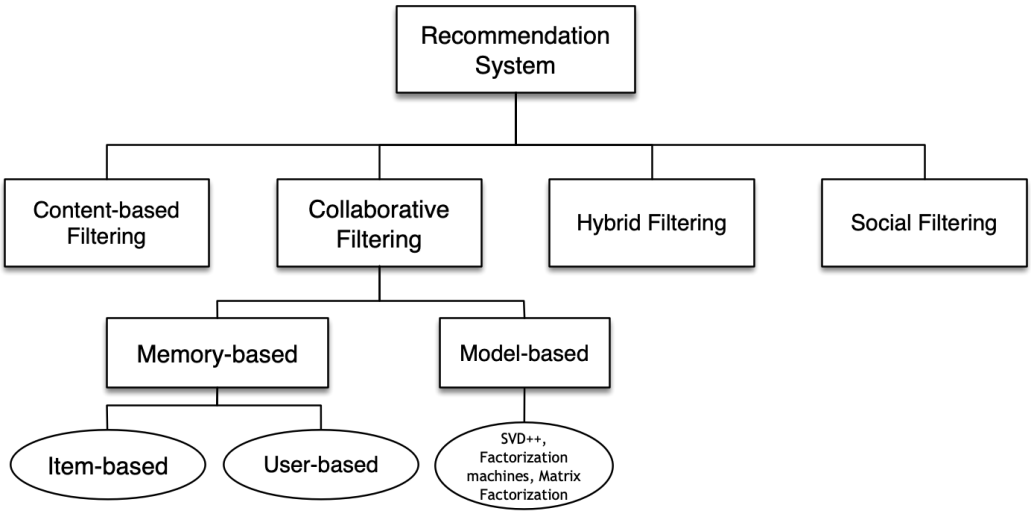


Figure 2.1: Filtering techniques used in Recommendation Systems.

2.2 Overview of Recommendation techniques

The idea behind CF is exploiting the preferences of a community. A RS can, however, follow different strategies. Figure 2.1 illustrates the most common filtering techniques used by RSs.

Model-based CF generally, relies on supervised machine learning to perform rating prediction [28]. Some models also perform rank prediction and try to identify how related a user is to each item, skipping the rating prediction stage [29]. These supervised machine learning algorithms have two phases: (1) Learning phase: where the latent features are fine-tuned to improve the overall performance of the RS, generally by minimizing the root squared error between rating predictions and the actual ratings. (2) Prediction (or recommendation) phase: where the model is already trained and able to provide rating predictions and recommendations to its users. The learning phase is the most computationally expensive. Model updates are achieved either by using an online approach, where the learning and prediction phases concurrently occur, or through a periodic model retrain-

ing to incorporate new ratings, e.g. after a certain threshold of new ratings has been registered.

One of the most common Model-based CF approaches is Matrix Factorization [30], since it was applied with great success on the Netflix challenge [31]. Matrix Factorization performs a decomposition of the rating matrix into two latent matrices, where one represents users based on latent features, and the other represents items based on their latent features [32]. These matrices are determined using optimization algorithms, such as stochastic gradient descent or alternating least squares.

Content-based filtering infers the correlation between the attributes of items and users. These attributes usually result from the description of an item such as its textual description or keywords associated with it. Users provide information by creating a profile rich with attributes, that can be compared to the items and exploited by the RS to generate valuable recommendations.

Hybrid filtering consists of a combination of different filtering techniques that often result in an improved recommendation quality, since recommendations are generated based on richer information regarding items and (or) users. Hybrid filtering techniques usually rely on CF combined with one or more filtering techniques that enrich the RS with contextual information.

Social filtering uses the community social network to infer a correlation between users, often to complement CF information, reducing data sparsity and the cold-start problem [33, 34]. The use of a social network allows improving the neighborhood formation of a traditional CF system, an essential component of memory-based CF [35]. Since CF does not work well for items and users with few ratings available [4], the use of an existing social network by, for example, using the degree of trust between peers, provides contextual information that helps to improve recommendations. Both Content-based and Social filtering help increase the infor-

mation available regarding users and items. It is by exploiting this information and combining it with CF that Hybrid filtering achieves better performing RSs.

2.3 Collaborative Filtering

Most Recommendation Systems rely upon Collaborative Filtering (CF), since it provides good quality recommendations and its implementation cost is low. CF relies on community knowledge and experience to better infer recommendations. CF can be split into two main categories: (1) Memory-based: an approach that computes similarities between items (or users), then predicts ratings for user-item pairs; (2) Model-based: an approach that relies on machine-learning algorithms to predict ratings, often resulting from the computation of latent features between variables. The Model-based approach is nowadays more supported by the scientific community, highly focused on comparing the recommendation quality between proposed solutions and often overlooking the computational requirement of such solutions in real-world applications [15].

Memory-based CF is split into two main approaches: (1) user-based, which compares users and tries to determine similarities between them; and (2) item-based, that measures the similarity between items instead. More specifically, user-based CF compares users to create groups of similar users (neighborhoods), using these to compute rating predictions for a given $(user, item)$ pair. It relies on the degree of similarity between neighboring users. Item-based CF, on the other hand, creates neighborhoods of items producing predictions for $(user, item)$ pairs also, but now depending only on comparing items that the user previously rated. Item-based has several advantages, since a RS usually has more users than items, and the computation of rating predictions relies on similarity metrics, causing the number of similarities to be computed to be much higher for user-based

than for item-based approaches.

This Thesis focuses on item-based approaches. The main advantage of item-based CF is its scalability in comparison to user-based CF. In well established real-world RSs, the number of users is far higher than items. This is highly relevant because a higher number of users also means a higher number of similarities to be computed. To illustrate, consider the dataset provided by Netflix (see Chapter 5) contains 20 000 users and 1 700 items. With a user-based approach, 399 980 000 similarities would have to be computed, as opposed to an item-based CF which would only need to calculate 2 888 300 similarities, approximately 138 times less. Besides, item-based CF approaches, in general, allow for better recommendations to be obtained, when compared to user-based CF, which makes it also preferable to use [10].

To address the high number of similarities to be computed and the fact that these are continually changing, a straightforward solution is to pre-compute similarities, which can be later updated with a given periodicity. This Thesis presents a less computationally demanding similarity metric that, while delivering good results, allows to improve the computational efficiency of such updates.

CF systems usually store the ratings given to items by users in a so-called *rating matrix*. They rely on such ratings to determine similarities between items (or users), through the use of a similarity metric. This allows the creation of neighborhoods of similar items (or users), to predict new ratings and consequently to be used by the recommendation strategy. Figure 2.2 illustrates the architecture of traditional Collaborative filtering with users, interface, rating matrix, similarity metric, prediction metric, and recommendation strategy.

Users of a RS interact with it via the user interface. The user interface (e.g. website or a mobile app) is responsible for being the intermediary between the RS and the user. Either by collecting ratings from users on

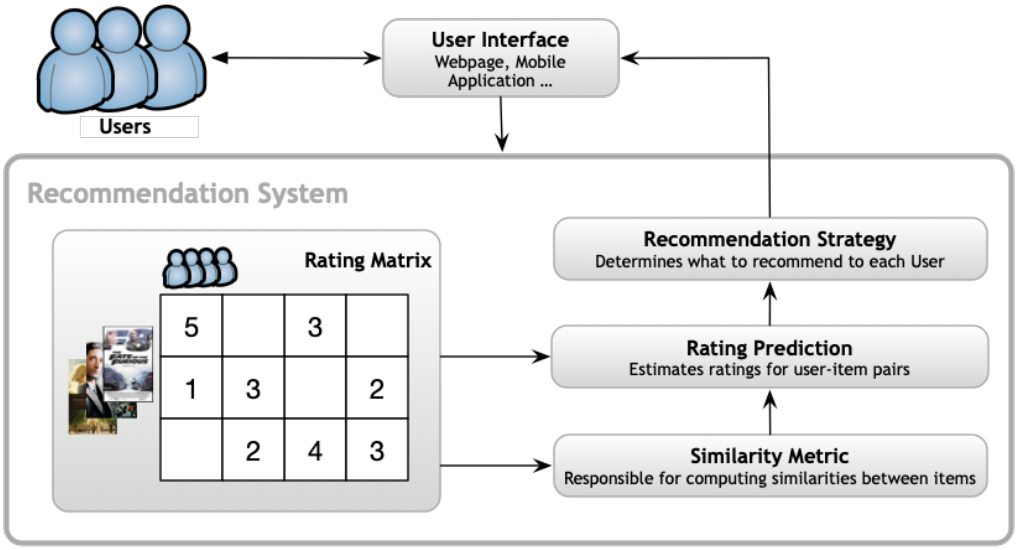


Figure 2.2: Memory-based Collaborative filtering.

items, as by providing recommendations to users. Using the ratings from its users, the RS builds the rating matrix, in the example of Fig. 2.2 items correspond to movies. It is by leveraging the rating matrix that both the similarity metric and rating prediction generate their results. A memory-based CF uses, exclusively, the similarity metric to compute similarities between items (or users). Rating predictions require the computation of similarities using the similarity metric and the rating matrix. Relying on the quality of rating predictions, the recommendation strategy generates sets of items to recommend to each user.

2.4 Memory-based Collaborative Filtering

Memory-based Collaborative filtering traditionally performs the following tasks to create recommendations: (1) compute the similarity between items (or users) depending on the used approach; (2) predict the possible rating that a user attributes to an item; (3) recommendation generation of

items to a user. This Section explains how this is achieved and provides formulas and calculation examples, for each.

2.4.1 Recommendation Strategy

The most simple and commonly used recommendation strategy is to sort rating predictions placing the highest rating predictions first. Assuming that only seven items exist in the RS, and that user v has already rated items x and y, and a set of item rating predictions $\hat{r}_{vi}$ were generated by a rating prediction formula. This set of rating predictions could be as illustrated in Fig. 2.3.

i	a	b	c	d	e
$\hat{\mathbf{r}}_{\mathbf{vi}}$	3.3	4.2	4.8	3.5	2.5

Figure 2.3: A set of rating predictions $\hat{r}_{vi}$ for user v.

Since RSs can generate a vast number of rating predictions, there should be a limit of the number of recommendations made to a user. In this example, if the maximum number of items to recommend is set to 3, the resulting recommendations would be items c, b and d. This is an important aspect since depending on the application of a RS more than ten or twenty recommendations could be too much for the user. The RS's goal is to alleviate and filter through the overload of item possibilities.

2.4.2 Rating prediction

To generate recommendations, rating predictions need to be computed for all items not rated by the user. The rating prediction formula is obtained using the neighborhood resulting from the similarity metric. More specifically, let $\widehat{r}_{vi}$ be the *predicted* rating that a given user v would assign to item i. Starting by computing the *neighborhood* $N_i(v)$ of item i , i.e. the set of n items in the database that are more similar to i. To determine

$N_i(v)$, the similarity between the item i and all other items rated by the v must be computed. The process is performed by computing the similarities and then ranking them from the most similar to the less similar. Figure 2.4 shows an example of a set of similarities between item i and all other items.

i	a	b	c	d	e
sim(i, j)	0.71	0.30	0.81	0.41	0.9

Figure 2.4: Similarities between item i and other items rated by user v.

The neighborhood is determined by ranking these items in Fig. 2.4. The resulting ranked neighborhood is e, c, a, d, b. Ranking the neighbors of i is important, since depending on the memory-based CF approach used [35], the number of neighbors used to generated rating predictions can be truncated either by: (1) setting maximum of neighbors; or (2) establishing a minimum similarity value to be considered a neighbor. The most common way to truncate the number of neighbors is to set the maximum number of neighbors and to fine-tune the RS according to an evaluation criteria.

With a known neighborhood of items, the value of $\widehat{r}_{vi}$ can be computed [35, 36] using, for example, the average algorithm (Eq. 2.1), weighted average algorithm (Eq. 2.2), or the adjusted weighted method (Deviation-from-mean) (Eq. 2.3).

$$\widehat{r}_{vi} = \frac{\sum_{g \in N_i(v)} r_{vg}}{|N_i(v)|} \tag{2.1}$$

$$\widehat{r}_{vi} = \gamma_{vi} \sum_{g \in N_i(v)} sim(i, g) \times r_{vg} \tag{2.2}$$

$$\widehat{r}_{vi} = \bar{r}_i + \gamma_{vi} \sum_{g \in N_i(v)} sim(i, g) \times (r_{vg} - \bar{r}_g) \tag{2.3}$$

where r_{vg} is the rating assigned by user u to item g, $\bar{r}_x$ is the average of all ratings assigned to item x and γ_{ui} is the normalization factor defined as:

$$\gamma_{vi} = \frac{1}{\sum_{g \in N_i(v)} sim(i,g)} \tag{2.4}$$

The average algorithm (Eq. 2.1) is an average of ratings of the neighbor items of item i. It is an extremely simplified approach to generate rating predictions. Note that Eq. 2.1 uses the similarity metric as it is required to compute the *neighborhood* $N_i(v)$. The weighted average algorithm (Eq. 2.2) is an improved version of the average algorithm. It uses the similarity metric to balance the relevance of each neighbor rating and is normalized using Eq. 2.4.

The last alternative to compute rating predictions, and the one used in this Thesis, is the adjusted weighted method (Deviation-from-mean) (Eq. 2.3), due to the better results obtained on experiments. Being an improvement over the weighted average algorithm, it accounts for the average rating of each item and how much a user enjoys or dislikes an item in relation to its average rating. As in the weighted average algorithm, ratings are also weighted by the similarity metric. The sum of this operation for all the neighbors, after being normalized, estimates how much the user is expected to like an item.

Rating prediction formula in Eq. 2.3 uses the average rating of the item, yet a vast number of combinations can be applied [31]. The value $\bar{r}_i$ refers to the average of the community rating on that given item, but the average rating of the user $\bar{r}_v$ and the global average rating of the rating matrix can also be used on Eq. 2.3. Nevertheless, as a rule of thumb, item-based approaches use $\bar{r}_i$ and user-based approaches use $\bar{r}_v$.

To generate recommendations for a user u, rating predictions are computed for all items not evaluated by the user. For example, in Fig. 2.5 item i can be recommended to users a and e, as the rating vector has no rating from them.

u	a	b	c	d	e
$\mathbf{r_{ui}}$	-	3	4	1	-

Figure 2.5: Rating vector of users on item i. Users a and e have not rated item i.

2.4.3 Computing similarities

Some of the most common similarity metrics used in RSs are Cosine similarity (COS) and the Pearson Correlation (PC) [6].

u	a	b	c	d	e
$\mathbf{r_{ui}}$	-	3	4	1	-
$\mathbf{r_{ug}}$	5	2	-	3	3

Figure 2.6: Rating vectors of item i and item g.

Figure 2.6 illustrates an example of two rating vectors from items i and g. It is used to explain how the similarities are computed between two items.

The Cosine similarity (COS), as defined in Equation (2.5), is the angle formed between two vectors.

$$sim_{COS}(i,g) = \frac{\sum_{u \in U} r_{u,i} \times r_{u,g}}{\sqrt{\sum_{u \in U} r_{u,i}^2} \times \sqrt{\sum_{u \in U} r_{u,g}^2}} \tag{2.5}$$

Results from the inner product between both items i and g normalized by their L2-norm, over the users in U that rated both movies. It yields a value between 0 and 1, where 0 corresponds to no similarity between them and 1 to exactly proportional ratings between both users. Using the rating vectors from Fig. 2.6 the COS value between the two items is computed as follows:

$$sim_{COS}(i,g) = \frac{3 \times 2 + 1 \times 3}{\sqrt{3^2 + 1^2} \times \sqrt{2^2 + 3^2}} = \frac{6+3}{\sqrt{10} \times \sqrt{13}} = 0.789 \tag{2.6}$$

Another metric often used as a similarity measure is the Pearson Correlation. This coefficient has been widely used in RSs, since it is simple to implement, intuitive and provides fair quality results [37]. PC is defined in Eq. 2.7. It results from the inner product between i and g, normalized by the average rating of each item. The inner product between the two is then normalized by the square root of the product of the variance of from two items.

$$sim_{PC}(i,g) = \frac{\sum_{u \in U}(r_{u,i} - \bar{r}_i) \times (r_{u,g} - \bar{r}_g)}{\sqrt{\sum_{u \in U}(r_{u,i} - \bar{r}_i)^2} \times \sqrt{\sum_{u \in U}(r_{u,g} - \bar{r}_g)^2}} \tag{2.7}$$

The result of Eq. 2.7 is within the interval $[-1, 1]$, where -1 corresponds to an inverse correlation and $+1$ to a positive correlation. Values near zero show that no linear correlation exists between the two items.

Using the rating vectors from Fig. 2.6 the PC value between the two items is computed as follows:

$$sim_{PC}(i,g) = \frac{(3-2) \times (2-2.5) + (1-2) \times (3-2.5)}{\sqrt{(3-2)^2 + (1-2)^2)} \times \sqrt{(2-2.5)^2 + (3-2.5)^2}} = 0.44353 \tag{2.8}$$

Note that RSs have rating matrices with high sparsity and therefore the vector of each item is also sparse. Thus, in both cases, similarities are computed using only existing ratings with respect to each item.

2.5 Content-based Filtering

The roots of Content-based filtering are from Information Retrieval [38], due to the earlier development of search engines. Search engines rely on text, which is represented using words. Likewise, Content-based filtering represents items using their textual description and keywords. It relies on

the description of interests of the user to determine the degree of interest on an item. It is composed of three general components [39]: (1) Content analyzer: pre-processing step to transform unstructured data (e.g. text) into structured data using approaches such as TF-IDF representation [28]; (2) Profile learner: collects data from the content analyzer component, to represent users or items profiles; (3) Filtering component: exploiting the users' profiles and items, to then match them and generate recommendations.

Instead of users ratings to represent items, textual descriptions, e.g. movie synopsis or keywords are used to represent the item. Term Frequency-Inverse Document Frequency (TF-IDF) is a common approach to create a Content analyzer responsible for transforming unstructured data into structured data that can be compared programmatically. The value of TF-IDF corresponds to how relevant is a term of an item taking into account not only the relevance of the word in the given item but also in the collection of items. TF-IDF weights of each term w.r.t. each item is obtained by computing the Term Frequency (TF) multiplied by the Inverse Document Frequency (IDF). TF is the frequency of a term, in this Thesis a word w, for a given item i and is computed using Eq. 2.9.

$$tf_{i,w} = \frac{f_{i,w}}{\sum_{i' \in I} f_{i',w}} \tag{2.9}$$

The number of occurrences of the word w in the item i is represented by $f_{i,w}$, where w correspond to a given word within the dictionary W, and I corresponds to the collection of existing items. The IDF of a word reflects how frequent is the word within the collection of items. The computation of the IDF is done using Eq. 2.10.

$$idf_w = \log\left(\frac{|I|}{|i \in I : w \in W|}\right) \tag{2.10}$$

To obtain the TF-IDF of a word in a given item Eq. 2.11 is used.

$$TF\text{-}IDF_{i,w} = tf_{i,w} \times idf_w \tag{2.11}$$

Profile learner approaches such as clustering techniques (e.g. k nearest neighbors [4]) exploit structured representations of items and can compare and establish neighborhoods of items that later can, with the use of a filtering component, provide a selection of recommendable items to a user.

Content-based approaches have drawbacks since they are limited by (1) reliability of contextual data and if this data contains enough information, but also the use of other sources of information such as images, videos or manual keywords attribution is very time and resource consuming; (2) profile learning approaches can become overspecialized since recommendations only occur due to the high similarity between users profile and item representation. Also, diversity is hard to incorporate due to this property. Content-based approaches shine when new users join a RS, i.e. when users show a preference for a few items, which are not enough for CF to generate good recommendations, but enough for content-based filtering approaches.

On Content-based approaches it is highly unlikely that new information to improve the representation of items is added, e.g. a description of a movie or the description of products on an online shop will not change over time. Due to these limitations, Content-based approaches are commonly used in conjunction with CF approaches, leading to Hybrid filtering solutions.

2.6 Hybrid Filtering

The use of Hybrid filtering aims to provide more contextual information to the RS, this extra information allows to, generally, improve recommendation quality. A RS using Hybrid filtering can be implemented using

different approaches. This Section discusses each one of these approaches briefly.

Most approaches using Hybrid filtering aim to compensate for the limitations of different filtering techniques. The most straightforward approach is weighting [40], where each filtering technique produces a rating prediction or recommendation and, given certain weighting criteria, rating predictions or recommendation are generated resulting from the weighted combination of these filtering techniques.

Another simple approach involves switching the filtering technique [40]. This approach requires a criteria to decide when to switch to other filtering technique, e.g. when one filtering technique is not producing valuable recommendations anymore. A different filtering technique should be able to produce different recommendations to a user than the previous technique.

A mixed Hybrid filtering technique can also provide diversity to recommendations [40]. The aim of such approaches is to compensate for the limitations of different filtering techniques by generating recommendations, where part of the items are generated from a content-based approach, other part generated by a CF approach and another part by a social filtering approach. The recommendations could be an aggregation of the three filtering techniques, this approach allows the creation of a large number of recommendations and avoids a possible overspecialization of a particular technique.

Hybrid filtering through feature combination [40] is the technique used in this Thesis. The technique requires the combination of different sources of information e.g. ratings and textual data, for a single model use to both without distinguishing between them.

Cascading is also a popular approach [40], where a filtering technique creates a set of recommendations. Then, the next filtering technique improves and modifies the recommendations created by the previous approach, leading, in the end, to an expected better set of recommendations

for users.

Feature augmentation is also a Hybrid filtering technique [40]. These systems use one filtering technique that creates, e.g. rating predictions. These ratings then join the actual ratings, into the rating matrix to be used by other filtering technique to produce recommendations.

2.7 Fuzzy Systems

Since the creation of fuzzy logic, Fuzzy systems have been a focus of continuous research [41]. Fuzzy systems are computational systems based on fuzzy logic [41]. Fuzzy systems are based on a mathematical model that creates a trust value representation that accommodates continuous values between 0 and 1, as opposed to the boolean logic, which uses only *true* and *false*. This is why fuzzy logic has a broad scope of applicability, such as pattern classification or information processing [42].

Fuzzy sets are the core element of fuzzy logic. Proposed and defined in 1965, by Lotfi Zadeh [42], a Fuzzy set characterizes a class of objects within a continuum of grades of membership (characteristics), using a function that assigns to each object a degree of membership ranging from 0 to 1 on a continuous scale. It allows for a natural way of representing problems that have imprecision on membership representation of the data.

Let X be a space and x an arbitrary element of X thus $X = \{x\}$. The fuzzy set (class) A in X is represented by a membership function $f_A(x)$. A membership function associates to any x a real number in $[0, 1]$, where $f_A(x)$ is the degree of membership of x in class A. In its ordinary form, the closer to 1 the value of $f_A(x)$ the more x belongs to class A. On the contrary, the closer to 0, the less it belongs, according to the membership function used to characterize class A. The notion of the membership function is entirely none statistical [42]. The step when x is transformed from space X into space A using $f_A(x)$ is also known as the *fuzzyfication of* x.

$$trimf(x;a,b,c) = max\left(min\left(\frac{x-a}{b-a}, \frac{c-x}{c-b}\right), 0\right) \tag{2.12}$$

$$trapmp(x;a,b,c,d) = max\left(min\left(\frac{x-a}{b-a}, \frac{d-x}{d-c}\right), 0\right) \tag{2.13}$$

$$gaussmf(x;a) = e^{\frac{1}{2}\left(\frac{x-a}{\sigma}\right)^2} \tag{2.14}$$

$$gbellmf(x;a,b,c) = \frac{1}{1+\left|\frac{x-c}{b}\right|^{2a}} \tag{2.15}$$

There are infinite possibilities for membership functions, and its correct selection depends on its application. Examples of commonly used membership functions are triangular (Eq. 2.12), trapeziodal (Eq. 2.13), gaussian (Eq. 2.14), and generalized bell (Eq. 2.15). Figure 2.7 illustrates these membership functions responsible for creating a fuzzy set. During this Thesis, membership functions are not combined, although it is possible to do so with two or more functions [43].

Having two fuzzy sets, it is possible to compute the t-norms (intersection) and t-conorms (union). A triangular norm is a binary operation $T : [0,1]^2 \rightarrow [0,1]$, and for all $x, y, z \in [0,1]$. The following properties apply:

- commutativity: $T(x,y) = T(y,x)$;
- associativity: $T(x,T(y,z)) = T(T(x,y),z)$;
- monotonicity: $T(x,y) \leq T(x,z)$, if $y \leq z$;
- boundary condition: $T(x,1) = x$;

Two popular triangular norms are Gödelian minimum and Product.

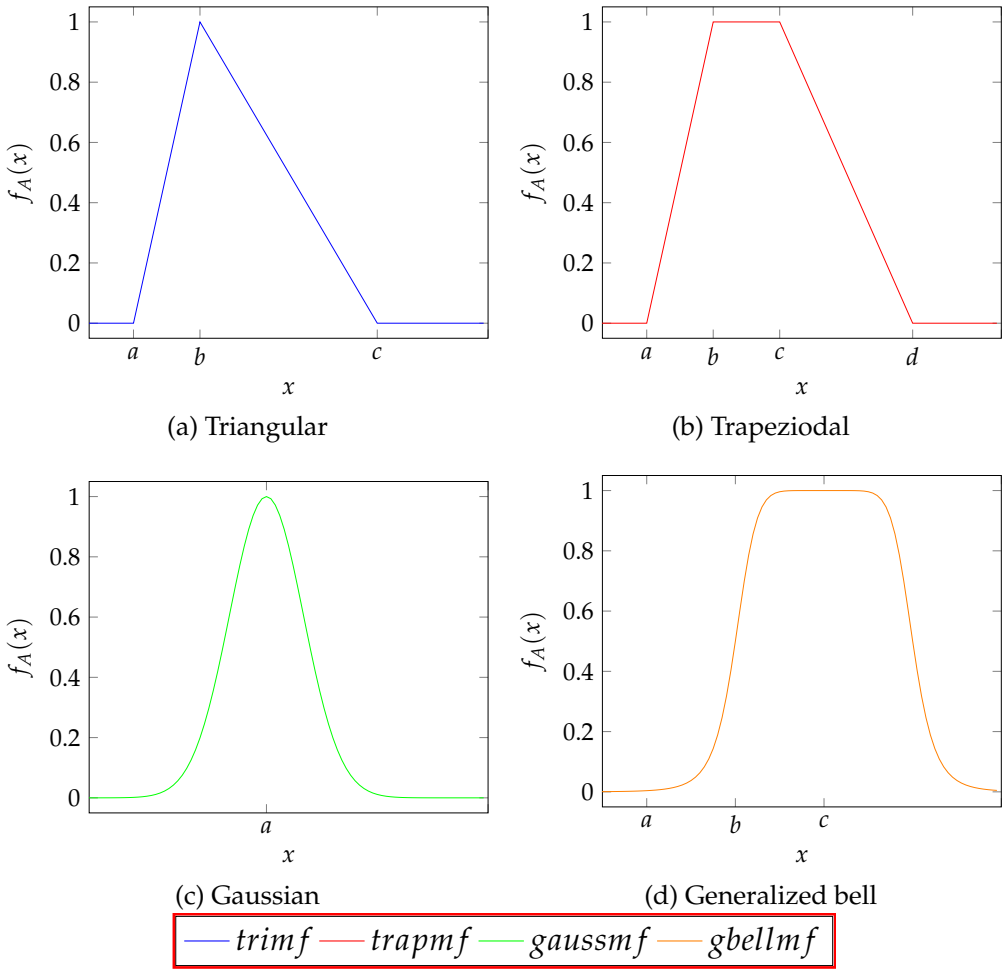


Figure 2.7: Membership functions: triangular (Fig. 2.7a), trapeziodal (Fig. 2.7b), gaussian (Fig. 2.7c), and generalized bell (Fig. 2.7d).

The Gödelian minimum triangular norm has the intersection operator i.e. t-norm $T_M(a,b) = min(a,b)$ and the union operator i.e. t-conorm $C_M(a,b) = max(a,b)$.

The Product triangular norm has the t-norm $T_P(a,b) = a \times b$ and its union operator i.e. t-conorm is $C_P(a,b) = a + b - a \times b$.

This Thesis relies on Gödelian minimum to form a similarity metric based on fuzzy logic, i.e. the Fuzzy Fingerprint similarity. This Section focused exclusively on topics relevant to the scope of this Thesis.

2.8 Ranking Fusion and Learning to rank

Ranking Fusion and Learning to rank are two techniques that aim to solve ranking problems. The most widely known application is ordering query results in a search engine. A search engine is responsible for, given a query, retrieving a list of documents. These documents need to be ranked regarding the relevance for that query.

2.8.1 Ranking Fusion

The goal of Ranking Fusion is when having a set of different ranked lists generated by different approaches, combine them into a final ranked list that contains the best ranking. Figure 2.8 illustrates this process applied to three different search engine approaches. Each generates a ranked reply to a given query, created via the search engine interface. The Ranking fusion algorithm combines the three ranked replies into an ideal set of ranked responses to that query.

There are several effects that ranking fusion algorithms can leverage to perform the task of re-ranking [44]:

- the skimming effect: different models represent differently the same items and consequently retrieve different results for the same query.

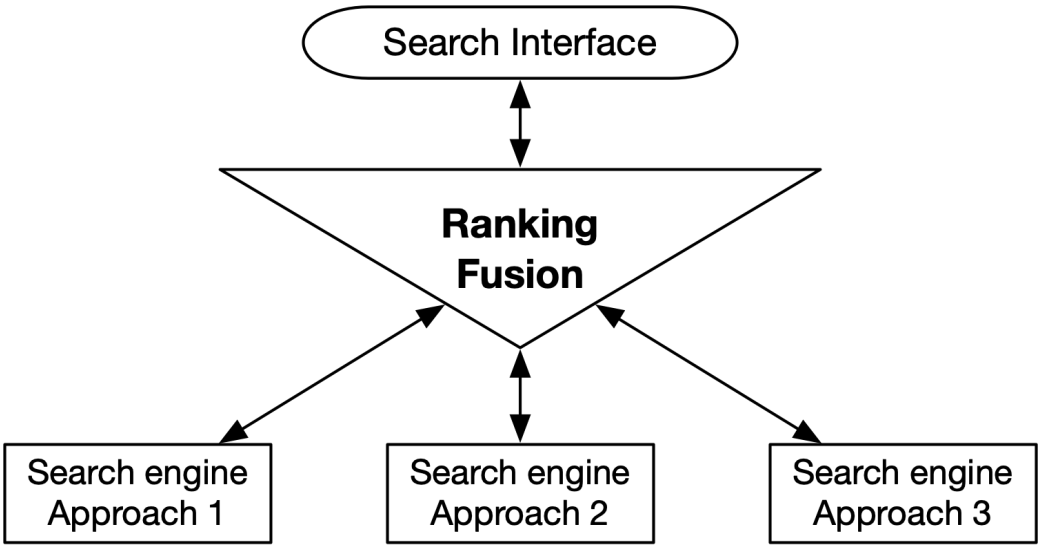


Figure 2.8: Ranking fusion combining three search engines replies to a query from a search engine interface.

Thus, top-ranked results from each approach can create a consensus of the most relevant elements, moving them to the higher ranked positions, while less relevant elements are moved to lower relevance positions;

- the chorus effect: elements retrieved for a query can be considered relevant by various approaches, thus tending to a confluence of more relevance than if only one approach finds it relevant;
- the dark horse: an approach in comparison to other approaches may determine for a set of queries a highly accurate or inaccurate ranking of elements.

Ranking fusion techniques exploit these effects with algorithms score based, e.g. Comb [45], and rank based algorithms, such as Borda-count [46] rank, Condorcet [47] and reciprocal ranking [48], among others. These algorithms are all unsupervised, meaning that they do not have a feedback of the correctness of the generated set or ranked elements.

Comb [45], a score based algorithm, has several variations: *CombMIN* (Eq. 2.16), *CombMAX* (Eq. 2.17), *CombSUM* (Eq. 2.18), *CombANZ* (Eq. 2.19) and *CombMNZ* (Eq. 2.20).

$$CombMIN(i) = min(s_{1i}, s_{2i}, \ldots, s_{ki}) \tag{2.16}$$

$$CombMAX(i) = max(s_{1i}, s_{2i}, \ldots, s_{ki}) \tag{2.17}$$

$$CombSUM(i) = \sum_{k}^{K} s_{ki} \tag{2.18}$$

$$CombANZ(i) = \frac{CombSUM(i)}{x_i} \tag{2.19}$$

$$CombMNZ(i) = CombSUM(i) \times x_i \tag{2.20}$$

As shown in Eq. 2.16- 2.20 the relevance value for a given the query is s_{ki}. The number of systems in K that retrieved i is denoted as x_i. This Thesis uses *CombMIN*, *CombMAX*, *CombSUM*, and *CombMNZ* to combine different similarity metrics using contextual information.

2.8.2 Learning to rank

Besides ranking fusion, it is possible to use machine learning to appropriately rank elements using learning to rank.

Learning to rank systems are split into supervised learning to rank and semi-supervised learning to rank. Supervised learning to rank relies on machine learning to train a model. The model is trained knowing the correct ranked list expected to be produced. Semi-supervised learning to rank also trains a model using machine learning. The data used to train has two parts, one which the correct ranked list expected to be produced is known

and the other which the correct ranked list is unknown. Meaning, the ranked list is unknown since it is not possible to verify if the truly correct rank list generated by the learning to rank system.

Due to the characteristics of the data used in this Thesis, rank fusion is more suitable to combine different neighborhoods, generated by different similarity metrics using different contextual sources of information. It is impossible to know the most accurate neighborhood to be used for rating prediction, and possibly it varies over time as new information enters the RS. For this reason, supervised learning to rank systems cannot be used for this specific application.

2.9 Summary

This Chapter allows to broadly understand some basic concepts and how diverse RS approaches are. It explains the key concepts of Fuzzy systems, in the context of this Thesis and discusses ranking fusion and learning to rank techniques. More specifically, this Chapter explains how a RS using CF operates to determine recommendations to its users, throughout the use of a similarity metric and rating prediction formula. It is followed by an explanation of Content-based filtering and Hybrid filtering approaches. Following, Fuzzy systems and concepts such as fuzzy logic, fuzzy sets, membership functions, t-norms, and t-conorms are explained. Finally, a brief discussion of learning to rank and ranking fusion approaches is presented and shown how they can be used to combine different ranking sets.

Chapter 3

Related work

This Chapter presents a literature review of four topics related to this Thesis: Memory-based Collaborative Filtering, similarity metrics, with focus on computational complexity, Model-based Collaborative filtering solutions, Recommendation systems using Fuzzy Systems, and previous works using Fuzzy Fingerprints in problems of other domains.

3.1 Memory-based Collaborative filtering

Similarity metrics between items (or users) are a central part of RS research [6]. Traditionally, the similarity is measured using metrics such as Pearson Correlation (PC) or the Cosine similarity (COS) [4]. Nevertheless, many other ways of measuring similarity have been proposed, ranging from simple variations of PC and COS, through the design of more complex functions [6, 30, 49, 50, 51].

In [37], for example, the authors argue that using PC alone is not enough to capture relevant statistical features of the data. To correct this, they propose a combination of the mean squared difference between the user's ratings and the Jaccard coefficient. Thus, capturing the similarity between users, while taking into account the number of items they have ranked in

common. The similarity metric proposed is called Jaccard Mean Squared Difference (JMSD) and is defined in Eq. 3.1:

$$sim_{JMSD}(i,g) = Jaccard(i,g) \times \big(1 - MSD(i,g)\big) \tag{3.1}$$

where *Jaccard* and *MSD* are defined as:

$$Jaccard(i,g) = \frac{|U_i \cap U_g|}{|U_i \cup U_g|} \tag{3.2}$$

$$MSD(i,g) = \frac{\sum_{u \in U}(r_{u,i} - r_{u,g})^2}{|U|} \tag{3.3}$$

where U_s is the set of items ranked by user s. Through experiments, the authors demonstrate that results are improved when compared to traditional CF. In particular, this metric allowed the RS to use fewer neighbors in comparison to COS and PC, according to the experiments carried on three datasets [37].

The JMSD similarity in [37] was later modified to be implemented via hardware allowing to speed up its calculation [15]. As in this Thesis, the authors argue that one of the problems with CF is the execution time required to produce recommendations. It relies on *k*-nearest neighbors and the non-scalable properties of the algorithm since the processing time increases quadratically as the number of users and items increase. The authors also argue that real-world applications have large demand spikes, thus further increasing the need for more computational efficient similarity metrics. Their solution presents a few drawbacks that limit its use: (1) the hardware solution does not allow easy modification to introduce other types of information, such as content-based, context-based information or social information; and (2) it does not improve nor maintain the recommendation quality achieved by the similarity metric that inspired it [37]. Instead, the presented similarity metric *HwSimilarity* [15] can speed up

the similarity computation by sacrificing the similarity precision. Their work goes in depth into the process of creation of the formulas BitJaccard and BitMSD, both adaptations from Eq. 3.2 and 3.3, respectively. These two allow creating BitJMSD a subtraction of BitMSD from BitJaccard. The BitJMSD is an intermediate metric, developed by the authors, which deals with Boolean data and logical operations and has natural numbers subtractions. While the *HwSimilarity* metric deals with Boolean data and logical operations only. The *HwSimilarity* can reduce the processing time required by 39.83% in comparison to JMSD and 47.92% in comparison to COS. The quality loss from JMSD to HwSimilarity is between 3.87% and 11.11% depending on the dataset.

The work in [52] presents a novel RS, that exploits social information to improve the results produced by traditional CF. This social information is the concept of *trust*, i.e. trusted neighbors explicitly specified by users. Cold-start problems lead to user preference models with lack of information. To address this, the authors use the trust between users within the system to help improve recommendation quality. The authors also show that users with few ratings also have few trusted neighbors. For this reason a *web-of-trust* is used, where trust is propagated between users. Their approach uses a weighting factor that reduces the trust degree as the distance between trusted users increases. Results show improvements of 53% in comparison to user-based CF using PC as the similarity metric.

The authors in [26] present a modified Pearson Correlation formula. This modified similarity metric adds constraints and positive or negative adjustments to the similarities. It considers the number of users and items existing in the rating matrix and uses a series of six equations selected depending on the constraints. The constraints take into account the number of co-rated items and relevance threshold of the PC. Another component of this work is the use of dynamic multi-levels of similarity. This solution requires providing characteristics regarding the database being used,

and manual experimentations of different parameters to optimize the similarity metric. The authors provide an extensive evaluation, showing that the proposed similarity metric allows for improvements in the quality of recommendations and rating prediction, in comparison with PC. The similarity metric was able to outperform in terms of MAE, Precision, Recall, and F1-score, all of baselines approaches on two datasets used for the experiments.

Liu et al. [6] evaluate various similarity metrics, pointing out their advantages and drawbacks. Taking these into account, they propose a new heuristic similarity (NHSM), which results from a combination of similarities including the Jaccard coefficient and Proximity-Significance-Singularity metric [51]. NHSM can provide good recommendation quality and combines heuristics to provide a meaningful similarity. The authors provide an in-depth comparison of Precision and Recall between NHSM and ten other similarity metrics, in which JMSD is included.

In [53] authors argue that the generation of user neighborhoods can be achieved through a Naive Random Neighbor selection, instead of the *k*-nearest neighbors as usual. Their algorithm selects a set of possible neighbors and stops when either the maximum number of neighbors is selected, or a certain threshold of similarity is reached. This filters neighbors with a lower correlation but, at the same time, reduces the number of similarities computed by setting a confidence interval of similarities used per rating prediction. Their results were the third best performing approach in comparison to the other baselines.

An alternative method, named *M-distance based recommendation* (MBR), allows the system to determine neighborhoods in linear time [17]. The authors leverage the average rating of each item and use the difference of such averages as the distance between items. The MBR metric uses a different principle, as shown in Eq. 3.4. It starts by computing the average rating $\bar{r}_j$ of each item j. The absolute value of the difference between these

average ratings (called *MBR*) determines the distance between the items. The set of neighbors H_i of item i is defined as all items $j \neq i$ such that $MBR(i,j) \leq T$, where T is a predefined threshold. Rating predictions $\hat{r}_{u,i}$ for user u and item i are the average of all ratings given by u to items in H_i.

$$MBR(i,j) = |\bar{r}_i - \bar{r}_j| \qquad \hat{r}_{u,i} = \frac{\sum_{j \in H_i \cap U_u} r_{u,j}}{|H_i \cap U_u|} \tag{3.4}$$

MBR has a lower computational cost than COS, PC, and JMSD. Since the average item rating can be pre-computed, the computational cost of MBR is O(1). It should be noted that MBR is not a similarity metric, but a distance metric between two items.

In [36] authors argue that contextual information can improve recommendation using a concept named singularity. The main idea of their work is that each item rating contribution to the similarity value of two users should not be considered absolute. As not every rating has the same relevance for every similarity computation. The authors explore the concept of singularity, which exploits embedded information of users rating patterns, e.g. if two users rated it negatively and every other user rated an item positively. Then these two users have a particularity in comparison to all the other users. Such information increases the quality of the similarity value between the two. The singularity metric can improve both prediction and recommendation quality in comparison to PC.

The above works, and others [15, 50, 51, 54, 55], show that improving the similarity metrics has a beneficial impact on the overall RS results. Nevertheless, this is often done at the expense of an increase in computational complexity. This Thesis introduces a similarity metric based on Fuzzy Fingerprint (FFP), adapted for item-based CF, that aims to improve time-efficiency, allowing for other sources of information besides ratings, while maintaining a low implementation effort and a comparable, or even better, recommendation accuracy.

3.2 Model-based Collaborative filtering

Instead of using Memory-based Collaborative filtering techniques, there is also the possibility of using Model-based [4] techniques, such as Matrix Factorization (MF). MF became well known after the results achieved in the Netflix challenge [31]. It can include, besides Collaborative filtering information, temporal information, social information among other contextual information [4].

In [31] the authors explain how MF is adapted to incorporate ratings and timestamps, and propose a Matrix Factorization model named SVD++. This model allowed the authors to be top-ranked during the 2007 Netflix Progress prize [31], achieving an improvement of 8.43% in comparison to the Netflix recommendation algorithm at that time. The temporal dynamics of ratings were taken into account using the assumption that, in general, item popularity varies over time but also each user taste changes over time, affecting their rating behavior. The authors show that temporal components have a crucial impact on Matrix Factorization models allowing them to improve rating prediction. When using the model with temporal dynamics, the rating prediction improvements were around 1% in RMSE.

The winning solution of the Netflix challenge, winning a prize of 1 million dollars, achieved an RMSE of 0.8563 in comparison to the Netflix RS which had an RMSE of 0.9514, a reduction of 10% in terms of RMSE. However, this solution was never used by Netflix, due to the engineering effort required to implement such a system. The winner solution resulted from the joint efforts of some of the top tier teams that participated in the Netflix Challenge.

The model was later improved creating the Asymmetric-SVD algorithm [56], which allows using fewer parameters and providing recommendations to new users, without requiring to re-train the model and estimate new parameters. It also allows explainability of recommendations,

by not abstracting users with an intermediate layer and the integration of implicit feedback to the model.

In [57], authors focus on the application of MF to RSs and propose a modification to the stochastic gradient descent optimization algorithm. It requires storing the entire data and continuously use and store the whole dataset while adding new data that enters the system. Instead of using stochastic gradient descent, they modify it creating an incremental stochastic gradient descent suitable for real-world applications, where there are streams of data continuously entering the RS. The incremental stochastic gradient descent presents the following differences: each new sample can be iterated multiple times to improve the accuracy, and the data used for learning does not require shuffling samples as in stochastic gradient descent. The proposed solution was tested on datasets using implicit feedback and shown to be faster and providing competitive accuracy to the baselines used. The authors in [57] focus on incremental solutions for RSs, such incremental solutions focus on providing fast RSs that support real-world applications with continuous streams of inputs and recommendation requests [58, 59, 60].

Over the years other alternatives have also been presented such as SocialMF [32], TrustMF [61], and TrustSVD [62]. These use Matrix Factorization and exploit the concept of trust. As previously explained, trust is defined as the degree of trustworthiness value between two users. The concept allows to correlate users better and often aids in the cold-start and sparsity problems [32]. This is useful for users that only interact on a social network and do not provide explicit information such as ratings.

Another highly relevant model-based CF is the use of Factorization Machines [63, 64]. This model-based approach is a modified polynomial regression, able to control the dimensional constraints. A significant difference between Factorization Machines and other factorization methods is that it was designed to be a general predictor. Thus, it can incorporate

arbitrary categorical and numerical variables, which are applied in a polynomial regression model. Such a model is capable of dealing with sparse data, by performing a factorization of the interactions between variables. Besides, variables in a Factorization Machine model can represent any information. One of the advantages of Factorization Machines over other factorization methods is that it allows the recommendation process to be treated as any other machine learning problem, where the model can be fine-tuned through feature engineering. Later, this model was adapted to easily incorporate relational data, allowing a reduction of computational complexity using a block structure for the input data [65].

In [66], the authors modified Factorization Machines, allowing the training phase to occur in a distributed fashion, using asynchronous stochastic gradient descent to perform the optimization. The proposed model allowed up to 8 times speedup (using 16 machines) while retaining a marginally lower prediction accuracy. Also, the model convergence required two times fewer iterations than the original Factorization Machine model according to the experiments.

3.3 Fuzzy Recommendation Systems

Fuzzy systems are present in many domains, such as the automotive [67], mobile networks [68], medical applications [69, 70, 71], among others. The main idea behind fuzzy systems is to represent truth values using continuous states, instead of only *true* or *false*, thus allowing a more accurate representation of the world. The ideas behind Fuzzy Systems have also been previously applied to RS. For example, in [72] a Neuro-Fuzzy Pedagogical Recommender, using an adaptive RS based on neuro-fuzzy inference, was proposed. This RS can aid students providing pedagogical content to them, but also teachers since they have the flexibility to modify the pedagogical model created. The system represents a user (student) using fuzzy

set theory, which allows for easy conversion from the pedagogical rules into a fuzzy computational model. The authors claim that the proposed system can automate the creation and configuration of the neural network based RS.

A Fuzzy-genetic approach is proposed in [73] to implement an Hybrid RS. Their work aims to reduce computational complexity while improving recommendation quality for movies. Each user is represented by a fuzzy model using features such as age, gender, occupation, and interest in each genre. These create a membership function and allow using fuzzy distance functions to compute the degree of similarity between two users. The model was then subjected to a genetic algorithm that captures the optimal weights of each feature, creating a hybrid fuzzy-genetic RS. The authors state, in their work, that the RS is limited by the computational complexity introduced by the genetic algorithm since each user requires its own genetic algorithm. The improvements achieved in terms of recommendation quality are significant, being as high as 25% of mean average error, in comparison to PC.

In [74], fuzzy set theory is combined with Bayesian networks to create intuitive representations of relations between users. To do so, a fuzzy representation of item ratings is used to create a probabilistic distribution of an expected rating. The fuzzy representation enables to represent the ambiguity and vagueness of ratings. A Bayesian network combines these representations to obtain a degree of relationship between users. The proposed RS improved the rating prediction and recommendation quality in comparison to the baselines.

In the work of Son [75], fuzzy sets are used to combine CF using fuzzy context to represent the demographic information of each user. This produces a similarity metric, later combined with Pearson Correlation (PC) and resulting in a Hybrid RS named HU-FCF. Results have shown that the system has higher accuracy depending on the configuration used. Their

system was able to outperform their baselines by up to 33%, while the computational complexity was increased approximately by 33% and 66% depending on the baseline used.

Later, Son addressed the cold-start problem and proposed an hybrid method named HU-FCF++ [76]. The system uses NHSM [6] on a rating matrix filled with rating predictions that are determined using Association Rules Mining. Association Rules Mining intends to find patterns, correlations, and associations between elements on a dataset [77]. These rating predictions result from a clustering process of users using demographic information, the clusters allow determining similar items to those for which a rating is being predicted. This set of techniques introduces a high demand for computational resources (in comparison NHSM requires only 4% of the time computational time) while the RMSE improvements of HU-FCF++ are only of 1.44%.

3.4 Fuzzy Fingerprints

This Thesis applies several concepts of Fuzzy Systems to the problem of item-based Collaborative Filtering. More specifically, the concepts of Fuzzy Fingerprints [18, 20] to represent items (or users) in a CF system.

Fuzzy Fingerprint (FFP) has been used for various tasks. In [20] the authors produced a text authorship identification system using Fuzzy Fingerprints. The proposed method allows matching texts with the corresponding author's text, according to features generated from words and stylometric characteristics, i.e. number of words used per clause or number of clauses per sentence. These features form a Fuzzy Fingerprint of each user and each text. Using a similarity metric, the most probable author for each text can be determined. Results have shown accuracy in detecting the correct author between 55% and 60%. The authors also concluded that this method could be adapted or modified to be applied to

other domains.

In [19] Fuzzy Fingerprints are used to identify mobile users based on call logs. The system requires a set of call logs from each user and then creates a Fuzzy Fingerprint for each. The features used to represent the user are a Top-N destination number call frequencies; and the representation is generated using a Filtered Space-saving, an algorithm that provides a fast and compact solution for the top-k problem but providing an approximate solution. By using it allows for a fast and compact approximation answer to the Top-N destination number calls, such approximation is not problematic for this application. The authors also state that the system can provide: fast comparison to identify a new session owner; scalability, i.e. the performance should not degrade, significantly, as the number of existing user sessions increases; flexibility allows for the incorporation of new users when enough information exists.

In [21] FFPs are used to create a topic detection method for microblogging (e.g. Twitter posts). Each Fuzzy Fingerprint represents a topic on Twitter, and each topic is represented using the most relevant words used of that topic. The selection of the most relevant words is made using the inverse document frequency (see Eq. 2.10) but now using Twitter topics as documents. The authors compared their method against k-Nearest Neighbours and Support Vector Machines. Their findings show that the proposed method provides a similar precision and 20% better recall in comparison to the best performing baseline which is Support Vector Machines. The training process is ten times faster in comparison to Support Vector Machines, and five times faster in comparison to k-Nearest Neighbours.

Later in [18], Fuzzy Fingerprints allowed to create a classification system that could classify unstructured texts into 42 categories and was able to outperform methods such as Support Vector Machines and Multinomial Naive Bayes algorithm. Unstructured texts are preprocessed and con-

verted to TF-IDF representation, on existing documents from the dataset Crunchbase. The dataset contains information about start-up companies, people and investors. The CrunchBase dataset contains about 650k profiles of people and companies and is maintained by tens of thousands of contributors. Descriptions consist of unstructured text, and all the information can be consulted using the CrunchBase API. The system relies on inverse *class* frequency, an adaptation of inverse document frequency to allow representing each existing category. The proposed method achieved between 7% and 20% accuracy improvements in comparison to baseline text classifiers. Experiments compare the results with exclusion or inclusion of the category 'others", by including the category, it degrades the accuracy on all the baselines. Fuzzy Fingerprints outperform by 7% the best performing baseline the Updatable Multinomial Naive Bayes.

In [22] the authors used Fuzzy Fingerprints to determine events mentioned in short sentences. The authors emphasize that the method outperforms Support Vector Machines, being able to determine all the twenty-six different types of events, while Support Vector Machines were able to determine only 60% of the existing type of events. The authors also refer to the computational advantages of using Fuzzy Fingerprints instead of Support Vector Machines, claiming it is more than 20 times faster.

Finally, in [78], fuzzy tools such as fuzzy sets and a fuzzy linguistic approach [79] are used to improve recommendation accuracy, by managing what the authors named *natural noise*. Natural noise results from the users' errors during item evaluation. Their work is compared to other noise reduction methods and shows superior performance. There is some similarity between [78] and the work developed on this Thesis as both aim to disregard information that does not help the RS to generate recommendations. However, this Thesis aims to filter unreliable ratings while improving the recommendation quality and seeking to reduce the computational complexity, whereas in [78] ratings are modified to reduce noise.

3.5 Summary

This Chapter provided a review of current similarity metrics being used in memory-based CF and interesting proposed solutions to be used in memory-based CF, followed by a review of state of the art on model-based CF. Next, it focused on current solutions that apply Fuzzy Systems to Recommendation Systems. The last Section focuses on how Fuzzy Fingerprints were applied to other domains, but also the advantages of using Fuzzy Fingerprints.

Chapter 4

Fuzzy Fingerprints for Recommendation

When users evaluate items on a Collaborative filtering system, often their preferences are provided as ratings. Ratings are a discretization of the user opinion and usually attributed on a discrete scale. Replacing ratings by Fuzzy Fingerprints allows for a reduction of ratings used when computing similarities, and consequently improving computational efficiency.

This Chapter starts by presenting how FFPs can be used in Recommender Systems (RSs), more specifically in memory-based Collaborative Filtering (CF). It proceeds to explore if the proposed RS can incorporate contextual information regarding items and users to improve the recommendation quality further, while still improving the computational efficency.

4.1 Building a Fuzzy Fingerprint representation

Fingerprints map an arbitrarily large object into a smaller and more compact representation [18]. This idea can be applied to items that are represented by a Fuzzy Fingerprint (FFP), which is then used to compute item

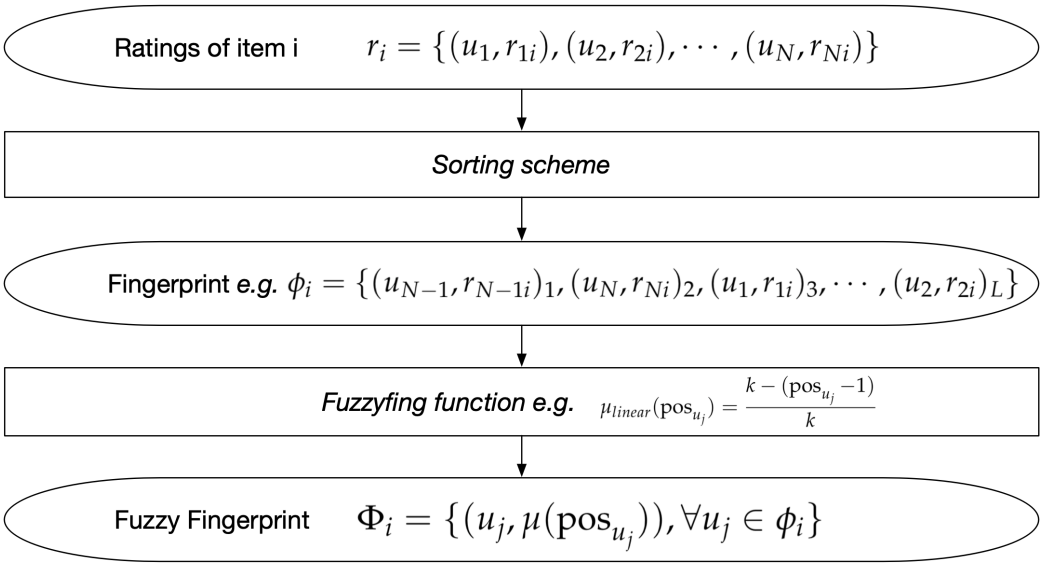


Figure 4.1: Steps required to create a Fuzzy Fingerprint. L corresponds to the total of ratings r_i.

similarities.

A Fingerprint, ϕ_i, of an item i is generated by using a criteria to select a subset of the ratings r_i assigned to i. To create ϕ_i a sorting scheme performs a ranking of the ratings relevance according to specific criteria. The produced ϕ_i is then reduced to only k ratings, these are the most relevant ratings based on the order determined by the sorting scheme. Once ϕ_i is determined, an FFP is computed, by applying a Fuzzyfying Function (FF) [80]. This function transforms the ordered set of ratings, i.e. ϕ_i into a fuzzy set, where a membership degree M is assigned to each rating. This set of M forms the FFP of item i, designated as Φ_i. Figure 4.1 illustrates the various steps and the corresponding result of each step. The following Sections go into detail of how a sorting scheme is created (Section 4.2) and how a fuzzifying function is applied (Section 4.3).

Traditionally, when predicting the rating that a given user u will assign to a given item i, item-based CF systems compute the items most similar

to i, by using the full set of ratings assigned to i. It is possible to replace this raw set of item ratings with the FFP of item i, while improving the recommendation quality and improving the computational efficiency, as detailed in the following Sections.

4.2 From Ratings to Fingerprints

Let r_i be the set of ratings that a given set of users $u_1 \cdots, u_N$ has provided for item i:

$$r_i = \{(u_1, r_{1i}), (u_2, r_{2i}), \cdots, (u_N, r_{Ni})\} \tag{4.1}$$

Building the Fingerprint ϕ_i for item i is a straightforward process. It starts by choosing a subset of k ratings in r_i, where k is the parameter that controls the size of the Fingerprint. The idea is that the selected ratings should be those that best represent item i, to this effect, selecting the k ratings with the highest value. However, on most RSs, users provide ratings on a small discrete scale (e.g. 1, 2, 3, 4, or 5 stars), and FFPs need to distinguish the relevance of ratings from different users. Thus, sorting equal ratings is a crucial part of this work. To sort such ratings sorting alternatives are proposed. Each of these sorting alternatives is named sorting scheme (SS).

The sorting schemes considered to rank ratings by their relevance for representing each item are:

- Random sorting scheme, which ranks ratings from the highest value to the lowest and, when two ratings are equal, orders them randomly;
- Higher to Lower (HL) sorting scheme which ranks ratings from the highest to the lowest value and, when equal, ratings from more active users are placed first;
- Lower to Higher (LH) sorting scheme that ranks ratings from the

lowest to the highest value, and when equal, ratings from less active users are placed first;

- Ratings and Words (R&W) sorting scheme that ranks ratings together with item descriptions (words) from the highest relevance to the lowest;
- Timestamp-aware sorting scheme that ranks ratings from the highest value to the lowest, and equal ratings are sorted according to their timestamp, i.e. more recent ratings are placed first;
- Timestamp-only sorting scheme that ranks ratings from most recent to the oldest.

The remainder of this Section contains a more detailed explanation of each of the proposed sorting schemes.

Figure 4.2 illustrates an example of r_i, where the first row contains the *user ids* and the second row the *ratings*. For the following explanation, it is assumed that $\#x$ is the number of ratings of user x, and that the number of ratings per user is ordered alphabetically, i.e. $\#a > \#b > \cdots > \#U$.

$\mathbf{u_j}$	a	b	c	d	e	f	g	h	i
$\mathbf{r_{ji}}$	5	2	-	5	4	2	-	1	2

Figure 4.2: The set of ratings r_i for item i.

4.2.1 Random sorting scheme

The first approach, Random can be considered as a baseline to determine if the rating scheme has any influence on the quality of the chosen Fingerprint. The Random SS, in Fig. 4.3 is, therefore, the most simplistic SS presented. The order of users d and a is selected *randomly* since both rated item i with 5. The users f, i and b are also *randomly* positioned since they all rated item i with 2.

$\mathbf{u_j}$	d	a	e	f	i	b	h
$\mathbf{r_{ji}}$	5	5	4	2	2	2	1

Figure 4.3: Fingerprint ϕ_i, resulting from a Random SS.

4.2.2 Higher to Lower sorting scheme

The Higher to Lower (HL) SS retains ratings from most active users, i.e. those with the highest number of ratings. The idea is that they are expected to be more engaged in providing accurate information, therefore being prioritized over less active users. The resulting Fingerprint is shown in Fig. 4.4. The order of the users a and d is selected taking into account who has rated *more* items. The same is done for sorting users b, f and i which rated item i with the same rating.

$\mathbf{u_j}$	a	d	e	b	f	i	h
$\mathbf{r_{ji}}$	5	5	4	2	2	2	1

Figure 4.4: Fingerprint ϕ_i, resulting from a Higher to Lower SS.

4.2.3 Lower to Higher sorting scheme

On the other hand, the Lower to Higher (LH) SS retains ratings from less active users. Based on the idea that they can be expected to be more coherent in providing accurate ratings, and thus more reliable information to compute similarities. In Fig. 4.5 the resulting Fingerprint for item i is presented, the order of the users d and a is selected taking into account who rated *less* items since both rated the item i. The same applies to users b, f and i.

$\mathbf{u_j}$	d	a	e	i	f	b	h
$\mathbf{r_{ji}}$	5	5	4	2	2	2	1

Figure 4.5: Fingerprint ϕ_i, resulting from a Lower to Higher SS.

4.2.4 Timestamp based sorting schemes

Considering the movie domain, popular movies tend to have a rating too high or too low when released. This is often caused by the hype around them and after some time, it converges to a more stable average rating, as the number of existing ratings increases. This Thesis presents two sorting schemes timestamp based that can capture this property: the Timestamp-aware SS and the Timestamp-only SS, where items are being described by users that more recently rated them. Both the Timestamp-aware SS and the Timestamp-only SS are evaluated using general evaluation conditions for time-aware RSs, i.e. by discarding any temporal overlapping among training and the test data, as done by authors in [81]. Assuming that the timestamp of each user rating is as presented in Fig. 4.6, the resulting Fingerprint after applying the Timestamp-aware SS is as shown in Fig. 4.7.

$\mathbf{u_j}$	a	b	c	d	e	f	g	h	i
$\mathbf{t_{ji}}$	5-Nov	3-Nov	-	16-Dec	30-Dec	1-Nov	-	1-Dec	16-Dec

Figure 4.6: The set of timestamps t_i for item i.

$\mathbf{u_j}$	d	a	e	i	b	f	h
$\mathbf{r_{ji}}$	5	5	4	2	2	2	1

Figure 4.7: Fingerprint ϕ_i, resulting from a Timestamp-aware sorting scheme.

The order of the users d and a is now selected taking into account who *more recently* rated item i, since they both rated it with 5. The same applies to the order of users i, b and f which also rated this item with the same value.

In contrast, the Timestamp-only SS generates a different Fingerprint. It sorts users considering more relevant those that most recently rated the item, without having any consideration for the rating value. The resulting Fingerprint is presented in Fig. 4.8. In the event that two users ratings are done at the same time on the same item, the one with a higher rating is considered more relevant.

$\mathbf{u_j}$	e	d	i	h	a	b	f
$\mathbf{r_{ji}}$	4	5	2	1	5	2	2

Figure 4.8: Fingerprint ϕ_i, resulting from a Timestamp-Only sorting scheme.

The experiments with the timestamp-based sorting schemes did not present promising recommendation quality nor computational complexity improvements. For that reason, they are not included in Chapter 5.

4.2.5 Dimensioning a Fingerprint

In all the above SSs the idea is that a Fingerprint stores only the *user ids* and neglects the ratings or timestamp information. As a result, after computing the order, it is legitimate to say that the rating (or timestamp) information is embedded in the degree of relevance of each user to the item Fingerprint. The Fingerprint creation is only complete when the k most relevant ratings are selected and the remaining ratings discarded. The dimensioning the Fingerprint is what allows the improvement of the similarity metric computational efficiency. The similarities can be computed using, therefore, only the order and most relevant users that compose ϕ_i.

A fuzzifying function $\mu(idx)$ generates the Fuzzy Fingerprint using the Fingerprints, designed by the sorting scheme and after reducing it to at most k features. The following Section provides more details on the fuzzfication process.

4.3 Fuzzifying a Fingerprint

The Fingerprint ϕ_i is, in fact, an *ordered set* of k users. This order, determined by one of the sorting schemes, defined in the previous Section, and reflects the importance of each rating to represent an item. It is by leveraging on this importance that the Fuzzy Fingerprint Φ_i, of item i, is determined.

A Fuzzyfying Function (FF) $\mu(idx)$, also refered to as membership function (see Section 2.7), assigns a weight to each position in a Fingerprint. In this case, the Fuzzyfying Function (FF) is used to assign a weight to each user in ϕ_i. The size of each Fingerprint is not constant, it varies on each Fingerprint depending on the number of ratings the item has. A maximum FFP size k is defined, for example using $k = 4$, to truncate the Fingerprint in Fig. 4.5 obtaining:

$\mathbf{u_j}$	d	a	e	f
$\mathbf{r_{ji}}$	5	5	4	2

Figure 4.9: Fingerprint ϕ_i, resulting from a Random SS (see Fig. 4.3) truncated to $k = 4$ ratings.

There are many alternatives to define a FF [80]. Here, three possible FFs are shown in Equations 4.2 through 4.4. In each equation, pos_{u_j} is the position of user u_j within ϕ_i.

Function μ_{one} (Eq. 4.2), assigns an equal membership degree to all user ratings. It is used mainly as a baseline for comparison.

$$\mu_{one}(\text{pos}_{u_j}) = 1 \tag{4.2}$$

Using function μ_{linear} (Eq. 4.3), the membership degree of a user decreases linearly, according to its position pos_{u_j}.

$$\mu_{linear}(\text{pos}_{u_j}) = \frac{k - (\text{pos}_{u_j} - 1)}{k} \tag{4.3}$$

Finally, function μ_{erfc} (Eq. 4.4), uses a variation of the *complementary error function* to yield a faster decrease in membership degrees.

$$\mu_{erfc}(\text{pos}_{u_j}) = 1 - \text{erfc}(\frac{2 \times \text{pos}_{u_j}}{k}) \tag{4.4}$$

To better illustrate their impact, a plot showing the behavior of these functions can be found in Fig. 4.10.

It is important to note that these functions are not the only available options [20], as shown in Section 2.7. However, preliminary experiments have indicated that using other variations does not significantly improve the quality of the recommendations for RSs. For this reason, no further alternatives are presented in this Thesis.

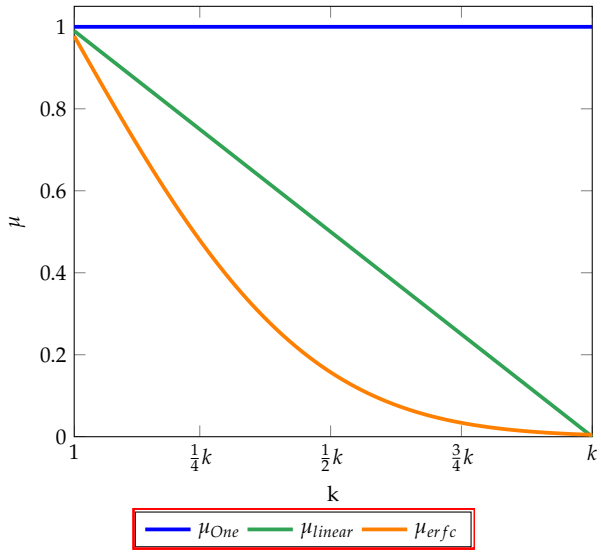


Figure 4.10: Plot of the three proposed Fuzzyfying Functions.

Using one of the above fuzzifying functions, FFP Φ_i is defined as:

$$\Phi_i = \{(u_j, \mu(\text{pos}_{u_j})), \forall u_j \in \phi_i\} \tag{4.5}$$

The FFP is the set of users in the Fingerprint, each with an associated membership degree, given by the Fuzzyfying Function. It is, in effect, a *fuzzy set of users* that ranked item i. An example is shown in Fig. 4.11, taking the Fingerprint from Fig. 4.3.

$\mathbf{u_j}$	d	a	e	f
μ	1	3/4	2/4	1/4

Figure 4.11: FFP obtained after applying μ_{linear}, to the Fingerprint in Fig. 4.3.

Notice that the FFP does not contain the item ratings. Instead, it contains only the value of membership degree, reflecting the rating position. In [20], when classifying textual documents, the authors argue that the relative weight of the words provides more useful information than their actual relevance. Similarly, this Thesis tries to exploit the relevance of ratings instead of their actual value.

4.4 Integrating Content-based Information

The detailed information regarding each item, e.g. item text description, can be used to represent the items better, leading to an improvement of recommendations. Moreover, content-based information can help address cold-start problems, where the item has not yet been rated. Either because it is not popular or because it is a new item listed in the RS. This information can be, for example, the synopsis of a movie or a detailed description of a product.

It starts by applying stemming and stop-word removal to create a dictionary of possible words. Then computing the respective $TF\text{-}IDF$ of each word, (see Eq. 2.11). Each item will thus be represented by a vector of $TF\text{-}IDF$ weights, as illustrated in Fig. 4.12.

word	w_1	w_5	w_{30}	w_{72}
TF-IDF$_{i,w_x}$	0.2	0.052	0.024	0.046

Figure 4.12: *TF*, *IDF* and *TF-IDF* representation of item i. Each w_j corresponds to a word present in the item's description.

Combining content-based information with user's ratings requires normalizing user's ratings to average zero. It replaces the original rating by a degree of how much the user liked the item when compared to his (or her) average rating $\bar{\mathbf{u}}_\mathbf{j}$.

To illustrate this process, Fig. 4.13 shows an example for an item i with ratings r_i, the average rating of each user is $\bar{\mathbf{u}}_\mathbf{j}$, and line $\mathbf{r_{ji}} - \bar{\mathbf{u}}_\mathbf{j}$ show the final value normalized.

$\mathbf{u_j}$	a	b	c	d	e
$\mathbf{r_{ji}}$	5	2	-	5	4
$\bar{\mathbf{u}}_\mathbf{j}$	3.5	3.4	3.4	3.6	3.5
$\mathbf{r_{ji}} - \bar{\mathbf{u}}_\mathbf{j}$	1.5	-1.4	-	1.4	0.5

Figure 4.13: Ratings (r_i) of item i, average rating of each user $\bar{\mathbf{u}}_\mathbf{j}$ and the rating of a user in reference to the average rating $\mathbf{r_{ji}} - \bar{\mathbf{u}}_\mathbf{j}$.

The next and final step is to combine the ratings and the item's *TF-IDF*. To do so, it is essential to normalize the values from both sources of information, so that they are on the same scale. This is achieved by performing a min-max normalization [82] to the ratings (having subtracted average rating) and the *TF-IDF* values, for all items, as shown in Eq. 4.6.

In Eq. 4.6, x_{new} is the resulting scaled value, x is the value being normalized, min the lowest value and max the highest value.

$$x_{new} = \frac{x - min}{max - min} \tag{4.6}$$

Experiments were also conducted using standardization (Eq. 4.7), where μ is the average of the data X and σ its standard deviation. Performing

standardization causes the standardized representation of the data to have zero as an average value and have a standard deviation of 1.

$$x_{new} = \frac{x - \mu}{\sigma} \tag{4.7}$$

The results obtained using standardization were outperformed by normalization. Thus, this Thesis only uses the min-max normalization.

Figure 4.14 shows the normalized vector obtained for the ratings from Fig. 4.13, considering $min = -2.0$ and $max = 2.5$. To determine the min and max values, the lowest value and highest value of X must be identified.

$\mathbf{u_j}$	a	b	c	d	e
Norm. $\mathbf{r_{ji}} - \bar{\mathbf{u}}_\mathbf{j}$	0.777	0.133	-	0.756	0.556

Figure 4.14: Min-max normalized ratings, for item i.

Figure 4.15 shows the normalized vector obtained for the text from Fig. 4.12, considering $min = 0.001$ and $max = 0.2$, these correspond to the min and max values, from the $TF\text{-}IDF$ matrix.

$word$	w_1	w_5	w_{30}	w_{72}
Norm. $TF\text{-}IDF_i$	1	0.2563	0.1156	0.2261

Figure 4.15: Min-max normalized $TF\text{-}IDF$, for item i.

Once the matrices resulting from the ratings and textual information are normalized, a Fingerprint can be generated by concatenating both representations, ordering them, and keeping only the k features with the highest values. Figure 4.16 shows the formation of a Fingerprint that combines the normalized ratings and the normalized $TF\text{-}IDF$ of item i before truncating the features to k.

Figure 4.17 shows the resulting Fingerprint for item i, with k equal to 5.

feature	w_1	a	d	e	w_5	w_{72}	b	w_{30}
ϕ_i	1	0.777	0.756	0.556	0.2563	0.2261	0.133	01156

Figure 4.16: Fingerprint ϕ_i using R&W sorting scheme with all its features.

feature	w_1	a	d	e	w_5
ϕ_i	1	0.777	0.756	0.556	0.2563

Figure 4.17: Fingerprint ϕ_i using R&W sorting scheme, and with k equal to 5.

After obtaining an item Fingerprint using the R&W sorting scheme, to compute the FFP a FF needs to be applied to the Fingerprint in Fig. 4.16. Using the μ_{linear} FF described in Section 4.3, the FFP of Fig. 4.17 is obtained.

4.5 Comparing Fuzzy Fingerprints

With two or more FFPs determined, it is possible to compute similarities between items. Consider Φ_i and Φ_j the FFPs of items i and j, respectively. Let U_i be the set of users in Φ_i and U_j be the set of users in Φ_j. The FFP similarity between items i and j is defined by:

$$\text{sim}(\Phi_i, \Phi_j) = \sum_{u_v \in U_i \cap U_j} \frac{\min(\Phi_i(u_v), \Phi_j(u_v))}{k} \tag{4.8}$$

where $\Phi_x(u_v)$ denotes the value associated with user u_v in Φ_x. Equation (4.8) defines the similarity between two FFPs.

The above similarity metric uses the sum of the lowest values from each FFP depending on the users in common between items i and j. This corresponds to a *Gödel t-norm* (see Section 2.7) between the two fuzzy sets represented by the FFPs. There are other possible t-norms such as product t-norm, drastic t-norm, or Hamacher t-norm among others [83].

The idea is that, for two items to be similar, they must have been rated

$\mathbf{u_j}$	a	i	d	h
μ	1	3/4	2/4	1/4

Figure 4.18: Item j FFP, Φ_j.

by the same users and the ratings given by those users must have the same relative importance. Equation 4.9 illustrates the computation of Eq. 4.8 for the similarity between the FFP from item i (Fig. 4.11) and FFP from item j (Fig. 4.18).

$$\begin{aligned} \text{sim}(\Phi_i, \Phi_j) &= \frac{\min(\Phi_i(a), \Phi_j(a))}{k} + \frac{\min(\Phi_i(d), \Phi_j(d))}{k} \\ &= \frac{\min(\frac{3}{4}, 1)}{4} + \frac{\min(1, \frac{2}{4})}{4} \\ &= \frac{\frac{3}{4}}{4} + \frac{\frac{2}{4}}{4} = 0.3125 \end{aligned} \tag{4.9}$$

4.6 Computational Efficiency

It is important to make some remarks on the computational efficiency of the similarity metric when compared to the traditional Cosine similarity and Pearson Correlation methods, and with the state of the art methods such as Jaccard Mean Squared Difference [37]. More specifically remarks in regard to the complexity of:

1. computing a rating prediction for a given user-item pair (u, i);
2. create a Fuzzy Fingerprint;
3. maintaining and updating FFPs as the rating matrix changes.

For the first point, the similarity between item i and all other items rated by user u must be computed. It occurs independently of the similarity metric used. Thus, this Thesis focuses on the complexity of computing each single similarity value between pairs of items, $\text{sim}(i, j)$. The

proposed approach, using FFP similarity with Eq. (4.8) costs $O(k)$, where k is the maximum size of the Fingerprints. For the baseline metrics, the average cost is $O(q)$, where q is the number of ratings between a pair of items. FFPs using solutions that rely on ratings or timestamps (e.g. LH SS and HL SS) always have $k \leq q$. For solutions such as R&W SS, k can be higher than q, since incorporating contextual features increases the size of each FFP. Nevertheless, it is expected that for any of the proposed SSs and RS approaches for k to be much smaller than q, and that there is a computational gain to be achieved, as shown later in Section 5.5.6.

Unlike the baseline metrics, the proposed FFP similarity has the additional cost of creating Fuzzy Fingerprints. Building an FFP, as explained in Section 4.1, requires collecting the top-N ratings of each item. It can be done with an average cost of $O(m \log k)$, where m is the number of ratings per item. It is higher than the baselines which have a cost of merely retrieving the ratings, i.e. $O(m)$. This operation can, however, be done offline, thus having no impact on similarity computation. This is related to the third point, maintaining the user-item rating matrix. Using an appropriate data structure, such as a B-tree, all ratings can be stored already ordered, thus reducing the cost of retrieval to $O(log(k))$. Updating and inserting new ratings is still entirely feasible, also having a logarithmic cost of $O(log(k))$ with a B-tree representation for each FFP.

Fuzzifying the Fingerprint into a Fuzzy Fingerprint has a cost of $O(k)$. The fuzzifying function is a continuous function yet, on this application, it maps the discrete position of the k features of any item, as result a simple table can store the corresponding values from 1 to k of the FF.

RSs have a high number of users and items, and these are constantly increasing. The FFP similarity metric can incorporate new items and users easily. If a new item enters the RS it starts with zero information in its FFP, as users rate the item or contextual information is added, it begins to form a Fingerprint with more and more features, until it reaches the maximum

k features.

The most relevant part regarding to computational complexity is the similarity computation cost, which is lower using the proposed solution in this Thesis (see Section 5.5.6). This fact, together with the low engineering effort required to implement it, makes this Thesis a practically off-the-shelf, effective solution for large-scale Recommender Systems.

4.7 Adapted Memory-based CF using Fuzzy Fingerprints

As discussed in Chapter 2, Memory-based Collaborative filtering can be user-based or item-based. This Thesis focuses mostly on item-based CF using FFPs. RSs traditionally rely on a rating prediction to determine which items to recommend to users. This Section proposes a solution for a user-based CF that does not make rating predictions. Instead, this solution identifies similar users using FFP similarity, based on keywords associated with items that users rated.

Let N be the total number of keywords in the system and let M be the total number of items in the system. Let also θ_i represent the set of keywords of a given item i: $\theta_i = (t_{1i}, t_{2i}, t_{3i}, \cdots, t_{Ni})$. Any element $t_{ni} \in \theta_i$ can assume the value 1 if the keyword is associated with that item, or 0 if it is not.

Let r_u be the set of ratings for a given set of items $i_1 \cdots, i_M$, provided by a user u: $r_u = (r_{1u}, r_{2u}, \cdots, r_{Mu})$. It is assumed that $r_{mu} \geq 0$ and that a value of zero means that the user has not yet rated item i_m. A Fingerprint ϕ_u is built by counting, for user u, the number of occurrences of each keyword in the items rated by u, multiplied by the respective item's rating,

i.e. $\phi_u = (c_{1u}, c_{2u}, \cdots, c_{Nu})$, where:

$$c_{nu} = \sum_{\forall i=1}^{M} t_{ni} \times r_{iu} \tag{4.10}$$

The rationale behind Eq. 4.10 is that keywords from items a user has rated higher should also get higher importance in the Fingerprint. The next step consists in ordering ϕ_u according to c_{nu} and keeping only the k highest values.

To illustrate the previous procedure, let $r_u = (5, 2, 4)$ for items a, b, and c. Assume there are only 5 keyword and let $\theta_a = (1, 0, 0, 1, 1)$, $\theta_b = (0, 1, 0, 0, 1)$, and $\theta_c = (0, 0, 1, 1, 0)$. Assuming that $k = 4$, the resulting Fingerprint ϕ_u will be $(c_{4u} = 9, c_{5u} = 7, c_{1u} = 5, c_{3u} = 4)$.

The Fingerprint ϕ_u is, therefore, an *ordered set* of keywords. The rank of each keyword reflects its importance for the user. This Fingerprint still needs to be fuzzified into a Fuzzy Fingerprint. As before, the fuzzification of the Fingerprint leverages the importance of the order (and not of the frequency) to represent users. The FFP of user u, Φ_u, is obtained by fuzzifying the rank (the position in the Fingerprint) of each keyword, as shown in Section 4.3. The FFP similarity computation, now user-based, is done using Eq. 4.8.

The recommendation process of the proposed RS, as stated previously, does not rely on rating predictions as in traditional Collaborative Filtering. Instead, it identifies the user's nearest neighbors (according to Eq. 4.8) and uses the items seen and liked by them to extrapolate possible items to recommend to the user.

The RS starts by computing which users are the nearest neighbors of user u, based on the FFP similarity metric. Users are considered neighbors if the similarity is higher than a defined threshold $sim_{\text{threshold}}$. It is considered that any item rated highly by a neighbor (e.g., 4 or 5 on a 0-5 scale) and rated higher than that same neighbor's item rating average, is suited

to be recommended.

The final step in the recommendation process consists of getting the difference between the rating of the recommendable item, the neighbor average rating, and multiplying it by the similarity between the user and the neighbor. It allows creating a ranking of recommendable items, for each user.

4.8 Multi-Context Fuzzy Fingerprints for RSs

Sorting schemes are responsible for ranking features according to their relevance. Ranking features that are on different domains such as ratings, timestamps of ratings, item textual descriptions and keywords, or users profile characteristics, among others are hard to combine into a single FFP. This is due to the different representation domain of each contextual information source. Two previous attempts to use contextual information proposed in this Thesis are the use of *R&W* sorting scheme (see Section 4.4) and in Section 4.7 through a different RS approach that skips the rating prediction task.

This Section proposes a new way to represent users. Instead of using normalization to combine contextual information into one FFP, create several FFPs, one for each of the contextual information source. Doing so requires computing the similarity between users according to each contextual information source.

The number of existing features limits the computational complexity involved. Even though there are more similarity metrics to be computed the total number of existing features is the same. On each FFP the available contextual information is less.

Each user has as many FFPs as the available contextual information sources. The similarity between two users is computed for each contextual information using Eq. 4.8. Figure 4.19 illustrates an example with the

similarity of user u_n with six other users over four context FFP similarities.

	u_1	u_2	u_3	u_4	u_5	u_6
sim_{C_1}	0.72	0.32	0.52	0.23	0.63	0.54
sim_{C_2}	0.27	0.64	0.33	0.72	0.58	0.18
sim_{C_3}	0.74	0.81	0.18	0.65	0.74	0.35
sim_{C_4}	0.98	0.90	0.46	0.27	0.22	0.52

Figure 4.19: Contextual Fuzzy Fingerprint similarities sim_{C_x}, between user u_n and all other users that rated item i i.e. u_n possible neighbors.

For each context, the top-n neighbors are computed, resulting in the neighborhoods shown in Fig. 4.20.

	u_1	u_2	u_3	u_4	u_5	u_6
sim_{C_1}	0.72	-	-	-	0.63	0.54
sim_{C_2}	-	0.64	-	0.72	0.58	-
sim_{C_3}	0.74	0.81	-	-	0.74	-
sim_{C_4}	0.98	0.90	-	-	-	0.52

Figure 4.20: Top-3 nearest neighbors for each contextual FFP similarities, between user u_n and all other users that rated item i i.e. u_n possible neighbors.

Each neighborhood of u_n is, in this example, limited to three users, i.e. the top-3 neighbors. Using the neighborhoods from each context, the computation of a ranking fusion is obtained. Figures 4.21-4.25 illustrate the resulting neighborhood of u_n depending on the ranking fusion approach desired which include: *CombMin*, *CombMax*, *CombSUM*, *CombMNZ* and *CombANZ*.

In Fig. 4.21 *CombMIN* is used as ranking fusion to combine the obtained similarities of each context, for each user from Fig. 4.20. *CombMIN* combines the similarity values by selecting the minimum value of similarity from all contexts, for each user. Therefore, the resulting ranked neigh-

	u_1	u_2	u_3	u_4	u_5	u_6
$sim_{CombMIN}$	**0.72**	**0.64**	0.00	**0.72**	0.63	0.52

Figure 4.21: Contextual Fuzzy Fingerprint similarities $sim_{CombMIN}$, between user u_n and the possible neighbors.

borhood is u_1, u_4 and u_2, as highlighted in the Figure. There are no criteria to untie neighbors such as u_1 and u_4 which have the same value of similarity. The likelihood of such occurrence is low in real cases and should have a small impact on the overall quality of recommendations generated by the RS.

	u_1	u_2	u_3	u_4	u_5	u_6
$sim_{CombMAX}$	**0.98**	**0.90**	0.00	0.72	**0.74**	0.54

Figure 4.22: Contextual Fuzzy Fingerprint similarities $sim_{CombMAX}$, between user u_n and the possible neighbors.

Figure 4.22 uses $CombMAX$ as ranking fusion to combine the obtained similarities of each context for each user. $CombMAX$ combines the similarity values by selecting the maximum value of similarity from all contexts, for each user. Thus, the resulting ranked neighborhood is u_1, u_5 and u_2, as highlighted in the Figure.

	u_1	u_2	u_3	u_4	u_5	u_6
$sim_{CombSUM}$	**2.40**	**2.25**	0.00	0.72	**1.95**	1.06

Figure 4.23: Contextual Fuzzy Fingerprint similarities $sim_{CombSUM}$, between user u_n and the possible neighbors.

Figure 4.23 uses $CombSUM$ as ranking fusion to combine the obtained similarities of each context for each user. $CombSUM$ combines the similarity values by performing a sum of the similarity values from all contexts,

for each user. Therefore, the resulting ranked neighborhood is u_1, u_2 and u_5, as highlighted in the Figure.

	u_1	u_2	u_3	u_4	u_5	u_6
$sim_{CombMNZ}$	**7.20**	**9.75**	0.00	0.72	**5.85**	2.12

Figure 4.24: Contextual Fuzzy Fingerprint similarities $sim_{CombMNZ}$, between user u_n and the possible neighbors.

Figure 4.24 uses $CombMNZ$ as ranking fusion to combine the obtained similarities from each context for each user. $CombMNZ$ combines the similarity values by performing a sum of the similarity values of all contexts, for each user, as $CombSUM$ does. The resulting values are then multiplied by the number of contexts, in which the user is considered neighbor, as in Eq. 2.20. The resulting ranked neighborhood is u_2, u_1 and u_4, as highlighted in the Figure.

	u_1	u_2	u_3	u_4	u_5	u_6
$sim_{CombANZ}$	**0.8**	**0.75**	0.00	**0.72**	0.65	0.53

Figure 4.25: Contextual Fuzzy Fingerprint similarities $sim_{CombANZ}$, between user u_n and the possible neighbors.

Finally, Figure 4.25 uses $CombANZ$ as ranking fusion to combine the obtained similarities of each context for each user. $CombANZ$ combines the similarity values by performing a sum of the similarity values of all contexts, for each user, as $CombSUM$ does. The resulting value is then divided by the number of contexts, in which the user is considered neighbor, as in Eq. 2.19. The resulting ranked neighborhood is u_1, u_2 and u_4, as highlighted in the Figure.

These five different ranking fusion approaches produce different rankings for the neighborhood of user u_n. The effects of ranking fusion are observable and exploited on these examples. The chorus effect is present

since three contexts consider users u_1 and u_2 relevant as a neighbor of user u_n. The dark horse effect is observable where C_2 considers u_4 as relevant while the other three contexts do not and, for that reason $CombMIN$, $CombMNZ$ and $CombANZ$ have u_4 in the top-3 neighbors of user u_n. The skimming effect is seen where different contexts obtain different similarity values for the same pair of users. User u_5 is an example of the skimming effect if this was an actual system.

Each ranking fusion approach, presented in this Section, is suitable to replace the similarity metric used on the RS proposed in Section 4.7. This is the final goal of the multi-context FFP similarity metric here proposed.

This FFP similarity metric, used in conjunction with the RS proposed in Section 4.7, has several parameters that need to be fine-tuned according to the data they represent. This is a challenging task to achieve as the parameters that must be fine-tuned include: the number of neighbors used; the number of features used for each individual context FFP; the sorting scheme and FF used for each individual context FFP; the selection of a ranking fusion algorithm; and the similarity threshold to determine if a neighbor is relevant or not of the RS algorithm;

4.9 Summary

This Chapter details the application of Fuzzy Fingerprint on RSs. The use of FFP aims to reduce the amount of data used to represent users and items. To do so, sorting schemes are responsible for ranking features thus allowing to remove less relevant information from the FFP. From the presented sorting schemes, it is important to highlight the High-Low and Low-High sorting schemes, both of which rely only on ratings to generate a FFP. One of the goals of this Thesis is also the combination of contextual information for RS using Fuzzy Fingerprints. This goal is attempted firstly by using the *R&W* SS, secondly by the proposed RS in Section 4.7,

and finally by creating an FFP which combines several contextual FFPs representing its users using a multi-context FFP.

Chapter 5

Evaluation

Evaluation is an important task to consider when developing Recommender Systems. However, there are many variations of evaluation metrics and procedures, making it difficult to replicate every single detail of the process [84]. Problems with the evaluation of RSs are not new and often authors assume crucial parts of their process. For example, do not report if they use cross-validation, what they do when a rating prediction cannot be determined or how recommendations are evaluated.

This Chapter is structured as follows. Section 5.1 presents the datasets used in this Thesis experiments and their properties. In Section 5.2, the evaluation metrics used to measure the quality of rating predictions, recommendations, and improvements in the computational complexity when using the FFP similarity metric. Finally, Section 5.5 presents the experiments with the approaches proposed in Chapter 4 and discussion on the quality of these approaches.

5.1 Datasets

To assert the effectiveness of FFPs when applied to RSs, the solutions presented in this Thesis are compared to the baseline similarity metrics Cosine

Similarity (COS), Pearson Correlation (PC) [4], and state-of-the-art methods, jaccard mean square distance (JMSD) [37] and MBR [17].

Experiments were performed on four standard datasets: (1) MovieLens-1M (ML-1M), a dataset from the movie domain; (2) Netflix, a large dataset, also from the movie domain with a rating matrix that has high sparsity; (3) Jester, a dataset for recommending jokes, with a high number of ratings per item; and (4) Hetrec2011-ML, a dataset containing movie ratings, and URLs for the Internet Movie Database[1] (IMDB) and Rotten Tomatoes[2] that allows extracting contextual information (movie synopsis). Table 5.1 shows statistics regarding these datasets. All the datasets provide timestamps for each rating.

Dataset	Ratings	Users	Items	sparsity	$\#\bar{r}_i$
ML-1M	1 000 209	6 040	3 706	95.53%	217
Jester	1 728 785	79 681	150	75.64%	12 348
Netflix	100 000 000	480 189	17 770	98.82%	5 576
Hetrec2011-ML	86 000	2 113	2 113	97.90%	85

Table 5.1: Statistics for the experimental datasets. Column *sparsity* shows the percentage of unrated items in the rating matrix and column $\#\bar{r}_i$ shows the average number of ratings per item.

To evaluate the use of the *R&W* sorting scheme the dataset Hetrec2011-ML is used. An appropriate web crawler was developed in order to obtain a synopsis for each movie using the IMDB website. A total of 52 392 terms were extracted to compose the TF-IDF of each movie.

To evaluate the RS proposed in Section 4.7, and to use the multi-context FFP similarity proposed in Section 4.8 the ML-1M dataset is used. The movie information is extracted from Dbpedia[3]. Dbpedia provides keywords for each movie via a web-crawler. It is a project that aims to retrieve

[1]IMDB: `http://imdb.com`
[2]Rotten Tomatoes: `https://www.rottentomatoes.com/`
[3]Dbpedia: `http://www.dbpedia.org`

structured information from Wikipedia. In December 2017, it provided a total of 29 942 unique keywords to represent items. These are used as contextual information on the experiments of this Thesis. Table 5.2 shows how many keywords exist on each contextual information source.

keyword type	number of keywords
basedOn	232
cinematography	693
director	1680
editing	704
musicComposer	1133
narrator	148
producer	2016
starring	6291
writer	2533
subject	6471
subject2	6477
type	3690

Table 5.2: Contextual information extracted from Dbpedia.

5.2 Evaluation Metrics

RSs provide item recommendations to users, e.g. products, movies, books, songs, and others. The quality of such recommendations is usually measured by evaluating the quality of the predicted ratings. To that effect, the Root Mean Square Error (RMSE) is commonly used and computed using the following equation:

$$RMSE = \sqrt{\frac{\sum_{u,i \in S} \left(r_{u,i} - \hat{r}_{u,i} \right)^2}{|S|}} \tag{5.1}$$

where S is a set of ratings to which an RS computes predictions, and $|S|$ corresponds to the number of ratings being predicted. The best value for RMSE is zero, corresponding to a perfect rating prediction $\forall u, i \in S$. The higher the RMSE, the worst the rating prediction capabilities of the RS. Due to cold-start problems, sometimes an RS are not able to predict ratings for some $u, i \in S$, as a result, the RMSE computation ignores these. For example, if no neighborhood can be determined, then a rating prediction also cannot be determined. This number of rating predictions that can not be determined by the RS are measured by coverage.

Coverage captures the ability of the RS to recommend using the whole collection of items instead of only a portion of the collection [37, 85]. Equation 5.2 shows how coverage is determined, where $|\hat{r}_{testset}|$ corresponds to the number of ratings predicted by the RS in the testset, and $|r_{testset}|$ the total number of ratings in the testset.

$$coverage = \frac{|\hat{r}_{testset}|}{|r_{testset}|} \tag{5.2}$$

The maximum coverage is 1, when all rating predictions are possible to compute. The closer to 0 coverage is the fewer rating predictions requests is the RS able to compute. The closer the value of coverage is to 1 the more rating predictions is the RS able to compute. For example, a RS with coverage of 0.1 indicates that it only predicts ratings for 10% of the requested rating predictions. Thus, even if it has an extremely good RMSE, i.e. close to 0, the usefulness of the recommendations is fairly limited.

RMSE measures only the quality of the rating predictions, it does not measure the quality of the actual recommendations. To do so Precision (PR) and Recall (RC) are often used. Precision reflects the amount of recommended items that are relevant to a user, as shown in Eq. 5.3.

$$PR = \frac{\#relevant\ recommended\ items}{\#items\ recommended} \tag{5.3}$$

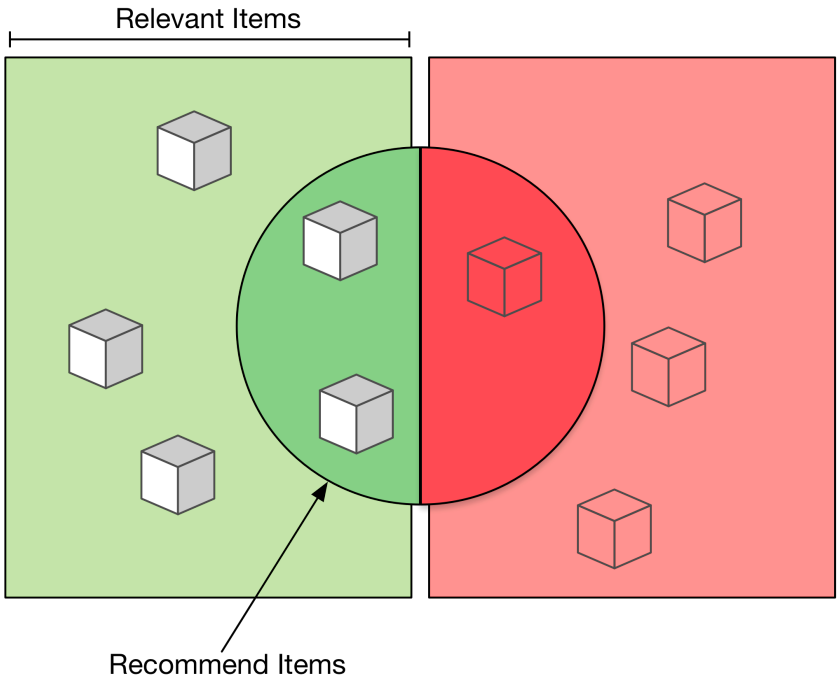


Figure 5.1: Evaluation of relevant items for recommendation, to a given user. Inside the circle are items recommend to the user.

Recall reflects the number of relevant items actually recommended to a user, as shown in Eq. 5.4.

$$RC = \frac{\#relevant\ recommended\ items}{\#relevant\ items} \tag{5.4}$$

A criteria to consider items as relevant or non-relevant for recommendations must be defined. Datasets provide ratings from users on items. Each user rates items as he or she sees fit. For some users, a 4 out of 5 stars is not a relevant item for a recommendation and for other users it is. This leads to the discussion of when to consider an item relevant or not. Usually, a rating threshold is set so that values above it are considered relevant, and below it are considered non-relevant. As expected, such threshold highly influences the recommendation evaluation. In this Thesis, if a rating, in the testset, is equal to or above 4, it is treated as relevant to that user.

There are other alternative approaches in order to set a threshold to determine if an item is relevant or irrelevant for recommendation. For example, for an item to be considered relevant, it must have a rating higher than the average rating of the user rating it. Another alternative is to use only items that have the maximum possible rating. The authors in [84] provide an in-depth discussion on evaluating RSs using precision based metrics.

Figure 5.1 illustrates a recommendation to a user wherein green are the items relevant for recommendation, i.e. ratings are above or equal to 4 in the testset, while in red are the items not relevant to recommend for this user. Inside the circle are the items that the RS recommends to the user, the ones inside the green semi-circle are the ones considered relevant by the RS and relevant according to the testset. Items inside the red semi-circle are items recommended by the RS that according to the testset are not relevant for recommendation to the user.

It is important to note that recall provides a different viewpoint of the RS capabilities than coverage. Recall measures the number of relevant items recommended, that are considered to be relevant by the testset. While coverage measures the ratio between the number of rating predictions computed and the number of rating predictions requested to the RS. Coverage does not provide any information about the quality of those rating predictions nor of the recommendation. It only informs if the RS is able to produce rating predictions or not.

Another recommendation quality metric is the F1-score (Eq. 5.5). It is valid to argue that precision is a good indicator for the quality of a RS, as long as the recall is within a range that allows the retrieval of a sufficient number of relevant items. The F1-score is a metric that combines both precision and recall into a single quality measure value, as shown by Eq. 5.5.

$$F1 = 2 \times \frac{PR \times RC}{PR + RC} \tag{5.5}$$

When evaluating recommendation quality, neither precision or recall account for the ranking position of the recommended items, i.e. if an item should appear first or after another item as a recommendation. The ranking correctness of recommendations can be measured using a metric such as Normalized Discounted Cumulative Gain (NDCG). This metric captures if a more relevant item is presented first than a less relevant item. NDCG results from computing the Discounted Cumulative Gain (DCG) (Eq. 5.6) that accounts for the order of the elements within the recommendations generated, divided by the Ideal Discounted Cumulative Gain (IDCG) (Eq. 5.7) that corresponds to perfect ranking of items to recommend for each user. NDCG is computed using the Eq. 5.8.

$$DCG@N = \sum_{i=1}^{P} \frac{2^{rel_i}}{\log(1+i)} \tag{5.6}$$

$$IDCG@N = \sum_{i=1}^{R} \frac{2^{rel_i}}{\log(1+i)} \tag{5.7}$$

$$NDCG@N = \frac{DCG@N}{IDCG@N} \tag{5.8}$$

In the Equations above, N is the number of recommendations provided, rel_i is the relevance of the recommendation at position i, P is the ordered set of recommendations being generated by the RS, and R is the correct ordered set of items to recommend to a user. Note the subtle difference between Eq. 5.6 and Eq. 5.7, which only differs by the use of sets P and R.

In the proposed RS of Section 4.7 and with the multi-context FFP similarity in Section 4.8, a threshold for a maximum number of recommendations is not set. The RS recommends as many relevant items to a user as it considers fit.

5.3 Computational Cost

This Thesis intends to demonstrate that using FFP similarity metric allows for a smaller computational cost than traditional similarity metrics. The metrics used to determine the reduction of complexity of this work are: the average number of iterations per similarity; the time required for the computation of a whole similarity matrix; and the time required to compute the testset rating predictions.

Similarity metrics such as COS and PC have the same average number of iterations per similarity. On the other hand, the proposed FFP similarity metric is limited to k, the maximum number of features an FFP can have. This parameter influences the quality of rating predictions, recommendations and the average number of iterations per similarity. A decrease in the average number of iterations per similarity indicates an improvement

of computational efficiency. Also, having a k smaller than the average number of ratings used by COS or PC indicates that the FFP similarity does have a smaller computational cost. The measurements of time required for the computation of a whole similarity matrix and time required to compute the testset rating predictions are performed using only a single thread.

The time required to create the FFPs is not computed since if the rating matrix is a sparse matrix, then it can be represented using a table, e.g. a hashtable. Such table has item ids as key and as value another hashtable, in which users ids are keys and the value is the corresponding rating. The list of existing keys (users) should be sorted, according to the sorting scheme used. Thus, not adding additional memory complexity and having a neglectable update cost over time, i.e. insertion of an element in a sorted list $O(log(n))$. The only memory complexity added is $O(k)$ for storing the FF values using a list. This is preferable than constantly computing the FF value for each position of the FFP.

5.4 Evaluation Framework

All experiments of this Thesis use RiVal [86], a framework that promotes a more uniform and unbiased RSs evaluation. The RiVal framework implements four modules: data split, recommendation, candidate selection, and evaluation. The user of the RiVal framework (developer of the RS) only needs to input the correct parameters to execute the experiments. RiVal is responsible for generating the dataset split (e.g. using cross-validation or separating a train and test set according to timestamps); and executing the recommendation module (i.e. uses the RS created by the developer to generate rating predictions and recommendations). The candidate selection module decides which items are used to evaluate the recommendation quality, an often overlooked aspect of RSs evaluation. The evalu-

ation module performs the computation of evaluation metrics using the recommendations produced by the recommendation module against the expected results from the testset.

This Thesis uses N-fold cross-validation with a N equal to five, i.e. a 5-fold cross-validation where each fold has two sets, a trainset and a testset which are randomly generated.

During the development of this Thesis large datasets such as Netflix (see Table 5.1) required modifications to the data split module of RiVal. More specifically, the generation of the folds for cross-validation was being all performed in memory, which made the process infeasible for large datasets. The solution was to write the folds directly to the disk, requiring only to have in memory one representation of the dataset. This modification was submitted to the framework, and it is now part of the master branch of the RiVal framework[4] allowing for a memory complexity of $O(1)$ instead of $O(N+1)$, when executing the data split module.

5.5 Experiments

This Section delivers the most relevant findings regarding the usage of the FFP similarity metrics for Recommender Systems. More specifically, the evaluation of the FFP similarity capabilities using different sorting schemes, fuzzifying functions, a comparison with baselines, a computational complexity analysis, an evaluation of the proposed RS in Section 4.7 and evaluation of the multi-context FFPs deployed into the RS of Section 4.7.

[4]RiVal framework: `https://github.com/recommenders/rival` Accessed on 2019 March 22.

5.5.1 Comparing Fuzzyfying Functions

To determine if the Fuzzy Fingerprint similarity is appropriate for traditional item-based CF using Eq. 2.3 for rating prediction, the first experiment presents a comparison of the fuzzifying functions μ_{One}, μ_{erfc} and μ_{linear} over a series of k features used to create the FFP. Afterward, using the best performing Fuzzifying Functions, a comparison of the sorting schemes Random, HL and LH, over a series of k features used to create the FFP is performed. This is followed by a comparison of the best performing combinations of fuzzifying functions and sorting schemes, to determine if these outperform the baselines.

The value of k features used to create an FFP depends on the dataset being used, i.e. the number of features each item uses, for representation, is taken into account during evaluation. When varying k, several changes in the dynamic of the FFPs and FFP similarity take place. Regarding to the FFPs, k influences the representation of the FF and, consequently, the relevance attributed to each feature. The number of features allowable to represent items with the FFP is capped by k thus it also influences the amount of information to represent each item.

The first experiment starts by comparing the rating prediction quality of the different Fuzzifying Functions. Figures 5.2-5.4 present the RMSE over three FFs, while varying the value of k. The FFPs use the LH sorting scheme.

Analysing Figure 5.2 using few features does not yield an optimal RMSE. The same also applies if too many features are used. The ideal number of features to use varies between 100 and 300, depending on FF selected. FF μ_{One} requires 100 features to obtain the minimal RMSE while μ_{linear} requires 200. The function μ_{erfc} requires 300, i.e. 3 times more features than μ_{One}, yet is the best performing in regards to RMSE, in the ML-1M dataset. Note that ML-1M has a total of 6040 users and each item has an average of 217 ratings, preferably FFPs should use less than 217 ratings in order to

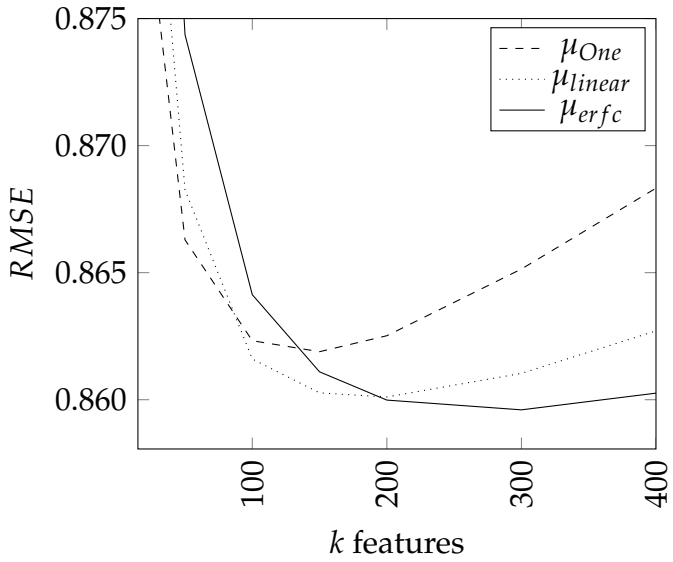


Figure 5.2: Experiment on ML-1M showing the impact on RMSE of different Fuzzyfying Functions. The CF uses 100 neighbors while varying k, i.e. the size of FFPs.

improve the computational complexity. Moreover, the cap that k features create influences only items with a high number of ratings (generally popular items), which have enough information to provide accurate similarity values. Items with few ratings are able to retain all features they have, thus not influencing the already low amount of information available.

In Figure 5.3, the experiment with the FFs shows different patterns of RMSE. The Netflix dataset has an average number of ratings per item of 5 576 and a total of 17 770 items. In comparison with the ML-1M, it has a higher number of users, thereby requiring to have a higher number of features per FFP. The FF μ_{One} uses 1 500 features to obtain the minimal RMSE while μ_{linear} uses 2 000. The μ_{erfc} uses 4 000 features, much more than the other two FFs, yet it is the best performing in regards to RMSE of the three, not only on the ML-1M, but also in the Netflix dataset. The μ_{erfc} has a different RMSE pattern than the other two FFs, as the number of k

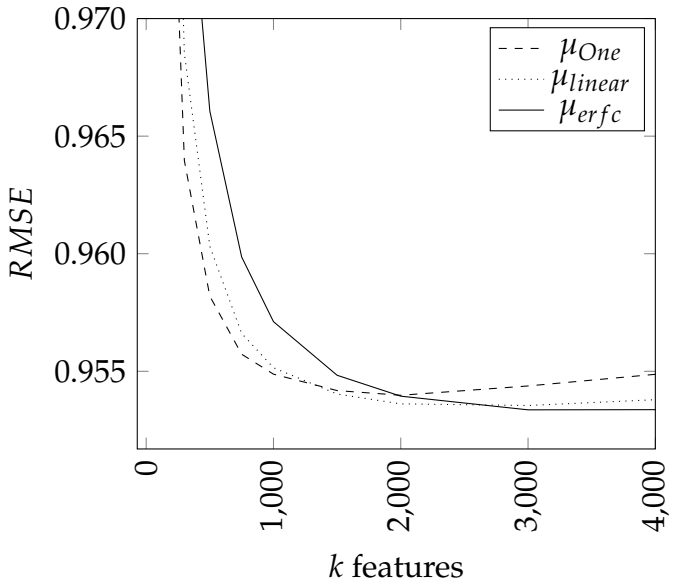


Figure 5.3: Experiment on Netflix showing the impact on RMSE of different Fuzzyfying Functions. The CF uses 200 neighbors while varying k.

features of the FFP increases.

Examining Figure 5.4, the Jester dataset enables the FFP similarity to use a low number of features in comparison to the average number of ratings per item 12 348. The ideal number of features to use varies between 100 and 300 depending on FF selected. The FF μ_{One} requires 175 features to obtain the minimal RMSE, while μ_{linear} requires 250, and the μ_{erfc} 450. The difference between the three FFs in terms of RMSE is marginal. As a result any of the FFs is suitable to use. Note that Jester has a total of 79 681 users and each item has an average of 12 348 ratings. This difference between the average number of rating per item and k features of FFPs allows for a good margin to improve the computational efficiency. A particularity of the Jester dataset is the rating scale from -10 to 10. This causes it to have a higher RMSE, much higher in comparison to the ML-1M and Netflix. This is expected and is only due to the scale of the ratings.

The results from Figures 5.2-5.4 lead to the conclusion that any of the

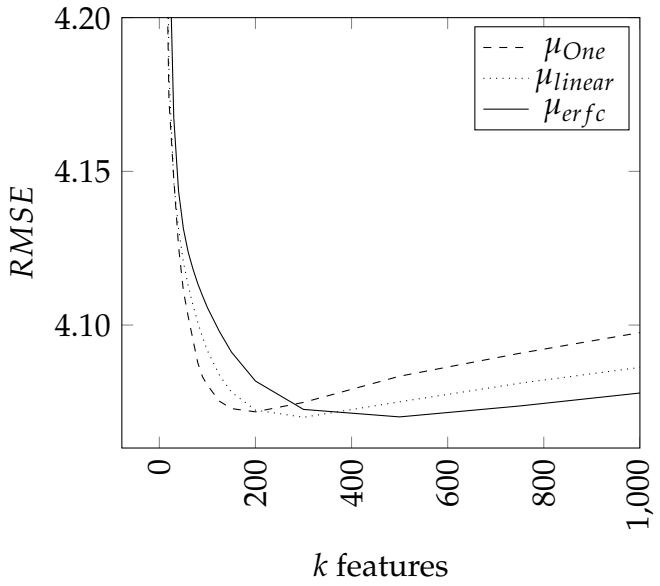


Figure 5.4: Experiment on Jester showing the impact on RMSE of different Fuzzyfying Functions. The CF uses 50 neighbors while varying k, i.e. the size of FFPs.

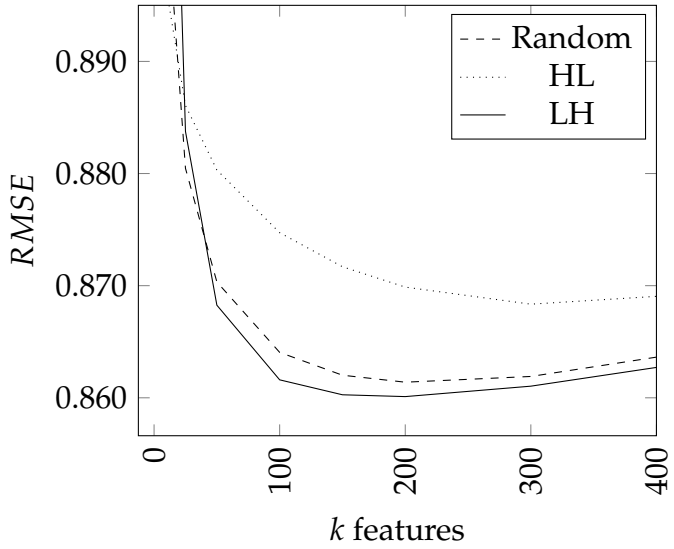


Figure 5.5: Experiment on ML-1M shows the impact on the RMSE of different SSs. The CF uses 100 neighbors while varying k.

FFs is suitable to use generate items FFPs. This small RMSE difference between FFs is, in fact, consistent with other results found in the literature [20]. Additionally, more extensive experiments show that the behaviour of the sorting schemes HL and Random is similar in terms of FFs to when using LH as SS.

5.5.2 Comparing Sorting Schemes

Sorting schemes are a core part of this Thesis, since creating an FFP requires to first rank features. Ratings are on a discrete scale, thus ranking them requires extra information regarding the items and users. The task of a sorting scheme is to use this information. The experiments on Figures 5.5-5.7 show the Random, HL, and LH sorting schemes while varying the number of k features used to form the FFP and FF.

The experiment in Fig. 5.5 shows that using an FFP with μ_{linear} FF with different SS causes different RMSE patterns. On the ML-1M dataset, LH

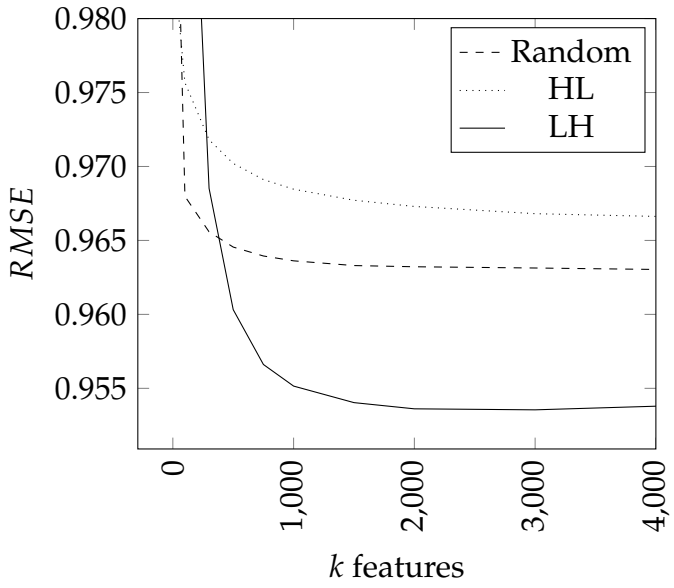


Figure 5.6: Experiment on Netflix shows the impact on the RMSE of different SSs. The CF uses 200 neighbors while varying k.

is the best performing SS, and the Random SS has a similar RMSE using $k = 200$ features. The HL sorting scheme requires $k = 300$ features to achieve peak quality in terms of RMSE. The HL SS ranks users considering more relevant the ones with a higher rating and unties equal user ratings based on which user has rated more items (see Section 4.2), yet has the worst RMSE values, this is an unexpected result.

In the experiment of Fig. 5.6, the Netflix dataset is used and FFPs use a μ_{linear} FF. The best performing SS is LH using $k = 2000$ features, while Random SS uses $k = 1500$ features. As in ML-1M, HL is the worst performing SS in terms of RMSE, requiring $k = 1900$ features to achieve peak RMSE quality. The convergence pattern in the Netflix dataset is different than in ML-1M, since increasing the FFP number of features after a certain point does not influence the RMSE value.

The experiment in Fig. 5.7 shows that using an FFP with a μ_{linear} FF with different SS causes different RMSE patterns. The best performing

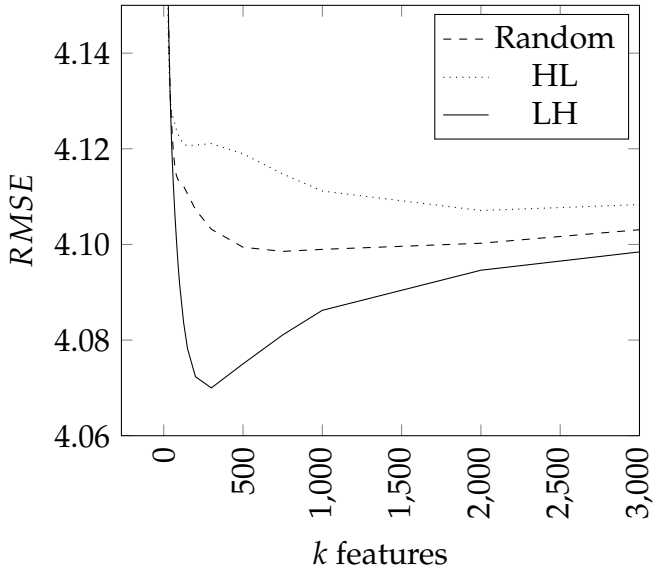


Figure 5.7: Experiment on Jester shows the impact on the RMSE of different SSs. The CF uses 50 neighbors while varying k.

SS is LH as in the two previous datasets. It requires using only $k = 300$ features to create the FFP which in comparison to the average number of features per item (12 348) is only 2.5%. Moreover, in comparison to the total number of users 79 681 is only 0.376%, this opens a clear opportunity to reduce the computational complexity in comparison to traditional similarity metrics such as COS or PC. The other two SSs require more features and have a higher RMSE, where the Random SS uses $k = 800$ features and the HL uses $k = 2000$ features.

The experiments carried in these three datasets lead to the conclusion that best performing SS is the LH when taking into account the RMSE performance and the number of k features required to create FFPs.

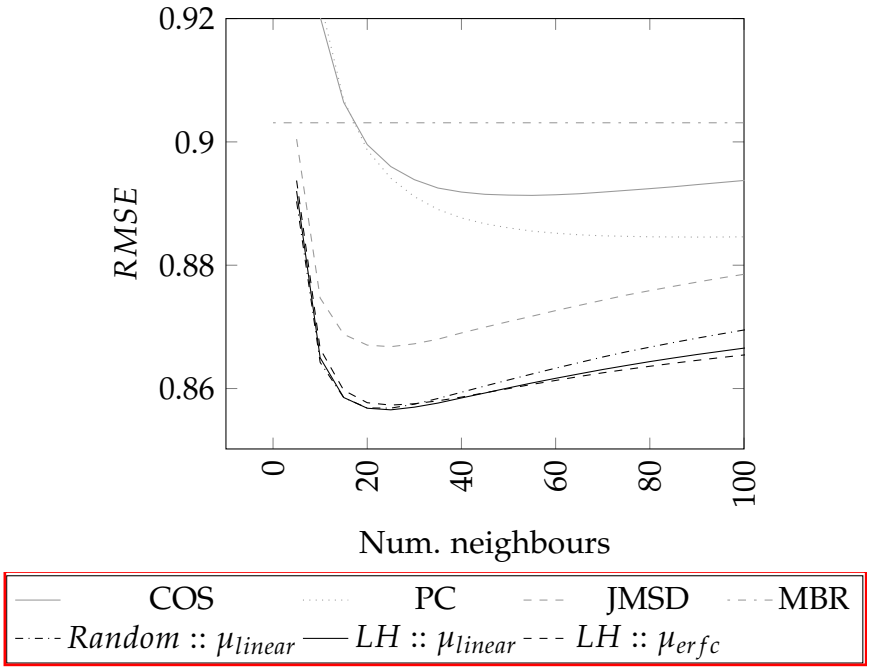


Figure 5.8: FFP similarity comparison with the baselines in the ML-1M dataset. Baseline metrics are represented in light grey, while the FFP metrics are represented in black.

5.5.3 Comparison to the baselines

Figures 5.8-5.10 compare the results between the best performing FFs and SSs (based on Figures 5.2-5.7) to the baseline similarity metrics, on the ML-1M, Netflix and Jester datasets.

The experiments from Fig. 5.8 show that the three alternatives presented of the FFP similarity metric are able to improve the recommendation quality on ML-1M, having a neglectable difference of RMSE among them. The best performing FFP similarity is the one using the LH sorting scheme, with μ_{linear} as FF and $k = 200$ features, with an improvement of 0.0105 in RMSE. The best performing baseline is JMSD with a RMSE of 0.8670 and requiring the same number of neighbors as the FFP similarity.

The experiments from Fig. 5.9 show that the three alternatives pre-

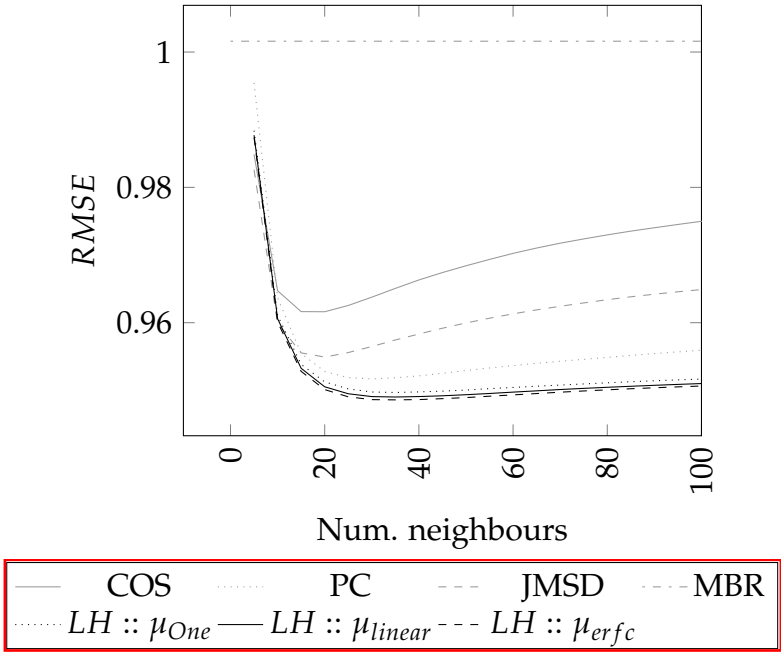


Figure 5.9: FFP similarity comparison with the baselines in the Netflix dataset. Baseline metrics are represented in light grey, while the FFP metrics are represented in black.

sented are able to improve the recommendation quality on the Netflix dataset, having a neglectable difference of RMSE among them. The best performing baseline is PC with a RMSE of 0.9517 and requiring 20 neighbors to determine rating predictions. The best performing FFP similarity is the one using the LH sorting scheme and the μ_{erfc} FF with $k = 200$ features and achieving the RMSE of 0.9486 while requiring 35 neighbors to determine rating predictions. Using the same number of neighbors for rating prediction still yields a better RMSE than any of the baselines, yet the best RMSE is achieved using 35 neighbors.

The experiments from Fig. 5.10 show that the three alternatives presented using the FFP similarity metric have similar RMSE on the Jester dataset. The best performing baseline is PC with a RMSE of 4.0419 and requiring 15 neighbors to determine rating predictions. The best perform-

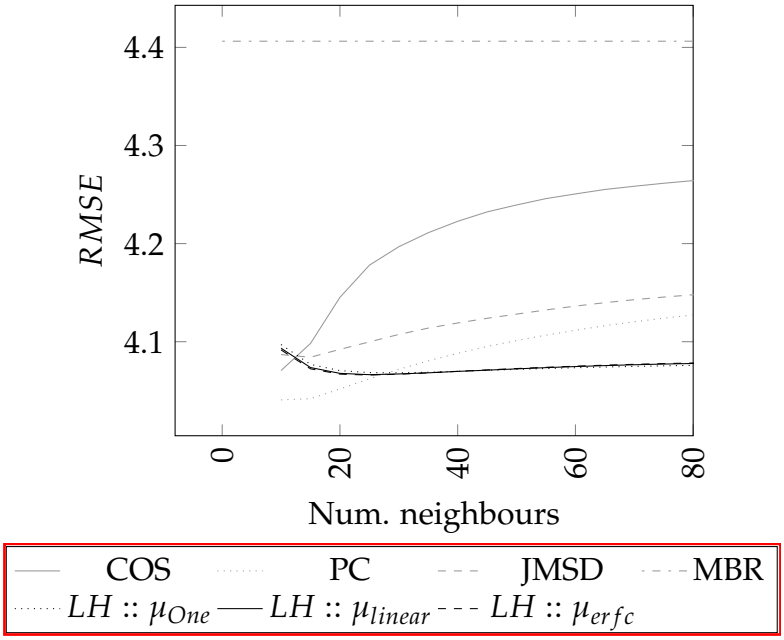


Figure 5.10: FFP similarity comparison with the baselines in the Jester dataset. Baseline metrics are represented in light grey, while the FFP metrics are represented in black..

ing FFP similarity is the one using the LH sorting scheme and the μ_{erfc} FF with $k = 300$ features and achieving the RMSE of 4.0660 while requiring 25 neighbors to determine rating predictions. Experiments on Jester show that PC outperforms the FFP similarity in regards to the RMSE and number of neighbors used. However, PC has to determine the similarity between two items comparing the 12 348 ratings that each item has on average. While, when using the FFP similarity at most 300 ratings are used to determine similarity values. This is a compromise in terms of RMSE that when performed improves the computational efficiency of the similarity metric. Jester has a low sparsity in comparison to the other two and is atypical dataset for RSs, due to its uncommon sparsity, as shown previously in Table 5.1.

Sim.	ML-1M				
	SS	μ	k	N	$RMSE$
FFP	LH	μ_{linear}	200	20	**0.8565**
	LH	μ_{erfc}	200	25	0.8577
	Rand	μ_{linear}	200	20	0.8568
COS	-	-	217	50	0.8914
PC	-	-	217	75	0.8847
JMSD	-	-	217	20	0.8670
MBR	-	-	217	-	0.9031

Table 5.3: Best results for FFPs and baselines, on ML-1M. Column N contains the number of nearest-neighbour items used to compute the predicted rating.

5.5.4 Summary of FFP Results

Tables 5.3-5.5 present a summary of the best performing FFP similarity combinations with the respective k features used and the number of neighbors used by the traditional CF, in comparison to the baselines. The best values of RMSE are highlighted using bold. The value of k for the baselines represents the average number of ratings per item. The coverage of all the similarity metrics in Tables 5.3-5.5 is always higher than 99.8%. Note that when computing a similarity value, the baselines must compute the degree of similarity across all the shared ratings between two items, instead of using the FFP similarity, which is limited to at most k features between the two items are computed.

The best results for the ML-1M dataset are obtained with the FFP similarity, which outperforms all four baselines. This is achieved using at most $k = 200$ features (ratings) to describe each item and 20 neighbors to compute rating predictions, as shown in Table 5.3.

The best results for the Netflix dataset are obtained with any of the FFP similarity approaches, which outperform all four baselines. This is achieved using at most $k = 3000$ features to describe each item and 35

Sim.	Netflix				
	SS	μ	k	N	$RMSE$
FFP	LH	μ_{One}	1500	35	0.9497
	LH	μ_{linear}	2000	35	0.9490
	LH	μ_{erfc}	3000	35	**0.9486**
COS	-	-	5 576	15	0.9616
PC	-	-	5 576	30	0.9517
JMSD	-	-	5 576	20	0.9549
MBR	-	-	5 576	-	1.0016

Table 5.4: Best results for FFPs and baselines, on Netflix. Column N contains the number of nearest-neighbour items used to compute the predicted rating.

Sim.	Jester				
	SS	μ	k	N	$RMSE$
FFP	LH	μ_{linear}	200	25	4.0664
	LH	μ_{erfc}	300	25	4.0660
	LH	μ_{One}	200	25	4.0685
COS	-	-	12 348	15	4.0983
PC	-	-	12 348	15	**4.0419**
JMSD	-	-	12 348	15	4.0842
MBR	-	-	12 348	-	4.4063

Table 5.5: Best results for FFPs and baselines, on Jester. Column N contains the number of nearest-neighbour items used to compute the predicted rating.

neighbors to compute rating predictions, as shown in Table 5.4. There is a possible trade-off between the number of k features used and the RMSE when choosing a FF for the FFP similarity, thus reducing by half the number of k features used.

The best RMSE in Jester dataset is obtained with PC, which is able to outperform the FFP similarities by 0.0266. However, the number of features used by the FFP similarity metric is less than by PC as shown by

Table 5.5. When using FFP similarities the maximum number of features used to determine similarities is between 200 and 300 depending on the FF, while the PC uses on average 12 348 ratings to determine a single similarity value, important characteristic since on average 95% less iterations are required to compute a similarity than by PC, as shown later in this Chapter (see Fig. 5.14).

5.5.5 Ratings & Words sorting scheme

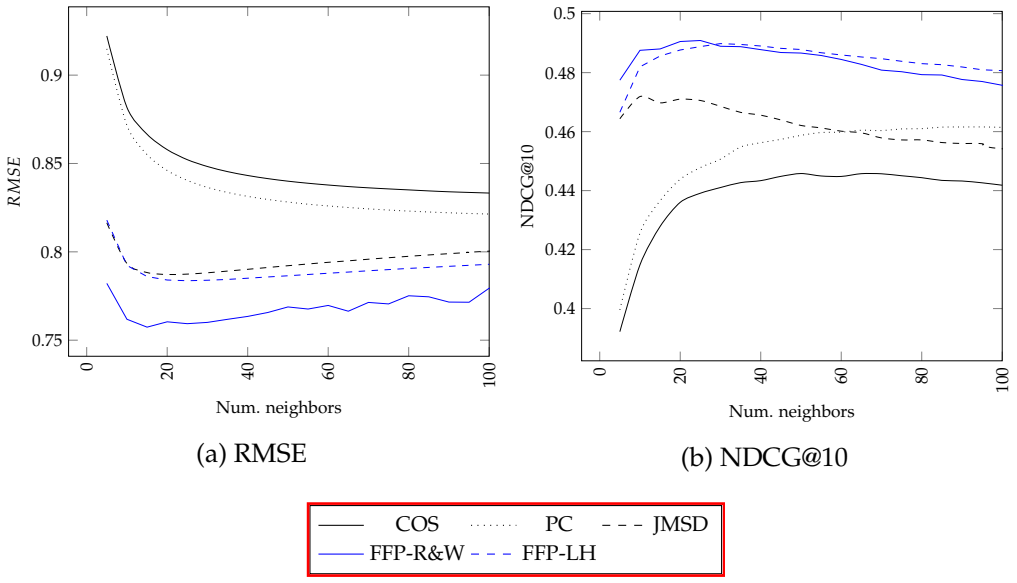
The R&W sorting scheme outperforms the baselines on the Hetrec-2011 dataset. The sorting scheme relies on the context-based information and collaborative filtering information to compute similarities, being able to improve the rating prediction and recommendation quality.

In comparison to the baselines COS, PC and JMSD the proposed similarity metric FFP-R&W improves the RMSE by 0.03 and the NDCG@10 by 0.023. It is also included the best performing FFP using ratings exclusively, more specifically using the LH sorting scheme and as FF the μ_{linear} with k equal to 75. The best performing baseline is the FFP using the LH SS followed by the JMSD as shown by Fig. 5.11. Table 5.6 provides a comparison of the proposed SS with the baselines.

The use of R&W as sorting scheme requires a higher computational complexity than when using Random, HL or LH as SSs due to the number of features used by the FFP. The number of features used by the FFP-R&W is 125 while the average number of rating per item is 85. One particularity is that FFP-R&W requires only 15 neighbors instead of the 25 required by the FFP-LH and the 20 required by the baseline JMSD.

5.5.6 Computational Efficiency

The proposed similarity metric shows consistent gains in recommendation quality. This Section aims to evaluate if the FFP similarity can improve



(a) RMSE

(b) NDCG@10

Figure 5.11: Comparison of the different baselines to the FFP-R&W similarity. RMSE is shown in Fig. 5.11a and NDCG@10 in Fig. 5.11b, while varying the number of neighbors.

Similarity	K	N	RMSE	NDCG@10
FFP-R&W	125	**15**	**0.757**	**0.488**
FFP-LH	75	25	0.783	0.489
COS	-	100	0.833	0.442
PC	-	100	0.821	0.461
JMSD	-	20	0.787	0.471

Table 5.6: FFP-R&W similarity metric in comparison to the baseline similarity metrics, on the dataset Hetrec-2011.

computational complexity and, if so, by how much. The experiments consider the average number of iterations per similarity, the time needed to compute a similarity matrix and the time required to compute the testset rating predictions.

Figures 5.12-5.14 show a comparison of the average number of iterations per similarity of the best performing FFP similarity metric on each dataset, while varying the number of k features. As the number of features that form the FFPs increases, so does the number of iterations per similarity increase in all datasets. The same does not apply to either of the baselines: COS, PC, JMSD, and MBR. The baselines are represented using the grey line at the top of each Figure. MBR has a computational complexity 1 as shown in the Figures and explained in Section 2.4.3. The vertical line that crosses the FFP iterations per similarity corresponds to the number of k features used by the best performing FFP in terms of RMSE. It is used as a reference to compare to the baseline similarity metrics. It is important to recall that none of the baselines can vary the k number of features. Thus, the horizontal line is only indicative of the average iterations per similarity of baseline similarity metrics. They do not change the number of features.

In ML-1M, the FFP similarity needs on average 100 iterations per similarity when using around 200 features. The baselines require, on average, 125 iterations per similarity, which is 20% more than FFPs.

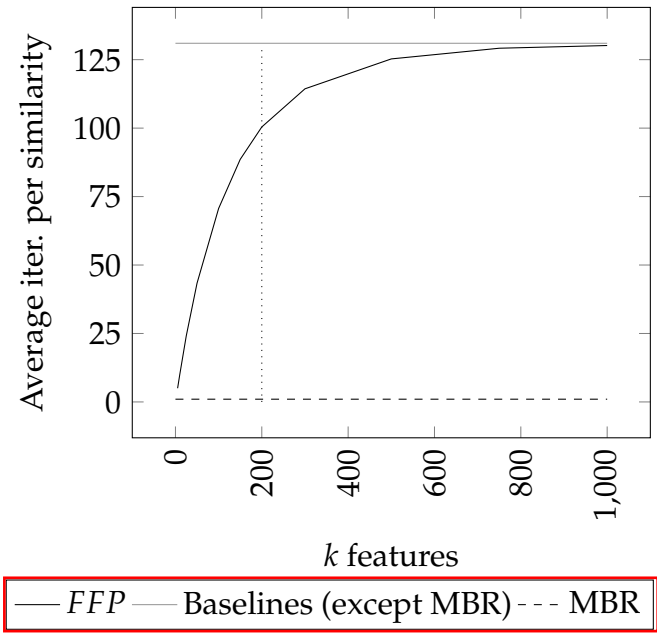


Figure 5.12: The average number of iterations performed per similarity computed in the ML-1M dataset, while varying the size of the FFP. Vertical lines show the value of k for which the best results were achieved.

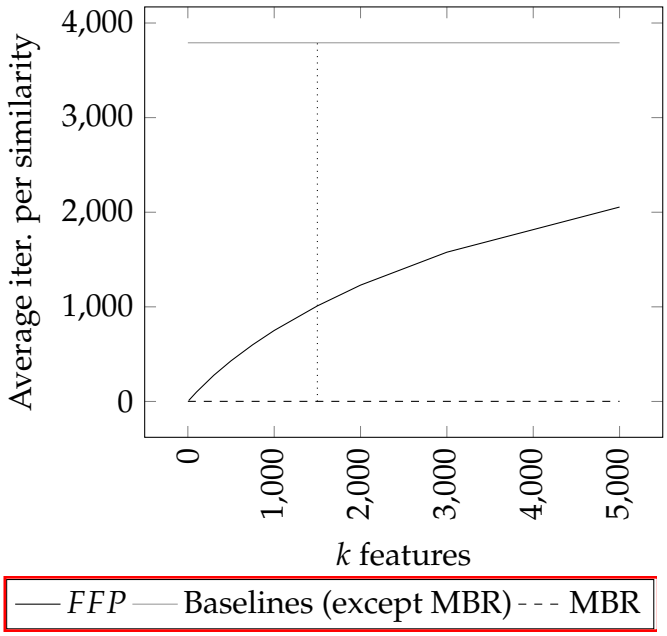


Figure 5.13: The average number of iterations performed per similarity computed in the Netflix dataset, while varying the size of the FFP. Vertical lines show the value of k for which the best results were achieved.

In Netflix the gains become more evident, as shown in Fig. 5.13. Using 1500 features (ratings) the FFP similarity metric only performs on average 1000 iterations per similarity. Baseline similarity metrics require 3800 iterations per similarity, on average. This corresponds to a reduction of iterations per similarity of almost 75%, combined with an improvement of RMSE, as shown in Figure 5.9.

The Jester dataset has the particularity of having an average of 12 348 ratings per item, and a sparsity of 75.64%, a low value for a RS. FFP similarity provides a good RMSE using only 300 features and requiring 281 iterations per similarity, on average. Such performance when compared to the baselines that require 5822 iterations per similarity allows for an improvement of at least 95% of the number of iterations per similarity.

Besides measuring the average number of iterations per similarity, other

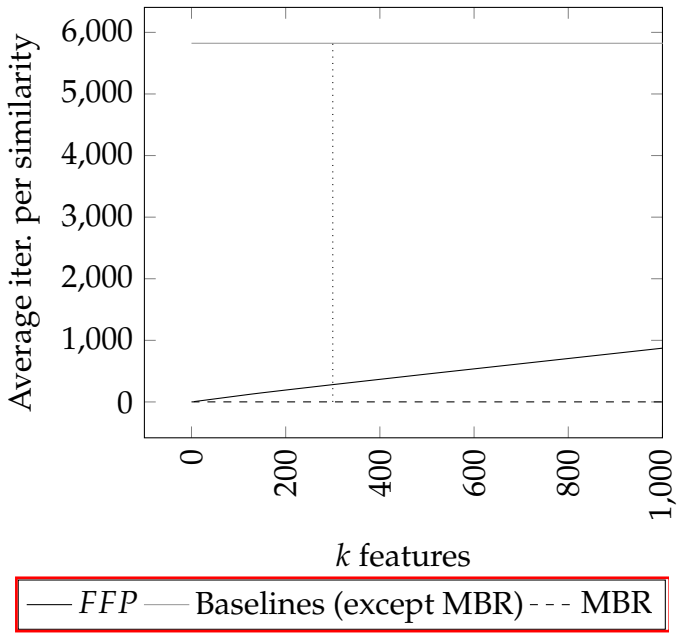


Figure 5.14: The average number of iterations performed per similarity computed in the Jester dataset, while varying the size of the FFP. Vertical lines show the value of k for which the best results were achieved.

Dataset	# items	# similarities	FFP	COS	PC	JMSD
ML-1M	6040	18 237 780	**28.88 s**	44.833 s	67.54 s	56.75 s
Netflix	17 770	157 877 565	**5 694 s**	21 157 s	39 707 s	37 990 s
Jester	150	11 175	**139 ms**	3 305 ms	5 092 ms	4 443 ms

Table 5.7: Time complexity to compute the similarity matrix.

valid measurement is the time required to compute rating predictions and a similarity matrix. Tables 5.7 and 5.8 show the results of these experiments, the highlighted times on the table correspond to the faster similarity metric on an item-based CF, for each dataset. In both experiments, the most relevant results to discuss are on Netflix due to the number of existing items and similarities that have to be computed.

Table 5.7 shows the number of items, the total of similarities that have to be computed, and the time required by each similarity metric to compute them. The FFP similarity outperforms the fastest baseline COS on Netflix requiring less 15 463 seconds. The baseline obtain the better RMSE is PC, yet it requires more 34 013 seconds than the FFP similarity. On ML-1M and Jester, the baselines are also outperformed by the FFP similarity, when considering to the time required to compute the similarities, by a large margin.

Table 5.7 shows the time required to compute the rating predictions in the testset. One of its columns shows the number of rating predictions contained in each testset as a reference to help understand the importance of improving the computational complexity of similarity metrics. The FFP similarity outperforms the fastest baseline COS on Netflix, as it requires less 17 768 seconds to determine the rating predictions. In Netflix the best performing baseline in regards to RMSE is PC, yet it requires more 35 073 seconds than the FFP similarity, it corresponds to roughly 4.5 times more time required, while achieving the worst RMSE in comparison to the FFP similarity. On ML-1M and Jester, the baselines are also outperformed by

Dataset	#rating predictions	FFP	COS	PC	JMSD
ML-1M	200 042	**483 s**	496 s	538 s	521 s
Netflix	1 408 393	**10 371 s**	28 139 s	45 444 s	43 241 s
Jester	345 757	**17.99 s**	23.89 s	28.18 s	26.39 s

Table 5.8: Time complexity to compute the testset with rating predictions.

the FFP similarity.

5.5.7 Adapted Memory-based CF results

In Section 4.7, a CF approach that does not rely on rating prediction is proposed. This Section presents an evaluation of this CF approach. The RS uses ratings and keywords to represent users with FFPs. For comparison, traditional CF with the baseline similarity metrics COS, PC, JMSD on both item-based (IB) and user-based (UB) CF are used. The FFP similarity metric is also included, where the LH SS is applied to a traditional item-based CF. The experiments are performed on the ML-1M dataset.

Figures 5.15-5.17 compare different sizes of the FFP, where for each size the number of neighbors used varies. According to the F1-score, the best results are obtained using k equal to 200. Knowing that, on average, each user has 637 keywords associated to rated movies, the FFP similarity metric uses only 31% of the existing keywords, thus being able to select the most relevant keywords to represent users. Note that the FFP does not use the ratings as features of the FFP, the features are exclusively keywords. The keywords are an aggregation of the available contextual information in Table 5.2, creating an FFP for each user using contextual information.

An important conclusion in Fig. 5.15 is that according to the F1-score using more features does not equate to better recommendations. The conclusions would be different if considering only Figures 5.16 and 5.17.

Table 5.9 shows how the different approaches perform. The proposed

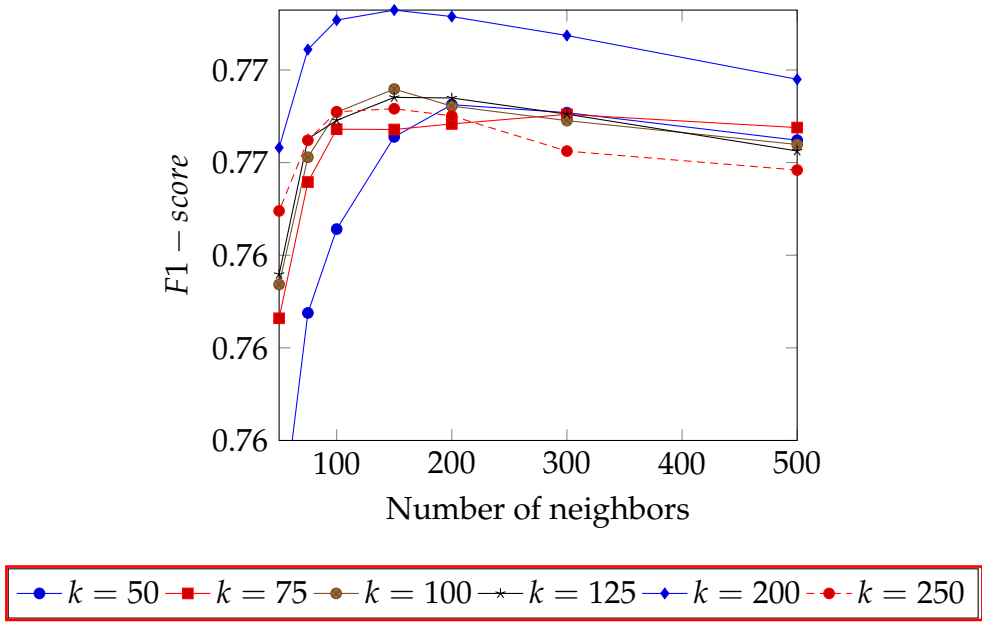


Figure 5.15: F1-score of different sizes of the FFP, while varying the number of neighbors used.

RS performs better overall than any other baseline, even when compared to the state-of-the-art JMSD, although the improvement is not statistically significant.

An interesting result is how much better the proposed approach is when compared to other previously proposed user-based approaches, leaving room to further developments in user-based RSs. It should be noted that item-based approaches have been thoroughly used in the past and have been highly optimized. Yet, user-based approaches are also viable. For example, it is straightforward to enrich the FFP using data other than simple keywords, from movie descriptions with a user's favorite actors, directors or genres.

Nevertheless, using contextual information besides ratings is more computationally demanding. The FFP using the LH sorting scheme and $k = 200$ features has an average number of iterations per similarity of 88. While

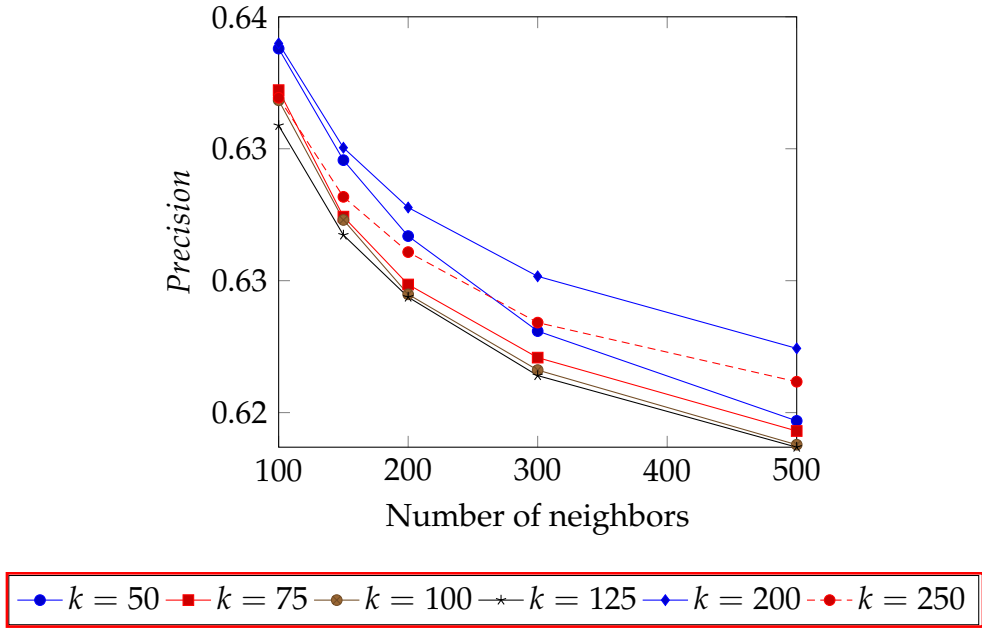


Figure 5.16: Precision of different sizes of the FFP, while varying the number of neighbors used.

Similarity Metric	Num. neighbors	F1-score	Precision	Recall
$FFP_{keywords}$-UB	150	**0.76929**	**0.63504**	0.97554
FFP_{LH}-IB	20	0.76623	0.62113	0.99979
COS-IB	50	0.76622	0.62112	**0.99978**
PC-IB	75	0.76621	0.62115	0.99969
JMSD-IB	20	0.76623	0.62112	0.99980
COS-UB	200	0.42356	0.26869	0.99989
PC-UB	100	0.42338	0.26854	0.99990
JMSD-UB	100	0.42356	0.26869	0.99989

Table 5.9: Summary results of $FFP_{keywords}$ (using $k = 200$), compared with several baselines using CF item-based (IB) and user-based (UB).

the solution presented in this Section using the FFP similarity, with a total of $k = 300$ features, requires an average of 213 iterations per similarity.

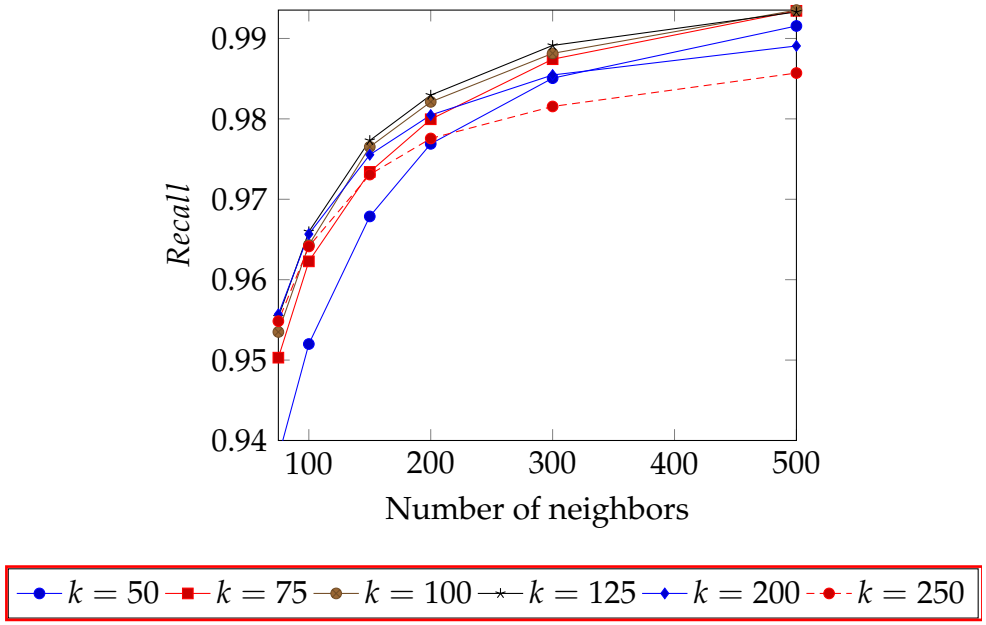


Figure 5.17: Recall of different sizes of the FFP, while varying the number of neighbors used.

5.5.8 Multi-Context Fuzzy Fingerprints for RSs

In Section 4.7 a CF approach that does not rely on rating prediction is proposed. Section 4.8 uses that CF approach and applies to it a multi-contextual FFP similarity metric (MC FFP). This similarity uses a ranking fusion algorithm to combine each contextual FFP similarity into a single similarity metric between two users. The relevant variables to take into consideration are the ranking fusion algorithm, which includes CombMIN, CombMAX, CombSUM, CombMNZ and CombANZ; the sources of contextual information used; the number of k features each contextual information uses; and the number of neighbors used by the CF system. All these variables play a role in the computational efficiency of the similarity metric and the recommendation quality.

The experiments use the ML-1M and each contextual-information from

Table 5.2. The multi-context FFP similarity is compared to traditional CF with the baseline similarity metrics COS, PC, JMSD on both item-based (IB) and user-based (UB) CF. The FFP similarity metric using the LH sorting scheme, on a traditional CF with an item-based approach is also a baseline similarity metric. So does the CF approach in Section 4.7 that is referred $FFP_{keywords}$.

The number of features used by each contextual FFP depends on the number of existing features on each contextual information source. Instead of manually selecting the appropriate k number of features for each FFP, this number is set using a percentage of the total existing features for each context. For example, if only 25% of features are used and one contextual information source has 100 different possible features, then the k number of features is set to 25 for those contextual FFPs.

The first experiment uses CombSUM as ranking fusion approach and varies the number of features used, as shown in Figures 5.18-5.20. The F1-score in Figure 5.18 is more informative than the ones showing precision and recall.

Precision shows that using fewer features yields better results, while recall shows that the increase of features causes it to improve. Using between 20% and 30% appears to be the best percentages of features to create multi-context FFPs. As for the number of neighbors used, the best F1-score value is between 75 and 100 neighbors. Increasing past 100 neighbors causes the recommendation quality to start deteriorating according to the values of F1-score.

The following experiment, shown in Figures 5.21-5.23, aims to determine if different ranking fusion algorithms influence the recommendation quality on the proposed RS using multi-context FFPs and which one is best suited for the task. The comparison of different ranking fusion algorithms is performed using only 20% of the k features for each context. CombMax, CombMNZ, and CombSUM all show good quality recommendations, ac-

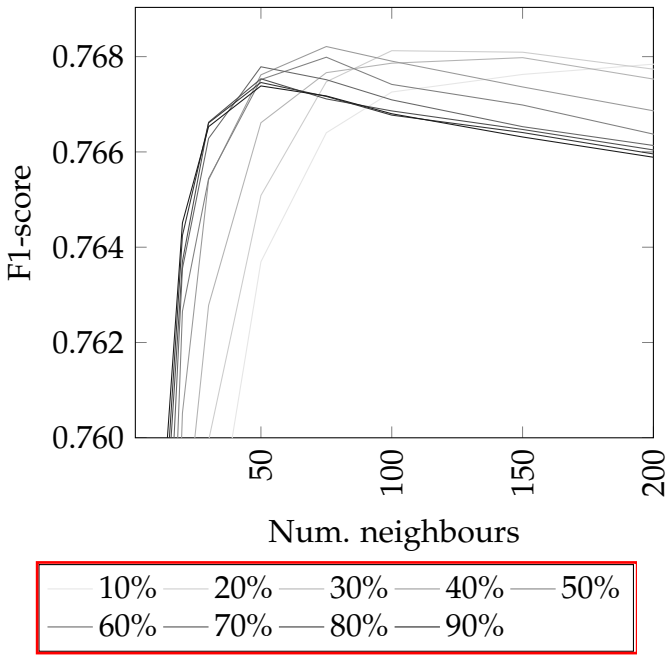


Figure 5.18: F1-score of the multi-context FFP similarity using CombMAX as ranking fusion algorithm depending on the percentage of features used and the number of neighbors of the RS.

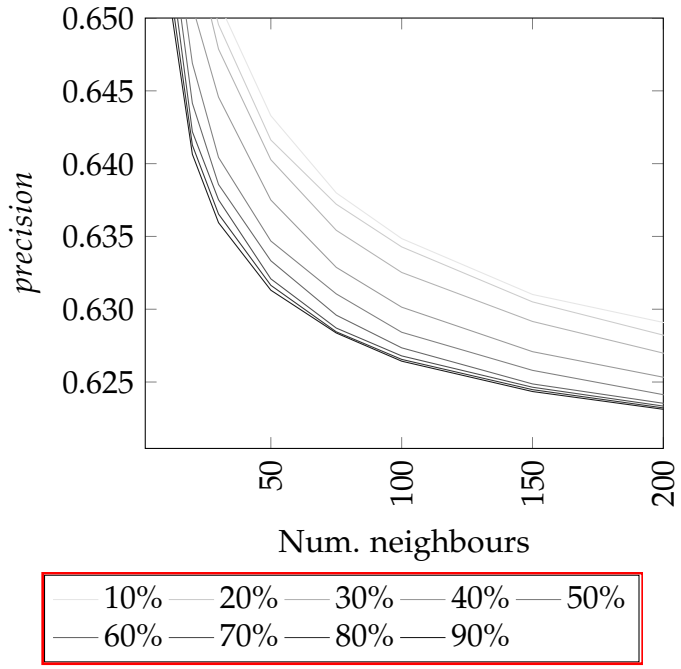


Figure 5.19: Precision of the multi-context FFP similarity using CombMAX as ranking fusion algorithm depending on the percentage of features used and the number of neighbors of the RS.

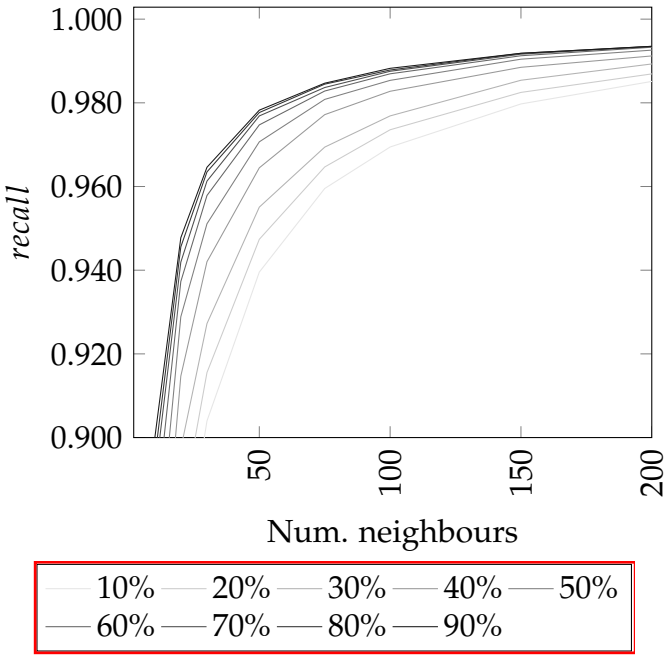


Figure 5.20: Recall of the multi-context FFP similarity using CombMAX as ranking fusion algorithm depending on the percentage of features used and the number of neighbors of the RS.

cording to the F1-score in Fig 5.21. While the ranking fusion CombMIN and CombANZ show a lower quality recommendation.

Deciding which ranking fusion algorithm to use and if to use 20% or 30% features for each contextual FFP must be done taking into consideration the average number of iterations required per similarity. Of the two, only the percentage of k features influences the number iterations per similarity. By using 10% the average number of iterations per similarity is 76, using 20% requires 101 iterations, and using 30% requires 194 iterations.

The multi-context FFP similarity using CombMAX with 20% of the features is the best performing alternative in regards to F1-score while using 100 neighbors to determine the recommendations. Recall that it requires on average 101 iterations per similarity computation. An alternative is to use CombMAX ranking fusion algorithm with also only 20% features, the RS is able to achieve good F1-score while requiring 50 neighbors to compute recommendations.

The baseline $FFP_{keywords}$ (see Table 5.9) uses 300 features and requires 213 iterations. The FFP similarity using the LH requires 88 iterations using 200 features. These results make the multi-context FFP similarity better than $FFP_{keywords}$ similarity in terms of computational efficiency, while still worst than the FFP similarity using the LH, in terms of computational efficiency.

Table 5.10 shows the comparison of the actual values obtained by the different approaches and baselines above discussed. The MC FFP-CombSUM-20% manages to have the highest precision on all the experiments while using only 50 neighbors and requiring only, on average, 101 iterations per similarity computation. As for the MC FFP-CombSUM-20%-UB the F1-score is closer to the $FFP_{keywords}$-UB which is the best performing approach in regards to F1-score. In comparison to baselines such as COS, PC, and JMSD that require 125 iterations per similarity, both the MC FFP-CombMAX-20% and the MC FFP-CombSUM-20%-UB are able to outper-

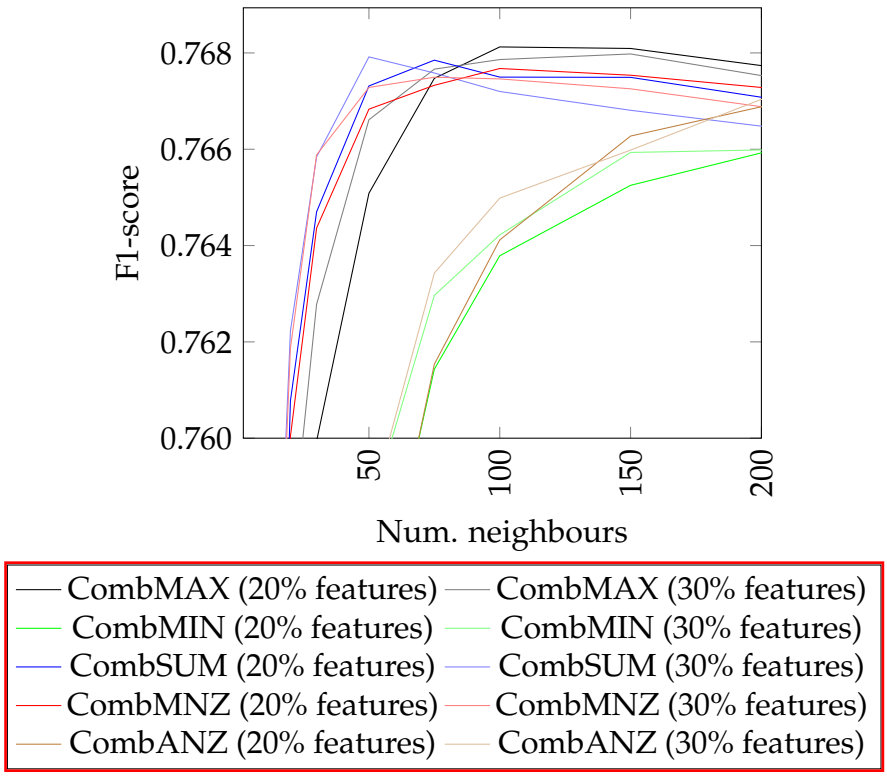


Figure 5.21: F1-score comparison of different ranking fusion algorithms using 20% and 30% of features to create FFPs, while varying the number of neighbors used by the RS.

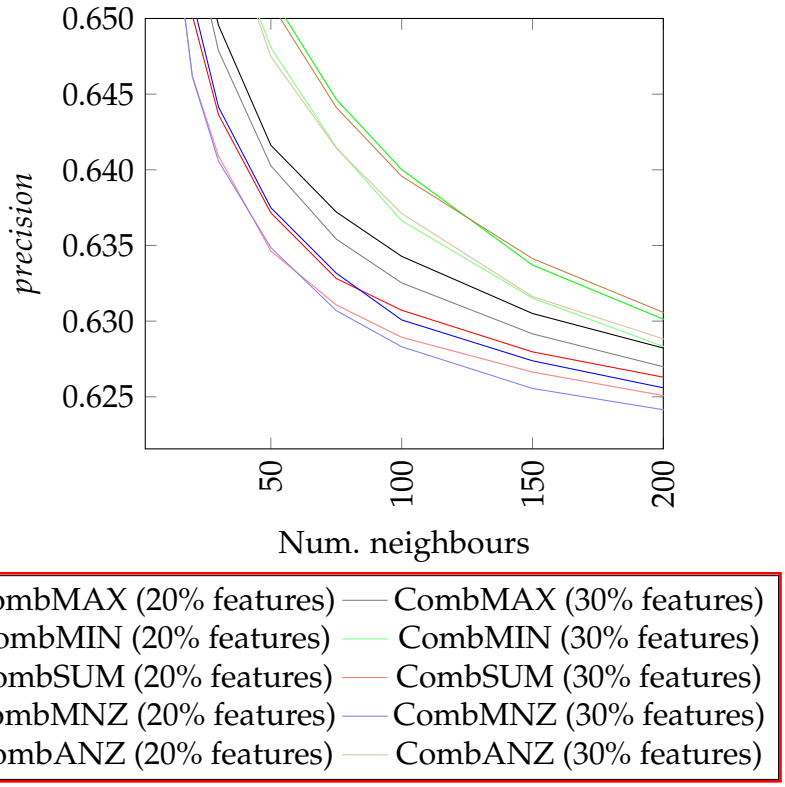


Figure 5.22: Precision comparison of different ranking fusion algorithms using 20% and 30% of features to create FFPs, while varying the number of neighbors used by the RS.

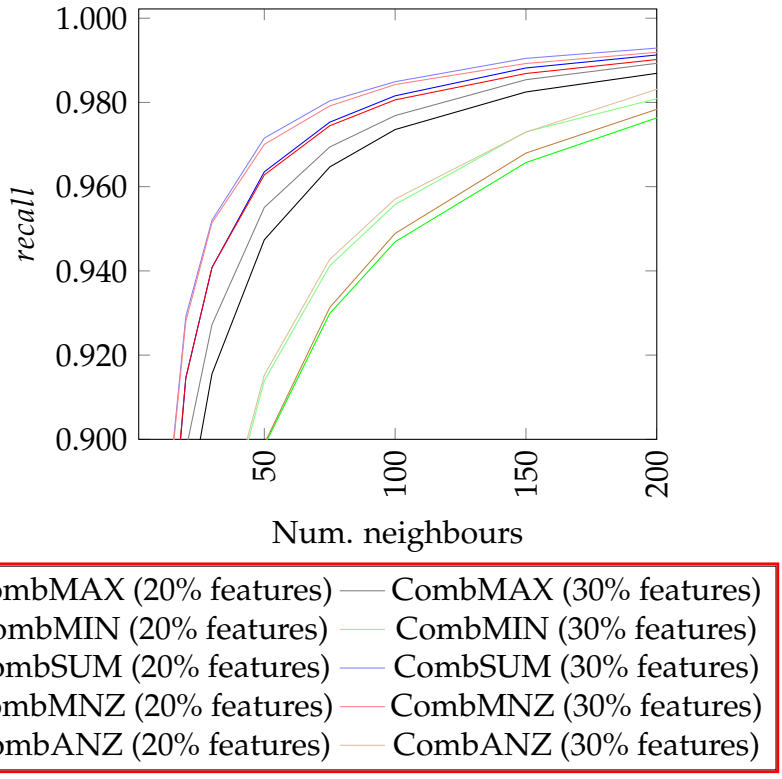


Figure 5.23: Recall comparison of different ranking fusion algorithms using 20% and 30% of features to create FFPs, while varying the number of neighbors used by the RS.

form them on F1-score, precision, and computational efficiency.

Similarity Metric	Num. neighbors	F1-score	Precision	Recall
MC *FFP*-CombSUM	50	0.76731	**0.63750**	0.96352
MC *FFP*-CombMAX	100	0.76812	0.63428	0.97356
$FFP_{keywords}$	150	**0.76929**	0.63504	0.97554
FFP_{LH}-IB	20	0.76623	0.62113	0.99979
COS-IB	50	0.76622	0.62112	0.99978
PC-IB	75	0.76621	0.62115	0.99969
JMSD-IB	20	0.76623	0.62112	0.99980
COS-UB	200	0.42356	0.26869	0.99989
PC-UB	100	0.42338	0.26854	**0.99990**
JMSD-UB	100	0.42356	0.26869	0.99989

Table 5.10: Summary results of Multi-context *FFP* similarity, compared with several baselines using CF item-based (IB) and user-based (UB). Both the MC *FFP*-CombSUM and the MC *FFP*-CombMAX use 20% of features on each contextual FFP and are a user-based approach.

5.6 Summary

This Chapter shows that using the FFP similarity can be beneficial for RSs. This work is able to improve recommendation quality while improving the computational efficiency of item-based CF systems. Experiments show that improvements in computational efficiency are still only possible using FFPs that rely exclusively on ratings. The computational efficiency of the similarity metric reduces when contextual information is added as features of FFP. This is due to the richness of information that contextual sources provide, as adding contextual information helps to improve the recommendation quality but at the expense of the FFP similarity computational efficiency. As a result, multi-context FFPs have a comparable computational efficiency to baseline similarity metrics and allow the RS to

produce higher quality recommendations to users.

Chapter 6

Conclusions and future work

This Chapter presents a summary of the achievements of this Thesis. Section 6.1 contains a closing analysis of the work developed. Following, Section 6.2 suggests future work to answer the research questions left open and possible complementary works.

6.1 Conclusions

This work proposed to create a computational efficient similarity metric for Recommender Systems, since large-scale RSs can significantly benefit from more efficient similarity metrics, while improving (or at least maintaining) the recommendation quality.

Recommender Systems work with sparse information. That sparsity is due to users rating only a few items in comparison to the extensive collection of items in a RS. Through the use of Fuzzy Fingerprints, the existing information is leveraged to improve the computational efficiency and improve recommendation quality.

The main research question is if it is possible to improve traditional CF methods using contextual information sources without increasing computational complexity? Two hypotheses were proposed. The first hypoth-

esis is to develop efficient similarity functions that can reduce the computational complexity of current CF solutions without impacting the recommendation quality. The second hypothesis is presenting a similarity function that combines contextual-information and improves the quality of recommendations while still maintaining low computational complexity.

The first hypothesis explored the FFP similarity metric adapted to RSs using CF. The second hypothesis explored several approaches: a similarity metric (using Fuzzy Fingerprints) for collaborative filtering that exploits items textual descriptions and ratings; the development of a RS that does not compute rating predictions, and in which the FFPs combine features sourced from ratings and items keywords; moreover, the last approach which uses the previously proposed RS that does not compute rating predictions only recommendations, while applying a multi-context Fuzzy Fingerprint similarity using a ranking fusion algorithm to generate similarity values.

The application of FFP similarity metric for collaborative filtering is not straightforward since ratings are on a discrete scale, commonly between 1 and 5, and FFPs needs to rank these ratings (features) by relevance. To rank them, three sorting schemes are presented: Low High, High Low, and Random. Sorting schemes leverage the number of ratings each user has and use that information to untie equal ratings. By ranking the users' ratings, a Fingerprint is formed. Fingerprints are modifiable in size, allowing to remove features to represent the items and this is why the sorting scheme is so important. To create a Fuzzy Fingerprint, a Fingerprint must then be fuzzified using a Fuzzyfying Function (FF).

Experiments with FFP similarity metric for collaborative filtering allow to draw four conclusions:

- It is possible to compute similarities using less information than is available;

- The FFP similarity can improve in most cases the RS recommendations in comparison to baselines;
- The more ratings exist in a dataset, the higher the computational improvements are possible;
- Netflix dataset has a reduction of near 75% of the average number of iterations required per similarity, the time required to compute the similarity matrix is at least 72% less, and at least 63% less time is required to compute rating predictions, in comparison with any other baseline with a comparable recommendation quality.

Such conclusions align with the first hypothesis of developing an efficient similarity function that can reduce the computational complexity of current CF solutions without impacting the recommendation quality.

To evaluate the validity of the second hypothesis, a similarity function that combines contextual information to improve the quality of recommendations while still maintaining a low computational complexity is an ambitious goal. The first proposed approach is the creation of a Fuzzy Fingerprint similarity metric that exploits both items textual descriptions and ratings using a sorting scheme named R&W. The richness of the item textual descriptions helps to improve the recommendation quality but also increases the size of FFPs. Consequently, the similarity metric computational complexity also increases. This solution uses almost the double of features that the LH SS uses. In its favor, it reduces the number of neighbors required from 25 to 15. The hypothesis requires the similarity metric to have low computational complexity. The trade-off between the improvements in recommendation quality and the added computation complexity lead to pursue other alternatives.

The second proposed approach involves the development of a RS that does not compute rating predictions only recommendations. The RS aims to produce valuable recommendations to users, without any constraint

for a maximum number of recommendations. To represent a user, the FFP registers as features the number of occurrences of each keyword on items the user rated, multiplied by the rating attributed. As a result, items a user enjoys lead also to a higher relevance of the keywords associated with that item. Relying on the neighborhood of users, if the user has not rated an item that a neighbor likes, the rating is multiplied by the similarity between the two, resulting in the value of relevance of that item for that user. The RS can generate as much item recommendations to each user as it sees fit. The experiments show that it has a better recommendation quality with an F1-score of 0.76929, while the best baseline similarity metric JMSD has an F1-score of 0.76623, an improvement of 0.00306 in F1-score. As a comparison, an item-based CF with an FFP similarity using the LH sorting scheme and $k = 200$ features, requires on average 88 iterations per similarity, while the proposed RS solution using the FFP similarity requires $k = 300$ features and has an average of 213 iterations per similarity.

The last test to the hypothesis is an approach that relies on the proposed RS that does not compute rating prediction, but only recommendations and applies to it a multi-context Fuzzy Fingerprint similarity. The similarity metric relies on ranking fusion algorithms to generate similarity values. Experiments show that the effort to create multi-context Fuzzy Fingerprints yields no significant improvements in recommendation quality, in comparison to the previously proposed approach that also endeavors to achieve the second hypothesis. Nevertheless, this last proposed approach has several variables and approaches that possibly can be better fine-tuned and optimized. These variables and approaches include the selection of a ranking fusion approach, the contextual information sources used, the individual k number of features for each contextual FFP, and the number of neighbors used by the RS.

Most RSs use CF, the rating information is sparse, but some of it can still be discarded allowing to reduce the computational load and improve

recommendations. Such improvements are due to the use of FFPs that rely exclusively on ratings. By introducing contextual information the computational complexity increases due to the additional information being used. The computational complexity is still comparable to the baselines, while providing a better recommendation quality. The baseline similarity metrics do not have the flexibility to vary the number of features used as FFP similarity has and, such characteristic is notably relevant for real-world applications. RSs have periods of peak usage, and having the flexibility to vary the k number of features allows to reduce the computation demand of the similarity metric, adjusting it to the traffic of the RS accordingly.

In conclusion, the FFP similarity uses a minimal number of features to represent an item (or user), in comparison to the original feature vector; it is easy to create and update, as new data enters the RS; there is a maximum size for each FFP, thus as new data enters the RS, the size of each FFP is limited; it is a faster similarity metric than traditional CF similarity metrics when no contextual information used; the FFP of each item (or user) is independent from each other, which also allows for an easy integration of new users and items to the system.

6.2 Future work

These are open questions left by this research and other ideas found during this work. In particular, these are the main ways to continue this work research.

Model refinement

The multi-context FFPs presented in Section 4.8 should improve the quality of recommendations and improve computational complexity. By using a different approach to select the number of features used per contextual

source instead of selecting a percentage based on the number of features that exist on each context information source. Also, the experiments include all the contextual information available. It is possible that not all the contextual information is beneficial to compute similarities.

Larger datasets

The experiments to achieve a RS with a low computational complexity using contextual information were conducted using the ML-1M dataset. The experiments on ML-1M with the FFP similarity using the LH sorting scheme do not have a meaningful improvement of computational complexity as in the datasets Netflix and Jester. For this reason, the size of the dataset used can be the root cause for being able to confirm that FFPs can provide a low complexity RS using contextual information.

Bibliography

[1] Chih-Fong Tsai and Chihli Hung. Cluster ensembles in collaborative filtering recommendation. *Applied Soft Computing*, 12(4):1417 – 1425, 2012. ISSN 1568-4946. doi: http://dx.doi.org/10.1016/j.asoc.2011.11.016. URL http://www.sciencedirect.com/science/article/pii/S1568494611004583.

[2] Dmitry Bogdanov, Martín Haro, Ferdinand Fuhrmann, Anna Xambó, Emilia Gómez, and Perfecto Herrera. Semantic audio content-based music recommendation and visualization based on user preference examples. *Information Processing and Management*, 49(1):13 – 33, 2013. ISSN 0306-4573. doi: http://dx.doi.org/10.1016/j.ipm.2012.06.004. URL http://www.sciencedirect.com/science/article/pii/S0306457312000763.

[3] G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. *Knowledge and Data Engineering, IEEE Transactions on*, 17(6): 734–749, 2005. ISSN 1041-4347. doi: 10.1109/TKDE.2005.99.

[4] J. Bobadilla, F. Ortega, A. Hernando, and A. Gutiérrez. Recommender systems survey. *Knowledge-Based Systems*, 46(0):109 – 132, 2013. ISSN 0950-7051. doi: http://dx.doi.org/10.1016/j.knosys.2013.03.012. URL http://www.sciencedirect.com/science/article/pii/S0950705113001044.

[5] Guohui Song, Shutao Sun, and Wen Fan. Applying user interest on item-based recommender system. In *Computational Sciences and Optimization (CSO) Fifth International Joint Conference on*, pages 635–638, 2012. doi: 10.1109/CSO.2012.145.

[6] Haifeng Liu, Zheng Hu, Ahmad Mian, Hui Tian, and Xuzhen Zhu. A new user similarity model to improve the accuracy of collaborative filtering. *Knowledge-Based Systems*, 56:156 – 166, 2014. ISSN 0950-7051. doi: http://dx.doi.org/10.1016/j.knosys.2013.11.006. URL http://www.sciencedirect.com/science/article/pii/S0950705113003560.

[7] Hao Ma, Tom Chao Zhou, Michael R. Lyu, and Irwin King. Improving Recommender Systems by Incorporating Social Contextual Information. *ACM Transactions on Information Systems*, 29(2):9:1–9:23, April 2011. ISSN 1046-8188. doi: 10.1145/1961209.1961212. URL http://doi.acm.org/10.1145/1961209.1961212.

[8] Tao Ye, Danny Bickson, Nicholas Ampazis, and Andras Benczur. Lsrs'15: Workshop on large-scale recommender systems. In *Proceedings of the 9th ACM Conference on Recommender Systems*, RecSys '15, pages 349–350, New York, NY, USA, 2015. ACM. ISBN 978-1-4503-3692-5. doi: 10.1145/2792838.2798715. URL http://doi.acm.org/10.1145/2792838.2798715.

[9] Christian Desrosiers and George Karypis. *A Comprehensive Survey of Neighborhood-based Recommendation Methods*, pages 107–144. Springer US, Boston, MA, 2011. ISBN 978-0-387-85820-3. doi: 10.1007/978-0-387-85820-3_4. URL http://dx.doi.org/10.1007/978-0-387-85820-3_4.

[10] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Item-based collaborative filtering recommendation algorithms. In

Proceedings of the 10th International Conference on World Wide Web, WWW '01, pages 285–295, New York, NY, USA, 2001. ACM. ISBN 1-58113-348-0. doi: 10.1145/371920.372071. URL http://doi.acm.org/10.1145/371920.372071.

[11] Alexander Felfernig, Michael Jeran, Gerald Ninaus, Florian Reinfrank, and Stefan Reiterer. Toward the Next Generation of Recommender Systems: Applications and Research Challenges. In George A. Tsihrintzis, Maria Virvou, and Lakhmi C. Jain, editors, *Multimedia Services in Intelligent Environments*, number 24 in Smart Innovation, Systems and Technologies, pages 81–98. Springer International Publishing, 2013. ISBN 978-3-319-00371-9, 978-3-319-00372-6. URL http://link.springer.com/chapter/10.1007/978-3-319-00372-6_5.

[12] Mukund Deshpande and George Karypis. Item-based top-n recommendation algorithms. *ACM Trans. Inf. Syst.*, 22(1):143–177, January 2004. ISSN 1046-8188. doi: 10.1145/963770.963776. URL http://doi.acm.org/10.1145/963770.963776.

[13] Ruzhi Xu, Shuaiqiang Wang, Xuwei Zheng, and Yinong Chen. Distributed collaborative filtering with singular ratings for large scale recommendation. *Journal of Systems and Software*, 95:231 – 241, 2014. ISSN 0164-1212. doi: http://dx.doi.org/10.1016/j.jss.2014.04.045. URL http://www.sciencedirect.com/science/article/pii/S0164121214001150.

[14] Rafael Pereira, Hélio Lopes, Karin Breitman, Vicente Mundim, and Wandenberg Peixoto. Cloud based real-time collaborative filtering for item–item recommendations. *Computers in Industry*, 65 (2):279 – 290, 2014. ISSN 0166-3615. doi: http://dx.doi.org/10.1016/j.compind.2013.11.005. URL http://www.sciencedirect.com/science/article/pii/S0166361513002352.

[15] Jesús Bobadilla, Fernando Ortega, Antonio Hernando, and Guillermo Glez de Rivera. A similarity metric designed to speed up, using hardware, the recommender systems k-nearest neighbors algorithm. *Knowledge-Based Systems*, 51(0):27 – 34, 2013. ISSN 0950-7051. doi: http://dx.doi.org/10.1016/j.knosys.2013.06.010. URL http://www.sciencedirect.com/science/article/pii/S095070511300186X.

[16] Noam Koenigstein and Yehuda Koren. Towards scalable and accurate item-oriented recommendations. In *Proceedings of the 7th ACM Conference on Recommender Systems*, RecSys '13, pages 419–422, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-2409-0. doi: 10.1145/2507157.2507208. URL http://doi.acm.org/10.1145/2507157.2507208.

[17] M. Zheng, F. Min, H. R. Zhang, and W. B. Chen. Fast recommendations with the m-distance. *IEEE Access*, 4:1464–1468, 2016. ISSN 2169-3536. doi: 10.1109/ACCESS.2016.2549182.

[18] F. Batista and J. P. Carvalho. Text based classification of companies in crunchbase. In *Fuzzy Systems (FUZZ-IEEE), 2015 IEEE International Conference on*, pages 1–7, Aug 2015. doi: 10.1109/FUZZ-IEEE.2015.7337892.

[19] Nuno Homem and João Paulo Carvalho. Mobile phone user identification with fuzzy fingerprints. In *EUSFLAT Conf.*, 2011.

[20] N. Homem and J. P. Carvalho. Authorship identification and author fuzzy 'fingerprints". In *Fuzzy Information Processing Society (NAFIPS), 2011 Annual Meeting of the North American*, pages 1–6, March 2011. doi: 10.1109/NAFIPS.2011.5751998.

[21] H. Rosa, F. Batista, and J. P. Carvalho. Twitter topic fuzzy fingerprints. In *2014 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, pages 776–783, July 2014. doi: 10.1109/FUZZ-IEEE.2014.6891781.

[22] Luís Marujo, Joao Paulo Carvalho, Anatole Gershman, Jaime Carbonell, João P. Neto, and David Martins de Matos. *Textual Event Detection Using Fuzzy Fingerprints*, pages 825–836. Springer International Publishing, Cham, 2015. ISBN 978-3-319-11313-5. doi: 10.1007/978-3-319-11313-5_72. URL http://dx.doi.org/10.1007/978-3-319-11313-5_72.

[23] Steffen Rendle. Learning Recommender Systems with Adaptive Regularization. In *Proceedings of the Fifth ACM International Conference on Web Search and Data Mining*, WSDM '12, pages 133–142, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-0747-5. doi: 10.1145/2124295.2124313. URL http://doi.acm.org/10.1145/2124295.2124313.

[24] V. Vijayakumar, V. Neelanarayanan, and Saikat Bagchi. Big data, cloud and computing challenges performance and quality assessment of similarity measures in collaborative filtering using mahout. *Procedia Computer Science*, 50:229 – 234, 2015. ISSN 1877-0509. doi: http://dx.doi.org/10.1016/j.procs.2015.04.055. URL http://www.sciencedirect.com/science/article/pii/S1877050915005566.

[25] C. Chen, J. Zeng, X. Zheng, and D. Chen. Recommender system based on social trust relationships. In *2013 IEEE 10th International Conference on e-Business Engineering*, pages 32–37, Sept 2013. doi: 10.1109/ICEBE.2013.5.

[26] Nikolaos Polatidis and Christos K. Georgiadis. A dynamic multilevel collaborative filtering method for improved recommendations. *Computer Standards & Interfaces*, 51:14 – 21, 2017. ISSN 0920-5489. doi: http://dx.doi.org/10.1016/j.csi.2016.10.014. URL //www.sciencedirect.com/science/article/pii/S0920548916301441.

[27] David Goldberg, David Nichols, Brian M. Oki, and Douglas Terry. Using collaborative filtering to weave an information tapestry. *Com-*

mun. ACM, 35(12):61–70, December 1992. ISSN 0001-0782. doi: 10.1145/138859.138867. URL http://doi.acm.org/10.1145/138859.138867.

[28] Ran Li and Xianjiu Guo. An improved algorithm to term weighting in text classification. In *Multimedia Technology (ICMT), 2010 International Conference on*, pages 1–3, Oct 2010. doi: 10.1109/ICMULT.2010.5630962.

[29] Enrico Palumbo, Giuseppe Rizzo, and Raphael Troncy. Entity2rec: Learning user-item relatedness from knowledge graphs for top-n item recommendation. In *Proceedings of the Eleventh ACM Conference on Recommender Systems*, RecSys '17, pages 32–36, New York, NY, USA, 2017. ACM. ISBN 978-1-4503-4652-8. doi: 10.1145/3109859.3109889. URL http://doi.acm.org/10.1145/3109859.3109889.

[30] Parivash Pirasteh, Dosam Hwang, and Jason J. Jung. Exploiting matrix factorization to asymmetric user similarities in recommendation systems. *Knowledge-Based Systems*, 83:51–57, July 2015. ISSN 0950-7051. doi: 10.1016/j.knosys.2015.03.006. URL http://www.sciencedirect.com/science/article/pii/S0950705115000982.

[31] Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *IEEE Computer*, 42(8):30–37, August 2009. ISSN 0018-9162. doi: 10.1109/MC.2009.263.

[32] Mohsen Jamali and Martin Ester. A matrix factorization technique with trust propagation for recommendation in social networks. In *Proceedings of the Fourth ACM Conference on Recommender Systems*, RecSys '10, pages 135–142, New York, NY, USA, 2010. ACM. ISBN 978-1-60558-906-0. doi: 10.1145/1864708.1864736. URL http://doi.acm.org/10.1145/1864708.1864736.

[33] Xiwang Yang, Yang Guo, Yong Liu, and Harald Steck. A survey of collaborative filtering based social recommender systems. *Computer Communications*, 41:1 – 10, 2014. ISSN 0140-3664. doi: https://doi.org/10.1016/j.comcom.2013.06.009. URL http://www.sciencedirect.com/science/article/pii/S0140366413001722.

[34] Guibing Guo, Jie Zhang, and Daniel Thalmann. Merging trust in collaborative filtering to alleviate data sparsity and cold start. *Knowledge-Based Systems*, 57:57 – 68, 2014. ISSN 0950-7051. doi: https://doi.org/10.1016/j.knosys.2013.12.007. URL http://www.sciencedirect.com/science/article/pii/S0950705113003870.

[35] Hael Al-bashiri, Mansoor Abdullateef Abdulgabber, Awanis Romli, and Hasan Kahtan. An improved memory-based collaborative filtering method based on the topsis technique. *PLOS ONE*, 13(10): 1–26, 10 2018. doi: 10.1371/journal.pone.0204434. URL https://doi.org/10.1371/journal.pone.0204434.

[36] Jesús Bobadilla, Fernando Ortega, and Antonio Hernando. A collaborative filtering similarity measure based on singularities. *Information Processing Management*, 48(2):204 – 217, 2012. ISSN 0306-4573. doi: https://doi.org/10.1016/j.ipm.2011.03.007. URL http://www.sciencedirect.com/science/article/pii/S0306457311000409.

[37] J. Bobadilla, F. Serradilla, and J. Bernal. A new collaborative filtering metric that improves the behavior of recommender systems. *Knowledge-Based Systems*, 23(6):520 – 528, 2010. ISSN 0950-7051. doi: http://dx.doi.org/10.1016/j.knosys.2010.03.009. URL http://www.sciencedirect.com/science/article/pii/S0950705110000444.

[38] Charu C. Aggarwal. *Recommender Systems: The Textbook*. Springer International Publishing, 2016. ISBN 978-3-319-29657-9. URL https://www.springer.com/kr/book/9783319296579.

[39] Pasquale Lops, Marco de Gemmis, and Giovanni Semeraro. *Content-based Recommender Systems: State of the Art and Trends*, pages 73–105. Springer US, Boston, MA, 2011. ISBN 978-0-387-85820-3. doi: 10.1007/978-0-387-85820-3_3. URL https://doi.org/10.1007/978-0-387-85820-3_3.

[40] Robin Burke. *Hybrid Web Recommender Systems*, pages 377–408. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007. ISBN 978-3-540-72079-9. doi: 10.1007/978-3-540-72079-9_12. URL https://doi.org/10.1007/978-3-540-72079-9_12.

[41] Jesus Alcala-Fdez and Jose Alonso. A survey of fuzzy systems software: Taxonomy, current research trends and prospects. *IEEE Transactions on Fuzzy Systems*, 24:40–56, 02 2016. doi: 10.1109/TFUZZ.2015.2426212.

[42] L.A. Zadeh. Fuzzy sets. *Information and Control*, 8(3):338 – 353, 1965. ISSN 0019-9958. doi: https://doi.org/10.1016/S0019-9958(65)90241-X. URL http://www.sciencedirect.com/science/article/pii/S001999586590241X.

[43] Mihai Florea, Anne-Laure Jousselme, Dominic Grenier, and Éloi Bossé. Combining belief functions and fuzzy membership functions. *Proceedings of SPIE - The International Society for Optical Engineering*, 04 2003. doi: 10.1117/12.487366.

[44] Christopher C. Vogt and Garrison W. Cottrell. Fusion via a linear combination of scores. *Information Retrieval*, 1(3):151–173, Oct 1999. ISSN 1573-7659. doi: 10.1023/A:1009980820262. URL https://doi.org/10.1023/A:1009980820262.

[45] Joon Ho Lee. Analyses of multiple evidence combination. *SIGIR Forum*, 31(SI):267–276, July 1997. ISSN 0163-5840. doi: 10.1145/278459.258587. URL http://doi.acm.org/10.1145/278459.258587.

[46] Jon Fraenkel and Bernard Grofman. The borda count and its real-world alternatives: Comparing scoring rules in nauru and slovenia. *Australian Journal of Political Science*, 49, 04 2014. doi: 10.1080/10361146.2014.900530.

[47] Gilbert Laffond, Jean François Laslier, and Michel Le Breton. Condorcet choice correspondences: A set-theoretical comparison. *Mathematical Social Sciences*, 30(1):23 – 35, 1995. ISSN 0165-4896. doi: https://doi.org/10.1016/0165-4896(94)00778-7. URL http://www.sciencedirect.com/science/article/pii/0165489694007787.

[48] Gordon V. Cormack, Charles L A Clarke, and Stefan Buettcher. Reciprocal rank fusion outperforms condorcet and individual rank learning methods. In *Proceedings of the 32Nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '09, pages 758–759, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-483-6. doi: 10.1145/1571941.1572114. URL http://doi.acm.org/10.1145/1571941.1572114.

[49] Dietmar Jannach, Markus Zanker, Alexander Felfernig, and Gerhard Friedrich. *Recommender Systems: An Introduction*. Cambridge University Press, 2010.

[50] G. Guo, J. Zhang, and N. Yorke-Smith. A novel bayesian similarity measure for recommender systems. In *Proceedings of the 23rd International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2619–2625, 2013.

[51] Hyung Jun Ahn. A new similarity measure for collaborative filtering to alleviate the new user cold-starting problem. *Information Sciences*, 178(1):37 – 51, 2008. ISSN 0020-0255. doi: http://dx.doi.org/10.1016/j.ins.2007.07.024. URL //www.sciencedirect.com/science/article/pii/S0020025507003751.

[52] Vahid Faridani, Mehrdad Jalali, and Majid Vafaei Jahan. Collaborative filtering-based recommender systems by effective trust. *International Journal of Data Science and Analytics*, 03 2017. doi: 10.1007/s41060-017-0049-y.

[53] A. T. Wibowo and A. Rahmawati. Naive random neighbor selection for memory based collaborative filtering. In *Intelligent Technology and Its Applications (ISITIA), 2015 International Seminar on*, pages 351–356, May 2015. doi: 10.1109/ISITIA.2015.7220005.

[54] Christian Matyas and Christoph Schlieder. A spatial user similarity measure for geographic recommender systems. In Krzysztof Janowicz, Martin Raubal, and Sergei Levashkin, editors, *GeoSpatial Semantics*, volume 5892 of *Lecture Notes in Computer Science*, pages 122–139. Springer Berlin Heidelberg, 2009. ISBN 978-3-642-10435-0. doi: 10.1007/978-3-642-10436-7_8. URL http://dx.doi.org/10.1007/978-3-642-10436-7_8.

[55] Yue Shi, Martha Larson, and Alan Hanjalic. Collaborative filtering beyond the user-item matrix: A survey of the state of the art and future challenges. *ACM Computing Surveys*, 47(1):3:1–3:45, May 2014. ISSN 0360-0300. doi: 10.1145/2556270. URL http://doi.acm.org/10.1145/2556270.

[56] Andreas Töscher, Michael Jahrer, and Robert M. Bell. The bigchaos solution to the netflix grand prize, 2009.

[57] João Vinagre, Alípio Mário Jorge, and João Gama. Fast incremental matrix factorization for recommendation with positive-only feedback. In Vania Dimitrova, Tsvi Kuflik, David Chin, Francesco Ricci, Peter Dolog, and Geert-Jan Houben, editors, *User Modeling, Adaptation, and Personalization*, pages 459–470, Cham, 2014. Springer International Publishing. ISBN 978-3-319-08786-3.

[58] João Vinagre, Alípio Jorge, and João Gama. Online gradient boosting for incremental recommender systems. pages 209–223, 10 2018. ISBN 978-3-030-01770-5. doi: 10.1007/978-3-030-01771-2_14.

[59] Alípio M. Jorge, João Vinagre, Marcos Domingues, João Gama, Carlos Soares, Pawel Matuszyk, and Myra Spiliopoulou. Scalable online top-n recommender systems. In Derek Bridge and Heiner Stuckenschmidt, editors, *E-Commerce and Web Technologies*, pages 3–20, Cham, 2017. Springer International Publishing. ISBN 978-3-319-53676-7.

[60] Catarina Miranda and Alípio Jorge. Item-based and user-based incremental collaborative filtering for web recommendations. pages 673–684, 10 2009. doi: 10.1007/978-3-642-04686-5_55.

[61] Bo Yang, Yu Lei, Dayou Liu, and Jiming Liu. Social collaborative filtering by trust. In *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence*, IJCAI '13, pages 2747–2753. AAAI Press, 2013. ISBN 978-1-57735-633-2. URL http://dl.acm.org/citation.cfm?id=2540128.2540524.

[62] Guibing Guo, Jie Zhang, and Neil Yorke-Smith. Trustsvd: Collaborative filtering with both the explicit and implicit influence of user trust and of item ratings. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, AAAI'15, pages 123–129. AAAI Press, 2015. ISBN 0-262-51129-0. URL http://dl.acm.org/citation.cfm?id=2887007.2887025.

[63] S. Rendle. Factorization Machines. In *2010 IEEE 10th International Conference on Data Mining (ICDM)*, pages 995–1000, December 2010. doi: 10.1109/ICDM.2010.127. URL http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5694074.

[64] Chen Chen, Wu Dongxing, Hou Chunyan, and Yuan Xiaojie. Exploiting Social Media for Stock Market Prediction with Factoriza-

tion Machine. In *IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT)*, pages 142–149. IEEE, August 2014. ISBN 978-1-4799-4143-8. doi: 10.1109/WI-IAT.2014.91. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6927618.

[65] Steffen Rendle. Scaling factorization machines to relational data. *Proceedings VLDB Endowment*, 6(5):337–348, March 2013. ISSN 2150-8097. doi: 10.14778/2535573.2488340. URL http://dx.doi.org/10.14778/2535573.2488340.

[66] Mu Li, Ziqi Liu, Alexander J. Smola, and Yu-Xiang Wang. Difacto: Distributed factorization machines. In *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining*, WSDM '16, pages 377–386, New York, NY, USA, 2016. ACM. ISBN 978-1-4503-3716-8. doi: 10.1145/2835776.2835781. URL http://doi.acm.org/10.1145/2835776.2835781.

[67] Bing Li, Pei lin Zhang, Hao Tian, Shuang shan Mi, Dong sheng Liu, and Guo quan Ren. A new feature extraction and selection scheme for hybrid fault diagnosis of gearbox. *Expert Systems with Applications*, 38(8):10000 – 10009, 2011. ISSN 0957-4174. doi: http://dx.doi.org/10.1016/j.eswa.2011.02.008. URL http://www.sciencedirect.com/science/article/pii/S0957417411002144.

[68] Emil J. Khatib, Raquel Barco, Ana Gómez-Andrades, Pablo Muñoz, and Inmaculada Serrano. Data mining for fuzzy diagnosis systems in {LTE} networks. *Expert Systems with Applications*, 42(21):7549 – 7559, 2015. ISSN 0957-4174. doi: http://dx.doi.org/10.1016/j.eswa.2015.05.031. URL http://www.sciencedirect.com/science/article/pii/S0957417415003590.

[69] B. Dennis and S. Muthukrishnan. Agfs: Adaptive genetic fuzzy

system for medical data classification. *Applied Soft Computing*, 25: 242 – 252, 2014. ISSN 1568-4946. doi: http://dx.doi.org/10.1016/j.asoc.2014.09.032. URL http://www.sciencedirect.com/science/article/pii/S1568494614004852.

[70] Thanh Nguyen, Abbas Khosravi, Douglas Creighton, and Saeid Nahavandi. Medical data classification using interval type-2 fuzzy logic system and wavelets. *Applied Soft Computing*, 30:812 – 822, 2015. ISSN 1568-4946. doi: http://dx.doi.org/10.1016/j.asoc.2015.02.016. URL http://www.sciencedirect.com/science/article/pii/S1568494615001076.

[71] Nguyen Tho Thong and Le Hoang Son. Hifcf: An effective hybrid model between picture fuzzy clustering and intuitionistic fuzzy recommender systems for medical diagnosis. *Expert Systems with Applications*, 42(7):3682 – 3701, 2015. ISSN 0957-4174. doi: http://dx.doi.org/10.1016/j.eswa.2014.12.042. URL http://www.sciencedirect.com/science/article/pii/S0957417414008215.

[72] Zoran Sevarac, Vladan Devedzic, and Jelena Jovanovic. Adaptive neuro-fuzzy pedagogical recommender. *Expert Systems with Applications*, 39(10):9797 – 9806, 2012. ISSN 0957-4174. doi: http://dx.doi.org/10.1016/j.eswa.2012.02.174. URL http://www.sciencedirect.com/science/article/pii/S095741741200437X.

[73] Mohammad Yahya H. Al-Shamri and Kamal K. Bharadwaj. Fuzzy-genetic approach to recommender systems based on a novel hybrid user model. *Expert Systems with Applications*, 35(3):1386 – 1399, 2008. ISSN 0957-4174. doi: http://dx.doi.org/10.1016/j.eswa.2007.08.016. URL http://www.sciencedirect.com/science/article/pii/S095741740700351X.

[74] Nicolás Marín, Olga Pons, Luis M. de Campos, Juan M. Fernández-Luna, and Juan F. Huete. Advances in intelligent databases and information systems a collaborative recommender system based on probabilistic inference from fuzzy observations. *Fuzzy Sets and Systems*, 159(12):1554 – 1576, 2008. ISSN 0165-0114. doi: http://dx.doi.org/10.1016/j.fss.2008.01.016. URL http://www.sciencedirect.com/science/article/pii/S0165011408000353.

[75] Le Hoang Son. Hu-fcf: A hybrid user-based fuzzy collaborative filtering method in recommender systems. *Expert Syst. Appl.*, 41(15):6861–6870, November 2014. ISSN 0957-4174. doi: 10.1016/j.eswa.2014.05.001. URL http://dx.doi.org/10.1016/j.eswa.2014.05.001.

[76] Le Hoang Son. Hu-fcf++: A novel hybrid method for the new user cold-start problem in recommender systems. *Engineering Applications of Artificial Intelligence*, 41:207 – 222, 2015. ISSN 0952-1976. doi: http://dx.doi.org/10.1016/j.engappai.2015.02.003. URL //www.sciencedirect.com/science/article/pii/S0952197615000330.

[77] Jaideep Vaidya and Chris Clifton. Privacy preserving association rule mining in vertically partitioned data. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '02, pages 639–644, New York, NY, USA, 2002. ACM. ISBN 1-58113-567-X. doi: 10.1145/775047.775142. URL http://doi.acm.org/10.1145/775047.775142.

[78] Raciel Yera, Jorge Castro, and Luis Martínez. A fuzzy model for managing natural noise in recommender systems. *Applied Soft Computing*, 40:187 – 198, 2016. ISSN 1568-4946. doi: 10.1016/j.asoc.2015.10.060. URL http://www.sciencedirect.com/science/article/pii/S1568494615007048.

[79] Rosa M. Rodríguez and Luis Martínez. An analysis of symbolic linguistic computing models in decision making. *International Journal of General Systems*, 42(1):121–136, 2013. doi: 10.1080/03081079.2012.710442. URL https://doi.org/10.1080/03081079.2012.710442.

[80] Vesselin Dimiev. Fuzzifying functions. *Fuzzy Sets and Systems*, 33(1):47 – 58, 1989. ISSN 0165-0114. doi: http://dx.doi.org/10.1016/0165-0114(89)90216-9. URL http://www.sciencedirect.com/science/article/pii/0165011489902169.

[81] Paolo Cremonesi, Yehuda Koren, and Roberto Turrin. Performance of recommender algorithms on top-n recommendation tasks. In *Proceedings of the Fourth ACM Conference on Recommender Systems*, RecSys '10, pages 39–46, New York, NY, USA, 2010. ACM. ISBN 978-1-60558-906-0. doi: 10.1145/1864708.1864721. URL http://doi.acm.org/10.1145/1864708.1864721.

[82] S GOPAL PATRO and Kishore Kumar Sahu. Normalization: A preprocessing stage. *IARJSET*, 03 2015. doi: 10.17148/IARJSET.2015.2305.

[83] M.M. Gupta and J. Qi. Theory of t-norms and fuzzy inference methods. *Fuzzy Sets and Systems*, 40(3):431 – 450, 1991. ISSN 0165-0114. doi: http://dx.doi.org/10.1016/0165-0114(91)90171-L. URL http://www.sciencedirect.com/science/article/pii/016501149190171L. Fuzzy Logic and Uncertainty Modelling.

[84] Alejandro Bellogin, Pablo Castells, and Ivan Cantador. Precision-oriented evaluation of recommender systems: An algorithmic comparison. In *Proceedings of the Fifth ACM Conference on Recommender Systems*, RecSys '11, pages 333–336, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0683-6. doi: 10.1145/2043932.2043996. URL http://doi.acm.org/10.1145/2043932.2043996.

[85] Mouzhi Ge, Carla Delgado-Battenfeld, and Dietmar Jannach. Beyond accuracy: evaluating recommender systems by coverage and serendipity. In *Proceedings of the fourth ACM conference on Recommender systems*, RecSys '10, pages 257–260, New York, NY, USA, 2010. ACM. ISBN 978-1-60558-906-0. doi: 10.1145/1864708.1864761. URL http://doi.acm.org/10.1145/1864708.1864761.

[86] Alan Said and Alejandro Bellogín. Rival: a toolkit to foster reproducibility in recommender system evaluation. In *RecSys '14 Proceedings of the 8th ACM Conference on Recommender Systems*, pages 371–372. ACM Press, 2014. ISBN 9781450326681. doi: 10.1145/2645710.2645712. URL http://dl.acm.org/citation.cfm?doid=2645710.2645712.

Appendices

Appendix A

Fuzzy Fingerprints for Item-Based Collaborative Filtering

Fuzzy Fingerprints for Item-Based Collaborative Filtering

André Carvalho[1], Pável Calado[2], and Joao Paulo Carvalho[3]

INESC-ID, Instituto Superior Técnico, Universidade de Lisboa, Portugal
[1] andre.silva.carvalho@tecnico.ulisboa.pt,
[2] pavel.calado@tecnico.ulisboa.pt,
[3] joao.carvalho@inesc-id.pt

Abstract. Memory-based Collaborative filtering solutions are dominant in the Recommender Systems domain, due to its low implementation effort and service maintenance when compared with Model-based approaches. Memory-based systems often rely on similarity metrics to compute similarities between items (or users). Such metrics can be improved either by improving comparison quality or minimizing computational complexity. There is, however, an important trade-off — in general, models with high complexity, which significantly improve recommendations, are computationally unfeasible for real-world applications. In this work, we approach this issue, by applying Fuzzy Fingerprints to create a novel similarity metric for Collaborative Filtering. Fuzzy Fingerprints provide a concise representation of items, by selecting a relatively small number of user ratings and using their order to describe them. This metric requires from 23% through 95% less iterations to compute the similarities required for a rating prediction, depending on the density of the dataset. Despite this reduction, experiments performed in three datasets show that our metric is still able to have comparable recommendation results, in relation to state-of-art similarity metrics.

1 Introduction

Users of the digital world are overloaded with information [13]. Recommender Systems (RSs) allow us to cope with this, by cataloging a vast list of items, that later can be recommended. Due to their success, RSs can be found in a number of services, providing recommendations for movies, music, news, products, events, services, among others [1].

However, turning state of the art solutions into real-world scenarios is still challenging, mainly due to a large amount of data available and the scalability issues that ensue. For this reason, more traditional approaches, such as item-based Collaborative Filtering (CF) are still the most widely used [16]. Despite its simplicity, item-based CF can provide quite accurate results, thus yielding an advantageous trade-off between engineering effort and user satisfaction.

In CF systems, the issue of scalability is closely related to the need to compute similarities between a high number of items in the database. To solve this,

two complementary types of solution are usually proposed. One is to provide scalability by distributing the storage and computational cost over several machines [15,10]. The other is to devise computationally efficient similarity metrics [3,8,14,18]. Our work focuses on the latter.

Our main contribution is a novel similarity metric for RSs, using the concept of Fuzzy Fingerprints (FFPs) [7]. More specifically, we propose to represent items by their low-dimensional Fingerprints, which can then be directly used to determine similarities between them. A similar idea has been previously applied to text authorship identification [7] with success. Our goal is to apply the same principle to RSs. This solution has three major advantages: (1) it has a smaller computational cost than traditional similarity metrics; (2) it requires a minimal implementation effort; and (3) the proposed representation of the items is also easily maintainable.

To demonstrate our claims, experiments were performed on three datasets. Results show that FFPs are a promising route to be applied for recommendations, requiring from 23% through 95% less iterations to compute the similarities for a rating prediction, depending on the density of the dataset. This improvement is achieved while maintaining a comparable quality of results.

The remainder of this paper is organized as follows. Section 2 contains literature review on similarity metrics for CF. Section 3 presents how FFPs can be applied to RSs. Section 4 presents an experimental evaluation. Finally, in Section 5 some conclusions are drawn from the results and directions for future work are proposed.

2 Related Work

Even thought Fuzzy systems have been previously applied to RSs, they have never been specifically used to improve the RS similarity metric [12,17]. Our proposal applies concepts of Fuzzy Systems to the problem of item-based Collaborative Filtering. More specifically, we use the Fuzzy Fingerprints to represent items in a CF system.

CF systems usually rely on the ratings given to items by users to determine similarities between items (or users), through the use of a similarity metric. This allows the creation of neighborhoods of similar items, to predict new ratings. Traditionally, the similarity is measured using metrics such as Pearson Correlation (PC) or the Cosine similarity (COS) [2]. Nevertheless, many other ways of measuring similarity have been proposed, ranging from simple variations of PC and COS, through the design of more complex functions.

An example is the work of [5], where ratings are combined with a measure of *trust* between users, which is inferred from social information. The authors show that such combination does improve the overall rating prediction. On a different approach, in [4], the authors propose a combination of the mean squared difference between the user's ratings with the Jaccard coefficient. Through experiments, they demonstrate that results are improved, when compared to traditional CF.

Liu et al. [9] also propose a new similarity metric, which attributes penalties to *bad* similarities, while rewarding *good* similarities. Defining a similarity as good or bad depends on several factors, such as the popularity of the rated items or the similarity of the rating to the other user's ratings. Finally, in [18], authors propose an alternative called M-distance-based recommendation (MBR). They leverage the average rating of each item and use the difference of such averages as the distance between items. Authors also have shown that it is possible to pre-compute such averages and compute the similar items neighborhood in linear time. Since this is the most time-efficient work that we know of, we use it as a baseline in our experiments.

The above works show that improving the similarity measures has a beneficial impact on the overall RS results. Nevertheless, this is often done at the expense of an increase in the computational complexity. In this work, we introduce a similarity metric based on FFPs, adapted for item-based CF, that aims to improve time-efficiency while maintaining a low implementation effort and a comparable, or even better, recommendation accuracy.

3 Fuzzy Fingerprints for Collaborative Filtering

We now explain how the Fingerprint of a given item is created and, following, how a Fuzzifying Function can be applied to obtain the corresponding FFP.

Let r_i be the set of ratings that a given set of users $u_1 \cdots, u_N$ has provided for item i be: $r_i = \{(u_1, r_{1i}), (u_2, r_{2i}), \cdots, (u_N, r_{Ni})\}$. To build the Fingerprint ϕ_i, we start by choosing a subset of k ratings in r_i, where k is the parameter that controls the size of the Fingerprint. The idea is that the selected ratings should be those that best represent item i. To this effect, we select the k highest ratings.

However, since users usually provide ratings on a small discrete scale (e.g. 1,2,3,4, or 5 stars), we still need to give a different importance to the possibly many ratings with the same value. Thus, when two ratings are equal, we use what we call a sorting scheme (SS) to decide which one will have a higher rank. In our experiments, we evaluated three different SSs. Let $\#u_j$ be the total number of items that user u_j has rated: (1) Random: equal ratings are ordered randomly (to be used as a baseline); (2) Higher to Lower (HL): equal ratings are sorted in descending order according to $\#u_j$; (2) Lower to Higher (LH): equal ratings are sorted in ascending according to $\#u_j$.

To illustrate, let $r_i = \{(a,5), (b,2), (d,5), (e,4), (f,2), (h,1), (i,2)\}$, assume that $k = 4$, and $\#a > \#b > \cdots > \#i$. The resulting Fingerprints ϕ_i, using each of the above sorting schemes would be $\phi_i^{(\text{random})} = \{(d,5), (a,5), (e,4), (f,2)\}$, $\phi_i^{(\text{HL})} = \{(a,5), (d,5), (e,4), (b,2)\}$, and $\phi_i^{(\text{LH})} = \{(d,5), (a,5), (e,4), (i,2)\}$.

The Fingerprint ϕ_i is, in fact, an *ordered set* of ratings. This order, determined by a SS and, reflects the importance of each rating to represent items. It is by leveraging on this importance that we determine the Fuzzy Fingerprint of item i, Φ_i.

A Fuzzyfying Function (FF) $\mu(idx)$ assigns a weight to each position in a Fingerprint. In this case, the FF is used to assign a weight to each user in ϕ_i.

There are many alternatives to define the FF [6]. Here, we have tested three possibilities, shown in Equation 1, where p_{u_j} is the position of user u_j within ϕ_i.

$$\mu_{one}(\mathrm{p}_{u_j}) = 1 \qquad \mu_{linear}(\mathrm{p}_{u_j}) = \frac{k - \mathrm{p}_{u_j}}{k} \qquad \mu_{erfc}(\mathrm{p}_{u_j}) = 1 - \mathrm{erfc}(\frac{2 \times \mathrm{p}_{u_j}}{k}) \quad (1)$$

The function μ_{one} assigns an equal weight to all users ratings. Using function μ_{linear}, the weight of a user decreases linearly, according to its position p_{u_j}. Finally, function μ_{erfc}, uses a variation of the *complementary error function* to yield a faster decrease in weights. It is important to note that these functions are not the only available options [7]. However, preliminary experiments have indicated that using other variations does not significantly improve the quality of the results. For this reason, we have not tested further alternatives.

Using one of the above fuzzifying functions, we can now define the FFP Φ_i as: $\Phi_i = \{(u_j, \mu(\mathrm{p}_{u_j})), \forall u_j \in \phi_i\}$. The FFP is, therefore, the set of users in the original Fingerprint, each with an associated weight, given by theFuzzyfying Function. It is, in effect, a *fuzzy set of users* that rated item i.

Once the FFP for each item is determined, it is possible to compute similarities between items.

Consider Φ_i and Φ_j the FFPs of items i and j, respectively. Let U_i be the set of users in Φ_i and U_j be the set of users in Φ_j. The FFP similarity between items i and j is defined as:

$$\mathrm{sim}(\Phi_i, \Phi_j) = \sum_{u_v \in U_i \cap U_j} \frac{\min(\Phi_i(u_v), \Phi_j(u_v))}{k} \quad (2)$$

where $\Phi_x(u_v)$ denotes the the value associated to user u_v in Φ_x. This similarity, in fact, corresponds to a *minimum t-norm* between the two fuzzy sets represented by the FFPs.

Rating predictions can now be obtained in a process similar to traditional Collaborative Filtering. More specifically, let $\widehat{r}_{vi}$ be the *predicted* rating that a given user u_v would assign to item i. We start by computing the *neighborhood* of item i, i.e. the set of n items in the database that are more similar to i, $N_i(v)$, using the similarity function defined in Eq. (2). The value of $\widehat{r}_{vi}$ is defined as:

$$\widehat{r}_{vi} = \bar{r}_i + \frac{\sum_{j \in N_i(v)} \mathrm{sim}(\Phi_i, \Phi_j) \times (r_{vj} - \bar{r}_j)}{\sum_{j \in N_i(v)} sim(\Phi_i, \Phi_j)} \quad (3)$$

where r_{vj} is the rating assigned by user u to item j, $\bar{r}_x$ is the average of all ratings assigned to item x. A RS will usually perform these predictions for a large set of items and return those with the highest rating predictions, thus creating recommendations for a user.

4 Evaluation

4.1 Experimental Setup

To assert the effectiveness of FFPs, experiments were performed using four baseline similarity metrics and three distinct datasets. The similarity metrics used as a baseline for comparison are the traditional Pearson Correlation (PC) and Cosine similarity (COS). In addition, we also include theJaccard Mean Squared Difference (JMSD) [4], an improvement on previous metrics that offers a high rating prediction accuracy, while using a lower number of neighbors. Finally, an alternative similarity metric, named MBR [18], is also compared since its authors have the exact same goal as ours.

The Pearson Correlation coefficient has been widely used since it is simple to implement, intuitive, and provides good quality results [4]. PC is defined in Eq. 4, where U is the set users that rated both items i and j.

$$sim_{PC}(i,j) = \frac{\sum_{u \in U}(r_{u,i} - \bar{r}_i) \times (r_{u,j} - \bar{r}_j)}{\sqrt{\sum_{u \in U}(r_{u,i} - \bar{r}_i)^2} \times \sqrt{\sum_{u \in U}(r_{u,j} - \bar{r}_j)^2}} \tag{4}$$

The resulting similarity will be in within the interval $[-1, 1]$, where -1 corresponds to an inverse correlation, $+1$ to a positive correlation, and values near zero show that no linear correlation exists between the two items.

Another often used similarity measure is the Cosine similarity, as defined in Eq. 5. COS will yield a value between 0 and 1, where 0 corresponds to no similarity between i and j and 1 to exactly proportional ratings between both users.

$$sim_{COS}(i,j) = \frac{\sum_{u \in U} r_{u,i} \times r_{u,j}}{\sqrt{\sum_{u \in U} r_{u,i}^2} \times \sqrt{\sum_{u \in U} r_{u,j}^2}} \tag{5}$$

The idea behind Jaccard Mean Squared Difference (JMSD) it to combine the Jaccard coefficient, which captures the number of ratings in common between items, with the Mean Square Difference (MSD) of those ratings, resulting in Eq. 6:

$$sim_{JMSD}(i,j) = Jaccard(i,j) \times \big(1 - MSD(i,j)\big); \tag{6}$$

where $Jaccard$ and MSD are defined as:

$$Jaccard(i,j) = \frac{|U_i \cap U_j|}{|U_i \cup U_j|} \qquad MSD(i,j) = \frac{\sum_{u \in U}(r_{u,i} - r_{u,j})^2}{|U|} \tag{7}$$

where U_s is the set of items ranked by user s.

The MBR metric uses a different principle, as shown in Eq. 8. It starts by computing the average rating $\bar{r}_j$ of each item j. The absolute value of the difference between these average ratings (called MBR) determines the similarity between the items. The set of neighbors H_i of item i is defined as all items $j \neq i$ such that $MBR(i,j) \leq T$, where T is a predefined threshold. Rating predictions

$\hat{r}_{u,i}$ can then be determined such that the rating predicted for user u and item i is the average of all ratings given by u to items in H_i.

$$MBR(i,j) = \left|\bar{r}_i - \bar{r}_j\right| \qquad \hat{r}_{u,i} = \frac{\sum_{j \in H_i \cap U_u} r_{u,j}}{|H_i \cap U_u|} \tag{8}$$

Evaluation was conducted using three standard datasets: (1) MovieLens-1M (ML-1M), a dataset from the movie domain; (2) Netflix, a large dataset, also from the movie domain with a very sparse user-items ratings matrix; and (3) Jester, a dataset for recommending jokes, with a high number of ratings per item. Table 1 shows some statistics regarding their contents.

Table 1. Statistics for the experimental datasets. Column *sparsity* shows the percentage of not rated items in the rating matrix and column $\#\bar{r}_i$ shows the average number of ratings per item.

Dataset	Ratings	Users	Items	sparsity	$\#\bar{r}_i$
ML-1M	1 000 209	6 040	3 706	95.53%	217
Jester	1 728 785	79 681	150	75.64%	12348
Netflix	100 000 000	480 189	17 770	98.82%	5576

All experiments were conducted using the RIVAL framework [11]. All measurements result from a 5 fold cross-validation, where the ratings are split on a user basis. The exception is the Netflix dataset, where we used the provided *probe* test set, to make our results comparable to those found in most literature.

4.2 Results

Recommendation Effectiveness We start by evaluating the results yielded by different sorting schemes. Figure 1 presents the RMSE for the three proposed sorting schemes, while varying the value of k, using μ_{linear} as theFuzzyfying Function. We note that the different scale for the Jester dataset is required since its ratings vary between 1 and 10, whereas the ratings for the ML-1M and Netflix datasets vary between 1 and 5.

We can see that, in all datasets, the three sorting schemes show a similar behavior. The best results are achieved, in general, by the LH sorting scheme, while the worst are achieved by the HL sorting scheme. This indicates that ratings given by the least active users are better sources of information. A possible explanation for this phenomena is that very active users tend to give the same rating to a large number of items that, in practice, vary widely in quality (as perceived by the user). On the other hand, less active users are more discriminatory when evaluating the items.

We now evaluate the impact on RMSE for differentFuzzyfying Functions. Figure 2 presents the values obtained, while varying the value of k, using the LH sorting scheme.

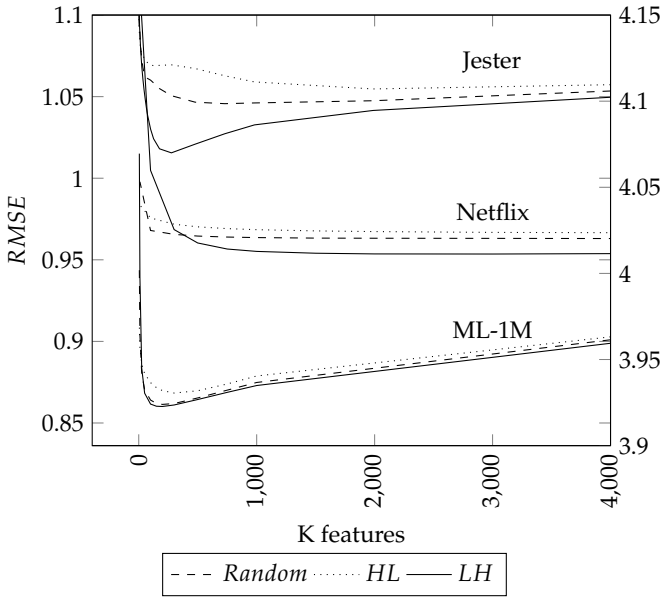


Fig. 1. Impact on RMSE of different sorting schemes. We use 100 neighbors for ML-1M, 200 for Netflix and 50 for Jester. The scale on the left y axis is for the ML-1M and Netflix datasets. The scale on the right y axis is for the Jester dataset.

As for the sorting schemes, all FFs show a similar behavior on all the datasets. In addition, all show a similar performance, with only very small differences in RMSE. The difference is slightly more evident in the ML-1M dataset, although still lower than 1% between the best (μ_{erfc}) and worst (μ_{One}) FFs. This small difference is, in fact, coherent with other results found in the literature [7].

It is also interesting to show a comparison between the results achieved by our proposed FFP similarity and the baselines described in Section 4.1. Figure 3 shows the results obtained, while varying the number of neighbors while computing rating predictions. We note that the number of neighbors is not applicable to the MBR similarity metric (see Section 4.1). Thus, results for MBR are shown as an horizontal line across the plot.

The Figure shows that results are clearly comparable, independently of the number of neighbors used. Furthermore, the lower RMSE was yielded, in general, by the FFP similarity metrics. We also note that FFP metrics seem to be more resilient to variations in the number of neighbors used, with results remaining almost constant as this number increases.

Computational Efficiency To measure computational efficiency, we count the number of iterations required to make a single rating prediction. Since computing the actual prediction, using Eq. (3) is independent of the similarity metric

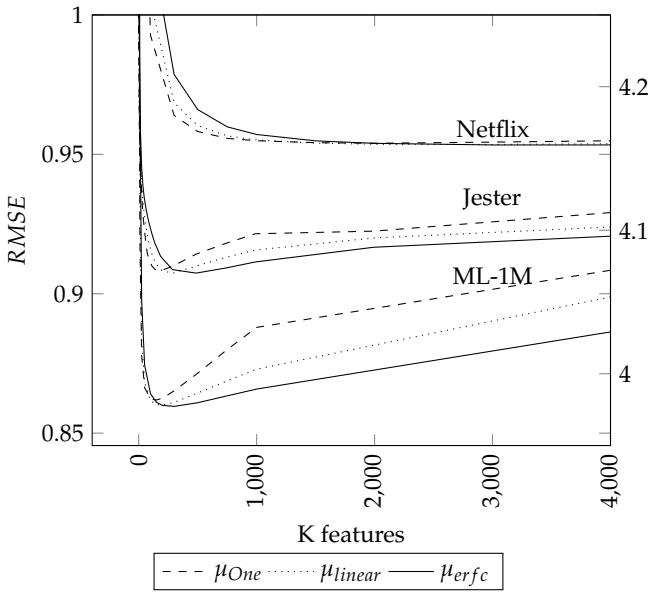


Fig. 2. Impact on RMSE of differentFuzzyfying Functions. We use 100 neighbors for ML-1M, 200 for Netflix and 50 for Jester. The scale on the left y axis is for the ML-1M and Netflix datasets. The scale on the right y axis is for the Jester dataset.

used (i.e. in all cases, the same items will be compared to the item whose rating is being predicted) we are only interested in the iterations required to compute the similarity between any two items.

For the purpose of this work, when computing the similarity between any pair of items i and j, we define an iteration as: (1) a comparison between a value in the FFP of item i and a value in the FFP of item j, as in Eq. (2); (2) a multiplication of a rating of item i by a rating of item j, as required for the Pearson correlation or Cosine similarity; or (3) a subtraction of a rating of item i from a rating of item j, as in Eq. (7). In practice, for the baselines, this will be the number of ratings in common between the two items being compared. For the FFP, this will be the highest value between k and the number of ratings in common between the two items. We expect the gain in our proposal to come from the fact that k will be lower.

Figure 4 shows a plot of the average number of iterations performed per similarity computed, on each dataset. The number of operations for the FFP metric is shown as a function of k. A vertical line is drawn where the best results were achieved. It should be noted that MBR only requires one iteration, since it compares items by simply computing the difference between their ratings average.

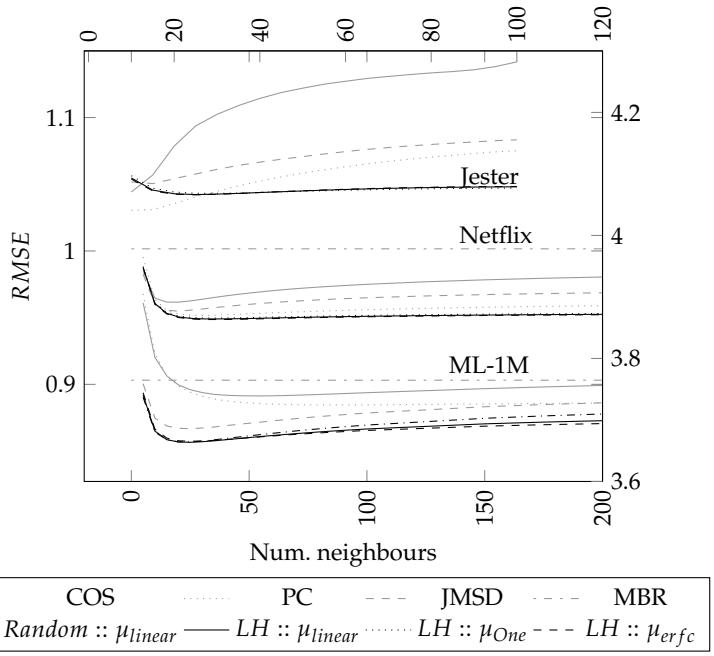
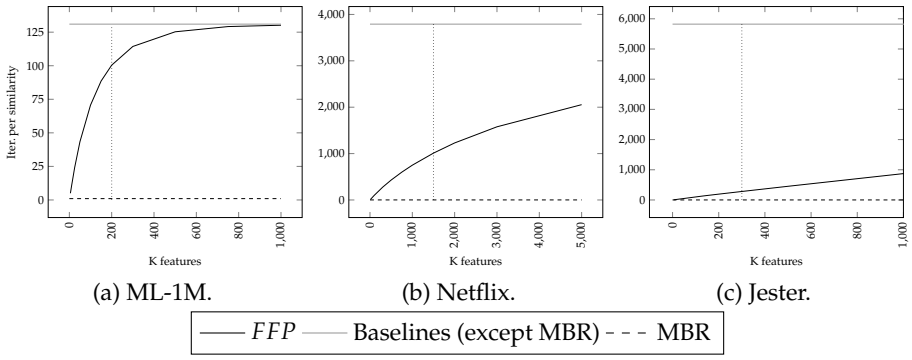


Fig. 3. Comparison of FFPs with the baseline similarity metrics. Baseline metrics are represented in light grey, while the FFP metrics are represented in black. The scales on the left y axis and bottom x axis are for the ML-1M and Netflix datasets. The scales on the right y axis and top y axis are for the Jester dataset.



(a) ML-1M. (b) Netflix. (c) Jester.

Fig. 4. Average number of iterations performed per similarity computed, on the three experimental datasets. Vertical lines show the value of k for which the best results were achieved.

In Figure 4a, we observe that, for the ML-1M dataset, the FFP similarity requires on average 100 iterations when $k = 200$. This corresponds to a reduction of about 23% per similarity, since the baselines require 130 iterations. In the Netflix dataset (Figure 4b), the gain is even more evident, with the FFP metric

Table 2. Best results for FFPs and baselines, on ML-1M. Column N contains the number of nearest-neighbour items used to compute the predicted rating.

Sim.	ML-1M					Netflix					Jester				
	SS	μ	k	N	$RMSE$	SS	μ	k	N	$RMSE$	SS	μ	k	N	$RMSE$
FFP	LH	μ_{linear}	200	20	**0.8565**	LH	μ_{One}	1500	35	0.9497	LH	μ_{linear}	200	25	4.0664
	LH	μ_{erfc}	200	25	0.8577	LH	μ_{One}	1500	35	0.9497	LH	μ_{erfc}	300	25	4.0660
	Rand	μ_{linear}	200	20	0.8568	LH	μ_{erfc}	3000	35	**0.9486**	LH	μ_{One}	200	25	4.0685
COS	-	-	-	50	0.8914	-	-	-	15	0.9616	-	-	-	15	4.0983
PC	-	-	-	75	0.8847	-	-	-	30	0.9517	-	-	-	15	**4.0419**
JMSD	-	-	-	20	0.8670	-	-	-	20	0.9549	-	-	-	15	4.0842
MBR	-	-	-	-	0.9031	-	-	-	-	1.0016	-	-	-	-	4.4063

requiring about 1009 iterations, when $k = 1500$, whereas the baselines use 3791 iterations — a reduction of 73%. Finally, for the Jester dataset (Figure 4c), the FFP similarity requires, on average, 281 iterations when $k = 300$, compared to the baseline, requiring 5822 iterations. The gain is, therefore, of 95%.

In conclusion, FFP has shown gains in all cases. This is, of course, dependent on the data. However, it is natural to expect that, the bigger the dataset, the most likely it is that items have a high number of ratings in common and, thus, the more gains can be achieved by our proposal.

Summary of Results To summarize our experiments, we now present the best results achieved by each tested similarity metric. Results for the ML-1M, Netflix, and Jester datasets are shown in Table 2. The lowest values for RMSE are highlighted using bold.

The best results for the ML-1M dataset were obtained with FFPs, which outperforms all four baselines. This was achieved using at most 200 ratings to describe the items and 20 neighbors to compute rating predictions. JMSD, the best performing baseline, uses the same number of neighbors, as the best FFP similarity, but still requires using all available ratings to compute the similarities.

Similarly, on the Netflix dataset, the best results were also achieved by the FFPs. However, the lowest value in RMSE was obtained using the μ_{erfc} Fuzzyfying Function and $k = 3000$.

On the Jester dataset, the best results were obtained using Pearson Correlation. Nevertheless, the results for our proposal are still highly relevant, for several reasons. First, Jester is a somewhat unusual dataset, with a highly number of ratings per item (see Table 1) thus, we could expect the similarity metrics to behave differently. Second, the difference in RMSE to the best FFP similarity is small (0.02). Finally, as shown in Figure 4c, the gain in efficiency obtained by the FFP is clearly significant, since on average we need 95% less iterations to compute a similarity than PC.

In conclusion, the use of FFPs allows the reduction of the similarity computational complexity, while improving, or at least maintaining, the quality of recommendations, in comparison with the baselines COS, PC, JMSD, and MBR.

The improvements become more noticeable in larger datasets, which translates to a better solution in real world RSs, where we can expect very sparse data and a higher number of users and items.

5 Conclusion

In this work, we have applied the concept of Fuzzy Fingerprints to item-based Collaborative Filtering. FFPs are used to create a new concise item representation and an efficient and effective similarity metric. They have a smaller computational cost than traditional similarity metrics while requiring a low engineering effort to implement.

We have experimentally compared our proposal to two traditional similarity measures, Pearson Correlation and Cosine similarity, and two state of the art similarity metrics, Jaccard Mean Squared Difference and MBR. Results show that FFPs are a promising approach since they can be applied with success in recommendation tasks. In fact, using FFPs we were able to obtain a reduction of the number of operations needed per similarity computation between 23% and 95%, depending on the density of the rating matrix. This was achieved with an overall improvement in $RMSE$.

Future work will be conducted with the goal of exploring further configuration options for the FFPs, such as new sorting schemes and Fuzzyfying Functions. Also, we will study the application of FFPs to content-based RS.

Acknowledgments. This work was supported by national funds through Fundação para a Ciência e a Tecnologia (FCT) with reference UID/CEC/50021/2013, by project GoLocal (ref. CMUPERI/TIC/0046/2014) and co-financed by the University of Lisbon and INESC-ID.

References

1. Adomavicius, G., Tuzhilin, A.: Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. Knowledge and Data Engineering, IEEE Transactions on **17**(6), 734–749 (2005). DOI 10.1109/TKDE.2005.99
2. Bobadilla, J., Ortega, F., Hernando, A., Gutiérrez, A.: Recommender systems survey. Knowledge-Based Systems **46**(0), 109 – 132 (2013). DOI http://dx.doi.org/10.1016/j.knosys.2013.03.012. URL http://www.sciencedirect.com/science/article/pii/S0950705113001044
3. Bobadilla, J., Ortega, F., Hernando, A., de Rivera, G.G.: A similarity metric designed to speed up, using hardware, the recommender systems k-nearest neighbors algorithm. Knowledge-Based Systems **51**(0), 27 – 34 (2013). DOI http://dx.doi.org/10.1016/j.knosys.2013.06.010. URL http://www.sciencedirect.com/science/article/pii/S095070511300186X
4. Bobadilla, J., Serradilla, F., Bernal, J.: A new collaborative filtering metric that improves the behavior of recommender systems. Knowledge-Based Systems **23**(6), 520 – 528 (2010). DOI http://dx.doi.org/10.1016/j.knosys.2010.03.009. URL http://www.sciencedirect.com/science/article/pii/S0950705110000444

5. Chen, S., Luo, T., Liu, W., Xu, Y.: Incorporating similarity and trust for collaborative filtering. In: Fuzzy Systems and Knowledge Discovery, 2009. FSKD '09. Sixth International Conference on, vol. 2, pp. 487–493 (2009). DOI 10.1109/FSKD.2009.720
6. Dimiev, V.: Fuzzifying functions. Fuzzy Sets and Systems **33**(1), 47 – 58 (1989). DOI http://dx.doi.org/10.1016/0165-0114(89)90216-9. URL http://www.sciencedirect.com/science/article/pii/0165011489902169
7. Homem, N., Carvalho, J.P.: Authorship identification and author fuzzy 'fingerprints". In: Fuzzy Information Processing Society (NAFIPS), 2011 Annual Meeting of the North American, pp. 1–6 (2011). DOI 10.1109/NAFIPS.2011.5751998
8. Koenigstein, N., Koren, Y.: Towards scalable and accurate item-oriented recommendations. In: Proceedings of the 7th ACM Conference on Recommender Systems, RecSys '13, pp. 419–422. ACM, New York, NY, USA (2013). DOI 10.1145/2507157.2507208. URL http://doi.acm.org/10.1145/2507157.2507208
9. Liu, H., Hu, Z., Mian, A., Tian, H., Zhu, X.: A new user similarity model to improve the accuracy of collaborative filtering. Knowledge-Based Systems **56**, 156 – 166 (2014). DOI http://dx.doi.org/10.1016/j.knosys.2013.11.006. URL http://www.sciencedirect.com/science/article/pii/S0950705113003560
10. Pereira, R., Lopes, H., Breitman, K., Mundim, V., Peixoto, W.: Cloud based real-time collaborative filtering for item–item recommendations. Computers in Industry **65**(2), 279 – 290 (2014). DOI http://dx.doi.org/10.1016/j.compind.2013.11.005. URL http://www.sciencedirect.com/science/article/pii/S0166361513002352
11. Said, A., Bellogín, A.: Rival: a toolkit to foster reproducibility in recommender system evaluation. In: RecSys '14 Proceedings of the 8th ACM Conference on Recommender Systems, pp. 371–372. ACM Press (2014). DOI 10.1145/2645710.2645712. URL http://dl.acm.org/citation.cfm?doid=2645710.2645712
12. Son, L.H.: Hu-fcf: A hybrid user-based fuzzy collaborative filtering method in recommender systems. Expert Syst. Appl. **41**(15), 6861–6870 (2014). DOI 10.1016/j.eswa.2014.05.001. URL http://dx.doi.org/10.1016/j.eswa.2014.05.001
13. Tsai, C.F., Hung, C.: Cluster ensembles in collaborative filtering recommendation. Applied Soft Computing **12**(4), 1417 – 1425 (2012). DOI http://dx.doi.org/10.1016/j.asoc.2011.11.016. URL http://www.sciencedirect.com/science/article/pii/S1568494611004583
14. Vijayakumar, V., Neelanarayanan, V., Bagchi, S.: Big data, cloud and computing challenges performance and quality assessment of similarity measures in collaborative filtering using mahout. Procedia Computer Science **50**, 229 – 234 (2015). DOI http://dx.doi.org/10.1016/j.procs.2015.04.055. URL http://www.sciencedirect.com/science/article/pii/S1877050915005566
15. Xu, R., Wang, S., Zheng, X., Chen, Y.: Distributed collaborative filtering with singular ratings for large scale recommendation. Journal of Systems and Software **95**, 231 – 241 (2014). DOI http://dx.doi.org/10.1016/j.jss.2014.04.045. URL http://www.sciencedirect.com/science/article/pii/S0164121214001150
16. Ye, T., Bickson, D., Ampazis, N., Benczur, A.: Lsrs'15: Workshop on large-scale recommender systems. In: Proceedings of the 9th ACM Conference on Recommender Systems, RecSys '15, pp. 349–350. ACM, New York, NY, USA (2015). DOI 10.1145/2792838.2798715. URL http://doi.acm.org/10.1145/2792838.2798715
17. Yera, R., Castro, J., Martínez, L.: A fuzzy model for managing natural noise in recommender systems. Applied Soft Computing **40**, 187 – 198 (2016). DOI 10.1016/j.asoc.2015.10.060. URL http://www.sciencedirect.com/science/article/pii/S1568494615007048
18. Zheng, M., Min, F., Zhang, H.R., Chen, W.B.: Fast recommendations with the m-distance. IEEE Access **4**, 1464–1468 (2016). DOI 10.1109/ACCESS.2016.2549182

Appendix B

Combining Ratings and Item Descriptions in Recommendation Systems using Fuzzy Fingerprints

Combining Ratings and Item Descriptions in Recommendation Systems using Fuzzy Fingerprints

André Carvalho
INESC-ID,
Instituto Superior Técnico,
Universidade de Lisboa
andre.silva.carvalho@tecnico.ulisboa.pt

Pável Calado
INESC-ID,
Instituto Superior Técnico,
Universidade de Lisboa
pavel.calado@tecnico.ulisboa.pt

João Paulo Carvalho
INESC-ID,
Instituto Superior Técnico,
Universidade de Lisboa
joao.carvalho@inesc-id.pt

Abstract— **Memory-based Collaborative filtering solutions are dominant in the Recommendation Systems domain, due to their low implementation effort and service maintenance, when compared to Model-based approaches. Memory-based systems often rely on similarity metrics to compute similarities between items (or users) using ratings, in what is often named neighbor-based Collaborative filtering. This paper applies Fuzzy Fingerprints to create a novel similarity metric. In it, the Fuzzy Fingerprint of each item is described with a ranking of users ratings, combined with words obtained from the items' description. This allows the presented similarity metric to use fewer neighbors than other well-known metrics such as Cosine similarity or Pearson Correlation. Our proposal is able to reduce RMSE by at least** 0.03 **and improve NDCG@10 by at least** 0.023 **when compared with the best baseline here presented.**

I. Introduction

Users of the digital world are overloaded with information [20]. Recommender Systems (RSs) allow us to cope with this, by cataloging a vast list of items, that later can be recommended. These recommendations can be determined by a large number of techniques, as highlighted by the scientific community [1, 5]. Due to their success, RSs can be found in a number of services, providing recommendations for movies, music, news, products, event, and services, among others [1].

RSs allow the discovery of new items that might please the user, among the diversity of items available. Some websites, such as Amazon, Ebay, Netflix, and LinkedIn are current examples of services that benefit from providing recommendations to their clients [1, 12, 19]. As a result, much research has been done in the area, contributing to the development of new techniques to improve recommendations, thus helping to increase the RS providers income.

Despite these efforts, there are still challenges to be addressed. For example, turning state-of-art solutions into real-world scenarios is still difficult, mainly because of the large amount of data and scalability issues that ensue [13]. For this reason, more traditional approaches, such as item-based Collaborative Filtering (CF) are still the most widely used [8, 10, 16]. Despite their simplicity, item-based CF can provide quite accurate results, thus yielding an advantageous trade-off between engineering effort and user satisfaction.

Our goal is to present a similarity metric using Fuzzy-Fingerprints (FFPs) [11], which combines item ratings and words information obtained from the item descriptions to produce a better similarity metric. More specifically, we propose to represent items by their ratings, combined with a textual description, which can then be directly used to determine similarities between them. Previously, FFPs have been used to classify text documents, using the words' relative weight to provide more useful information than their actual importance. Our work combines such approach with CF techniques. The main contribution of this work is, therefore, the application of FFP similarity metric for the RS domain, by combining ratings and items descriptions within the same FFP.

Results show that our FFP similarity metric is a promising solution to be applied for recommendations, achieving improvements of up to 0.073 in RMSE and in NDCG@10 of up to 0.048.

The remainder of this paper is organized as follows. Section II contains the literature review in Fuzzy systems and RSs, more specifically on the use of similarity metrics for CF. Section III presents how Fuzzy Fingerprints can be applied to RSs and how text can be combined with ratings. Section IV presents the experimental evaluation of our proposal. Finally, in Section V we draw some conclusions from our results and propose directions for future work.

II. Related Work

Similarity metrics between items (or users) are a central part of RS research [12]. Traditionally, similarity is measured using metrics such as Pearson Correlation (PC) or the Cosine similarity (COS) [5]. Nevertheless, many

other ways of measuring similarity have been proposed, ranging from simple variations of PC and COS, through the design of more complex functions [5, 7].

An example is the work of [7], where a novel similarity metric is proposed, with the goal of exploiting social information to improve the results produced by traditional CF. In their work, user ratings are combined with a measure of *trust* between users, which is inferred from social information. The combination is performed through a variation of PC and is able to improve the rating predictions.

In [6], the authors argue that using PC alone is not enough to capture relevant statistical features of the data. To correct this, they propose a combination of the mean squared difference between the user's ratings with the Jaccard coefficient, thus capturing the similarity between users, while taking into account the number of items they have ranked in common. Through experiments, the authors demonstrate that their metric allows the RS to use a lower number of neighbors to determine rating predictions than traditional metrics, thus reducing rating prediction complexity.

Fuzzy systems have also been previously applied to RS [2, 17]. An example is the work of Son [18], where fuzzy sets were used to combine CF with demographic information. The author also proposes a formal definition for a *Fuzzy Recommender System*.

In [14], fuzzy principles are combined with probabilistic inference. To do so, a fuzzy representation of item ratings is used to create a probabilistic distribution. The fuzzy representation enables to represent ambiguity and vagueness of ratings. A Bayesian network combines these representations to obtain a relationship between users, allowing an improvement of the system accuracy.

In a recent work [21], fuzzy tools are used to improve recommendation accuracy, by managing what the authors call *natural noise*. Natural noise results from the users' own errors during item evaluation. Their work is compared to other noise reduction methods and is shown to have a superior performance..

There is some similarity between [21] and our work. However, our aim is to filter unreliable ratings, whereas in [21] ratings are modified to reduce noise.

The above works, as many others show that improving the similarity metrics has a beneficial impact on the overall RS results. In this work, we introduce a similarity metric based on FFPs, adapted for RSs, that aims to improve recommendations and allows the combination of different types of information.

III. Proposal

Fingerprints map an arbitrarily large object into a smaller and more compact representation [3, 11]. In our work, items in a Recommender System are represented by their Fuzzy-Fingerprint (FFP), which is then used to compute item similarities. Before obtaining the FFPs, Fingerprints must be computed. A Fingerprint, ϕ_i, of an item i is generated by ranking a set of features assigned to it. Once ϕ_i is determined, a FFP is computed, through the application of a Fuzzyfing Function (FF) [9]. This function transforms sets of ratings into fuzzy sets, which form the FFP of item i, designated as Φ_i. In the following, we explain this process in detail.

A. Creating Fingerprints

Neighbor-based CF computes similarities using the ratings given by users to different items. In this work, besides using the ratings, we add textual information taken from the item's descriptions. We argue that ratings and text can be combined into an FFPand improve the quality of the recommendations.

Representing Ratings

Let r_i be the set of ratings, for which N users provided a rating for item i defined as:

$$r_i = \{(u_1, r_{1i}), (u_w, r_{wi}), \cdots, (u_N, r_{Ni})\} \tag{1}$$

where each u_j is a user identifier and r_{ji} is the rating assigned by user u_j to item i.

In general, users provide explicit feedback by giving a rating, i.e. by evaluating the item using a given scale. Since ratings are usually based on a discrete limited scale (e.g. 1 to 5 stars), items often get the same rating from many users. However, the meaning of a particular value may be different for different users. Thus, it is important to normalize the ratings. To achieve this, we simply subtract the average of all ratings given by the user.

To illustrate this process, Fig. 1 shows an example for an item i with ratings r_i, the average rating of each user is $\bar{\mathbf{u}}_\mathbf{j}$, and line $\mathbf{r_{ji}} - \bar{\mathbf{u}}_\mathbf{j}$ show the final normalized value.

$\mathbf{u_j}$	a	b	c	d	e
$\mathbf{r_{ji}}$	5	2	-	5	4
$\bar{\mathbf{u}}_\mathbf{j}$	3.5	3.4	3.4	3.6	3.5
$\mathbf{r_{ji}} - \bar{\mathbf{u}}_\mathbf{j}$	1.5	-1.4	-	1.4	0.5

Figure 1: Ratings (r_i) of item i, average rating of each user $\bar{\mathbf{u}}_\mathbf{j}$ and the rating of a user in reference to the average rating $\mathbf{r_{ji}} - \bar{\mathbf{u}}_\mathbf{j}$.

Representing Item Descriptions

We now need to apply a similar process to the item descriptions. An item description can be any piece of textual information associated with an item (e.g. a movie synopsis).

We start by applying stemming and stop-word removal, followed by computing the respective TF-IDF of each word. Each item will, thus be represented by a vector of TF-IDF weights, as illustrated in Fig. 2.

word	w_1	w_5	w_{30}	w_{72}
TF-IDF$_i$	0.2	0.052	0.024	0.046

Figure 2: TF-IDF representation of item i. Each w_j corresponds to a word present in the item's description.

Building the Fingerprint

Finally, we combine the ratings and item information. To do so, we first need to transform the values so that both types of information are on the same scale. We achieve this using Eq. 2, where x_{new} is the resulting scaled value, x is the value being normalized, min the lowest value max the highest value. This normalization is applied to each item representation separately.

$$x_{new} = \frac{x - min}{max - min} \tag{2}$$

Figure 3 shows the normalized vector obtained for the ratings from Fig. 1, considering $min = -2.0$ and $max = 2.5$. Figure 4 shows the normalized vector obtained for the text from Fig. 2, considering $min = 0.001$ and $max = 0.2$.

$\mathbf{u_j}$	a	b	c	d	e
Norm. $\mathbf{r_{ji}} - \bar{\mathbf{u}}_\mathbf{j}$	0.777	0.133	-	0.756	0.556

Figure 3: Normalized ratings for item i.

word	w_1	w_5	w_{30}	w_{72}
Norm. TF-IDF$_i$	1	0.2563	0.1156	0.2261

Figure 4: Normalized TF-IDF for item i.

Once the ratings and textual information are normalized, a Fingerprint can be generated by simply joining both representations, ordering them, and keeping only the k features with the highest values. The value for k can be configured and optimized. Figure 5 shows the resulting Fingerprint for item i, with k equal to 5.

feature	w_1	a	d	e	w_{72}
ϕ_i	1	0.777	0.756	0.556	0.2261

Figure 5: Fingerprint ϕ_i, using k equal to 5.

B. Fuzzifying a Fingerprint

The Fingerprint ϕ_i, shown in the previous Section, is an *ordered set* of features. This order reflects the importance of each rating and word in representing the items. It is by leveraging this importance that we determine the Fuzzy-Fingerprint of item i, Φ_i.

A Fuzzyfing Function assigns a weight to each feature in ϕ_i. There are many alternatives to define a FF [9]. Here, we present one suggestion, shown in Eq. 3, where pos_{f_j} is the position of feature f_j within ϕ_i.

$$\mu(\text{pos}_{f_j}) = \frac{k - \text{pos}_{f_j}}{k} \tag{3}$$

Using Eq. 3, the weight of a feature decreases linearly, according to its position. Preliminary experiments have indicated that using other functions does not influence significantly the quality of the results for this problem. For this reason, we do not present the alternatives in this paper.

Using the above FF, we can now define the FFP Φ_i as:

$$\Phi_i = \{(f_j, \mu(\text{pos}_{f_j})), \forall f_j \in \phi_i\} \tag{4}$$

The FFP is, therefore, the set of features in the Fingerprint and its associated weight, given by the FF. It is, in effect, a *fuzzy set* of features for item i.

An example is shown in Fig. 6, resulting from fuzzifying the Fingerprint in Fig. 5.

f_j	w_1	a	d	e	w_{72}
Φ_i	0.8	0.6	0.4	0.2	0

Figure 6: FFP for item i, with k equal to 5.

Note that the FFP does not contain the item ratings nor the actual TF-IDF weights of the words. Instead, it contains the value corresponding to the position in the FF. In [11], when classifying textual documents, the authors argue that the relative weights of words, resulting from a FF, provides more useful information than their actual value. We apply the same idea in this work.

C. Comparing Fuzzy Fingerprints

Once the FFPs are determined, it is possible to compute similarities between items. Consider Φ_i and Φ_g the FFPs of items i and g, respectively. Let F_i be the set of features in Φ_i and F_g be the set of features in Φ_g. The FFP similarity between items i and g is defined as:

$$\text{sim}_{FFP}(\Phi_i, \Phi_g) = \sum_{u_v \in F_i \cap F_g} \frac{\min(\Phi_i(f_v), \Phi_g(f_v))}{k} \tag{5}$$

where $\Phi_x(v)$ denotes the the value associated with feature f_v in Φ_x.

Equation (5) defines the similarity between two FFPs [11]. This similarity uses the sum of the lowest values from each FFP depending on the features in common between the two items i and g. This corresponds to a minimum *t-norm*, where we intersect the two fuzzy sets. The idea is that, for two items to be similar, they must have been rated by the same users and the ratings given by those users must have the same relative importance, or the words in their descriptions also have a similar importance.

f_j	d	w_1	s	y	w_{30}
Φ_g	0.8	0.6	0.4	0.2	0

Figure 7: FFP for item g, with k equal to 5.

We illustrate this using the previously determined FFP for item i (Fig. 6) and the FFP of a new item g, shown in Fig. 7. The computation is shown in Eq. (6).

$$\begin{aligned}\text{sim}_{FFP}(\Phi_i, \Phi_g) &= \frac{\min(\Phi_i(d), \Phi_g(d))}{k} \\ &+ \frac{\min(\Phi_i(w_1), \Phi_g(w_1))}{k} \\ &= \frac{\min(0.4, 0.8)}{5} + \frac{\min(0.8, 0.6)}{5} = 0.2 \end{aligned} \quad (6)$$

D. Rating Prediction and Recommendation

In this work, predictions are obtained using the conventional neighbor-based Collaborative Filtering. More specifically, let $\widehat{r}_{vi}$ be the *predicted* rating that a given user u_v would assign to item i. We start by computing the *neighborhood* of item i, i.e. the set of n items in the database that are more similar to i, $N_i(v)$. The value of $\widehat{r}_{vi}$ is defined as:

$$\widehat{r}_{vi} = \bar{r}_i + \gamma_{vi} \sum_{g \in N_i(v)} \text{sim}(\Phi_i, \Phi_g) \times (r_{vg} - \bar{r}_g) \quad (7)$$

where r_{vg} is the rating assigned by user u to item g, $\bar{r}_x$ is the average of all ratings assigned to item x and γ_{ui} is defined as:

$$\gamma_{vi} = \frac{1}{\sum_{g \in N_i(v)} sim(\Phi_i, \Phi_g)} \quad (8)$$

To generate recommendations for a user, rating predictions are computed for all items not evaluated by the user. Those predictions are then sorted and the highest rated are selected and recommended.

IV. Evaluation

This Section contains the experimental procedure used to validate our work. Our aim is to determine similarities between items by combining ratings and content-based information. These are embedded in a Fuzzy-Fingerprint and the similarity between FFPs is used to compare the items.

To assert the effectiveness of our approach, we have conducted experiments with three baseline similarity metrics: Pearson Correlation (PC), Cosine similarity (COS) and a state-of-the-art similarity metric named *Jaccard-Mean Squared Difference (JMSD)* [6].

The Pearson Correlation coefficient has been widely used in RSs, since it is simple to implement, intuitive and provides fair quality results [6]. PC is defined in Eq. 9, where U is the set users that rated both items i and g.

$$sim_{PC}(i,g) = \frac{\sum_{u \in U}(r_{u,i} - \bar{r}_i) \times (r_{u,g} - \bar{r}_g)}{\sqrt{\sum_{u \in U}(r_{u,i} - \bar{r}_i)^2} \times \sqrt{\sum_{u \in U}(r_{u,g} - \bar{r}_g)^2}} \quad (9)$$

Another often used similarity measure is the Cosine similarity (COS), as defined in Equation (10).

$$sim_{COS}(i,g) = \frac{\sum_{u \in U} r_{u,i} \times r_{u,g}}{\sqrt{\sum_{u \in U} r_{u,i}^2} \times \sqrt{\sum_{u \in U} r_{u,g}^2}} \quad (10)$$

To further demonstrate the generality of our solution, the similarity metric proposed in [6], designated as the JMSD is included. It combines the Jaccard coefficient, to capture the number of ratings in common, with the mean squared difference (MSD) of those ratings. It offers a good rating prediction accuracy, using a lower number of neighbors than PC and COS. JMSD is defined in Eq. 11:

$$sim_{JMSD}(i,g) = Jaccard(i,g) \times \big(1 - MSD(i,g)\big) \quad (11)$$

where $Jaccard$ and MSD are defined as:

$$Jaccard(i,g) = \frac{|U_i \cap U_g|}{|U_i \cup U_g|} \quad (12)$$

$$MSD(i,g) = \frac{\sum_{u \in U}(r_{u,i} - r_{u,g})^2}{|U|} \quad (13)$$

where U_s is the set of items ranked by user s.

The evaluation was conducted using the Hetrec2011-movielens dataset[1]. This dataset, from the movie domain, has 86 000 ratings, from 2113 users to 10 197 movies. Content-based information was extracted from the movies synopsis.

A. Evaluation Metrics

In this Section, we are interested in determining how effective is our proposed similarity and how many features are needed to build the FFP. To do so, we use the values of Root Mean Square Error (RMSE) and Normalized Discounted Cumulative Gain (NDCG) [5]. RMSE evaluates the quality of rating predictions. NDCG evaluates the recommendations considering not only the elements being recommended, but also their ranking within the recommendation. For NDCG we consider the 10 highest (predicted) rated items to recommend to users. An item is considered relevant only if it has the highest possible rating, which in this dataset is the rating of 5 stars.

No users or items were discarded, if even they contained only very few ratings. Although this might lead to what is called the *cold start problem*, we were interested in observing the performance of our approach under such conditions. All experiments were conducted using RIVAL [15], a framework to evaluate Recommender Systems. All measurements result from a 5 fold cross-validation experiment, where the ratings are split on a user basis. All predictions are made using only the items in the test set [4].

[1]The dataset can be obtained at http://grouplens.org/

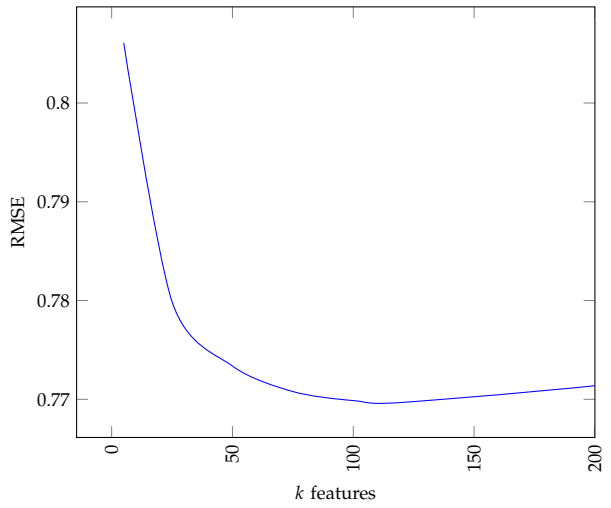


Figure 8: Value of RMSE for varying sizes of the FFP. Predicitons are made using 50 neighbors.

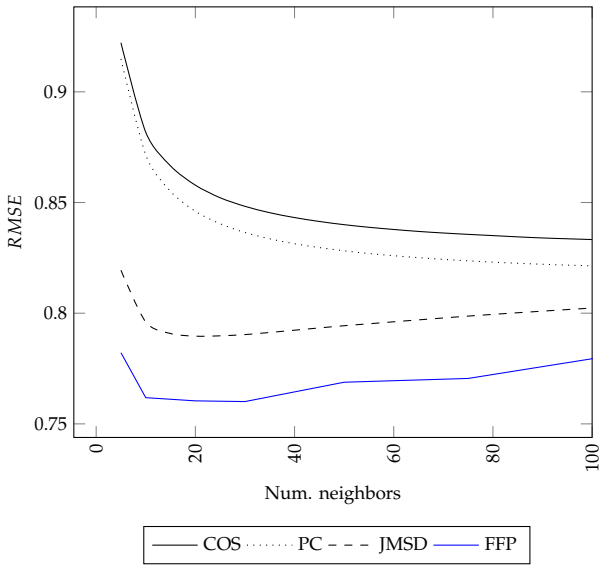


Figure 9: Value of RMSE for varying number of neighbors, comparing the different baselines with the FFP similarity.

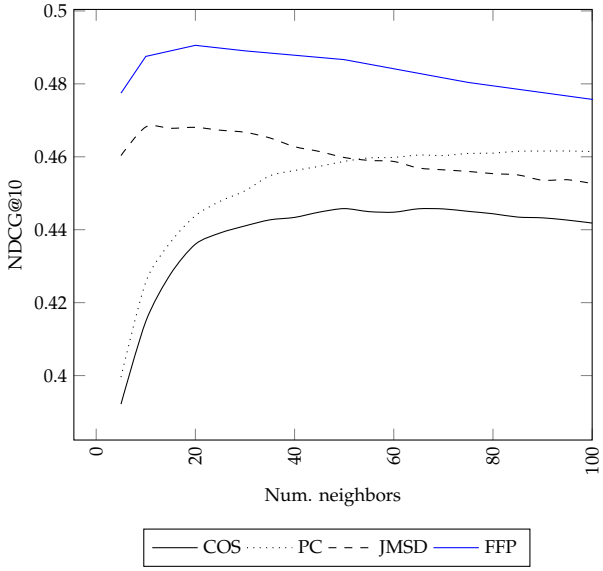


Figure 10: Value of NDCG@10 for varying number of neighbors, comparing the different baselines with the FFP similarity.

B. Results

Our experiments start by analyzing the impact of the FFP size, i.e. the number k of features used to represent items. Results are shown in Fig. 8.

We can see that there is an optimal value for k, achieved when k is equal to 125, in this dataset. Figure 8 also shows that FFPs with more than 125 features start to introduce noise and, thus, lead to an slight increase in RMSE.

Using this ideal number of features, Fig. 9 and Fig. 10 show a comparison of our solution to the baselines, in terms of RMSE and NDCG, respectively.

We can see that the FFP similarity is able to improve both rating predictions and recommendation quality. Moreover, according to the obtained NDCG, not only more relevant items are being recommended to users, those most relevant are being recommended first, when compared to the baselines. This is achieved using fewer neighbors than all the baseline similarity metrics. As result, our proposal not only can increase the quality of the results, but also improve recommendation performance.

Similarity	K	N	RMSE	NDCG@10
FFP	125	**20**	**0.760**	**0.490**
COS	-	100	0.833	0.442
PC	-	100	0.821	0.461
JMSD	-	25	0.790	0.467

Table I: Summary table with FFP similarity and baselines.

To summarize our evaluation, Table I shows the best results achieved by each similarity metric. The values shown confirm our observations. More specifically, the FFP similarity requires five neighbors less, achieves an RMSE that is 0.03 lower and an NDCG that is 0.023 higher than the best baseline here presented, JMSD.

We can therefore conclude that using FFPs is an effective way to combine different types of information to improve the quality of RSs.

V. Conclusion

In this work, we apply the concept of Fuzzy Fingerprints to RSs, creating a new item representation and similarity metric. Fuzzy Fingerprints use not only user ratings, as common similarity metrics do, but also allow the introduction of other types of information. In this

work, we experiment with textual information, extracted from movie synopsis.

A set of experiments on a standard dataset shows that FFPs are a promising approach. When compared with two traditional and one state-of-the-art baseline, we achieved improvements of 0.03 in RMSE and 0.023 in NDCG, when compared to the best results achieved by the baselines.

Future work will be conducted with the introduction of more information sources, such as social network data and user demographics.

Acknowledgments

This work was supported by national funds through Fundação para a Ciência e a Tecnologia (FCT) with reference UID/CEC/50021/2013, by project GoLocal (ref. CMUPERI/TIC/0046/2014) and co-financed by the University of Lisbon and INESC-ID.

References

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

[2] M. Y. H. Al-Shamri and K. K. Bharadwaj. Fuzzy-genetic approach to recommender systems based on a novel hybrid user model. *Expert Sys. with Appl.*, 35(3):1386 – 1399, 2008.

[3] F. Batista and J. P. Carvalho. Text based classification of companies in crunchbase. In *Fuzzy Systems (FUZZ-IEEE), 2015 IEEE Inter. Conf. on*, pages 1–7, Aug 2015.

[4] A. Bellogin, P. Castells, and I. Cantador. Precision-oriented evaluation of recommender systems: An algorithmic comparison. In *Proc. of the 5th ACM Conf. on Recommender Systems*, RecSys '11, pages 333–336, New York, NY, USA, 2011. ACM.

[5] J. Bobadilla, F. Ortega, A. Hernando, and A. Gutiérrez. Recommender systems survey. *Knowl-Based Syst*, 46(0):109 – 132, 2013.

[6] J. Bobadilla, F. Serradilla, and J. Bernal. A new collaborative filtering metric that improves the behavior of recommender systems. *Knowl-Based Syst*, 23(6):520 – 528, 2010.

[7] S. Chen, T. Luo, W. Liu, and Y. Xu. Incorporating similarity and trust for collaborative filtering. In *Fuzzy Systems and Knowledge Discovery, 2009. FSKD '09. Sixth Inter. Conf. on*, volume 2, pages 487–493, Aug 2009.

[8] C. Desrosiers and G. Karypis. *A Comprehensive Survey of Neighborhood-based Recommendation Methods*, pages 107–144. Springer US, Boston, MA, 2011.

[9] V. Dimiev. Fuzzifying functions. *Fuzzy Sets and Systems*, 33(1):47 – 58, 1989.

[10] A. Felfernig, M. Jeran, G. Ninaus, F. Reinfrank, and S. Reiterer. Toward the Next Generation of Recommender Systems: Appl. and Research Challenges. In G. A. Tsihrintzis, M. Virvou, and L. C. Jain, editors, *Multimedia Services in Intel. Environ.*, number 24 in Smart Innovation, Systems and Technologies, pages 81–98. Springer Inter. Publishing, 2013.

[11] N. Homem and J. P. Carvalho. Authorship identification and author fuzzy 'fingerprints". In *Fuzzy Information Processing Society (NAFIPS), 2011 Annual Meeting of the North American*, pages 1–6, March 2011.

[12] H. Liu, Z. Hu, A. Mian, H. Tian, and X. Zhu. A new user similarity model to improve the accuracy of collaborative filtering. *Knowl-Based Syst*, 56:156 – 166, 2014.

[13] H. Ma, T. C. Zhou, M. R. Lyu, and I. King. Improving Recommender Systems by Incorporating Social Contextual Information. *ACM Trans. on Information Systems*, 29(2):9:1–9:23, Apr. 2011.

[14] N. Marín, O. Pons, L. M. de Campos, J. M. Fernández-Luna, and J. F. Huete. Advances in intel. databases and information systems a collaborative recommender system based on probabilistic inference from fuzzy observations. *Fuzzy Sets and Systems*, 159(12):1554 – 1576, 2008.

[15] A. Said and A. Bellogín. Rival: a toolkit to foster reproducibility in recommender system evaluation. In *RecSys '14 Proc. of the 8th ACM Conf. on Recommender Systems*, pages 371–372. ACM Press, 2014.

[16] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Item-based collaborative filtering recommendation algorithms. In *Proc. of the 10th Inter. Conf. on World Wide Web*, WWW '01, pages 285–295, New York, NY, USA, 2001. ACM.

[17] Z. Sevarac, V. Devedzic, and J. Jovanovic. Adaptive neuro-fuzzy pedagogical recommender. *Expert Systems with Appl.*, 39(10):9797 – 9806, 2012.

[18] L. H. Son. Hu-fcf: A hybrid user-based fuzzy collaborative filtering method in recommender systems. *Expert Syst. Appl.*, 41(15):6861–6870, Nov. 2014.

[19] G. Song, S. Sun, and W. Fan. Applying user interest on item-based recommender system. In *Computational Sciences and Optimization (CSO) Fifth Inter. Joint Conf. on*, pages 635–638, 2012.

[20] C.-F. Tsai and C. Hung. Cluster ensembles in collaborative filtering recommendation. *Applied Soft Computing*, 12(4):1417 – 1425, 2012.

[21] R. Yera, J. Castro, and L. Martínez. A fuzzy model for managing natural noise in recommender systems. *Applied Soft Computing*, 40:187 – 198, 2016.

Appendix C

Tag-based User Fuzzy Fingerprints for Recommender Systems

Tag-based User Fuzzy Fingerprints for Recommender Systems

André Carvalho[1], Pável Calado[2], and Joao Paulo Carvalho[3]

INESC-ID, Instituto Superior Técnico, Universidade de Lisboa, Portugal
[1] andre.silva.carvalho@tecnico.ulisboa.pt,
[2] pavel.calado@tecnico.ulisboa.pt,
[3] joao.carvalho@inesc-id.pt

Abstract. Most Recommender Systems rely exclusively on ratings and are known as Memory-based Collaborative Filtering systems. This is currently dominant approach outside of academia due to the low implementation effort and service maintenance, when compared with more complex Model-based approaches, Traditional Memory-based systems have as their main goal to predict ratings, using similarity metrics to determine similarities between the users' (or items) rating patterns. In this work, we propose a user-based Collaborative Filtering approach based on tags that does not rely on rating prediction, instead leveraging on Fuzzy Fingerprints to create a novel similarity metric. Fuzzy Fingerprints provide a concise and compact representation of users allowing the reduction of the dimensionality usually associated with user-based collaborative filtering. The proposed recommendation strategy combined with the Fuzzy Fingerprint similarity metric is able to outperform our baselines, in the Movielens-1M dataset.

Keywords: Recommender System, Collaborative Filtering, Fuzzy Fingerprint, Tags

1 Introduction

Users of the digital world are overloaded with information [16]. Recommender Systems (RSs) allow us to cope with this, by cataloging a vast list of items, that later can be recommended. Due to their success, RSs can be found in a number of services, providing recommendations for movies, music, news, products, events, services, among others [1]. However, turning state of the art solutions into real-world scenarios is still challenging, mainly due to a large amount of available data and the ensuing scalability issues. For this reason, more traditional approaches, such as Collaborative Filtering (CF) are still the most widely used [18]. Despite its simplicity, CF can provide quite accurate results, thus yielding an advantageous trade-off between engineering effort and user satisfaction.

Memory-based Collaborative Filtering can usually be implemented using one of two different strategies: *user-based* CF, which compares users ratings to

determine a neighborhood of similar users; and *item-based* CF, which instead computes item similarities and forms item neighborhoods to produce the rating predictions.

Over the years, item-based CF has replaced user-based CF, given its better scalability properties [11]. Since the number of users grows over time, and generally at a faster rate than the items, so does the number of similarities, thus posing a scalability problem. Similarities between users also vary more over time than similarities between items, since individual users tend to change their preferences, while the global opinion on a given item tends to remain stable.

In this work, we argue that an effective and efficient user-based CF system can be implemented. To this effect, we use Fuzzy Fingerprints (FFPs) to represent users based on item *tags* and ratings. *Tags*, i.e. short textual labels attached by the users to the items, provide an item description or categorization and are a common resource in current online RSs. They allow us to create a more detailed user representation than traditional CF, in a controlled manner, i.e. by controlling the number of tags used in the FFPs, we can easily fine-tune our system to improve recommendation quality or to speed up the similarity computation. In this work, we mainly focus on obtaining an improved recommendation quality.

Our main contributions are, therefore, (1) a new way to determine relevant items to recommend to users without requiring the computation of rating predictions for user-based CFs, and (2) a novel similarity metric for RSs, using the concept of FFPs cite5751998 to represent users based on tags from rated items. More specifically, we propose to represent users by their low-dimensional Fingerprints, which can then be directly used to determine similarities between them. A similar idea has been previously applied to text authorship identification [9] with success. Our goal is to apply the same principle to RSs using tags from the items rated by each user, to obtain better recommendations. This solution has three major advantages: (1) provides overall better recommendations to users; (2) requires a minimal implementation effort; and (3) its representation of the users is scalable and easily maintainable.

To demonstrate our claims, experiments were performed on a movie dataset providing movies metadata information, allowing the creation of users FFPs.

The remainder of this paper is organized as follows: Section 2 contains literature review on similarity metrics for CF; Section 3 presents how FFPs can be applied to RSs; Section 4 presents an experimental evaluation; finally, in Section 5 some conclusions are drawn from the results and directions for future work are proposed.

2 Related Work

Fuzzy systems approaches have been previously used to improve the RS similarity metric [6] focusing exclusively on item-based CF. Our proposal applies concepts of Fuzzy Systems to the problem of user-based Collaborative Filtering. More specifically, we use Fuzzy Fingerprints, in a CF system, to represent users in a more compact way.

CF systems usually rely on the ratings given to items by users to determine similarities between users (or items), through the use of a similarity metric. This allows the creation of neighborhoods of similar users, to predict new ratings. Traditionally, the similarity is measured using metrics such as Pearson Correlation (PC) or the Cosine similarity (COS) [2]. Nevertheless, many other ways of measuring similarity have been proposed, ranging from simple variations of PC and COS, through the design of more complex functions.

An example is the work of [7], where ratings are combined with a measure of *trust* between users, which is inferred from social information. By introducing the degree of trust between users the authors show that it does improve the overall rating prediction. On a different approach, in [3], the authors propose a combination of the mean squared difference between the user's ratings with the Jaccard coefficient. Through experiments, they demonstrate that results are improved, when compared to traditional CF.

To determine the neighborhood of each user, usually, similarities are computed between the user and all other users, which are then sorted by their degree of similarity and only the top k are kept. In [17] an alternative way to determine neighborhoods is proposed. The authors randomly choose a possible neighbor from the set of all users. This neighbor is kept only if its similarity is above a given threshold. The process is then repeated until a certain amount of neighbors is obtained. Their work has two threshold variables that depend on the data and must be fine-tuned: (1) the minimum similarity for a user to be considered a neighbor; (2) the minimum number of users in the neighborhood.

Combining Recommender systems and tags is not a novel idea [13,15,10]. Tags can help alleviate the so-called *cold-start* and *data sparsity* problems. The cold-start problem occurs when new items, not yet rated by any user, or new users, who have not rated any item yet, cannot receive recommendations since they cannot be compared to other items/users. The *data sparsity* problem is also associated with CF systems since it is common for users and items to have very few ratings, and thus not enough information to produce valuable recommendations [4]. Tags can help address these issues, they only depend on the availability of metadata, for each item. Our RS takes advantage of tags to more accurately represent each user and, therefore, improve the quality of the user similarity computation.

Liu et al. [12] also propose a new similarity metric, which assigns penalties to *bad* similarities, while rewarding *good* similarities. Defining a similarity as good or bad depends on several factors, such as the popularity of the rated items or the similarity of the rating to the other user's ratings.

In [5] a FFP was applied to item-based CF using also movies synopsis to represent items. The FFP results from ratings and synopsis words that are also added as features. A normalization is applied to both ratings and synopsis words, separatly, resulting in FFP which combines both.Note that in this work, we are currently creating a user-based CF to represent users with item tags weighted by the ratings, and not represent item using FFP.

The above works show that the selection process of neighbors and the improvement of the similarity measures have a beneficial impact on the overall RS results. This work presents a similarity metric based on FFPs, adapted for user-based CF, using the tags associated to each item, with the main goal of improving the recommendation quality.

3 Tag-Based User Fuzzy Fingerprints for Collaborative Filtering

A Fuzzy Fingerprint (FFP) is a fuzzified ranked vector containing information based on frequencies of occurrence of the elements being encoded [9]. In this Section, we explain how to build and apply a tag-based FFP to represent users in a CF recommender system.

Let N be the total number of tags in the system and let M be the total number of items in the system. Let θ_i represent the set of tags of a given item i: $\theta_i = (t_{1i}, t_{2i}, t_{3i}, \cdots, t_{Ni})$. Any element $t_{ni} \in \theta_i$ can assume the value 1 if the respective tag occurs in the item, or 0 if it does not.

Let r_u be the set of ratings for a given set of items $i_1 \cdots, i_M$, provided by a user u: $r_u = (r_{1u}, r_{2u}, \cdots, r_{Mu})$. We assume, without loss of generality, that $r_{mu} \geq 0$ and that a value of zero means that the user has not yet rated item i_m.

A Fingerprint ϕ_u is built by counting, for user u, the number of occurrences of each tag in the items rated by u, multiplied by the respective item's rating, i.e. $\phi_u = (c_{1u}, c_{2u}, \cdots, c_{Nu})$, where:

$$c_{nu} = \sum_{\forall i=1}^{M} t_{ni} \times r_{iu} \tag{1}$$

The rationale behind Eq. (1) is that tags from items a user has rated higher should also get a higher importance in the Fingerprint. The next step consists in ordering ϕ_u according to c_{nu} and keeping only the k highest values. The Fingerprint size k is a parameter of the system and can be optimized offline.

To illustrate the previous procedure, let $r_u = (5, 2, 4)$ for items a, b, and c. Assume there are only 5 tags and let $\theta_a = (1, 0, 0, 1, 1)$, $\theta_b = (0, 1, 0, 0, 1)$, and $\theta_c = (0, 0, 1, 1, 0\}$. Assuming that $k = 4$, the resulting Fingerprint ϕ_u will be $(c_{4u} = 9, c_{5u} = 7, c_{1u} = 5, c_{3u} = 4)$.

The Fingerprint ϕ_u is, therefore, an *ordered set* of tags. The rank of each tag reflects its importance in representing the user. This Fingerprint still needs to be transformed into a Fuzzy Fingerprint. The fuzzification of the Fingerprint leverages the importance of the order (and not of the frequency) to distinguish between users. The FFP of user u, Φ_u, is obtained by fuzzifying the rank (the position in the Fingerprint) of each tag.

The choice of the fuzzifying function can affect the obtained results [8,9]. Here, we have tested the linear approach, shown in Equation 2, where p_{u_j} is the rank of tag t_n within ϕ_u (starting with t=0).

$$\mu_{linear}(\mathrm{p}_{t_n}) = \frac{k - \mathrm{p}_{t_n}}{k} \tag{2}$$

Preliminary experiments indicate that using other fuzzifying functions does not significantly improve or degrade the quality of the results in this approach.

After the fuzzification step, we can now define the FFP Φ_u as:

$$\Phi_u = \{(t_n, \mu(\mathrm{p}_{t_n})), \forall t_n \in \phi_u\} \tag{3}$$

The FFP is, therefore, a ranked set of tags, each of which is associated with a membership value, built based on the description of the items rated by the user.

Once the FFP for each user is determined, it is possible to compute similarities between users.

Consider Φ_u and Φ_j the FFPs of users u and j. The FFP similarity between users u and j is defined as:

$$sim_{FFP} = (\Phi_u, \Phi_j) = \sum_{t_n \in U_i \cap U_j} \frac{\min(\Phi_u(t_n), \Phi_j(t_n))}{k} \tag{4}$$

where $\Phi_x(t_u)$ denotes the membership value associated to tag t_n in Φ_x. Note that the use of k in this equation as a normalization factor is only needed to facilitate development and parameter optimization. It can be omitted during system operation when computing similarities, largely improving computational efficiency.

The recommendation process of the proposed RS does not rely upon rating predictions as in traditional Collaborative Filtering (see Section 4). Instead, it identifies the user's nearest neighbors (according to Equation 4) and uses the items seen and liked by them to extrapolate possible items to recommend to the user.

The RS starts by computing which users are the nearest neighbors of user u, based on the FFP similarity metric. Users are considered neighbors if the similarity is greater than a defined threshold $sim_{\text{threshold}}$.

We consider that any item rated highly by a neighbor (e.g., 4 or 5 on a 0-5 scale) and rated higher than that neighbor's item rating average, is recommendable to the user.

The final step in the recommendation process consists in getting the difference between the rating of the recommendable item, the average rating given to that item by the neighbor, and multiplying it by the similarity between the user and the neighbor. This allows to create a ranking of recommendable items.

4 Evaluation

To assert the effectiveness of the proposed RS experiments were performed using a movie dataset. Precision, Recall, and F1-score are used as evaluation metrics.

The similarity metrics used as baselines for comparison are the traditional Pearson Correlation (PC) and Cosine similarity (COS). In addition, we also include

the Jaccard Mean Squared Difference (JMSD) [3], an improvement on previous metrics that offers a high rating prediction accuracy, while using a lower number of neighbors. Finally, a similarity metric, that uses FFPs [6] yet is only applicable to traditional item-based CF and which only relies upon ratings to compute similarities. We refer to this baseline [6] throughout the rest of this document as FFP_{rating}. While the FFP proposed in this document will be referenced as FFP_{tags}. All similarity metrics baselines use both user-based and item-based, except FFP_{rating} that is only applicable to item-based CF.

Pearson Correlation coefficient has been widely used since it is simple to implement, intuitive, and provides good quality results [3]. PC is defined in Eq. 5, where I is the set items both user u and j rated.

$$sim_{PC}(u,j) = \frac{\sum_{i \in I}(r_{u,i} - \bar{r}_u) \times (r_{j,i} - \bar{r}_j)}{\sqrt{\sum_{i \in I}(r_{u,i} - \bar{r}_u)^2} \times \sqrt{\sum_{i \in I}(r_{j,i} - \bar{r}_j)^2}} \tag{5}$$

The resulting similarity will be in within the interval $[-1, 1]$, where -1 corresponds to an inverse correlation, $+1$ to a positive correlation, and values near zero show that no linear correlation exists between the two users.

Another often used similarity measure is the Cosine similarity, as defined in Eq. 6. COS will yield a value between 0 and 1, where 0 corresponds to no similarity between u and j and 1 to exactly proportional ratings between both users.

$$sim_{COS}(u,j) = \frac{\sum_{i \in I} r_{u,i} \times r_{j,i}}{\sqrt{\sum_{i \in I} r_{u,i}^2} \times \sqrt{\sum_{i \in I} r_{j,i}^2}} \tag{6}$$

The idea behind Jaccard Mean Squared Difference (JMSD) is to combine the Jaccard coefficient, which captures the number of ratings in common between users, with the Mean Square Difference (MSD) of those ratings, resulting in Eq. 7:

$$sim_{JMSD}(u,j) = Jaccard(u,j) \times \big(1 - MSD(u,j)\big); \tag{7}$$

where $Jaccard$ and MSD are defined as:

$$Jaccard(i,j) = \frac{|I_u \cap I_j|}{|I_u \cup I_j|} \qquad MSD(i,j) = \frac{\sum_{i \in I}(r_{u,i} - r_{j,i})^2}{|I|} \tag{8}$$

where I_s is the set of items rated by user s.

The FFP_{rating} metric uses an approach that is totally different to the one proposed in this work: each item has its own FFP and the recommendation is based exclusively on ratings. The user's ratings constitute the item Fingerprint and ratings are sorted taking into consideration the total amount of ratings from each user.

We now explain how a traditional CF computes rating predictions. Let $\hat{r}_{ui}$ be the *predicted* rating that a given user u would assign to item i. We start by computing the *neighborhood* N_u, of user u, i.e. the set of n users in the database

that are more similar to u, using a similarity function. The value of $\hat{r}_{ui}$ is defined as:

$$\hat{r}_{ui} = \bar{r}_u + \frac{\sum_{v \in N_u} \text{sim}(u,v) \times (r_{vi} - \bar{r}_v)}{\sum_{v \in N_v} \text{sim}(u,v)} \tag{9}$$

where r_{vi} is the rating assigned by user v to item i, $\bar{r}_x$ is the average of all ratings assigned to user x. A traditional CF system usually performs these predictions for a large set of items and returns those with the highest rating predictions, as recommendations.

An evaluation was conducted using MovieLens-1M (ML-1M) dataset, from the movie domain. By using Dbpedia[1], Tags and other meta-data, regarding each movie, were collected. In this work, we focus exclusively on Tag information.

The ML-1M dataset has 1 million ratings, 6040 users, 3706 items, a sparsity of 95.53% and has an average of 125 ratings per user.

The evaluation process was performed through 5-fold cross-validation, using RiVal [14], a framework to make RSs evaluation fair process, completely separating the recommendation task of a RS from the Evaluation of the recommendations.

We define any item with rating greater than or equal to 4 as a relevant (i.e. should be recommended) to the user.

Precision can be computed using Eq. 10 and Recall using Eq. 11. In this work, we do not set a threshold for a maximum number of recommendations i.e. the RS can recommend as many relevant items to a user as possible. Even though we calculate the F1-score (Eq. 12.), we support the idea that Precision is a far better indication for a good RS, as long as Recall is within a range that allows the retrieval of a sufficient number of relevant items (in the tested cases, all approaches fulfill the Recall criteria).

$$PR = \frac{\#TruePositives}{\#TruePositives + \#FalsePositives} \tag{10}$$

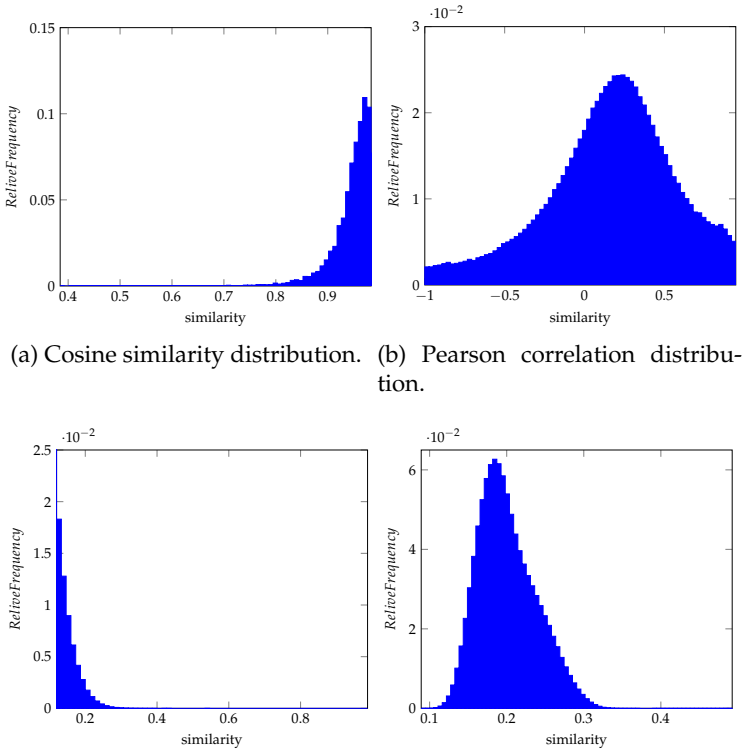
$$RC = \frac{\#TruePositives}{\#TruePositives + \#FalseNegatives} \tag{11}$$

$$F1 = 2 \times \frac{PR \times RC}{PR + RC} \tag{12}$$

We start by comparing the similarity distribution using our similarity metric and the baselines, this allows us to determine the best $sim_{\text{threshold}}$ when selecting the neighborhood. We then vary the number of neighbors used by the FFP_{tags} over different sizes of k. This allows us to determine not only the best k for the FFP_{tags} but also the most adequate number of neighbors to use. Finally, we present a summary table with baselines and how do they perform in comparison to the proposed RS.

[1] Dbpedia: http://www.dbpedia.org

Figure 1 shows the similarity distributions. By analyzing Fig. 1d, we notice that the average similarity is around 0.2. This provides a good indicator to experiment different $sim_{\text{threshold}}$ around 0.2. Experimentally, we determined that using 0.25 provides good results, for this dataset.



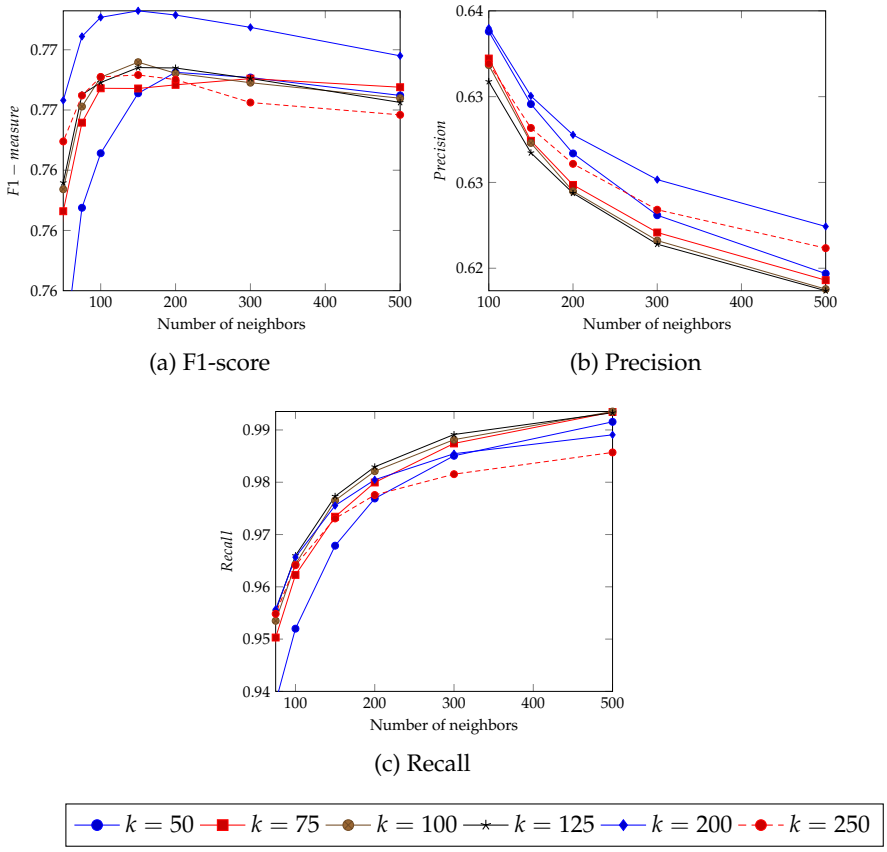
(a) Cosine similarity distribution. (b) Pearson correlation distribution.

(c) JMSD similarity distribution. (d) FFP_{tags} similarity distribution.

Fig. 1. Histograms show the similarity distribution of different similarity metrics when applied to user similarity computation. FFP_{tags} uses $k = 200$ tags to represent a user FFP.

Figure 2 compares different sizes for the FFP and for each size we vary the number of neighbors used by the RS. According to the $F1 - measure$ the best results are obtained using k equal to 200. Knowing that, on average, each user has 637 tags associated to rated movies, the proposed FFP similarity metric uses only 31% of existing tags, being able to correctly select relevant tags to represent each user.

Table 1 shows how the different tested approaches perform. The proposed Tag-user based FFP performs better overall than any other approach, even when



(a) F1-score

(b) Precision

(c) Recall

Fig. 2. Comparison between different sizes of the FFP, while varying the number of neighbors used.

compared to the state-of-the-art JMSD, although the improvement is not significant.

An interesting result is how much better the proposed approach is when compared to other previously proposed user-based approaches, thus opening the door to further developments in user-based RS. It should be noted that item-based approaches have been thoroughly used in the past and have been highly optimized. Yet user-based approaches are also viable. For example, it is very easy to enrich the FFP using data other than simple tags, from movie descriptions to a user's favorite actors, directors or genres.

Similarity Metric	Approach	Num. neighbors	F1-score	Precision	Recall
FFP_{tags}	User-based	150	**0.76929**	**0.63504**	0.97554
COS	Item-based	50	0.76622	0.62112	**0.99978**
PC	Item-based	75	0.76621	0.62115	0.99969
JMSD	Item-based	20	0.76623	0.62112	0.99980
$FFP_{ratings}$	Item-based	20	0.76623	0.62113	0.99979
COS	User-based	200	0.42356	0.26869	0.99989
PC	User-based	100	0.42338	0.26854	0.99990
JMSD	User-based	100	0.42356	0.26869	0.99989

Table 1. Summary results in which FFP_{tags} (using $k = 200$) combined with the proposed recommendation algorithm is compared with several baselines using item-based and user-based CF.

5 Conclusion

In this work, we have applied the concept of Fuzzy Fingerprints to user-based Collaborative Filtering and represented users based on tags according to the items they rated. FFPs are used to create a new concise user representation that improves the F1-score and Precision of an RS. The best result for the proposed approach was obtained for $k = 200$. In this dataset, each user has on average 637 tags, which shows that the FFPs are able to reduce the problem complexity while still improving recommendation quality.

We have experimentally compared our proposal to two traditional similarity measures, Pearson Correlation and Cosine similarity, and a state-of-the-art similarity metrics such as Jaccard Mean Squared Difference.

Results show that FFPs are a promising approach since they can be applied with success in recommendation tasks. In fact, using FFPs we are able to represent a user using, on average, 68% less features. In addition, and even though we do not address such issue in this paper, FFP similarity is a much more computationally efficient process than any of the other similarity measures. This can be arguably enough to compensate for the fact that there are usually much more users than items in RS, as we will try to show in a future work.

Future work includes more extensive parameter optimization, enriching the FFP with other features, and improving the last step of the recommendation algorithm by using more sophisticated ways to aggregate the influence of each neighbor.

Acknowledgments. This work was supported by national funds through Fundação para a Ciência e a Tecnologia (FCT) with reference UID/CEC/50021/2013, by project GoLocal (ref. CMUPERI/TIC/0046/2014) and co-financed by the University of Lisbon and INESC-ID.

References

1. Adomavicius, G., Tuzhilin, A.: Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. Knowledge and Data Engineering, IEEE Transactions on **17**(6), 734–749 (2005). DOI 10.1109/TKDE.2005.99
2. Bobadilla, J., Ortega, F., Hernando, A., Gutiérrez, A.: Recommender systems survey. Knowledge-Based Systems **46**(0), 109 – 132 (2013). DOI http://dx.doi.org/10.1016/j.knosys.2013.03.012. URL http://www.sciencedirect.com/science/article/pii/S0950705113001044
3. Bobadilla, J., Serradilla, F., Bernal, J.: A new collaborative filtering metric that improves the behavior of recommender systems. Knowledge-Based Systems **23**(6), 520 – 528 (2010). DOI http://dx.doi.org/10.1016/j.knosys.2010.03.009. URL http://www.sciencedirect.com/science/article/pii/S0950705110000444
4. Bogdanov, D., Haro, M., Fuhrmann, F., Xambó, A., Gómez, E., Herrera, P.: Semantic audio content-based music recommendation and visualization based on user preference examples. Information Processing and Management **49**(1), 13 – 33 (2013). DOI http://dx.doi.org/10.1016/j.ipm.2012.06.004. URL http://www.sciencedirect.com/science/article/pii/S0306457312000763
5. Carvalho, A., Calado, P., Carvalho, J.P.: Combining ratings and item descriptions in recommendation systems using fuzzy fingerprints. In: 2017 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE), pp. 1–6 (2017). DOI 10.1109/FUZZ-IEEE.2017.8015604
6. Carvalho, A., Calado, P., Carvalho, J.P.: Fuzzy fingerprints for item-based collaborative filtering. In: J. Kacprzyk, E. Szmidt, S. Zadrożny, K.T. Atanassov, M. Krawczak (eds.) Advances in Fuzzy Logic and Technology 2017, pp. 419–430. Springer International Publishing, Cham (2018)
7. Chen, S., Luo, T., Liu, W., Xu, Y.: Incorporating similarity and trust for collaborative filtering. In: Fuzzy Systems and Knowledge Discovery, 2009. FSKD '09. Sixth International Conference on, vol. 2, pp. 487–493 (2009). DOI 10.1109/FSKD.2009.720
8. Dimiev, V.: Fuzzifying functions. Fuzzy Sets and Systems **33**(1), 47 – 58 (1989). DOI http://dx.doi.org/10.1016/0165-0114(89)90216-9. URL http://www.sciencedirect.com/science/article/pii/0165011489902169
9. Homem, N., Carvalho, J.P.: Authorship identification and author fuzzy 'fingerprints". In: Fuzzy Information Processing Society (NAFIPS), 2011 Annual Meeting of the North American, pp. 1–6 (2011). DOI 10.1109/NAFIPS.2011.5751998
10. Kim, H.N., Ji, A.T., Ha, I., Jo, G.S.: Collaborative filtering based on collaborative tagging for enhancing the quality of recommendation. Electronic Commerce Research and Applications **9**(1), 73 – 83 (2010). DOI http://dx.doi.org/10.1016/j.elerap.2009.08.004. URL http://www.sciencedirect.com/science/article/pii/S1567422309000544. Special Issue: Social Networks and Web 2.0
11. Linden, G., Smith, B., York, J.: Amazon.com recommendations: item-to-item collaborative filtering. Internet Computing, IEEE **7**(1), 76–80 (2003). DOI 10.1109/MIC.2003.1167344
12. Liu, H., Hu, Z., Mian, A., Tian, H., Zhu, X.: A new user similarity model to improve the accuracy of collaborative filtering. Knowledge-Based Systems **56**, 156 – 166 (2014). DOI http://dx.doi.org/10.1016/j.knosys.2013.11.006. URL http://www.sciencedirect.com/science/article/pii/S0950705113003560
13. Osmanli, O., Toroslu, I.: Using tag similarity in svd-based recommendation systems. In: Application of Information and Communication Technologies (AICT), 2011 5th International Conference on, pp. 1–4 (2011). DOI 10.1109/ICAICT.2011.6111034

14. Said, A., Bellogín, A.: Rival: a toolkit to foster reproducibility in recommender system evaluation. In: RecSys '14 Proceedings of the 8th ACM Conference on Recommender Systems, pp. 371–372. ACM Press (2014). DOI 10.1145/2645710.2645712. URL http://dl.acm.org/citation.cfm?doid=2645710.2645712
15. Smith, G.: Tagging: People-powered Metadata for the Social Web, first edn. New Riders Publishing, Thousand Oaks, CA, USA (2007)
16. Tsai, C.F., Hung, C.: Cluster ensembles in collaborative filtering recommendation. Applied Soft Computing **12**(4), 1417 – 1425 (2012). DOI http://dx.doi.org/10.1016/j.asoc.2011.11.016. URL http://www.sciencedirect.com/science/article/pii/S1568494611004583
17. Wibowo, A.T., Rahmawati, A.: Naive random neighbor selection for memory based collaborative filtering. In: Intelligent Technology and Its Applications (ISITIA), 2015 International Seminar on, pp. 351–356 (2015). DOI 10.1109/ISITIA.2015.7220005
18. Ye, T., Bickson, D., Ampazis, N., Benczur, A.: Lsrs'15: Workshop on large-scale recommender systems. In: Proceedings of the 9th ACM Conference on Recommender Systems, RecSys '15, pp. 349–350. ACM, New York, NY, USA (2015). DOI 10.1145/2792838.2798715. URL http://doi.acm.org/10.1145/2792838.2798715

Appendix D

Poster - Symposium'17 CMU Portugal - Fuzzy Fingerprints for Item-based Collaborative Filtering

Fuzzy Fingerprints for Item-based Collaborative Filtering

André Carvalho
PhD Student
INESC-ID / IST

ADVISOR: **Pável Calado**
INESC-ID / IST

ADVISOR: **João Paulo Carvalho**
INESC-ID / IST

Introduction

Recommendation systems generally use Collaborative Filtering (CF), a technique where ratings given to items by users are exploited to determine rating predictions for other yet unrated items. In CF, recommendations depend on a similarity metric to compare items, a rating prediction equation, and a recommendation strategy (e.g. selection of the Top-K rated items).

We propose a novel similarity metric for Recommender Systems using Fuzzy Fingerprints and item-based Collaborative Filtering.

Our current solution allows to:

- Improve rating prediction and recommendation quality;
- Reduce the complexity required to compute item similarity, in comparison to the current state of the art metrics;
- Provide a concise representation of items, by selecting a relatively small number of user ratings and using their order to describe them.

Methodology

Item Ratings

$\mathbf{u_j}$	a	b	c	d	e	f	g	h	i
$\mathbf{r_{ji}}$	5	2	-	5	4	2	-	1	2

Fingerprint

$\mathbf{u_j}$	d	a	e	i
$\mathbf{r_{ji}}$	5	5	4	2

; Fuzzy-Fingerprint

$\mathbf{u_j}$	d	a	e	i
Φ_i	3/4	2/4	1/4	0

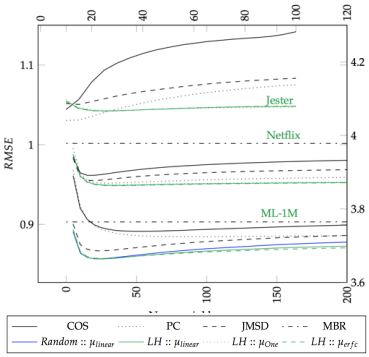
This similarity metric does not use the ratings directly instead uses the order of those ratings and an ordering criteria based on rating value and user popularity: (1) Random or (2) High-Low or (3) Low-High.

Fuzzy Fingerprint item similarity: $$\mathrm{sim}(\Phi_i, \Phi_j) = \sum_{u_v \in U_i \cap U_j} \frac{\min(\Phi_i(u_v), \Phi_j(u_v))}{k}$$

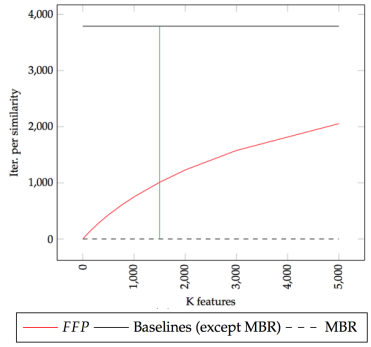
The number of ratings used to define the Fingerprint depends only on k, which highly influences the similarity computational complexity.

Rating prediction equation $$\hat{r}_{vi} = \bar{r}_i + \frac{\sum_{j \in N_i(v)} \mathrm{sim}(\Phi_i, \Phi_j) \times (r_{vj} - \bar{r}_j)}{\sum_{j \in N_i(v)} sim(\Phi_i, \Phi_j)}$$

IMAGES:



Comparison of similarity metrics.



Netflix dataset similarity metric computational analysis by varying k.

Results

Experiments have shown that the ordering criteria have a high impact on the recommendation quality.

The best performing feature sorting scheme is Low-High, which solves rating ties by placing the user with fewer ratings first. Users with less ratings are more reliable than users with more ratings. We are currently working on this question and exploring new ways to perform this sorting scheme using unsupervised learning to rank.

Regarding the similarity computational complexity, we achieved a significant reduction of the number of operations per similarity computation, between 23% and 95% less than other state of the art metrics.

Appendix E

Poster - Ciência 2017 - Fuzzy Fingerprints for Item-based Collaborative Filtering

Information and Communication Technologies Institute
Carnegie Mellon | PORTUGAL

Jan-2015 / Jan-2019

Fuzzy Fingerprints for Item-based Collaborative Filtering

André Carvalho
PhD Student
INESC-ID / IST

ADVISOR: **Pável Calado**
INESC-ID / IST

ADVISOR: **João Paulo Carvalho**
INESC-ID / IST

PARTICIPANTS

Objectives

Recommendation systems generally use Collaborative Filtering (CF), a technique where ratings given to items by users are exploited to determine rating predictions for other yet unrated items. In CF, recommendations depend on a similarity metric to compare items, a rating prediction equation, and a recommendation strategy (e.g. selection of the Top-K rated items).

We propose a novel similarity metric for Recommender Systems using Fuzzy Fingerprints and item-based Collaborative Filtering.

Our current solution allows to:

- Improve rating prediction and recommendation quality;
- Reduce the complexity required to compute item similarity, in comparison to the current state of the art metrics;
- Provide a concise representation of items, by selecting a relatively small number of user ratings and using their order to describe them.

Methodology

Fingerprint $\mathbf{u_j}$ / $\mathbf{r_{ji}}$:

d	a	e	i
5	5	4	2

; Fuzzy-Fingerprint $\mathbf{u_j}$ / Φ_i:

d	a	e	i
3/4	2/4	1/4	0

This similarity metric does not use the ratings directly instead uses the order of those ratings and an ordering criteria based on rating value and user popularity: (1) Random or (2) High-Low or (3) Low-High.

Fuzzy Fingerprint item similarity: $$\mathrm{sim}(\Phi_i, \Phi_j) = \sum_{u_v \in U_i \cap U_j} \frac{\min(\Phi_i(u_v), \Phi_j(u_v))}{k}$$

The number of ratings used to define the Fingerprint depends only on k, which highly influences the similarity computational complexity.

Rating prediction equation: $$\hat{r}_{vi} = \bar{r}_i + \frac{\sum_{j \in N_i(v)} \mathrm{sim}(\Phi_i, \Phi_j) \times (r_{vj} - \bar{r}_j)}{\sum_{j \in N_i(v)} sim(\Phi_i, \Phi_j)}$$

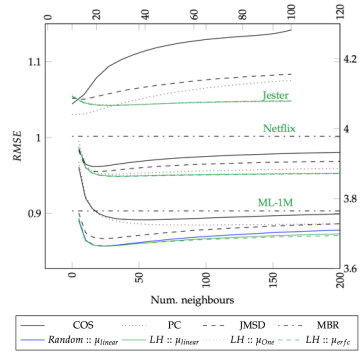
Results

Experiments have shown that the ordering criteria have a high impact on the recommendation quality.

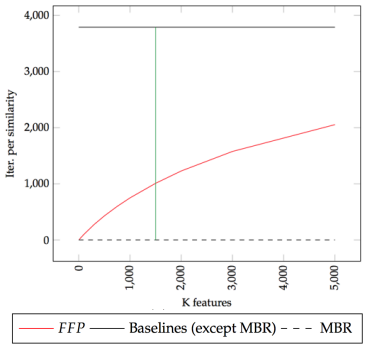
The best performing feature sorting scheme is Low-High, which solves rating ties by placing the user with fewer ratings first. Users with less ratings are more reliable than users with more ratings. We are currently working on this question and exploring new ways to perform this sorting scheme using unsupervised learning to rank.

Regarding the similarity computational complexity, we achieved a significant reduction of the number of operations per similarity computation, between 23% and 95% less than other state of the art metrics.

IMAGES:



Comparison of Fuzzy Fingerprint similarity with the baseline similarity metrics.



Netflix dataset similarity metric computational analysis by varying k.