UNIVERSIDADE DE SÃO PAULO
Instituto de Ciências Matemáticas e de Computação

Content recommendation based on semantic descriptions extracted from user annotations

Rafael Martins D'Addio

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

ICMC USP SÃO CARLOS

Rafael Martins D'Addio

Content recommendation based on semantic descriptions extracted from user annotations

Doctoral dissertation submitted to the Institute of Mathematics and Computer Sciences – ICMC-USP, in partial fulfillment of the requirements for the degree of the Doctorate Program in Computer Science and Computational Mathematics. *FINAL VERSION*

Concentration Area: Computer Science and Computational Mathematics

Advisor: Prof. Dr. Marcelo Garcia Manzato

USP – São Carlos
November 2020

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

D'Addio, Rafael Martins
D121c Content recommendation based on semantic descriptions extracted from user annotations / Rafael Martins D'Addio; orientador Marcelo Garcia Manzato. -- São Carlos, 2020.
137 p.

Tese (Doutorado - Programa de Pós-Graduação em Ciências de Computação e Matemática Computacional) -- Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, 2020.

1. Recommender Systems. 2. Information Retrieval. 3. Natural Language Processing. I. Manzato, Marcelo Garcia, orient. II. Título.

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

Rafael Martins D'Addio

Recomendação de conteúdo baseada em descrições semânticas extraídas de anotações de usuários

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

Área de Concentração: Ciências de Computação e Matemática Computacional

Orientador: Prof. Dr. Marcelo Garcia Manzato

USP – São Carlos
Novembro de 2020

This work is dedicated to the people that helped it become true.
To my parents, girlfriend, friends and colleagues, I say: thank you for believing in me.

ACKNOWLEDGEMENTS

First and foremost, I would like to deeply thank my mother, Viviane. Without her love and support, the words of encouragement and her blessings, I wouldn't be able to come this far. In fact, I would like to thank all the people that accompanied me throughout this journey, my family, friends and girlfriend. Their support means a lot to me.

I would like to thank my doctorate supervisor, Marcelo Manzato, who accompanied me through seven years, since my Master's degree. He was the one that, through stimulant discussions, careful orientation and a lot of patience ultimately allowed me to reach the end of this cycle.

I would like to thank the people from my institute: the professors, lab partners and colleagues. They were part of my education as a researcher, a teacher and as a better person overall. I would also like to give a special thanks to the people of our research group and external collaborators: Arthur da Costa, Eduardo Fressato, Ronnie Marinho, Arpit Rana and Derek Bridge. You guys helped me advance my research in many ways through our years of collaboration.

Finally, I would like to thank the University of São Paulo for the opportunity, CAPES for funding my research and FAPESP for the financial support on acquiring equipment and publishing our results.

“All we have to decide is what to do with the time that is given us.”
(J.R.R. Tolkien, The Fellowship of the Ring)

RESUMO

D'ADDIO, R. M. **Recomendação de conteúdo baseada em descrições semânticas extraídas de anotações de usuários**. 2020. 137 p. Tese (Doutorado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, 2020.

Sistemas de recomendação surgiram como forma de reduzir o problema de sobrecarga de informações. Eles necessitam de dados para funcionar corretamente, ou seja, seus usuários precisam fornecer interações para que seus perfis possam ser construídos, enquanto itens precisam de descrições para serem diferenciados, especialmente em abordagens baseadas em conteúdo. Essas abordagens geralmente dependem de fontes externas, que podem ser metadados estruturados ou textos não estruturados, como sinopses, notícias ou resenhas de usuários. As resenhas de usuários têm sido cada vez mais usadas como fonte de informações devido à sua capacidade de transmitir características do item, bem como a opinião do autor em relação a elas. Esta pesquisa de doutorado se concentra no design de representações ricas e semânticas de itens a partir de resenhas de usuários, a fim de aumentar a precisão das recomendações. Além disso, objetiva-se analisar sua aplicação em vários domínios, algoritmos e tarefas de recomendação, verificando sua qualidade e diferenças. Propõem-se cinco representações de itens baseadas em conceitos, que são projetadas no modelo de espaço vetorial. Todas elas compartilham, em algum nível, o mesmo conjunto de características, variando na semântica que seu esquema de ponderação transmite. Elas foram exploradas numa variedade de abordagens, algoritmos e configurações de avaliação, desde experimentos offline até uma avaliação online com usuários reais. Os resultados revelam a necessidade de projetar boas representações semânticas de itens, que ajudam o recomendador a diferenciar corretamente os itens da coleção e, finalmente, produzir sugestões mais relevantes para seus usuários.

Palavras-chave: Sistema de Recomendação, Representações de Itens, Anotações de Usuário, Recuperação da Informação.

ABSTRACT

D'ADDIO, R. M. **Content recommendation based on semantic descriptions extracted from user annotations**. 2020. 137 p. Tese (Doutorado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, 2020.

Recommender systems emerged as means of reducing the information overload problem. They require data to correctly function, i.e., users need to provide interactions so their profiles can be constructed, while items need descriptive data to be differentiated, especially in content-based approaches. Those approaches often rely on external sources, which can be structured metadata or unstructured texts such as synopses, news or user reviews. User reviews have increasingly been used as source of information due to their capability of conveying item characteristics as well as the author's opinion towards them. This doctorate research focuses on designing rich, semantic item representations from user reviews in order to increase recommendation accuracy. Beyond that, we aim to analyze their application in several domains, algorithms and recommendation tasks, verifying their quality and differences. We have proposed five concept-based item representations, which are designed in the vector space model. They all share, in some level, the same feature set, varying in the semantics which their weighting scheme convey. We explore them in a plethora of approaches, algorithms and evaluation settings, from offline experiments to an online user trial. Results reveal the necessity of designing good, semantic item representations, which aid the recommender in correctly differentiating the items in the collection and ultimately producing more relevant suggestions to its users.

Keywords: Recommender Systems, Item Representations, User Annotations, Information Retrieval.

LIST OF FIGURES

LIST OF ALGORITHMS

LIST OF TABLES

LIST OF ABBREVIATIONS AND ACRONYMS

CB	the classic content-based algorithm
fb	unweighted feature-based formulation in *r-by-e*
k-NN	*k* -Nearest Neighbor
nb	unweighted neighbour-based formulation in *r-by-e*
r-by-e	Recommendation-by-Explanation
wfb	weighted feature-based formulation in *r-by-e*
wnb	weighted neighbour-based formulation in *r-by-e*
BPR	Bayesian Personalized Ranking with Matrix Factorization
BPR	Bayesian Personalized Ranking
CBF	content-based filtering
CBMF	Content-Boosted Matrix Factorization
CF	collaborative filtering
CT-Sentiment	Classification Terms weighted by Sentiment
CT-TFIDF	Classification Terms weighted by TFIDF
DF	document frequency
EL	entity linking
EM	Expectation-Maximization algorithm
FCM	Fuzzy C-Means algorithm
HT-Sentiment	Heuristic Terms weighted by Sentiment
HT-TFIDF	Heuristic Terms weighted by TFIDF
IF	item frequency
IIB	Iterative Information Bottleneck algorithm
IMDb	Internet Movie Database
IR	information retrieval
Item-MSMF	Item Most Similar based on Matrix Factorization
ItemAttrKNN	Item Attribute *k*-NN
LIME	Local Interpretable Model-Agnostic Explanations
MF	Matrix Factorization
ML-100k	MovieLens 100k dataset
NASARI	Novel Approach to a Semantically-Aware Representation of Items
NLP	natural language processing

PLTRB	Paralleled Personalized Pairwise Lear-to-Rank with block partitioning
PLTRN	Paralleled Personalized Pairwise Lear-to-Rank with no partitioning
POS	part-of-speech
RMSE	root mean square error
RS	Recommender systems
SF-IDF	synset frequency - inverse document frequency
TF-IDF	term frequency - inverse document frequency
WSD	word sense disambiguation
XML	Extensible Markup Language

LIST OF SYMBOLS

— ———————————- General Common Notation ———————————

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

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

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

D, K — the set of rated items

i — an item

I — the set of items

k — the number of neighbors, for k-NN algorithms

m_j — the number of relevant items in a ranking in MAP@N equation

O_u — the predicted items set for a user u, in RMSE equation

$p - value$, p_0 — the value used to determine statistical significance

q_j — a query in MAP@N equation

Q — the set of queries in MAP@N equation

r_{ui} — a rating or feedback from a user u to an item i

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

R_{jk} — the set of ranking results returned from the first item until the kth item in MAP@N equation

s_{ij} — a similarity score between two items

$S^k(i;u)$ — the set of k nearest neighbors of item i rated by user u

u — a user

U — the set of users

λ_n — several regularization factors that are used in multiple algorithms

μ — the global average rating in the dataset

— ———————————- Item Representation Notation ———————————

f — a feature

F — the set of features

IF_f — the item frequency value of a feature

k_{if} — a binary value that indicates whether an item has (1) or not (0) a feature in IF equation

n_s, $n_{s,i}$ — the frequency of synset s in the item reviews

N — the global average number of synset frequencies

s — a synset

s_c — a sense embedding

s_i — an item embedding

$S_c(i)$ — the set of sense embeddings of the features of an item i

$S(s,i)$ — sentiment value of a synset s in relation to an item i

w_{fi} — the weight of a feature f for an item i

— ———————————————- Chapter 2 specifics ———————————————

a_i — an aspect that an item contains in an item representation based on aspects

A — the set of aspects

c_k — a cluster

g_i — a genre that an item contains in an item representation based on genres

G — the set of genres

$i(c_k)$ — a relevance degree of an item i to a cluster ck

l — a Gaussian, in EM algorithm

m — a fuzzification value, > 1, in FCM algorithm

R_v^i — a set of users that have rated item i

R_j^u — a set of items rated by a user u

t_i — a term that an item contains in an item representation based on terms

T — the set of terms

u_i — users that rated an item in an item representation based on ratings

$u(c_k)$ — a relevance degree of a user u to a cluster ck

v_i — a cluster centroid in FCM and EM algorithms

x_j — an object in FCM and EM algorithms

μ_{ij} — the degree of pertinence of an object x_j to a cluster c_i in FCM algorithm

π_l — the priori probability that a random object was generated by a Gaussian l, in EM algorithm

Σ_l — the weighted covariance matrix of the Gaussian l

— ———————————- Chapter 4 specifics ———————————

$N_i^k(i;u)$ — the subset of most similar items to candidate item i that are not new

p_u — latent vector representing user u's interactions

q_i — latent vector representing item i's interactions

$\mathbb{R}_f$ — latent vector space representing the system's interactions

σ_i — a vector in the cosine similarity metric

— ———————————- Chapter 5 specifics ———————————

D_k — the set of triples (u,i,j) where item i is known to user and item j is not

$N_u(i)$ — The set formed by the intersection between k most similar items to i and the set of items that user u interacted with

$N_u(i)^S$ — the sentiment neighborhood

$N_u(i)^{SF}$ — the SF-IDF neighborhood

p_{ij} — the Pearson correlation coeficient value between items i and j

$\hat{r}_{ul}^S$ — a prediction value generated by sentiment item representations

$\hat{r}_{ul}^{SF}$ — a prediction value generated by SFIDF item representations

$R(u)$ — the final ranking for user u

$R_S(u)$ — the sentiment ranking for user u

$R_{SF}(u)$ — the SF-IDF ranking for user u

w_n^i — the value of feature n towards item i in Pearson correlation coefficient formulation

$\overline{w_i}$ — The average values of the features of item i in Pearson correlation coefficient formulation

w_{fi}^S — the sentiment weight of a feature f for an item i

w_{fi}^{SF} — the SF-IDF weight of a feature f for an item i

$\hat{x}_{uij}$ — a real value that captures the relationship between a triple $(u,i,j) \in D_k$

α_n — a weight assigned to predictions from a recommender n in the BPR-based ranking combination techniques

η — a learning rate constant

Θ — the parameters to be trained in BPR formulation

— ———————————- Chapter 6 specifics ———————————

C — an Explanation Chain

F_i, F_j, F_p — the set of features of an item, which can be the candidate item i, the candidate predecessor p or the predecessor j

I — the set of candidate items

j — the chosen predecessor to be added in an Explanation Chain C

J — the set of candidate predecessor items p

L — a list of Explanation Chains for candidate items

$L*$ — a ranked list of top-n Explanation Chains

N_i, N_p — the set of neighbours of a candidate item i or predecessor p

o_{fi} — the frequency of a feature f for an item i

p — a candidate predecessor item

P — the set of items in a user profile

r — the Pearson correlation between explanation-ratings and actual-ratings

w_{ji}, w_{jp} — the weight of a neighbour towards candidate item i or predecessor p

w_{max} — the maximum value that the difference between the weights of a feature of two items can assume

$\mathbb{I}$ — the set of all items

$\mathbb{I}_f$ — the set of items that contain feature f

α — a parameter which controls the influence of item or profile overlap in chain selection of *r-by-e*

ε — a marginal gain threshold

θ — a similarity threshold

μ_d — the mean difference between explanation-ratings and corresponding actual-ratings

σ — the standard deviation between explanation-ratings and corresponding actual-ratings

CONTENTS

CHAPTER

1

INTRODUCTION

Since the advent of Web 2.0 (O'REILLY, 2005), it became easy for users to create and provide content. That effect influenced to the aggravation of the *information overload* problem (AGGARWAL, 2016a; RICCI; ROKACH; SHAPIRA, 2015), where there is so much information online that users are not able to filter it themselves. Recommender systems (RS) arrived to soften this problem, and are researched even before the Web 2.0 movement, nearly completing three decades of research, with the first works as independent research area appearing in mid-1990s (RICCI; ROKACH; SHAPIRA, 2015). Nowadays, e-commerce and streaming services are the ones that most benefit from RS, but other applications can indeed use them, such as spam filters, trip planning agencies, food ordering applications, and even dating apps.

Recommender systems take users feedback to learn their tastes in order to suggest relevant items, softening the information overload problem and facilitating users to choose what to buy or consume (AGGARWAL, 2016a; RICCI; ROKACH; SHAPIRA, 2015). User feedback can vary between thumbs up/down buttons, star ratings (e.g., 1 to 5 stars), bookmarking and/or building a wish list; or be more subtle as capturing a user's browsing history or mouse clicking pattern (NING; DESROSIERS; KARYPIS, 2015). User feedback is crucial to recommender systems; without them, there is no way that the system is able to produce a personalized list of suggestions. In this case, recommendation is just a matter of most popular or highest rated items.

Suggestions that a recommender may produce can be evaluated in two main scenarios: rating prediction and top-N recommendation (AGGARWAL, 2016a). Earlier research in recommender systems tried to reduce the rating prediction error, i.e. strengthen the system's capability of guessing the rating that a determined user would give to a determined item. But user confidence towards RS are not related with its ability to predict ratings; in fact, users are more interested in receiving lists of suggestions and their confidence on the system raises as it agrees with the set of suggestions it receives (BOBADILLA *et al.*, 2013). With that in mind, top-N recommendation has been taking a growing interest by the research community, while researches focused on rating prediction have diminished. Beyond accuracy, researches that regard different

aspects such as diversity, novelty, serendipity, etc. have gained attention in the past few years (KAMINSKAS; BRIDGE, 2016).

Several techniques can be used to create recommender systems. Among them, the best known and most used are collaborative and content-based filtering paradigms. The first elects its recommendations based on the users' previous interactions, deriving relationships between users and/or items (NING; DESROSIERS; KARYPIS, 2015; KOREN; BELL, 2015). The second suggests items based on their descriptions, which must be similar to the user's interest profile (GEMMIS *et al.*, 2015).

The most important problem that collaborative filtering algorithms must deal is what is called cold start, i.e. new items are difficult to recommend because they have not been classified; similarly, new users are not able to receive suggestions because they do not have enough rating history (NING; DESROSIERS; KARYPIS, 2015). Content-based recommendation systems often have difficulty to obtain semantic and meaningful item information (GEMMIS *et al.*, 2015). Both techniques characterize most of the current recommendation systems, and it is still possible to intertwine aspects of each, generating a hybrid system (ADOMAVICIUS; TUZHILIN, 2005). Such systems take advantage of the benefits of both approaches in order to generate better recommendations, e.g., collaborative filtering algorithms can incorporate item descriptions in order to reduce the cold start problem (RICCI; ROKACH; SHAPIRA, 2015).

Beyond collaborative and content-based filtering, there are several other approaches in which recommender systems can be built. For instance, knowledge-based recommenders deal with unusual item recommendation, which require detailed item representations, which are often constructed as knowledge graphs, and user feedback, which is often captured by means of user queries (AGGARWAL, 2016b). Context-based recommenders rely on context information, such as time of day, mood, pricing, location, among others (HARUNA *et al.*, 2017). There is even a growing effort by the community to research better ways to explain recommendations to users, improving its trust towards the RS (ZHANG; CHEN, 2020).

Regardless of the method, in order for a RS to be successful, it must rely on its main aspect: its available data. Users need to provide feedback and, in most of the aforementioned approaches, items must be well described so that the system can make good suggestions.

This research focus on the item aspect of Recommender Systems. In our research, we aim to increase recommendation accuracy by improving the descriptions of the items which are likely to be recommended.

1.1 Motivation and Problem Statement

Recommender Systems need data to correctly function. With no user feedback, collaborative filtering systems would not be able to generate suggestions (KOREN; BELL, 2015). Similarly,

content-based recommender systems (GEMMIS *et al.*, 2015), alongside other content-aware RS paradigms such as context (HARUNA *et al.*, 2017) and knowledge-based (AGGARWAL, 2016b) recommendations, need as much information about the items as possible. Content and knowledge-based systems need semantic information about the items' features, so that the systems are able to differentiate them and correctly match them with the users' profiles or their explicit information need, respectively. Context-based recommenders usually require circumstantial information such time of the day, location, etc., but also may require to know items information to derive contextual recommendations (HARUNA *et al.*, 2017). There has also, in the past few years, a growing effort on refining recommendations explanations or justifications, where some successful approaches relied mostly on item features (ZHANG; CHEN, 2020).

In this sense, it is natural that researches may focus on methods for better describing items. Early works attempted to use structured item information (such as genres and/or personnel for movies and music) to describe items (PAZZANI; BILLSUS, 2007), which often required a domain expert to correctly catalog them or the access to external databases. Other works started to use unstructured information, i.e. texts, to extract features for their items, such as the news recommender developed by Capelle *et al.* (2015), or the cross-lingual recommender from Narducci *et al.* (2016). Another step towards enhancing semantics of content-based systems was the adoption of user reviews as the primary source of information, a growing trend according to the survey performed by Chen, Chen and Wang (2015).

Indeed, reviews provide great semantics since they are able to convey both items' characteristics and users' opinions towards them. With the aid of methods, tools and techniques derived from the areas of natural language processing (NLP) and information retrieval (IR), several works were able to extract information from these sources in order to describe users (TERZI *et al.*, 2014; ZHAO *et al.*, 2019), items (QUMSIYEH; NG, 2012; MCAULEY; PANDEY; LESKOVEC, 2015) or both (CONTRATRES *et al.*, 2018). Moreover, user reviews were also used to aid in the process of explaining or justifying recommendations, such as the works of Chen and Wang (2017) and Musto *et al.* (2019).

But, to the best of our knowledge, most of these works tend to use this information as means to refine their own recommender systems and algorithms, not delegating the required attention to correctly refine and properly design structures and/or representations that are able to convey the semantics found within these texts. Moreover, linguistic issues such as synonymy (several word conveying the same meaning) and polisemy (one word containing several meanings) are often disregarded, which ultimately may hurt the system's capability of acknowledging similarities between items or even disregarding fundamental differences between them. For example, a recommender that have representations with synonymous features may regard a movie with the feature *"cop"* as not related to a movie with the feature *"policeman"*, since the two words are regarded as different features even though they share the same meaning. Similarly, two movies with the feature *"stars"* can be regarded as related, while one can relate to "celestial

corpses" and the other could be using the word as a verb, in which it means "play a role".

Even though we have worked previously in this research theme (D'ADDIO; MANZATO, 2015; D'ADDIO; DOMINGUES; MANZATO, 2017), we have not addressed the aforementioned linguistic issues. Also, we also did not focus on the semantics that different feature weighting schemes may present to recommenders. Finally, we have previously restricted our efforts to test our representations in a single recommender algorithm and item domain.

This doctoral work is a continuation of our previous efforts, with the goal to address the issues we have presented so far.

1.2 Research Questions and Hypothesis

In order to develop our research, we raised an important question that were drawn from our previous work as well as from careful reading of the literature:

- Research Question 1: How can we effectively use user-provided unstructured information such as user reviews to better describe items?
  - Hypothesis 1: Allied with state-of-the-art natural language processing tools, we can design rich and semantic item representations that allow recommender systems to better discern items qualities and flaws, ultimately matching them with users' tastes.

As one can see, the literature presents a plethora of works that try to address the same issue. But as far as we know, most of those works refrain themselves in trying to construct representations with a higher degree of semantics, using them as tools to enhance the recommending solutions they propose. In this sense, our work focuses on the item representation side of the problem, devising semantic alternatives to describe them.

Moreover, the research question and Hypothesis presented above are broad and difficult to be directly addressed in experiments. To better address this question, we have formulated two more research questions that can be directly addressed in our experiments, leaving Question 1 to be indirectly answered. The research questions are:

- Research Question 2: Does increasing the semantics of item representations improve recommendation accuracy?
  - Hypothesis 2: Increasing item's semantics leads to a greater capability of differentiating items, which is crucial for recommender systems that rely on content to make predictions.
- Research Question 3: Is a kind of item representation more relevant to use in determined recommender system or scenario than another?

– Hypothesis 3: Each recommender system and scenario in which it is being applied on has their particularities, which should be addressed accordingly. Different semantics may benefit differently in each of these The descriptive power of a representation may dictate its applicability in each scenario.

1.3 Objectives

Given the aforementioned Research Questions and their respective Hypotheses, this research project has as main objective the creation of more rich and semantic item representations based on user-annotated texts, such as user reviews, with the aid of well established natural language processing and information retrieval methods and tools. These representations can aid recommender systems in their decision making processes to generate more meaningful suggestion for their users. Beyond producing them, we have as secondary objective the goal of analysing their application in different recommender algorithms, settings, data domains and approaches, to better ascertain their capabilities, advantages and limitations.

1.4 Contributions

This research produced the following main contributions:

- We have defined a methodology for producing item representations based on opinionated texts, i.e. user reviews. This methodology considers reviews as items' descriptive information and considers mainly the consensus of user opinion towards items. In this sense, user reviews can be extracted from different sources instead of just relying on information from users within the system itself. This directly diminishes the limited content analysis (GEMMIS *et al.*, 2015) since information can be easily gathered from free and open platforms on the Web. We follow the same methodology in the construction of all types of item representations proposed in this research. First, we gather reviews for the available items, which may be obtained directly from the users or from external sources, depending on the dataset; then, we extract the feature set that will represent the dimensions of most of our vector-based representations and, finally, we devise a weighting scheme for them that will be based on different semantics.

- We have developed five different item representations, which are produced following the methodology described previously, constructed with the aid of state-of-the-art NLP and IR methods and tools. Those representations are all based on the same feature extraction technique, which extracts features as concepts instead of simple terms, varying on the weighting scheme. We used BabelFy (MORO; RAGANATO; NAVIGLI, 2014) to disambiguate the items' reviews, allowing us to extract the relevant concepts for the representations. We made different representation constructions, all based on the concepts

we extracted. The first one was based on an extension of the well known term frequency - inverse document frequency (TF-IDF) (MANNING; RAGHAVAN; SCHüTZE, 2008) weighting scheme, and is depicted on experiments detailed on Chapters 3 and 5. Another one, based on sentiment analysis, and considered our main approach, is used in experiments on Chapters 4, 5 and 6. The other three representations were all based on concept *embeddings*, and they vary on the semantics they depict. One of them plots the item in the same latent space as the embeddings, and the other two are based on similarity between item and concepts. They are detailed in the experiments conveyed in Chapter 5.

- We applied the representations in a wide array of algorithms, data domains and methods. Even though we performed several tests and comparisons, we restrict our presentation to the most expressive results, which were or are in the process of being published, and are reported in the rest of this thesis. In the majority of our experiments, we applied our approaches in the classic item-based k -Nearest Neighbor (k-NN) algorithm, since it is data-dependent and ideal to evaluate the differences between representations. We also applied them in a matrix factorization algorithm, as well as an explanation-oriented RS. We evaluated them in both the rating prediction and top-N recommendation tasks, which are the tasks in which most of the works on recommender systems are based, as well as into several different datasets. We also applied our approaches in a simulated cold-start scenario and a user trial.

- We have prepared four datasets with our pre-processing approaches, which are available to use on a repository[1]. These pre-processing steps were very resource intensive, since there was a great volume of text to be processed and some tools, such as Babelfy, had limited daily access to its server. The already pre-processed datasets are available to use, easing preparation time for future experiments.

- This research produced 2 full papers and 1 short paper published in conferences, as well as 1 published and one submitted journal article. Beyond them, we produced other 2 full and 1 short conference papers, and 2 journal articles during this period. Even though they are related to this research, those papers were not inserted in this document, since some of them reflect contributions made previously during the author's master degree, while others relate to user profile modeling. They are listed for reference in Section 7.3.

1.5 Outline

This doctoral thesis is organized in the form of a collection of articles, which were written during this research and summarize its contributions. The four conference papers and the two journal articles presented here all relate to the main goal of this research: exploring rich

[1] https://github.com/RafaelDaddio/DatasetsSysRec

and semantic item descriptions in recommender systems. Each article is depicted in its entirety in each chapter, enveloped by an introduction and a conclusion which places them into this research's context. The articles summarize the most relevant results obtained throughout the research, and are organized as follows:

- Chapter 2 presents a new study based on previous attempts into producing sentiment-based item representations (D'ADDIO; MANZATO, 2015), this time applied into a recommender based on soft-clustering techniques. The paper depicted here, "Exploiting Item Representations for Soft Clustering Recommendation", was published in the Proceedings of the 22nd Brazilian Symposium on Multimedia and the Web (WebMedia '16) (D'ADDIO; MANZATO, 2016).

- Chapter 3 presents a collaboration with Ronnie Shida Marinho, in which we produced item representations based on disambiguated concepts using state-of-the-art natural language processing tools. The paper is entitled "Semantic Organization of User's Reviews Applied in Recommender Systems" and was published as a short paper in the Proceedings of the 23nd Brazilian Symposium on Multimedia and the Web (WebMedia '17) (MARINHO; D'ADDIO; MANZATO, 2017).

- Chapter 4 presents another collaboration between our research group, in which we use four different item representations in two recommender systems on a simulated cold-start scenario. The paper, entitled "Incorporating Semantic Item Representations to Soften the Cold Start Problem", was published in the Proceedings of the 24nd Brazilian Symposium on Multimedia and the Web (WebMedia '18), receiving the Best-Paper Award (D'ADDIO *et al.*, 2018).

- Chapter 5 reports efforts into combining two of the proposed item representations in order to increase recommender accuracy. We explore ten different techniques which can be divided into pre, post and neighbor combinations. The article, entitled "Combining different metadata views for better recommendation accuracy" was published in the Elsevier's Information Systems Journal (D'ADDIO; MARINHO; MANZATO, 2019).

- Chapter 6 reports a collaboration with Arpit Rana and Derek Bridge from University College Cork, Ireland, in which we apply one of our item representations in their state-of-the-art recommender system with the focus of checking whether this representation helps produce better explanations for users. The paper, entitled "Extended Recommendation-by-Explanation" is currently submitted to Springer's User Modeling and User-Adapted Interaction.

Finally, Chapter 7 presents the conclusions of this research, as well as its limitations and future directions.

CHAPTER

2

EXPLOITING ITEM REPRESENTATIONS FOR SOFT CLUSTERING RECOMMENDATION

2.1 Contextualization

Recommender systems are known to capture and process the interactions that their users perform to better learn what to suggest (RICCI; ROKACH; SHAPIRA, 2015). Content-based or content-aware hybrid recommenders consider extra information that better describes their items to, allied with its users' interactions, improve suggestions. Even new items have a chance to be recommended if they have features in common with other previously liked items (AGGARWAL, 2016a). Thus, a crucial design aspect of such recommenders is the quality of their items' representations, which must correctly describe them so that the system is capable of finding similarities and discrepancies, correctly aligning them with users' interests.

As we have stated in Chapter 1, the main goal of this research is to design such rich representations, as well as explore its impacts on content-aware recommender systems. In earlier works (D'ADDIO; MANZATO, 2015; D'ADDIO; DOMINGUES; MANZATO, 2017), we have designed some review-based item representations. Our main goal was to find semantic methods to extract item features, which would comprise the dimensions of vector space representations. They were applied to the well known item k-NN algorithm, yielding interesting results.

In this work (D'ADDIO; MANZATO, 2016), we apply some of those representations in another recommender algorithm. With that, we analyzed whether the representations yield the same result patterns in other contexts, as well as if they indeed help better describe items. We apply them into an adapted algorithm from the work of Ganu, Kakodkar and Marian (2013), which uses soft clustering solutions to predict ratings of unknown items for a user.

Exploiting Item Representations for Soft Clustering Recommendation

Rafael M. D'Addio and Marcelo G. Manzato
Institute of Mathematics and Computer Science
University of São Paulo
São Carlos, SP, Brazil
{rdaddio,mmanzato}@icmc.usp.br

ABSTRACT

Recommender systems help dealing with the information overload problem since they provide personalized content for users. There are two major paradigms in recommendation: content-based and collaborative filtering. Regardless of the paradigm, there has been a great effort into finding additional information to better describe items and/or users, which in turn helps to increase the personalization power of the system. User's reviews turn out to be a great source of information, since they provide information about the characteristics of the items as well as insights about the opinion of the user towards them. In previous works, we explored some techniques for extracting information from reviews in order to generate items' representations and applied them into an item k-NN algorithm. In this work, we explore the impact that those representations, alongside with rating and genre-based representations, can cause into a soft clustering-based recommender system. We compare our findings with the item k-NN algorithm and observe that they are better in some cases, but the soft clustering recommender has lower computational cost.

CCS Concepts

•**Information systems** → **Recommender systems;**

Keywords

Recommendation; Soft Clustering; Item Representation

PUBLICATION INFORMATION

Published in proceedings of the 22nd Brazilian Symposium on Multimedia and the Web (Webmedia '16). Volume 2. 2016. Pages 271–278. DOI:10.1145/2976796.2976858

1. INTRODUCTION

Recommender systems emerged to deal with the well-known information overload problem by producing personalized content to their users. Theses systems can be traditionally divided into two main paradigms: content-based [15], where users' profiles are matched with items' representations using similarity measures; and collaborative filtering [8, 13], where two main approaches are adressed: the neighborhood and the latent factors space models. Beyond these two paradigms, there is an effort to combine them into a third hybrid approach, where the flaws of each other are compensated by their strengths [1].

Specifically in neighborhood models, there are many forms to represent the entities (users or items) in order to obtain their similarities. The most common approach is to use the user x item rating matrix as features for a similarity metric [12], but one can use other information such as items' metadata or users' demographics depending on the context.

In fact, given the current scenario of the Web, where users can provide content by producing annotations, comments and reviews about any subject, there is a great amount of rich and detailed information available that is created collaboratively by the community. Recent works focus on extending the traditional recommendation paradigms by using this user-provided unstructured information [9, 11, 21, 23]. This is a great source of information, since it is able to describe items in detail.

However, dealing with such unstructured user-generated data raises a set of challenges [2]. First, reviews are prone to the occurrence of noise, such as misspelling and false information. Secondly, there is a requirement for natural language processing (NLP) tools to analyze, extract and structure relevant information about a subject from texts. Finally, there is a lack of research about how to organize and use additional data provided by users in order to enhance items' representations, and consequently, to improve the accuracy of recommendations.

In previous works [5, 6, 7], we explored different methods to extract relevant information from users' reviews in order to produce items' representations for an item-based k-nearest neighbors (Item k-NN) [12]. Although the results were promising, the study was restricted to one specific recommendation algorithm. Even more, the algorithm takes a considerable time to process, since it builds a neighborhood for each user x item pair by taking into account item x item similarities among those rated by the user.

A possible alternative to reduce computational cost and time is to cluster closely related items. Since those items have similar characteristics, their contribution to a final recommendation score may be close, thus allowing to group them and use a single score as their cluster contribution.

Even though, items may have a big variety of characteristics, making it virtually impossible to group them into well-defined and hard partitioned clusters. With this, a possible solution is to use a clustering solution which produces overlapping clusters, with items having probabilities, or pertinence degrees, of belonging to each of the clusters. With this, items can relate with each other in different aspects and with different degrees.

In this study, we explore the impact that different types of items' representations can cause to an algorithm where the relatedness between items is governed by a soft clustering of them. We adopt the terminology "soft" to address partitions with cluster overlapping, but test our findings with fuzzy (Fuzzy c-means [3]) and probabilistic (Expectation-Maximization [4]) clustering algorithms. We apply the partitions obtained by these algorithms into a recommender system which we adapt from the work of Ganu et al. [9]. We evaluate our proposal by considering the error between predicted and real ratings, and the results show that the recommendations generated by considering soft clusters of items are better to those produced by the Item k-NN model with two of the four items' representations addressed. Nevertheless, the best soft-clustering configuration yields better results than the best item k-NN configuration, with the advantage of requiring less computational time to be executed.

This paper is structured as follows: in Section 2 we discuss some related work; in Section 3 we provide more details into which items' representations we have considered in this study; in Section 4 we detail the recommendation algorithm; in Section 5 we describe our experimental setting and our results and, finally, in Section 6 we provide our final remarks.

2. RELATED WORK

In this section, we present some works that use users' reviews to obtain additional information in order to generate better recommendations. We also present some works that aggregate clustering solutions to calculate recommendations.

Some recent works use reviews to extract sentiment related to the characteristics of the items in order to characterize them for a content-based recommendation scenario. For example, Qumsiyeh and Ng [21] proposed a system capable of generating recommendations for various multimedia items, using information such as genres, actors and reviews, extracted from multiple trusted Web sites. Their method is based on mathematical and statistical formulations using the sentiment (positive, neutral or negative) and degree (ratings) of every aspect it considers to predict scores of unclassified items.

Unlike the work described above, other works use reviews for the construction of user profiles, applying them in collaborative filtering and hybrid approaches. Kim et al. [11], for example, proposed a personalized search engine for movies, called MovieMine, based on reviews and user-provided ratings. In this system, the user types a query, which is expanded by adding keywords taken from earlier reviews provided by himself, allowing the search key to be customizable. Ganu et al. [9] proposed a review-based recommendation system for restaurants. This system performs a soft clustering of users based on topics and sentiment present in the reviews.

Clustering methods applied into collaborative filtering recommendation have been studied previously in the literature. In the work of Gong [10], for instance, the k-means algorithm was used to cluster users based on their ratings in order to find nearest neighbors and smooth the prediction. Then, k-means is applied to cluster items and finally produce the recommendations. Park and Tuzhilin [19] address the Long Tail problem problem by splitting the data into head (items with many ratings) and tail (new or unpopular items), and apply the EM clustering in the tail set. Recommendation is thus produced by creating predictive models for the groups in the tail set, while the items in the head set have each a unique predictive model. Pham et al. [20], in turn, proposed a hierarchical clustering of users based on their social information found in a network topology.

Related works also use the degree of appreciation of users towards items features as alternatives to build clustering solutions for recommenders. As stated before, Ganu et al. [9] performed a soft clustering of users based on features extracted from their reviews. Wang and Chen [23] also used reviews to derive information about users, which is used in CF and clustering techniques. Liu et al. [14], in turn, use clustering solutions in a multi-criteria recommender systems. The main notion is that users have their preferences defined by different criteria, such as price, quality, location. Users with similar criteria preferences are thus clustered, and recommendation is based on those groups.

Our approach differs from the aforementioned since it explores different item representations derived from reviews, ratings and genres to cluster items in a soft-partitioned scenario. These partitions are applied in a recommender algorithm that makes direct use of the pertinence degrees of the items in each of the clusters.

3. ITEM REPRESENTATIONS

In this study, we consider four items' representations that describe the items in different senses. Two of them are obtained directly from the database, being ratings and item genres used as features. The other two where obtained from outer sources of user reviews, being terms and aspects used as features. Figure 1 gives examples of these representations, where $u_i \in U$ represent the users that gave the rating, $g_i \in G$ are the genres that an item may or may not contain, $t_i \in T$ are the terms and $a_i \in A$ are the aspects used as features that may or may not have a sentiment value related to them.

The following subsections detail better the four representations we address in this study.

3.1 Traditional Representations

The first kind of representation consists solely in the ratings obtained in the database. Each item is represented by a vector where each position consists of a user. The score of each position is the rating that a specific user provided to that item. The scores are discrete numbers that range from 1 to 5, and if a user u did not provide a rating for the item i, $(i, u) = 0$. Thus, each item's vector represent which users evaluated it and, for those that indeed evaluated it, which ratings they assigned.

The second representation uses the items' genres available in our database. Since in this study we are dealing with a movie recommendation scenario, each position represents a movie genre, such as "action", "suspense" and "drama". The scores are binary, and reflect whether an item has or not determined genre.

The other two representations were constructed by con-

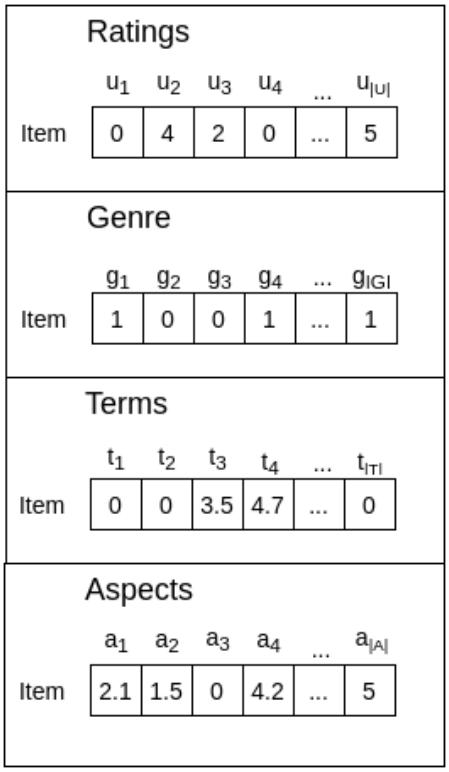


Figure 1: Examples of the Item Representations.

sidering user reviews found on the Web and are described in the following subsection.

3.2 Review-based Representations

The main goal of these representations is to capture the overall sentiment (good, bad or neutral) of many reviewers towards different characteristics of the movies. This is done by following the premise that users analyze reviews from other users to decide whether or not to consume certain product. When reading the reviews, a user can verify whether the item meets his/her expectations in certain aspects, analyzing if the majority of the reviewers appreciates or not the features that he/she thinks interesting. Thus, we encapsulate into one representation the average sentiment of other users towards several aspects of the items, obtained in their texts.

In this study, we consider two different feature granularities: terms and aspects; which were proposed in previous works [6, 7]. Terms are words that represent actual characteristics of an item [16], while aspects are collections of terms that together represent a generic concept [2]. For instance, suppose that among the terms extracted from a review, there are the words **actor**, **star**, and **artist**. While in the term-based representation they would be treated as separate features, in the aspect-based representation they may be aggregated into a single feature, namely **ACTOR**.

Regardless of the granularity of the representation, one may need to pre-process the texts in order to obtain candidate words (tokens) that may constitute terms or aspects. We pre-process the whole reviews set (gathered from the Web) with the well-known Stanford CoreNLP[1] [17], a natural language processing toolkit that contains several NLP routines. There, we perform routines such as tokenization, lemmatization, stemming, part-of-speech (POS) tagging and sentence splitting. The sentiment analysis algorithm is also executed in this toolkit, and a brief description of it can be seen in Section 3.2.3.

Having the texts pre-processed, we apply either the term or the aspect extraction techniques, obtaining a set of features that will constitute the item representation. We detail them in the following subsections, and later present the scoring method for the representations.

3.2.1 Term Extraction

The term extraction technique involves the application of two filters in the set of lemmatized words: one linguistic and another statistical.

First, we select only words with the nouns POS tag as candidate terms. One of the problems with part-of-speech taggers is that unknown words tend to be classified as nouns. This problem is aggravated when using texts produced by users, due to misspellings, Internet slangs and abbreviations.

Therefore, we select from the set of candidate words those that are more common among the item reviews, assuming that these may be, in fact, features. Since an item has n reviews, instead of using the document frequency (DF) [16], we decided to use a similar metric called item frequency (IF) [6, 7]. Considering F as the candidate words set and I the items set, the item frequency IF_f of a candidate word f is given by

$$IF_f = \sum_{i}^{|I|} k_{if}, \tag{1}$$

where k_{if} is equal to 1 if an item i has the candidate word in at least one of its reviews. The IF_f is then compared to a threshold, and if its value is greater than it, the candidate word is maintained in the term set. In an earlier experiment, we considered four different thresholds for constructing lists of terms: 1, 30, 100 and 200 [6]. The results indicated that the threshold of 30 is a good value since it produces a shorter set of terms with interesting results, performing better than the baseline.

3.2.2 Aspect Extraction

In this approach, we apply a set of heuristics, similarly to those applied in the term extraction, to reduce the number of candidate words and create aspects from the reviews. We apply in the set of lemmatized word both linguistic and statistical filters applied in the first term extraction technique: first we select only nouns and then apply the item frequency, discarding the candidate words that have an IF_f value lower than a certain threshold.

From the set of remaining terms, we aggregate those that contain the same or similar stem. For example, when performing the pre-processing step, we may obtain the lemmas "director" and "direction", but both share the same stem "direct". By joining the lemmas that share the same stem, we often reduce the number of features that have relation to the same topic.

The last step, performed semi-automatically, is to aggre-

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

gate synonymous topics. We use a lexicon as a basis to obtain the synonyms of the lemmas for each existing topic, and group those topics who share the synonyms. After performing this step, we make a manual check to remove errors and noise.

We explored this technique in a previous experiment [7]. The results showed that the produced aspect set was very small, which affected the capability of the Item k-NN recommender to distinguish the items and hence to locate appropriate neighbors to produce adequate suggestions.

3.2.3 Building the Representations

Regardless of the method, the resulting feature set will constitute the items' representations dimensionality. In the next step, the sentiment value for each of those features is computed. Thus, an item is represented by the average sentiment of many users' reviews towards each of its characteristics.

In order to do that, first we apply a sentiment analysis algorithm in the item's reviews, obtaining the sentiment for each sentence. The main reason for using a sentence-level sentiment analysis is that most of the features extracted from the reviews is nouns, specially in a movie recommendation domain. Nouns have neutral sentiment, hence we rely on the context and sentiment existing in sentences containing these nouns.

We use the sentiment analysis algorithm available in the Stanford CoreNLP toolkit [22] to obtain the sentiment of all reviews' sentences. In this approach, recursive neural networks models are used to build representations that capture the structure of the sentences, obtaining in this way their sentiment based on the meaning of each of words. The Stanford CoreNLP sentiment analysis tool classifies sentences in five sentiment levels: "Very Negative", "Negative", "Neutral", "Positive" and "Very Positive". We convert this classification into a $[1, 5]$ rating system, being 1 equals to "Very Negative" and 5 equals to "Very Positive".

Next, the system analyzes the feature set and checks if they are terms or aspects. If they are terms, the system finds and stores which sentences are related to them. If they are aspects, the system finds the set of terms that each aspect represents, and then checks which sentences contain them.

After obtaining the sentences related to each feature, the next step is the sentiment attribution to them. For each feature of each item it is calculated the average sentiment of the related sentences. Thus, the final value represents the collective level of appreciation or depreciation of certain attribute of an item. A zero value indicates that an item simply does not have that feature.

4. RECOMMENDATION

With the representations at hand, each of them will describe the items in a specific manner. The goal of this paper is to analyze which of these descriptions can organize better the items into clusters, which will in turn serve as single weights to contribute to the recommendation, thus reducing the computational cost. As mentioned before, hard partitioned clusters are not a good solution, since the nature of the representations describe items as belonging to different cluster with varied degrees of persistence.

Thus, we apply them into a soft clustering algorithm which will produce partitions with overlapping clusters. These clusters, as well as the probabilities that each item is contained in those clusters, are given to a recommender algorithm which will use this information to predict ratings for unknown user x item pairs.

For this work, we adapted a recommender algorithm proposed by Ganu et al. [9], which uses the premise of soft clusters of users in its calculation. The changes were made in order to the algorithm perform its calculations with a soft clustering of items. A brief description of the original algorithm follows, and then we detail the changes we made in order to produce a soft clustering of items.

4.1 Original Algorithm

The main idea of the original algorithm is that each user has a value that describes the probability, or the degree of relevance, that it is contained in a cluster. The predicted rating for an unknown pair (u, i) will be an average of all ratings from other users who evaluated the item in question, weighted by the degree of relevance of them.

In order to obtain the clusters, the authors used the Iterative Information Bottleneck (IIB) in users' profiles that were produced based on the sentiment analysis of several aspects of reviews produced by themselves.

Once the groups and the degree of relevance of each user are defined, the recommendation algorithm uses this information to predict the rating of an unknown user-item pair. Considering that the set of users who have rated an item i is called R_v^i, and that each user has a relevance degree for the cluster c_k, denoted as $u(c_k)$, the contribution score $Contr(c_k, i)$ of each cluster c_k given a fixed item i is calculated by the following equation:

$$Contr(c_k, i) = \frac{\sum_{v \in R_v^i} v(c_k) * r_{vi}}{\sum_{v \in R_v^i} v(c_k)}, \tag{2}$$

where r_{vi} is the rating that user v gave to the item i. Thus, the contribution of each cluster is an average of the scores given by other users to the item, weighted by their pertinence in the cluster.

The final rating $\hat{r}_{ui}$ is predicted by calculating the average contribution weighted by the relevance degree of the user u in relation to the n existing cluster:

$$\hat{r}_{ui} = \frac{\sum_{k=1}^{n} u(c_k) * Contr(c_k, i)}{\sum_{k=1}^{n} u(c_k)} \tag{3}$$

4.2 Item-based Soft Clustering Recommender

We adapted the previously detailed algorithm to compute the predicted ratings from soft clusters of items. The clustering solutions are produced from the items' representations described in the previous section. With them, the algorithm tries to predict a rating in a similar fashion of the original algorithm, but instead of computing contribution from clusters of users, it computes from clusters of items. In the following subsections, we detail each change we perform in the original algorithm.

4.2.1 Soft Clustering Algorithms

The first difference worth noting is that while the original work uses the IIB algorithm, we decided to test two other soft clustering algorithms: one based in fuzzy clustering (fuzzy c-means, or FCM) and one based on probabilis-

tic clustering (expectation-maximization, or EM), since they are well-known and successful soft clustering algorithms.

The FCM is a fuzzy version of the well known k-means algorithm, by allowing the overlapping of clusters and setting degrees of cluster persistence to the data. As well as the k-means, it attempts to minimize the intra-cluster variance:

$$\min_{\mu_{ij},v_i} J = \sum_{j=1}^{N}\sum_{i=1}^{c} \mu_{ij}^{m} \parallel x_j - v_i \parallel^2, \tag{4}$$

where x_j is an object, v_i is a cluster centroid, μ_{ij} is the degree of pertinence of an object j to a cluster i with a value between 0 and 1, $\sum_{i=1}^{c} \mu_{ij} = 1$ and $m > 1$ is a fuzzification value.

The EM algorithm attempts to probabilistically model the data through Gaussian mixtures. In this sense, the clusters are Gaussians distributed into the data space, and the objects have probabilities of belonging to those Gaussian. The algorithm maximizes the log-likelihood function:

$$ln(p(X\|\pi,\Sigma,v)) = \sum_{j=1}^{N} ln(\sum_{l=1}^{k} \pi_l N(x_j\|v_l,\Sigma_l)), \tag{5}$$

where π_l is the a priori probability that a random object was generated by a Gaussian l, $N(x_j\|v_l,\Sigma_l)$ and Σ_l is the weighted covariance matrix of the Gaussian l.

4.2.2 Recommendation algorithm

Regarding the recommendation algorithm, we perform a conversion similar to the one done from the User k-NN algorithm into the Item k-NN. In this sense, the contribution of a cluster c_k is calculated by considering the score and relevance degree of all other items rated by the user. We also aggregate into the calculation the baseline estimates b_{ui}, which represent user and item biases and are commonly used in neighborhood and factorization models [12]. A baseline estimate for an unknown rating $\hat{r}_{ui}$ is denoted by:

$$b_{ui} = \mu + b_u + b_i, \tag{6}$$

where μ is the global average rating, b_i and b_u are the item's and user's deviations from the average. To estimate b_u and b_i one can solve a least squares problem. We adopted a simpler approach which will iterate a number of times the following equations:

$$b_i = \frac{\sum_{u:(u,i)\in K}(r_{ui} - \mu - b_u)}{\lambda_1 + |\{u|(u,i)\in K\}|}, \tag{7}$$

$$b_u = \frac{\sum_{i:(u,i)\in K}(r_{ui} - \mu - b_i)}{\lambda_2 + |\{i|(u,i)\in K\}|}, \tag{8}$$

where K is the set of rated items and r_{ui} is a rating given by a user u to an item i.

Given those changes, the algorithm works as follows. Considering R_j^u as the set of items that a user u evaluated, and $i(c_k)$ as the pertinence degree of an item in a cluster, we rewrite the contribution score $Contr(c_k,u)$ of a cluster c_k given a fixed user u as:

$$Contr(c_k,u) = \frac{\sum_{j\in R_j^u} j(c_k) * (r_{uj} - b_{uj})}{\sum_{j\in R_j^u} j(c_k)}. \tag{9}$$

With this, the contribution of a cluster is the average of the variation between the ratings of the other items evaluated by the user and their baseline estimate, weighted by their pertinence degree on the cluster. The final rating $\hat{r}_{ui}$ is given as:

$$\hat{r}_{ui} = b_{ui} - \frac{\sum_{k=1}^{n} i(c_k) * Contr(c_k,u)}{\sum_{k=1}^{n} i(c_k)}, \tag{10}$$

which is its baseline estimate adjusted by the average of the clusters contributions weighted by the pertinence degree of the item.

5. EXPERIMENTAL EVALUATION

In the following subsections, we detail the experiments conducted. We first present the dataset used, then we detail our experimental setting, and finally present the results obtained.

5.1 Dataset

We performed our experiments on a database related to movies, generated from the MovieLens Web site[2] and enhanced with information contained in the IMDb Web site[3].

For this study, we used the well-known MovieLens 100k (ML-100k) database. The ML-100k consists of 100,000 ratings (from 1 to 5) performed by 943 users for 1,682 movies. It also categorizes the movies into genres and provides users demographics.

We also collected up to 10 reviews per item for the ML-100k database, resulting in a total of 15,863 documents. Unfortunately, not every movie had the maximum number of reviews, in fact, there were some movies that did not have reviews at all.

5.2 Experimental Setting

Regarding the size of the representations, we have obtained 3,085 terms by selecting the IF_f threshold of 30, accordingly to our findings in earlier experiments [6]. Each item contains sentiment in an average of 223.32 terms. As for the aspect-based representation, it contains a set of 78 features, with each item having sentiment in an average of 22.59 aspects. Regarding the other representations, the genre-based contains 18 features, with each item being assigned on average to 1.72 genres; while the rating-based contains 943 features, which is the number of users in the database, and each item was rated by an average of 59.45 users.

Regarding the clustering methods, we selected as 3, 5 and 7 the number of clusters and run each of them 10 times for each representation, selecting the partitions that had the smallest intra-cluster variation for FCM and biggest log-likelihood function for EM.

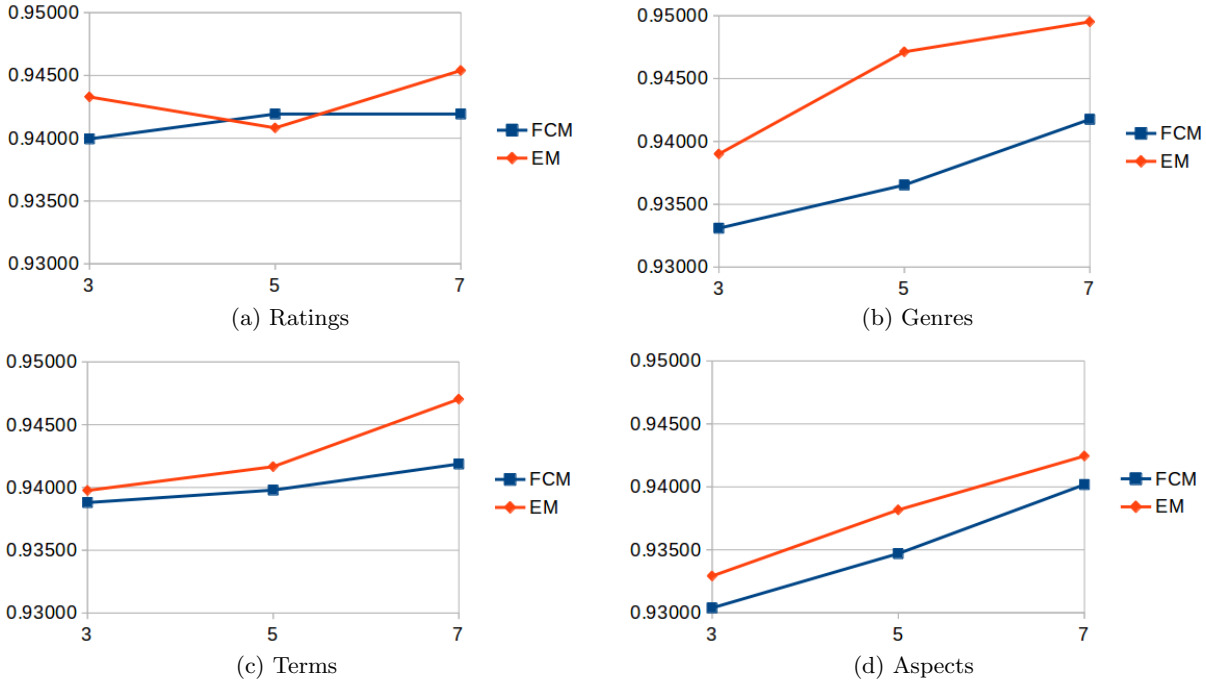
Regarding the recommendation algorithm, we learned the baseline estimates by iterating 10 times the Equations 7 and 8, with $\lambda_1 = 10$ and $\lambda_2 = 15$, as suggested by the literature [12].

We evaluate our approach in the rating prediction scenario, using the root mean square error (RMSE) between a predicted rating and its real value in the test set, which is

[2]http://movielens.umn.edu
[3]http://www.imdb.com

Table 1: RMSE results obtained by applying the clustering methods into the item representations.

Representations	Number of Clusters					
	3		5		7	
	FCM	EM	FCM	EM	FCM	EM
Ratings	**0,93995**	0,94329	0,94193	0,94082	0,94193	0,94539
Genres	**0,93310**	0,93905	0,93654	0,94713	0,94176	0,94952
Terms	**0,93883**	0,93976	0,93979	0,94166	0,94187	0,94703
Aspects	**0,93039**	0,93292	0,93470	0,93817	0,94018	0,94245



(a) Ratings (b) Genres

(c) Terms (d) Aspects

Figure 2: Graphics comparing the results obtained by FCM and EM clustering for each representation.

defined as:

$$\text{RMSE} = \frac{1}{|U|} \sum_{u \in U} \sqrt{\frac{1}{|O_u|} \sum_{i \in O_u} (\hat{r}_{ui} - r_{ui})^2}, \qquad (11)$$

where O_u is the predicted items set that the user u evaluated. By evaluating in this scenario, we aim to minimize the error between the predicted ratings and those on the test set, thus, lower RMSE values represent better results.

All experiments were carried out in a 10-fold cross validation setting, and the values displayed are the average results of the folds. In order to check the significance of the results, we applied the Student's t test [18].

5.3 Results

Table 1 and Figure 2 presents the results obtained by executing the recommendation algorithm with all previously detailed configurations. Values in bold indicate the best configuration results for each representation.

As it can be seen, the best results were obtained using the partitions with fewer clusters. By analyzing the FCM partitions, we have observed that as we raise the number of clusters, the algorithm tends to assign similar relevance degrees in every cluster for each item. As for the EM algorithm, the behaviour is the opposite: it tends to assign a high probability to only one cluster, while the others remain with low probabilities. We believe that this happens due to the nature of the representations. Since all of them are quite sparse, as we try to segregate more the data by raising the number of clusters, the centroids of the FCM tend to move to the center of the data space, encompassing the whole data with similar probabilities, while the EM tend to describe the data by producing Gaussians that rarely overlap, giving clustering solutions that are nearly hard partitioned.

Another thing worth noting is that FCM provided better results for every kind of items' representation in comparison with EM (with p-value < 0.01 in all cases). This leads us to conclude that FCM handles better the situations where the data are not well behaved, i.e., the objects in the vector space do not form natural clusters. This can be backed up by the sparse nature of our data, where there are not small and separated natural clusters.

We compare our best results (FCM with 3 clusters) with those produced by executing the same representations into the item k-NN. We have trained the baseline estimates with the same configurations previously reported, set $k = 40$ and used the Pearson correlation to find the neighbours. Table 2 and Figure 3 present the comparison.

Table 2: Comparison of the best results with item k-NN results.

Representations	Recommender	
	FCM	Item k-NN
Ratings	0,93995	**0,93583**
Genres	**0,93310**	0,94018
Terms	0,93883	**0,93135**
Aspects	**0,93039**	0,94262

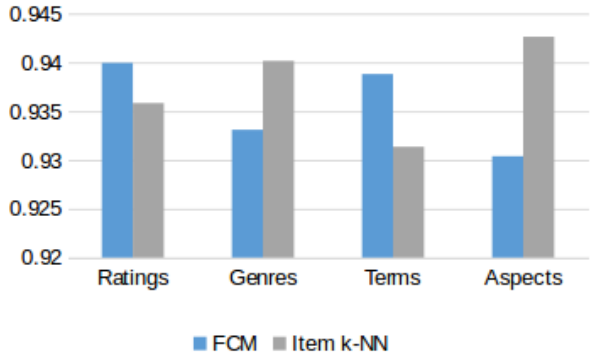


Figure 3: Chart comparing the best results (3 clusters) with item k-NN results.

By comparing it with item k-NN, one can see that the soft clustering recommendation provided significantly better results (p-value < 0.005) for two of the four representations: genres and aspects. Both representations use a smaller set of attributes, which can lead us to the conclusion that the clustering algorithms considered in this study behaved better with fewer features, providing more accurate groups. We argue that it happens mainly because of the nature of the distance metric used in those algorithms. Since FCM uses Euclidean and EM uses Mahalanobis, and both distance metrics consider zero values in their calculations, they tend to perform poorly in matrices with high dimensionality and sparsity.

The opposite can be seen in the item k-NN algorithm, which seems to perform better using representations with high dimensionality. The main drawback of this is that the system needs more computational resources and time to perform the recommendations, since there are larger matrices to process. This scenario tends to get worse as the database grows.

Nonetheless, the result obtained by using the aspect-based representation with FCM clustering is statistically superior (p-value < 0.01) to the results obtained by the item k-NN with rating and term-based representations. This shows that, even though the soft clustering recommender provided better results in two out of four representations, it still yields the best result with a significantly smaller set of features.

Another indication of the strength of this algorithm is that it in overall performs faster, as it can be seen in Table 3 and Figure 4[4]. Even though the similarity, as well as the partitions, can be computed offline and be given to the recommender itself which operates online, the recommendation part by itself performs faster in the soft clustering-based algorithm than in the item k-NN. The main reason for that difference is that the item k-NN requires an additional step before computing the predictions to find the k nearest neighbors, which involves sorting items by similarity.

[4]Experiments were executed in a Linux Mint 17 Cinnamon 64-bits, with Intel Core i7-4930K CPU @ 3.40GHz x 6 and 16gb of RAM.

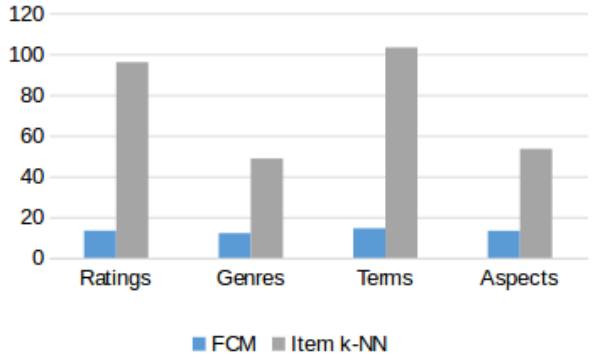


Figure 4: Chart comparing the overall execution time of FCM with 3 clusters and item k-NN, in seconds.

6. CONCLUSIONS

In this paper, we analyzed the application of four different items' representations in a recommender system that uses partitions produced by soft clustering solutions. This algorithm was adapted from the work of Ganu et al. [9], and we tested two classic algorithms of clustering with partitions superposition. Experimentation indicates that the algorithm provides, in some cases, better results than the well-known item k-NN algorithm, making the choice of the kind of representation crucial for the system. Nevertheless, the algorithm provides the best results when considering a smaller set of features (the aspects representation), outperforming even the best results produced by the item k-NN. This proves to be advantageous since it requires fewer computational resources and time. Another advantage is that the soft clustering recommendation algorithm itself performs faster than the item k-NN.

As future work we plan on testing partitions produced by other state-of-the-art soft clustering algorithms into the recommender system. We also plan to apply the proposed system in other data domains, such as music, products and turism.

7. ACKNOWLEDGMENTS

The authors would like to thank the financial support from CAPES and FAPESP.

8. 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. *IEEE Transactions on Knowledge and Data Engineering*, 17(6):734–749, 2005.

[2] C. C. Aggarwal and C. X. Zhai. *Mining Text Data*. Springer Publishing Company, Incorporated, 2012.

Table 3: Execution time comparison between FCM with 3 clusters and item k-NN, in seconds.

Representations		Similarity / Partitioning	Recommendation	Overall
Ratings	FCM	1.390	11.903	**13.293**
	Item k-NN	46.420	49.525	95.945
Genres	FCM	0.0476	12.039	**12.086**
	Item k-NN	0.507	48.232	48.739
Terms	FCM	2.603	11.890	**14.493**
	Item k-NN	54.744	48.453	103.197
Aspects	FCM	0.128	13.093	**13.221**
	Item k-NN	3.897	49.509	53.406

[3] J. C. Bezdek. *Pattern Recognition with Fuzzy Objective Function Algorithms*. Kluwer Academic Publishers, Norwell, MA, USA, 1981.

[4] C. M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[5] R. D'Addio, M. Conrado, S. Resende, and M. Manzato. Generating recommendations based on robust term extraction from users' reviews. In *Proceedings of the 20th Brazilian Symposium on Multimedia and the Web (WebMedia '14)*, pages 55–58, 2014.

[6] R. M. D'Addio and M. G. Manzato. A collaborative filtering approach based on user's reviews. In *2014 Brazilian Conference on Intelligent Systems (BRACIS '14)*, pages 204–209, 2014.

[7] R. M. D'Addio and M. G. Manzato. A sentiment-based item description approach for knn collaborative filtering. In *Proceedings of the 30th ACM/SIGAPP Symposium On Applied Computing (SAC '15)*, pages 1060–1065, 2015.

[8] C. Desrosiers and G. Karypis. A comprehensive survey of neighborhood-based recommendation methods. In F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, editors, *Recommender Systems Handbook*, pages 107–144. Springer US, 2011.

[9] G. Ganu, Y. Kakodkar, and A. Marian. Improving the quality of predictions using textual information in online user reviews. *Information Systems*, 38(1):1–15, Mar. 2013.

[10] S. Gong. A collaborative filtering recommendation algorithm based on user clustering and item clustering. *Journal of Software*, 5(7):745–752, 2010.

[11] H. Kim, K. Han, M. Yi, J. Cho, and J. Hong. Moviemine: personalized movie content search by utilizing user comments. *IEEE Transactions on Consumer Electronics*, 58(4):1416–1424, 2012.

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

[13] Y. Koren and R. Bell. Advances in collaborative filtering. In F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, editors, *Recommender Systems Handbook*, pages 145–186. Springer US, 2011.

[14] L. Liu, N. Mehandjiev, and D.-L. Xu. Multi-criteria service recommendation based on user criteria preferences. In *Proceedings of the Fifth ACM Conference on Recommender Systems*, RecSys '11, pages 77–84. ACM, 2011.

[15] P. Lops, M. de Gemmis, and G. Semeraro. Content-based recommender systems: State of the art and trends. In F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, editors, *Recommender Systems Handbook*, pages 73–105. Springer US, 2011.

[16] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA, 2008.

[17] C. D. Manning, M. Surdeanu, J. Bauer, J. Finkel, S. J. Bethard, and D. McClosky. The stanford corenlp natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60, 2014.

[18] T. M. Mitchell. *Machine Learning*. McGraw-Hill, Inc., 1997.

[19] Y.-J. Park and A. Tuzhilin. The long tail of recommender systems and how to leverage it. In *Proceedings of the 2008 ACM Conference on Recommender Systems*, RecSys '08, pages 11–18, New York, NY, USA, 2008. ACM.

[20] M. C. Pham, Y. Cao, R. Klamma, and M. Jarke. A clustering approach for collaborative filtering recommendation using social network analysis. *Journal of Universal Computer Science*, 17(4):583–604, 2011.

[21] R. Qumsiyeh and Y.-K. Ng. Predicting the ratings of multimedia items for making personalized recommendations. In *Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '12)*, pages 475–484, 2012.

[22] R. Socher, A. Perelygin, J. Wu, J. Chuang, C. D. Manning, A. Y. Ng, and C. Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP '13)*, pages 1631–1642, October 2013.

[23] F. Wang and L. Chen. Recommending inexperienced products via learning from consumer reviews. In *2012 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology (WI-IAT '12)*, pages 596–603, 2012.

2.2 Final Remarks

In this study, we found some insights related to Research Questions 2 and 3, i.e. "Does increasing the semantics of item representations improve recommendation accuracy?" and "Is a kind of item representation more relevant to use in determined recommender system or scenario than another?". We compare four item representations in a soft clustering-based recommendation algorithm against their application in a neighborhood-based recommendation algorithm. Two of them were based solely on dataset structured information, i.e., user ratings and item genre; while the other two were based on user reviews, i.e., terms and aspects.

Related to Question 3, one can see that indeed the representations can impact differently the recommender system in which they are used. Representations that had smaller dimensionality performed better in the soft-clustering recommender, and this is due to the fact that the clustering algorithms take into account zeroes in the representations' vectors to calculate their distances. Meanwhile, representations with higher dimensions perform better in the item-k-NN algorithm, since they are able to present more detailed information to accurately differentiate each item and thus produce better neighborhoods.

Related to Question 2, by analysing the pairs of high and low dimensionality item representations (i.e. aspect vs. genre and terms vs. ratings), one can see that in most cases representations with a higher level of semantics (i.e. those based on user reviews and sentiment analysis) performed better. The only case where this was not true was comparing aspects and genres in k-NN. Nevertheless, this is a strong indication that semantics play an important role on designing content-based recommender systems.

Even though we do not fully answer the aforementioned research questions, and we have indeed achieved interesting results, some issues arise from this experiment. The most prominent one is mostly related to Question 2 and the semantics that review-based item representations attain. While those representations have a significant semantic load due to sentiment analysis, their feature extraction methods are bound to suffer linguistic problems such as synonymy, where multiple terms define the same concept, and polysemy, where a term may have different meanings, thus referencing multiple concepts. Moreover, the method disregards n-grams (i.e., terms with more than one word) and named entities, such as famous people, places, brands, products, and so on. Another issue was that the experiment was performed on a relatively small dataset, thus requiring further experimentation on other datasets and/or domains.

In the next chapters, we continue to attempt to answer our research questions by investigating techniques to increase semantics in item representations and address the aforementioned problems.

CHAPTER

3

SEMANTIC ORGANIZATION OF USER'S REVIEWS APPLIED IN RECOMMENDER SYSTEMS

3.1 Contextualization

Taking into consideration the main drawback presented in last chapter, i.e., the linguistic problems present in our current feature extraction method, we proposed a new method which relies on state-of-the-art natural language processing tools.

This method takes the advantage of using Babelfy (MORO; RAGANATO; NAVIGLI, 2014), a disambiguation and entity linking tool which would not return terms as features, but concepts and named entities. Concepts can be seen as linguistic units that describe an idea and are composed by synonym words (synsets). Named Entities are a special kind of concept that describes things that have a proper name, such as famous people, places, organizations, movies, music artists and so on. Babelfy uses the BabelNet (NAVIGLI; PONZETTO, 2012) knowledge graph, which is a vast multilingual natural language resource that unifies several knowledge bases and sources such as WordNet, Wikipedia, Wikidata, Wikitionary and others.

By applying Babelfy into user reviews, the texts are disambiguated and the item features that we would extract from them become BabelNet concepts (synsets). With that, the issues of polysemy and synonymy are reduced. We applied this new set of features weighted by SF-IDF (CAPELLE *et al.*, 2012), which is an extension to the well known TF-IDF weighting scheme (MANNING; RAGHAVAN; SCHüTZE, 2008) where synsets are used instead of terms. We wrote a short paper describing our findings (MARINHO; D'ADDIO; MANZATO, 2017), which can be read in the following. The original paper is written in Brazilian Portuguese, but for the sake of consistency we provide an English translated version. The original version of the article can be seen in Appendix A.

Semantic organization of user's reviews applied in recommender systems

Ronnie S. Marinho, Rafael M. D'Addio and Marcelo G. Manzato
Institute of Mathematics and Computer Science
University of São Paulo
São Carlos, São Paulo - Brazil
ronnieshida@usp.br,{rdaddio,mmanzato}@icmc.usp.br

ABSTRACT

Recommender systems are widely used to minimize the information overload problem. A great source of information is users' reviews, since they provide both item descriptions and users' opinions. Recent works that process reviews often neglect problems such as polysemy and sinonimy. On the other hand, systems that rely on word sense disambiguation focus their efforts on items's static descriptions. In this paper, we propose a hybrid recommender system that uses word sense disambiguation and entity linking to produce concept-based item representations extracted from users' reviews. Our findings suggest that adding such semantics to items' representations have a positive impact on recommendations.

CCS CONCEPTS

• **Information systems → Content analysis and feature selection**; **Personalization**;

KEYWORDS

Recommender systems, item representation, word sense disambiguation

PUBLICATION INFORMATION

Published in proceedings of the 23rd Brazilian Symposium on Multimedia and the Web (Webmedia '17). 2017. Pages 271–278. DOI:10.1145/2976796.2976858

1 INTRODUCTION

Due to the Internet growth and the wide range of information available on the Web, recommender systems (RS) have emerged. Such systems assist users in the search for items of interest.

In general, RS can be classified into two important paradigms: i) content-based filtering, where items are recommended to a user based on the similarity of their content to the user's profile; and ii) collaborative filtering, where recommendations are inferred from relationships created through user interactions with the system [1]. Hybrid methods have gained prominence for combining aspects of both filters, minimizing their problems [1].

A recurring problem in these systems is the lack of information to describe the items in the collection and the preferences of its users. Recently, studies have studied the possibility of using user reviews [4–6, 8, 12, 18], however, such texts present several problems, such as noise and misspelled words. Although there are efforts related to the correct extraction of information [5–7], such systems suffer the aggravating factor called word sense ambiguity. Many words extracted from the texts can have different meanings and contexts (polysemy), in the same way that several words can refer to the same concept and be treated as different characteristics (synonymy).

Several efforts have been made to add techniques that perform the concept extraction instead of term extraction in recommender systems [2, 10, 14, 15]. However, most of these works use texts of a static nature as information sources, such as synopses and Wikipedia pages, neglecting the source of detailed information in user annotations. In particular, user reviews can provide, at the same time, a detailed description of the item and the users' opinions about the product, however, its use depends on the correct processing of information.

Thus, this work proposes a system that extracts concepts from user reviews to build item representations. For this, we used a tool that performs both word sense disambiguation (WSD) and entity linking (EL) called BabelFy [1] [13], which extracts concepts in the form of synsets from BabelNet[2] [16] knowledge base. Such synsets make up the item representations, which are processed by a hybrid recommender algorithm. Results show the efficiency of this approach, which surpasses the approaches based on terms with which this work is compared.

This article is organized in the following way: Section 2 lists works that follow the same line of research as this project; Section 3 presents the architecture of the proposed system; Section 4 presents the experiments carried out; finally, Section 5 presents conclusions and suggestions for future work.

2 RELATED WORKS

Several efforts have been made in order to use unstructured user annotations in recommendation systems. Chen et al. [3] conducted a survey related to recommender systems that explore user reviews.

Some recent works use reviews to characterize the items in the system, producing item representations [5, 6] or building relational graphs [12]. Other works also make use of reviews to describe users, building profiles that explain their preferences in relation to items features [8], or using their texts to find similarities between them [18]. Finally, some works describe both user and items [4], making use of reviews to rank the features that are most relevant to the user's preference. The biggest limitation of these approaches is that none of them deals with problems of ambiguity in the texts, such as polysemy and synonymy.

[1] http://babelfy.org/
[2] http://babelnet.org/

The WordNet[3] taxonomy and WSD algorithms based on it have been actively explored in the last decade by content-based recommender systems [2, 10]. On the other hand, some recommender works use EL techniques to extract features [14]. Finally, efforts have been made to incorporate techniques that do both in content-based RS, facilitating the creation of item representations [15, 17].

This work differs from the others presented in this section in the following points. First, works that use WSD and/or EL are based on pre-defined descriptions, such as synopses or Wikipedia pages. On the other hand, works that use user reviews do not extract concepts, focusing only on the use of keywords. The work presented here, in turn, focuses on extracting concepts discussed by users in their reviews. Second, the research presented is focused on content-based recommendations. This work, in turn, produces item descriptions that will be used in a hybrid system based on item neighborhood.

3 PROPOSED SYSTEM

As previously mentioned, the system proposed in this work uses WSD and EL in order to extract concepts from the items. From this, it is possible to characterize the items more efficiently, favoring the recommender's calculation. Thus, we developed modules that perform specific functions, as illustrated in Figure 1. In general, user reviews are used to feed the concept extraction module, which produces a list, or vocabulary, that will be used by the item representation construction module. Having produced the representations, they will be processed by the recommendation module, which makes use of an algorithm based on item neighborhoods to generate suggestions for users. Such modules will be further detailed in the following subsections.

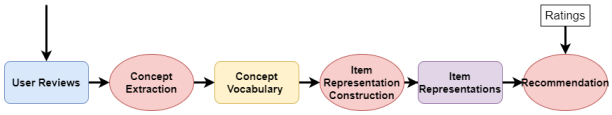


Figure 1: General architecture of the proposed system.

3.1 Concept Extraction

The concept extraction module is responsible for carrying out various tasks aimed at structuring user reviews. In this module, with the aid of Babelfy, the following tasks are performed:

- Noise Removal: words with little semantic load are removed, that is, the *stop-words*, in addition to words with special characters, dates and numeric characters.
- Word Sense Disambiguation and Entity Linking: identification of named concepts and entities is carried out. Thus, a BabelNet synset is assigned for each disambiguated term. In addition, the grammatical class of each synset is obtained.
- Morphosyntactic filtering: only noun synsets are selected, due to the fact that item characteristics in most domains are nouns.

The process described above is applied to all item reviews, producing a set of candidate synsets. As the set generated is very large, a feature filtering is carried out, removing those that are infrequent. As an item has several reviews, the item frequency (IF) metric was used [6]. The IF metric is an adaptation from the traditional document frequency (DF) metric [11], but we consider every item review as a single document.

After calculating the IF of each synset, the filter consists of removing those whose IF value is below a certain threshold. In this work, several threshold values were tested, and their results in relation to the final recommendation are reported in Section 4.2.1. The result of this filtering produces a set of synsets, the vocabulary, which will be used in the representation construction module, detailed below.

3.2 Representation construction

This module is responsible for using the synset vocabulary to build item representations. Such representations are modeled in a vector space, where each synset corresponds to a dimension of that model, and each vector represents an item.

As well as [2], this work adopts the SF-IDF technique as a vector weighing scheme, an extension of the TF-IDF [11] technique, applied to synsets instead of terms.

This technique analyzes how frequent a synset is in an item (SF) and how rare it is among all items (IDF). Formally, assume that $|I|$ is the total number of items in the system, and that a synset s appears in $|i : t \in i|$ items. Also assume that $n_{s,i}$ is the synset s frequency in item i's reviews. The synset frequency $SF(s, i)$ can be defined as [2]:

$$SF(s, i) = \frac{n_{s,i}}{\sum_k n_{k,i}}, \tag{1}$$

where $\sum_k n_{k,i}$ corresponds to the total frequency of the other k synsets in the item i's reviews.

The inverse document frequency, IDF_i, is calculated as:

$$IDF(s) = \log \frac{|I|}{|i : t \in i|}. \tag{2}$$

Finally, the SF-IDF of a synset s in an item i can be calculated as:

$$SF - IDF(s, i) = SF(s, i) \times IDF(s). \tag{3}$$

3.3 Recommendation

As a recommendation algorithm, we opted to use an item-based neighborhood method (item k-NN) [9], and it was adjusted to use the previously generated item representations in the process of obtaining neighbors.

The item k-NN algorithm aims to predict ratings of items unknown by the user, based on the ratings of other similar items already evaluated by him. For this, a measure of similarity between item representations is used. The similarity measure chosen is Pearson's correlation coefficient s_{ij} [9].

With similarity values, the algorithm identifies the k items rated by u that are most similar to i, that is, the k nearest neighbors. This set is denoted as $S^k(i; u)$. Using this set, the final predicted rating is an average of the ratings of the most similar k items, adjusted to their baseline estimates[9]:

$$\hat{r}_{ui} = b_{ui} \quad \frac{\sum_{j \in S^k(i;u)} s_{ij}(r_{uj} - b_{uj})}{\sum_{j \in S^k(i;u)} s_{ij}}, \tag{4}$$

[3] https://wordnet.princeton.edu/

where b_{ui} represents the baseline estimate, a value that encapsulates the bias of both users and items.

4 METHODOLOGY AND EVALUATION

In this section, the experiments performed to validate this proposal are presented. First, some information about the dataset used is presented, then the configurations of the experiments are described and, finally, the results are presented.

4.1 Configuração dos experimentos

In order to evaluate the proposed system, experiments were carried out on the MovieLens 100k [4] dataset. It has been expanded by adding user reviews of IMDb [5]. On average, the first 10 reviews of each item were collected, ordered by utility, resulting in a total of 15963 documents.

The proposal was compared with other approaches, also based on neighborhood in the rating prediction scenario. It was compared with the following baseline representations:

- *Genres:* this representation uses the metadata of genres of the films, where each position of the vectors refers to a genre that the item contains (value 1) or not (value 0);
- *Heuristic Terms:* this representation, proposed in [6], makes use of heuristics to extract terms from user reviews. This technique selects lemmatized terms with the noun part-of-speech, and also applies the IF (with a threshold of 30) for filtering less frequent terms. Two different weights for the terms are considered here: TF-IDF and, as proposed in the original work, sentiment analysis. The sentiment represents the general opinion of the reviewers with respect to each characteristic of the item. The terminology HT-TFIDF and HT-Sentiment is adopted for each version of the representations.
- *Classified Terms:* in this representation, proposed in [5], a transductive classification algorithm is used to extract stemmed terms, which also have the two weights described in the previous representation. The nomenclature CT-TFIDF and CT-Sentiment is adopted for each version of representations.

For the development, execution and comparison of the representations, the recommendation algorithm described in Section 3.3 was used, with the number of neighbors set to k = (20, 40, 60, 80, 100), and the similarity between the items is calculated by the shrunk Pearson correlation coefficient [9].

The systems were submitted to a 10-fold cross-validation, and the results are evaluated in the rating prediction scenario, whose metric used is the root mean square error (RMSE). Finally, the Wilcoxon test was used to verify whether the results obtained by the approach are statistically different from those obtained by baseline representations.

4.2 Results

To validate the proposal, two experiments were carried out. In the first, different synset vocabulary sizes were analyzed, defined by cuts based on the IF of the characteristics. In the second experiment, the quality of the proposed technique was compared to other baseline representations, described in the previous section. Such experiments are detailed in the following subsections:

4.2.1 Experiment 1: finding an optimal threshold. In this experiment, several thresholds based on the IF were verified in order to find the most appropriate set of synsets, eliminating the features that are less frequent and, consequently, add little or no information to the representations. Thus, the vocabulary extracted by BabelFy was filtered with thresholds defined as l_1= 1, l_2= 10, l_3= 30 e l_4= 50. The representations are called: *i*) Babelfy-threshold 1, *ii*) Babelfy-threshold 10, *iii*) Babelfy-threshold 30, e *iv*) Babelfy-threshold 50, respectively. In addition, the filtered representations are compared with the representation without applying any filtering (Babelfy-full). Table 1 shows the results obtained.

Table 1: RMSE values for representations with different thresholds. Bold values indicate the best results.

Algorithm	Features	k=20	k=40	k=60	k=80	k=100
Babelfy-full	60466	0,9197	0,9200	0,9208	0,9216	0,9224
Babelfy- threshold 1	28182	0,9189	0,9192	**0,9201**	**0,9209**	**0,9217**
Babelfy- threshold 10	6238	**0,9183**	**0,9190**	0,9203	0,9214	0,9224
Babelfy-threshold 30	2747	0,9212	0,9217	0,9230	0,9243	0,9254
Babelfy-threshold 50	1797	0,9225	0,9231	0,9246	0,9260	0,9271

Note that representations with lower thresholds showed better results, that is, the vocabularies produced by l_1 = 1 e l_2 = 10. Both vocabularies have similar results, the first being statistically superior only for k = 100 with p-value < 0.05. In addition, compared to the second, the size of the first vocabulary is significantly larger (29,000 characteristics compared to just over 6000). Thus, preference is given to the use of the Babelfy-threshold 10, given that having a smaller dimension reduces computational resources, in addition to generating faster recommendations. Therefore, this approach is used for the next experiment.

4.2.2 Experiment 2: comparison against baseline representations. In this experiment, the Babelfy-threshold 10 approach was compared with the representations described in Section 4.1. Both HT representations have 3085 terms, while the CT representations have 8433 terms. Gender-based representation has a total of 18 features.

According to previous works [6, 7], the size of the representation is an important factor in the similarity calculation and, consequently, recommendation of the item k-NN algorithm. To assess the quality of the representation regardless of the size of its vocabulary, experiments were carried out with the same number of features as both baseline representations. The results are reported in Table 2.

At first, Babelfy-threshold 10 was compared with the representations HT-TFIDF, HT-Sentiment, CT-TFIDF, CT-Sentiment and Genres. There is a statistically superior performance for p-value < 0.01 against all of them.

In addition, versions with the same vocabulary size as the baseline representations were built, producing the Babelfy-3085 representation to be compared with both HT representations, and Babelfy-8433 to be evaluated against the CT representations. It is observed that both approaches based on Babelfy present statistically superior results with p-value < 0.01. This reinforces the premise

[4] https://grouplens.org/datasets/movielens/100k/
[5] http://www.imdb.com/

Table 2: RMSE values for proposed representations and baselines. Bold values indicate the best results.

Algorithm	Features	k=20	k=40	k=60	k=80	k=100
Babelfy-threshold 10	6238	0,9183	0,9190	0,9203	0,9214	0,9224
Genres	18	0,9404	0,9401	0,9401	0,9401	0,9401
Babelfy-3085	3085	0,9202	0,9208	0,9222	0,9235	0,9246
HT-TFIDF		0,9435	0,9407	0,9402	0,9403	0,9404
HT-Sentiment		0,9310	0,9314	0,9330	0,9347	0,9361
Babelfy-8433	8433	**0,9180**	**0,9186**	**0,9199**	**0,9210**	**0,9219**
CT-TFIDF		0,944	0,9436	0,9434	0,9432	0,9431
CT-Sentiment		0,9301	0,9305	0,9327	0,9345	0,936

that, by adding more semantics to the representations, the items are better characterized and, consequently, the system is capable of producing more accurate recommendations.

Finally, when analyzing the results between the sentiment-based and TF-IDF versions of the baseline representations, it can be seen that there is an improvement in the RMSE values. This is indicative that sentiment analysis provides an additional semantic load for representations, which can be incorporated into the strategy proposed in this article in future works.

5 CONCLUSION

This article presented a recommender system based on disambiguated concepts extracted from user reviews. The experiments were focused on the movie domain, however, the system is generalizable to other domains.

The results show that the use of disambiguated concepts, associated with named entities, can improve the quality of item representations, consequently improving the effectiveness of a recommendation system.

As future work, it is expected to integrate the sentiment analysis to the proposal, considering that it adds a greater semantic load to the representations. It is also intended to extend the applicability of the technique to other recommendation algorithms that make use of content-based item representations.

6 ACKNOWLEDGEMENTS

The authors would like to thank the financial support from: CAPES, CNPq and FAPESP (grant 2016/20280-6).

REFERENCES

[1] J. Bobadilla, F. Ortega, A. Hernando, and A. Gutiérrez. 2013. Recommender systems survey. *Knowledge-Based Systems* 46, 0 (2013), 109–132.

[2] Michel Capelle, Flavius Frasincar, Marnix Moerland, and Frederik Hogenboom. 2012. Semantics-based News Recommendation. In *Proceedings of the 2Nd International Conference on Web Intelligence, Mining and Semantics (WIMS '12)*. ACM, New York, NY, USA, Article 27, 9 pages.

[3] Li Chen, Guanliang Chen, and Feng Wang. 2015. Recommender systems based on user reviews: the state of the art. *User Modeling and User-Adapted Interaction* 25, 2 (2015), 99–154.

[4] Xu Chen, Zheng Qin, Yongfeng Zhang, and Tao Xu. 2016. Learning to Rank Features for Recommendation over Multiple Categories. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '16)*. ACM, New York, NY, USA, 305–314. https://doi.org/10.1145/2911451.2911549

[5] Rafael D'Addio, Merley Conrado, Solange Resende, and Marcelo Manzato. 2014. Generating recommendations based on robust term extraction from users' reviews. In *Proceedings of the 20th Brazilian Symposium on Multimedia and the Web*. ACM, 55–58.

[6] Rafael Martins D'Addio and Marcelo Garcia Manzato. 2014. A collaborative filtering approach based on user's reviews. In *Intelligent Systems (BRACIS), 2014 Brazilian Conference on*. IEEE, 204–209.

[7] Rafael M D'Addio and Marcelo G Manzato. 2016. Exploiting Item Representations for Soft Clustering Recommendation. In *Proceedings of the 22nd Brazilian Symposium on Multimedia and the Web*. ACM, 271–278.

[8] Gayatree Ganu, Yogesh Kakodkar, and AméLie Marian. 2013. Improving the Quality of Predictions Using Textual Information in Online User Reviews. *Information Systems* 38, 1 (March 2013), 1–15. https://doi.org/10.1016/j.is.2012.03.001

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

[10] Pasquale Lops, Cataldo Musto, Fedelucio Narducci, Marco De Gemmis, Pierpaolo Basile, and Giovanni Semeraro. 2010. MARS: A MultilAnguage Recommender System. In *Proceedings of the 1st International Workshop on Information Heterogeneity and Fusion in Recommender Systems (HetRec '10)*. ACM, New York, NY, USA, 24–31. https://doi.org/10.1145/1869446.1869450

[11] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to Information Retrieval*. Cambridge University Press.

[12] J. McAuley, R. Pandey, and J. Leskovec. 2015. Inferring Networks of Substitutable and Complementary Products. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '15)*. ACM, New York, NY, USA, 785–794.

[13] Andrea Moro, Alessandro Raganato, and Roberto Navigli. 2014. Entity linking meets word sense disambiguation: a unified approach. *Transactions of the Association for Computational Linguistics* 2 (2014), 231–244.

[14] Cataldo Musto, Giovanni Semeraro, Pasquale Lops, and Marco de Gemmis. 2014. *Combining Distributional Semantics and Entity Linking for Context-Aware Content-Based Recommendation*. Springer International Publishing, Cham, 381–392. https://doi.org/10.1007/978-3-319-08786-3_34

[15] Fedelucio Narducci, Pierpaolo Basile, Cataldo Musto, Pasquale Lops, Annalina Caputo, Marco de Gemmis, Leo Iaquinta, and Giovanni Semeraro. 2016. Concept-based Item Representations for a Cross-lingual Content-based Recommendation Process. *Inf. Sci.* 374, C (Dec. 2016), 15–31. https://doi.org/10.1016/j.ins.2016.09.022

[16] Roberto Navigli and Simone Paolo Ponzetto. 2012. BabelNet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network. *Artificial Intelligence* 193 (2012), 217–250.

[17] Sergio Oramas, Vito Claudio Ostuni, Tommaso Di Noia, Xavier Serra, and Eugenio Di Sciascio. 2016. Sound and Music Recommendation with Knowledge Graphs. *ACM Trans. Intell. Syst. Technol.* 8, 2, Article 21 (Oct. 2016), 21 pages. https://doi.org/10.1145/2926718

[18] Maria Terzi, Matthew Rowe, Maria-Angela Ferrario, and Jon Whittle. 2014. *Text-Based User-kNN: Measuring User Similarity Based on Text Reviews*. Springer International Publishing, Cham, 195–206. https://doi.org/10.1007/978-3-319-08786-3_17

3.2 Final Remarks

In this work we have extracted a more meaningful, semantically richer set of features that are used to describe items. Those features are now disambiguated concepts and named entities, and thus do not suffer the problems of synonymy and polysemy that previous set of features could suffer. We create item representations based on that vocabulary, weighted by means of SF-IDF (CAPELLE *et al.*, 2012), which is a statistical weighting scheme that provides a score that reflects the intra and inter-frequencies of features towards items.

While it was a small scale experiment, it showed some interesting results that directly address Research Question 2: "Does increasing the semantics of item representations improve recommendation accuracy?". The results show that by refining the features in which the item representations are based on, we have a significant increase in recommendation accuracy. The new feature set is compared against two term-based feature sets that were produced in previous works (D'ADDIO; MANZATO, 2015; D'ADDIO; DOMINGUES; MANZATO, 2017), in both TF-IDF and sentiment-based weighting schemes. Results show a significant increase in recommendation accuracy against baselines, even when regarding similar vocabulary sizes.

But such small scale experiment is not sufficient evidence to answer the aforementioned question. Indeed, one of the main drawbacks of this work is that it just considers experiments in a single, small and well-behaved dataset.

Another issue is that SF-IDF weighting is one of many semantic ways to describe a feature's value towards items. SF-IDF relies on a statistical kind of semantics, where a feature's worth towards an entity (in this case, an item) is measured by how frequent it is mentioned by the entity and how rare it is mentioned by the other entities of the collection. As it is stated on the paper, sentiment-based weighting, in which a feature's value towards an entity is measured by how *positive* or *negative* a feature is regarded, provides better descriptions for term-based vocabularies. A natural next step in our research is to also apply such weighting scheme with the concept set of features, as well as other semantically different weighting schemes.

In the next chapter, we continue our research by generating new item representations. In that work, we do not focus on extracting a new vocabulary, but we focus on creating new weighting schemes for our features, including the aforementioned sentiment-based weighting. Also, from this point on, we direct our efforts towards different recommendation scenarios, algorithms and datasets.

CHAPTER

4

INCORPORATING SEMANTIC ITEM REPRESENTATIONS TO SOFTEN THE COLD START PROBLEM

4.1 Contextualization

We continue our research aiming to further expand the semantics of our item representations. As we have seen in Chapter 3, the vocabulary of our representations now comprises concepts and named entities instead of simple terms, minimizing the synonymy and polysemy problem. This kind of vocabulary will be used for the remainder of our research, and the focus is shifted into exploring more semantically rich weighting schemes for the features, as well as the application of the item representations in different algorithms and scenarios.

We have experimented with several weighting techniques and methods, with some of them yielding interesting results. The paper which we depict in this chapter (D'ADDIO *et al.*, 2018) describes the results obtained with the four more relevant approaches we have developed: one sentiment-based approach (using Stanford CoreNLP's sentiment algorithm (MANNING *et al.*, 2014; SOCHER *et al.*, 2013)), and three other based on word sense embeddings, i.e. latent vectors that describe each concept. The three approaches make use of an extension to the NASARI embeddings (CAMACHO-COLLADOS; PILEHVAR; NAVIGLI, 2016; PILEHVAR *et al.*, 2017) in the following way: i) creation of an item embedding; ii) creation of item-concept similarity, which can be constructed by either filling the similarity between the item and *every* concept of the vocabulary, or only with the ones that were *mentioned* in the items' reviews. Those four representations are applied into content-aware versions of two recommender algorithms in a simulated experiment that deals with the cold start problem (AGGARWAL, 2016a). The paper received the Best Paper Award in the conference it was published.

Incorporating Semantic Item Representations to Soften the Cold Start Problem

Rafael M. D'Addio, Eduardo P. Fressato, Arthur F. da Costa and Marcelo G. Manzato
Institute of Mathematical and Computer Sciences - ICMC
University of São Paulo - USP, São Carlos, SP, Brazil
{rdaddio, fortes, mmanzato}@icmc.usp.br and eduardofressato@usp.br

ABSTRACT

Recommender systems have been extensively used to provide meaningful and personalized content to users. A recurring issue, especially in collaborative filtering methods, is the cold-start problem, which can be related to new items or new users. This problem can be smoothed by aggregating item information into the recommender calculation, thus the semantics behind these items representations are important. In this paper, we propose four rich item representations, based on three kinds of semantics: sentiment analysis, sense embeddings and similarities. The items' features are disambiguated concepts extracted from textual users' reviews, which are known for possessing a great information load with both item descriptions and user preferences. We apply these four representations in two classic collaborative filtering algorithms, which were adapted to be attribute aware. We compare our approach against the original recommenders, and evaluate our results in two very different datasets to show the generality of our approach. Results show a very positive influence of the item representations to reduce prediction error.

CCS CONCEPTS

• **Information systems** → **Recommender systems**; Document representation;

KEYWORDS

Recommender systems, Item representation, cold-start

ACM Reference format:
Rafael M. D'Addio, Eduardo P. Fressato, Arthur F. da Costa and Marcelo G. Manzato. 2018. Incorporating Semantic Item Representations to Soften the Cold Start Problem. In *Proceedings of Brazilian Symposium on Multimedia and the Web, Salvador-BA, Brazil, October 16–19, 2018 (WebMedia '18),* 8 pages.
https://doi.org/10.1145/3243082.3243112

PUBLICATION INFORMATION

Published in proceedings of the 24th Brazilian Symposium on Multimedia and the Web (Webmedia '18). 2018. Pages 157–164.
DOI:10.1145/3243082.3243112

1 INTRODUCTION

Recommender systems (RS) have extensively been used by websites, social networks and applications that provide personalized content for their users. Using recommenders, the content provider is able to capture the users' preferences through their feedback, analyze them and return interesting suggestions about what users should consume [1].

The recommendation process can be performed mainly in two manners [1]: by analyzing the content of the items which the user has regarded as interesting (called content-based filtering), or by analyzing strictly the users' interactions and their relationships (collaborative filtering). The two paradigms are not exclusive, and often their are intermingled into a hybrid filtering [1, 14].

An occurring problem in recommender systems, especially with collaborative filtering algorithms, is the cold-start problem [1]. Items or users are considered cold if they have few or none interactions, leading to the system not being capable to recommend a cold item or to provide proper suggestions to a cold user. The item cold-start problem, also called new item problem, can be smoothed by aggregating item information into its calculation, thus performing a hybrid recommendation solution.

In this context, some efforts have been made in matrix factorization and neighborhood approaches to improve the quality of recommendations, using structured item metadata [10, 12, 17]. Usually, theses studies adopt the following strategies to incorporate metadata in the recommendation process: i) compute new similarity matrices to replace or assist the traditional rating-based similarities; or ii) aggregate or supplement additional information in traditional recommendation models. Nevertheless, in both cases, the quality and semantics of these items' descriptions play a crucial role in granting more accurate recommendations. In addition, there have been reports correlating the level of details of these descriptions with the performance of recommenders, where more enriched information can indeed benefit the accuracy of the model [8, 18].

Towards the representation of items with better semantics and richer information, nowadays there has been a growing effort to exploit user reviews for better descriptions of items or users in recommender systems [4]. Reviews are a great source of information since they contain relevant descriptions about items, as well as personal opinions regarding the item and its features [4, 8].

Works that use reviews direct their efforts to better describe items and/or users [5, 8, 9, 18, 26], but they mainly disregard text ambiguity problems such as synonymy and polysemy. On the other hand, works that address these issues apply their efforts on impersonal texts such as synopses and Wikipedia[1] descriptions [3, 21, 23]. Moreover, these works do not evaluate their findings directly in an item cold-start scenario, thus not dealing with this problem directly.

In this work we focus on the creation of high-quality and semantic item representations to improve the performance of hybrid recommenders in the item cold-start scenario. We propose four different item descriptions, with each carrying different semantics ranging from sentiment analysis, word sense embeddings to item-concept similarities. We use user reviews as source of information for the enriched representations, and extract from them disambiguated concepts (word senses or entities) as features. The main

[1] https://www.wikipedia.org/

advantage in using user content to build our representations is that in a sense we condense in each representation the aspects of the item that are relevant to the users themselves. Also, the semantics imbued in each of our representations are different from each other and each serves a determined purpose.

We apply those representations into two collaborative filtering algorithms that were adapted to incorporate item information to help minimize the new item problem and, unlike the aforementioned works, we evaluate our approach directly in a simulated item cold-start scenario. We compare the representations against each other and against the original recommender systems. Our study shows that their use can help reducing the prediction error of recommender systems in the item cold-start scenario.

Thus, our work's main contributions are:

- Creation of four different, rich and semantic item representations that are able to accurately describe items for recommender systems;
- A pre-processing module capable of extracting meaningful features from users' reviews using state-of-the-art natural language processing (NLP) toolkits and resources;
- Use of the proposed item representations to smooth the item cold-start problem in recommendation.

This paper is structured as follows: in Section 2 we overview research results related to text pre-processing, feature extraction and cold-start problem related to recommender systems. In Section 3 we review the NLP tools and the adapted recommender algorithms that we use in our study. We then present our item representation approaches in Section 4 and discuss the experimental results in Section 5. In Section 6 we present our conclusions and future work.

2 RELATED WORK

In this section, we review studies that apply additional item information to enrich or replace traditional representations, thus implicitly smoothing the item cold-start problem.

By combining a collaborative method with a content-based method, Forbes and Zhu [10] propose the Content-Boosted Matrix Factorization (CBMF) algorithm to incorporate structured item metadata directly into the MF approach. This approach do not deal directly with the cold-start problem and items descriptions are binary structured metadata, such as genres, which may carry little semantics.

Recent works have been using more semantic sources of information, such as user reviews. Those works tend to extract features from these texts and use them to better describe the items according to their characteristics. In the works proposed by D'Addio et al. [8, 9], different item representations based on terms and aspects were constructed and applied in two different recommenders, one based on neighborhood and the other on soft clustering. Works that build user profiles from their reviews, in turn, can generate profiles of finer granularity which explain the users' preferences in relation to the items' characteristics. An example is the work of Terzi et al. [26], which proposed a user-based neighborhood algorithm whose similarities are calculated based on user reviews. The authors explore six similarity metrics based on the WordNet[2] taxonomy. However, none of the aforementioned approaches deal with text ambiguity problems such as polysemy and synonymy.

While there are efforts to minimize these problems, they focus majorly on impersonal texts such as synopses and Wikipedia descriptions. In this sense, Capelle et al. [3] proposed a content-based news recommender system that makes use of a word sense disambiguation (WSD) algorithm based on the WordNet taxonomy for the construction of item representations and user profiles. They also extend the TF-IDF metric [15], incorporating in the calculation a similarity based on page count when querying for named entities present in the text. Systems that use the WordNet senses repository are limited by the low coverage of named entities.

Finally, efforts have been made to incorporate techniques that accomplish both WSD and entity linking (EL) in content-based recommendation systems, such as the work of Narducci et al. [21] and Oramas et al. [23]. In [21], the authors evaluated four different types of representations in the multilingual recommendation task, two based on Wikipedia and two based on BabelNet[3] [22]. In [23], the BabelFy[4] [20] tool is used to extract concepts related to music and sounds, which are used to automatically produce a knowledge base in the form of a graph. The item representations are constructed by mapping the graph in a vector space.

The majority of the previous works were not designed to explicitly address the item cold-start problem in the rating prediction scenario. With this work, in turn, we explicitly address the problem by designing our experiments to simulate a new item scenario. We focus our work into building rich and semantic representations that can correctly describe items, positively influencing the outcome of recommenders. Our item representations provide richer semantics than those present in the aforementioned works since they not only deal with WSD and EL techniques to extract features, but they also carefully explore weighting alternatives based on different kinds of semantics, such as sentiment analysis, word sense embeddings and item-concept similarities. In addition, differently from the aforementioned works, we do not add complexity to the prediction rule of the recommenders by guaranteeing that the item-item similarities can be computed offline as a separate step and then be plugged into any attribute aware recommender.

3 BACKGROUND

This section elaborates on the main concepts and algorithms involved this paper. Specifically, in the following we revise the natural language processing tools and recommender algorithms that will be used later on our experiments.

3.1 Notation

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

[2] https://wordnet.princeton.edu/

[3] http://babelnet.org/

[4] http://babelfy.org/

3.2 Natural Language Processing Tools

We use three different NLP tools or resources that allow semantic and accurate text processing, resulting in rich, more structured information ready to be consumed by our proposed representations.

BabelFy [20] uses the BabelNet [22] knowledge base to perform the tasks of word sense disambiguation and entity linking, extracting concepts called Babel synsets, which represent both word senses found on WordNet and entities from the Wikipedia.

NASARI[5], proposed by Camacho-Collados et al. [2], is a NLP resource that provides vector representations for a large portion of BabelNet synsets. They present them in three forms: a) lexical, where the features of the synsets are lemmas; b) unified, where the features are the BabelNet synsets themselves; and embedded, where the synsets are represented in a 300 dimensions latent space.

We also use Stanford CoreNLP [16], a NLP toolkit that contains several NLP routines, such as tokenization, lemmatization, parsing, part-of-speech (POS) tagging and sentence splitting. This toolkit also contains a sentiment analysis algorithm [25], which uses recursive neural networks models to classify sentences in five sentiment levels: "Very Negative", "Negative", "Neutral", "Positive" and "Very Positive". It outputs XML files structuring the texts into sentences and their components, as well as their parser and sentiment.

3.3 Item K-Nearest Neighbors

The first recommender algorithm used in our experiments is Item-KNN [1, 13], which uses the concept of nearest neighbors to generate recommendations. Item-KNN predicts the unknown rating $\hat{r}_{ui}$ based on u's ratings of the k most similar items to i. In order to find similar items, a similarity measure (e.g. Pearson, Jaccard, Cosine) is employed to generate an item-item similarity matrix.

Using the similarity matrix, we identify the set of k items that are most similar to i, the so-called k-nearest neighbors, and select the subset of those items that have been rated by u. Using this subset, denoted as $S^k(i;u)$, the final predicted rating can be calculated as:

$$\hat{r}_{ui} = b_{ui} + \frac{\sum_{j \in S^k(i;u)} s_{ij}(r_{uj} - b_{uj})}{\sum_{j \in S^k(i;u)} s_{ij}} \tag{1}$$

where s_{ij} is the similarity between items i and j and b_{ui} is a baseline estimate based on the ratings provided by users, proposed by [13] for neighborhood models, and defined as:

$$b_{ui} = \mu + b_u + b_i \tag{2}$$

where μ is the global average rating and b_u and b_i indicate the observed deviations of user u and item i from the average, respectively. The main purpose of b_{ui} is to capture systematic tendencies of certain users to give higher/lower ratings than others, and certain items to receive higher/lower ratings than others. The estimates b_u and b_i can be calculated by solving a least squares problem [13].

In order to incorporate enriched information about items to reduce the cold-start problem, we used the content-based version of the Item-KNN algorithm [12], which uses a similarity matrix based on item attributes to make predictions (called ItemAttrKNN in our experiments). The main difference between Item-KNN and ItemAttrKNN is that the latter replaces the traditional item representation ($item \times user$) with enriched and representative information about items to compute the similarity matrix. In this way, the algorithm takes into account the characteristics of each item and not the users' interactions, which means that items that have no interaction in the dataset have a chance to be recommended.

[5] http://lcl.uniroma1.it/nasari/

3.4 Matrix Factorization

The second recommender used in our experiments is Matrix Factorization [14]. In this algorithm, each item i is associated with a latent vector $q_i \in \mathbb{R}_f$ and each user u is associated with a latent vector $p_u \in \mathbb{R}_f$. Each component of the item's vector (q_i) represents the degree of relevance of the latent factor in item i. For user vector components (p_u), each factor represents the degree of interest (appreciation or depreciation) that user u has in the factor. The rating prediction is made based on the product of the transpose of the item's vector q_i with the user's vector p_u, defined by [14]:

$$\hat{r}_{ui} = \mu + b_u + b_i + q_i^T p_u. \tag{3}$$

where, as from Item-KNN, μ represents the average rating of all known ratings, and b_u and b_i indicate the observable deviation of user u and item i from the global mean.

During the training process, the parameters are trained and adjusted for better accuracy in prediction, minimizing the quadratic error:

$$\min_{p_*, q_*, b_*} \sum_{(u,i) \in K} (r_{ui} - \mu - b_u - b_i - q_i^T p_u)^2 + \lambda(b_u^2 + b_i^2 + ||q_i||^2 + ||p_u||^2), \tag{4}$$

where λ is the regularization parameter, usually defined by cross validation. Solving Equation 4 is typically accomplished through Stochastic Gradient Descent [14].

It is worth mentioning that in cases of pure item cold-start, the new item i will not have any interaction during training, which causes factor vector q_i to be filled with zero or random values, depending on implementation. Thus, the prediction rule in Equation 3 for item i will be reduced to $\hat{r}_{ui} = \mu + b_u$, which does not consider any additional information about i. In order to smooth this problem, we extend the presented matrix factorization approach in order to incorporate enriched information about the items. We call this approach Item-MSMF (Items Most Similar based on Matrix Factorization)[11]. This extension consists in a post-processing approach, which is applied after the training process. First, an item-item similarity matrix must be constructed using any content-based item representation. Next, we compute a weighted average of the latent factors of the k most similar items of each cold item, and replace the vector of latent factors of the new items, by their respective new vectors. Thus, for a new item i, its latent vector q_i is replaced as follows:

$$q_i \Leftarrow \frac{\sum_{j \in Ni^k(i;u)} q_j \cdot sim(i,j)}{\sum_{j \in Ni^k(i;u)} sim(i,j)}, \tag{5}$$

where $Ni^k(i;u)$ is the subset of most similar items to i that are not new, i.e., those that have received at least one rating.

In contrast to well-known matrix factorization algorithms that attempt to alleviate the cold-start problem by means of additional complexity of the model, this extension approach introduces changes

only in the prediction step rather than in the training process. This is an advantage because we add only one additional step at the end of the matrix factorization approach.

4 GENERATING ITEM REPRESENTATIONS

In this section we describe the steps performed to generate our item representations. In total there are four representations that carry different semantics to the items. One of them is based on the quality of the features that compose it, i.e. whether they are appreciated or not by the users that consumed the item. The other, in turn, is based on latent semantics observed in each of the concepts present in the items. Finally, the remaining two are based on similarities between pre-computed latent item representations and their features, also represented in a latent vector space.

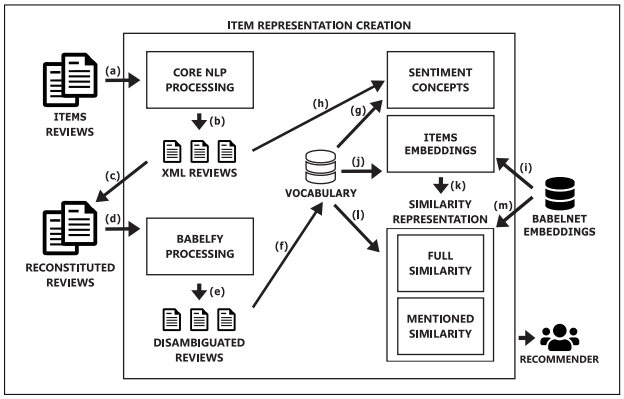


Figure 1: Schematic view of the proposed method.

Figure 1 shows the whole item representations creation process, which is better detailed in the next subsections. We first describe how we pre-process the texts so they can be processed by our approaches, and how we extract a vocabulary that will serve as features for our representations. Then, we describe each item representation approach separately in details. In this section, we use the words "concepts", "synsets" and "senses" interchangeably, i.e. they all mean features of items.

4.1 Text Pre-processing and Feature Extraction

Since in this work we use two different NLP tools (described in Section 3.2) for processing texts, i.e. one for sentiment analysis and another for word sense disambiguation, we need to perform steps that guarantee their communication. First, we process the items' reviews with Stanford CoreNLP [16] (Figure 1, arrows (a) and (b)), using the following annotators: tokenization, lemmatization, part-of-speech (POS) tagging, parsing, sentiment analysis and sentence splitting. The main goal here is to be able to split the texts into sentences and obtain their sentiments.

From there, we reconstruct the reviews into plain text (Figure 1, arrow (c)) so they can be processed by BabelFy [20]. The reason for reconstructing the documents is because we preserve the Stanford CoreNLP sentence splitting, which is necessary to match BabelFy's output with the sentiments produced by the previous tool, thus allowing us to build one of our item representations.

The documents are processed by BabelFy (Figure 1, arrows (d) and (e)), and transformed into text files where words are replaced by disambiguated Babel synsets. From there, we extract our vocabulary (Figure 1, arrow (f)), which will be the features of the majority of our representations. The selected synsets that compose our vocabulary are only nouns that appear in a predetermined minimum number of items. The intuition behind this is that features that appear in a very small number of items can be viewed as noise and thus will not largely interfere in the outcome of the algorithm [7]. Thus, we filter concepts that are present in a number of items smaller than a predetermined threshold, removing them from the final vocabulary. The remaining synsets after this step will constitute the vocabulary, which will be used to produce the majority of the item representations that are detailed in the next subsections. These representations are modeled in a vector space, where each feature corresponds to a dimension in that space.

4.2 Sentiment-based Representations

The first item representation, which we call **sentiment concepts**, focuses on measuring the quality of the features of an item. In this context, each concept from the vocabulary is weighted according to the average sentiment (e.g. positive, negative or neutral) that users assign in their reviews toward them.

In order to do that, we use the sentiment analysis algorithm available at the Stanford CoreNLP toolkit [25], which performs a sentence-level analysis. We use this kind of analysis mainly because all of the features extracted from the reviews are nouns, which in most cases have neutral sentiment. Thus, a reasonable approach to obtain their sentiment is to use the context in which they are inserted in the text. Moreover, since features are extracted from a tool other than CoreNLP, and they are concepts instead of words, it is justifiable to use the sentiment of the sentences they occur.

Recall from Section 3.2 that this algorithm classifies sentences in five sentiment levels: "Very Negative", "Negative", "Neutral", "Positive" and "Very Positive". We convert this classification into a $[1, 5]$ rating system, being 1 equals to "Very Negative" and 5 equals to "Very Positive".

We use the following strategy to obtain the sentiments of the synsets, adapted from [8]. First, the system analyzes the vocabulary, then, for each item, reads the disambiguated texts, storing the sentences IDs that are related to each of the synsets (Figure 1, arrow (g)). Then, it matches the IDs with those of the CoreNLP files, storing their corresponding sentiments (Figure 1, arrow (h)).

After obtaining the sentences related to each feature, the next step is the sentiment attribution. For each feature of each item it is calculated the average sentiment of the related sentences. Thus, this value represents the collective level of appreciation or depreciation of a certain attribute of an item. This approach does not consider the level of confidence that a final sentiment score may have: a score calculated from several mentions to a feature from several different reviews is more reliable than a score calculated from a single mention. In order to address this, we devised a heuristic where we dampen the sentiment value, i.e., approximate it to 3 (the neutral value), if the number of times it was mentioned is lower than the global average number of mentions. Formally, a sentiment

$S(s,i)$ of a synset s in relation to an item i can be dampen by the following formula, if its mentions are lower than the global average:

$$S(s,i) = \begin{cases} S(s,i) + log_{10}(n_s/N) & \text{if S(s,i)} > 3 \\ S(s,i) - log_{10}(n_s/N) & \text{if S(s,i)} < 3 \end{cases}, \quad (6)$$

where n_s is the number of times the synset was mentioned in the item's reviews, and N is the global average number of mentions. With this heuristic, we are able to dampen the sentiment strength to a maximum of 1, still preserving its original orientation.

Finally, a zero value indicates that an item simply does not have that feature.

4.3 Embedding-based Representations

In recent years, there has been a growing interest in representing words or senses (synsets) into semantic dense vectors called word/sense embeddings [24], due to their fast processing and generalization capability [2]. In this representation, we construct item embeddings derived from the embeddings of their own features.

As synset embeddings, we use an extension of the NASARI embedded vectors [2], proposed by Pilehvar et al. [24] (Figure 1, arrow (i)). They use the original NASARI embeddings to feed their technique called DeConf to produce new sense embeddings in the same vector space, thus expanding the original set of senses and covering the remaining WordNet synsets that NASARI did not cover. Those embeddings are vectors inserted in a 300-dimensions latent space, and can be used to find relationships between concepts.

In this sense, we try to put our items into that same latent space by using the embeddings of their features, thus creating **item embeddings**. The system analyses the vocabulary (Figure 1, arrow (j)), and, as a first step, for each item it produces a binary vector where each position represents a concept from our vocabulary. It sets a feature as 1 if it is mentioned in the items' reviews, and 0 if it is not. With these binary vectors, for each item i it obtains the sense embedding s_c of those concepts that are set as 1 and sets the item embedding as the centroid of these vectors, i.e. the average vector. Formally, let $S_c(i)$ be the set of sense embeddings of the features an item has, and $|S|$ be the size of this set. The item embedding s_i is defined as:

$$s_i = \frac{\sum_{s_c \in S_c(i)} s_c}{|S|} \quad (7)$$

Thus, the item representation will be portrayed as a vector with 300 latent features that encompasses characteristics of all of the concepts found in its reviews.

4.4 Similarity-based Representations

From the previous item embedding representation (Figure 1, arrow (k)), we further expand its semantics by proposing a third category of representations: based on item-concept similarity. In this section we present two versions of this type of representation. The main notion behind them is that now we try to describe items according to their relationships with their concepts.

Having calculated the item embedding vectors, the goal here is to define how close it is from its features, i.e., how similar it is from the concepts present on the vocabulary. To define its closeness, a similarity measure can be employed. We employ the cosine similarity, vastly used in recommender systems and information retrieval [15]. This metric calculates the cosine of the angle formed between two vectors, producing values from 1 ($cos(0)$) to -1 ($cos(180)$). Formally, the cosine similarity of two vectors v_i and v_j in the same space can be defined as:

$$cos(v_i, v_j) = \frac{v_i \cdot v_j}{\|v_i\|\|v_j\|} \quad (8)$$

We build two different representations from this similarity premise. Both of them have as features the concept vocabulary defined earlier, but they differ on how the vectors are filled.

In the first version, which we name as **full similarity**, we calculate the cosine similarities between the item embedding (generated in the previous representation) and the sense embeddings from all concepts in the vocabulary (Figure 1, arrows (l) and (m)), even those that are not present in the item's reviews. The weights of each position in the item-concept similarity representation are then filled with the calculated cosine values, producing a completely filled representation. With that, items now are discerned according to their closeness or distance towards each feature of our vocabulary.

The second version, called **mentioned similarity**, restricts the vector positions that are filled for only the concepts that are present in the texts related to the item. We calculate the cosine similarity in the same fashion as the other version of this representation, but we also use a binary vector that details which concepts are mentioned in the item's reviews. Only those mentioned in the reviews have their position filled with the similarity. The effect in this is that the vectors' disposition (positions with scores) remain similar to the binary matrix, and even similar to our sentiment-based representation, but the scores convey another type of semantics.

5 EVALUATION

To evaluate the efficiency of the proposed item representations, we employ them in two recommender systems in a simulated item cold-start scenario. We also compare and provide insights about the strengths of our representations.

Both algorithms were originally designed for collaborative filtering, but their extensions allow them to implicitly use item content through item-item similarity matrices. We calculate the similarity between items through our representations and pass them to the algorithms. The neighborhood-based model, presented in Section 3.3, uses our item-item similarity matrices instead of its original representation based on users' feedback, while the matrix factorization model, discussed in Section 3.4, uses our similarity matrices to select related items for a cold item so that its latent vector can be calculated based on them.

The following subsections detail our experiments. First, we present the two datasets used in this study; then we detail the experimental setup and the metrics used to evaluate our results; and, finally, we present and discuss our findings.

5.1 Datasets

We used two largely different data sets to evaluate our proposal.

The first is the well-known MovieLens 100k (ML-100k)[6] data set, which consists of 100, 000 ratings performed by 943 users for 1, 682 movies. In order to perform our experiments, we collected up to 10 reviews per item from the IMDb[7] website, resulting in a total of 15, 863 documents. The reviews were selected as the website's top-10, ordered by their helpfulness. The representations built on this dataset were constructed on top of these reviews and contain a vocabulary of 6238 Babelfy synsets. The item embeddings, in turn, were built on a latent space with 300 dimensions.

The second is one of the Amazon[8] datasets extracted by McAuley et al. [19][9]. From several datasets of different domains, we selected the Apps for Android dataset, which is the fifth biggest dataset from the collection. The original dataset has 2, 638, 172 ratings from 1, 323, 884 users for 61, 275 items. Given that it is a very large data set, we filtered it by maintaining only ratings that contain reviews and eliminated items and users that had less than 10 interactions, resulting in 16, 201 users and 4, 869 items, totaling 264, 047 interactions. The representations built on this dataset contain a vocabulary of 8978 Babelfy synsets. The item embeddings, similarly to the previous dataset, were built on a latent space with 300 dimensions.

As one can see, the two data sets are very different. To begin with, the user-item proportion is different, with the first data set having 1.78 times more items than users, while the second has 3.33 times more users. Furthermore, the second data set is significantly more sparse than the first: while it has 2.64 times more interactions than the previous, its user-item matrix is significantly larger: 17.18 times more rows (users) and 2.89 times more columns (items).

5.2 Experimental Setup and Evaluation Metrics

In order to check the strength of our representations, we apply them in two recommender systems adapted to the cold-start scenario, and we compare our findings against those same algorithms without the representations.

To simulate a pure item cold-start scenario, the available items in the datasets were divided into 10 different partitions randomly, (e.g. on a dataset with 1,000 items, each partition will have 100 distinct items). In this way, each partition will be composed by the triples $(u, i, r_{u,i})$ of their respective items. In the following step, we created the training and test sets, whose test set is composed by one partition and the training set by the other nine. Therefore, the items present in the test set will not be present in the training set. We swap these partitions 10 times, getting 10 different folds, which are used in a 10-fold cross-validation protocol.

Since it is a simulated cold-start scenario, we have used the reviews available within the Amazon Apps for simplicity. But in a real cold-start scenario, those reviews would not exist, since a new item do not have any kind of user interaction. We argue that, in those cases, the reviews must be gathered from external sources, as was the case of our other dataset, ML-100k. In that dataset, the reviews were gathered from a external source, the IMDb website.

To build the representations, we extracted a vocabulary by executing BabelFy [20] in the reviews set, then we restricted the extracted concepts to only nouns and filtered those that appeared in less than 10 items, resulting in 6238 and 8978 concepts for the ML-100k and Amazon Apps datasets, respectively. The item embeddings representation contain only 300 dimensions, similarly to the concept embeddings found on [24].

Regarding the parameters of the recommender algorithms, we determined a collection of values which performed well for all datasets. For Item-KNN, we used cosine to generate the similarity matrices, since the results indicate that this similarity is superior to other relevant measures that we tested, such as Pearson correlation and Euclidean. For the Matrix Factorization algorithm, we used 10 as factor value as suggested by Koren [13]. For other parameters, we used the default values of the chosen recommender tool.

Regarding our experiments, we generate item-item similarity matrices with each representation and apply them in both algorithms, testing them in both datasets. To construct the item-item similarity matrices, we used the cosine similarity. The only parameter that we test different configurations is the one directly related to the size of the item's neighborhood. For both algorithms, we show the results of our tests with $k = \{10, 20, 40\}$.

We evaluate our results with 10-fold cross-validation adapted to the item cold-start scenario, and analyze them by means of the average error between a predicted rating and its real value in the test set. We use root mean square error (RMSE), and calculate it for each test set of the 10 folds, but display here only the average results. In order to check statistical relevance between the results we applied the Wilcoxon test with 99% and 95% of confidence [27].

The baseline algorithms belong to the Case Recommender Framework [6] version 1.0.9[10], an open source tool developed in *Python*, with numerous features and algorithms for RS. We used the default framework settings in our evaluations.

5.3 Results and Discussion

This subsection presents and discusses the results of our experiments, showing the impact of our approach for item cold-start recommendation.

The tables here present the results of four recommender algorithms and four item representations. To recall their names, the algorithms are:

MF and Item-MSMF: the recommenders based on matrix factorization described in Section 3.4;

Item-KNN and ItemAttrKNN: the neighborhood algorithms described in Section 3.3;

And the item representations are:

Sentiment concepts: the representations based on the average sentiment that users assign to concepts, described in Section 4.2;

Item embeddings: the representations based on embeddings of concepts mentioned in the items' reviews, described in Section 4.3;

Full similarity: the representations based on the similarity between the items and all the concepts on the vocabulary, described in Section 4.4;

Mentioned similarity: the representations based on the similarity between the items and the concepts that they have on their reviews, also described in Section 4.4;

[6] https://grouplens.org/datasets/movielens/100k/
[7] http://www.imdb.com/
[8] https://www.amazon.com/
[9] http://jmcauley.ucsd.edu/data/amazon/
[10] https://pypi.python.org/pypi/CaseRecommender

5.3.1 Results. Table 1 shows the results of the representations applied in Item-MSMF and ItemAttrKNN and both datasets, which compares three neighborhood sizes, i.e. 10, 20 and 40. Highlighted values show the best value of k for each representation.

Table 1: Comparison of RMSE on two Datasets, using three different number of neighbors.

Rec.	Metadata	k=10	k=20	k=40
Movielens 100k				
Item-MSMF	Sentiment concepts	**1.0118**	1.0148	1.0188
	Item embeddings	**1.0311**	1.0316	1.0337
	Full similarity	1.0404	1.0402	**1.0396**
	Mentioned similarity	**1.0159**	1.0185	1.0225
ItemAttrKNN	Sentiment concepts	1.1120	1.1022	**1.1010**
	Item embeddings	1.1114	1.1003	**1.0998**
	Full similarity	1.1188	1.1060	**1.1034**
	Mentioned similarity	1.1074	**1.0990**	1.0991
Amazon Apps				
Item-MSMF	Sentiment concepts	1.2225	1.2178	**1.2171**
	Item embeddings	1.2509	1.2421	**1.2387**
	Full similarity	1.2456	1.2398	**1.2368**
	Mentioned similarity	1.2218	1.2162	**1.2153**
ItemAttrKNN	Sentiment concepts	1.2762	1.2682	**1.2670**
	Item embeddings	1.2813	1.2695	**1.2675**
	Full similarity	1.2791	1.2696	**1.2678**
	Mentioned similarity	1.2747	1.2675	**1.2663**

Bold typeset indicates the best performance.

As it can be seen, the k value plays a role in decreasing the prediction error. For the Item-MSMF in the ML-100k dataset, we select as optimal $k = 10$, since 3 out of 4 representations present better results with that value. As for the ItemAtrrKNN in the same dataset, we select as optimal $k = 40$. In the Amazon Apps dataset, in turn, all results show that $k = 40$ is the optimal value for both algorithms.

Tables 2 and 3 compare the item representations with the optimal k values against the baselines candidates, the original MF and Item-KNN, respectively.

Table 2: Matrix Factorization Algorithms

Rec.	Item Representation	ML-100k	Amazon Apps
MF	-	1.0664	1.2497
Item-MSMF	Sentiment concepts	**1.0118***	1.2171*
	Item embeddings	1.0311*	1.2387
	Full similarity	1.0404*	1.2368**
	Mentioned similarity	1.0159*	**1.2153***

Bold typeset indicates the best performance. * indicates statistical significance at $p < 0.01$, and ** indicates statistical significance at $p < 0.05$, pairwise compared to MF result.

Table 3: Neighborhood Algorithms

Rec.	Item Representation	ML-100k	Amazon Apps
Item-kNN	-	1.1256	1.3411
ItemAttrKNN	Sentiment concepts	1.1010*	1.2670*
	Item embeddings	1.0998*	1.2675*
	Full similarity	1.1034*	1.2678*
	Mentioned similarity	**1.0991***	**1.2663***

Bold typeset indicates the best performance. * indicates statistical significance at $p < 0.01$ pairwise compared to Item-kNN result.

It can be seen that almost all representations provide statistically superior results with a confidence level of 99% in both datasets, except for the full similarity with Item-MSMF in Amazon Apps, with 95% confidence, and item embeddings with Item-MSMF, also in the Amazon Apps dataset, with no confidence.

5.3.2 Discussion. As it has been pointed out earlier, the quality of item representations plays a crucial role in aiding content-based recommenders to provide meaningful suggestions. With our experiments, it can be concluded that they also considerably help smoothing the item cold-start problem, since these representations add semantics and/or real users' opinions to enrich the recommendations process.

A careful item characterization brings benefits to recommender systems with cold-start problem, where items that otherwise would have no means to be properly suggested can be better analyzed by the system. For instance, the Item-KNN recommender in a full item cold-start scenario is restricted to predict ratings using only the baseline estimates, since cold items will have empty representations and thus will only be similar to other cold items. By adding the content-based item representations, we solve this problem and can see a great improvement in the prediction error.

But the semantics behind those representations are largely responsible to improve the quality of the results. From the four item representations, two of them showed the best results with Item-MSMF in both datasets: sentiment concepts and mentioned similarity. They have a similar distribution, i.e., both representations have the same assigned concepts, but they vary in the score of these concepts: one's semantic is directly related to the quality of the features, while the other is based on how close, i.e. how well described, is the item by its features. With ItemAttrKNN, in turn, the representations presented very close results among each other, making us unable to decide which is the best. Nevertheless, the lowest prediction error was achieved by mentioned similarity, being 0.0007 lower than the next low result.

This closeness in the results between all of them in ItemAttrKNN, and two of them in Item-MSMF, is a positive point. That makes every one of them a good candidate to be applied in a recommender system, according to its requirement. For instance, the sentiment concept representation can only be used with recommenders where there are user reviews, since opinions form the core of the semantics of this representation. On the other hand, the other three representations can be used with impersonal texts such as synopses, Wikipedia articles or any other kind of texts.

Additionally, the full similarity representation may be regarded as the most space-consuming representation, since the vectors are all fully filled, but if little space is a requirement, the item embeddings are the most lightweight representations, and the best choice in this case.

Finally, all item representations can be constructed in an offline pre-processing step, not harming the complexity of training and prediction steps of the recommenders.

6 CONCLUSIONS

In this work we have proposed four rich item representations that use three kinds of semantics: sentiment analysis, sense embeddings and item-concept similarities. The representations were applied in recommender systems to help smoothing the item cold-start problem. Results from our experiments show that the item descriptions do help minimizing the prediction error for new items.

All item representations provide good results, making each of them interesting to be applied in recommender systems, leaving the decision of which to use only dependent on the available data and space requirements.

In future works, we aim to fuse sentiment and sense embeddings in two ways: a) enriching or inferring unknown sentiments towards unmentioned concepts by using the sentiments of related known concepts in the embeddings' latent space; or b) restricting the construction of item embeddings to use only concepts viewed as positive by the users. Additionally, we plan to test these representations in other recommender systems and scenarios, such as top-n recommendation, and also compare our approach against other techniques aimed in solving the item cold-start problem.

ACKNOWLEDGMENTS

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

REFERENCES

[1] Charu C. Aggarwal. 2016. *Recommender Systems: The Textbook* (1st ed.). Springer Publishing Company, Incorporated. http://www.springer.com/us/book/9783319296579

[2] José Camacho-Collados, Mohammad Taher Pilehvar, and Roberto Navigli. 2016. Nasari: Integrating explicit knowledge and corpus statistics for a multilingual representation of concepts and entities. *Artificial Intelligence* 240 (2016), 36 – 64. https://doi.org/10.1016/j.artint.2016.07.005

[3] Michel Capelle, Marnix Moerland, Frederik Hogenboom, Flavius Frasincar, and Damir Vandic. 2015. Bing-SF-IDF+: A Hybrid Semantics-driven News Recommender. In *Proceedings of the 30th Annual ACM Symposium on Applied Computing (SAC '15)*. ACM, New York, NY, USA, 732–739. https://doi.org/10.1145/2695664.2695700

[4] Li Chen, Guanliang Chen, and Feng Wang. 2015. Recommender systems based on user reviews: the state of the art. *User Modeling and User-Adapted Interaction* 25, 2 (2015), 99–154.

[5] Xu Chen, Zheng Qin, Yongfeng Zhang, and Tao Xu. 2016. Learning to Rank Features for Recommendation over Multiple Categories. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '16)*. ACM, New York, NY, USA, 305–314.

[6] Arthur F da Costa and Marcelo G Manzato. 2016. Case Recommender: A Recommender Framework. *In: Workshop de Teses e Dissertações (WTD), 16., 2016, Teresina. Anais do XXII Simpósio Brasileiro de Sistemas Multimídia e Web. Porto Alegre: Sociedade Brasileira de Computação, 2016. v. 2.* (2016).

[7] Rafael D'Addio, Merley Conrado, Solange Resende, and Marcelo Manzato. 2014. Generating recommendations based on robust term extraction from users' reviews. In *Proceedings of the 20th Brazilian Symposium on Multimedia and the Web*. ACM, 55–58.

[8] Rafael M. D'Addio, Marcos A. Domingues, and Marcelo G. Manzato. 2017. Exploiting feature extraction techniques on users' reviews for movies recommendation. *Journal of the Brazilian Computer Society* 23, 1 (05 Jun 2017), 7.

[9] Rafael M D'Addio and Marcelo G Manzato. 2016. Exploiting Item Representations for Soft Clustering Recommendation. In *Proceedings of the 22nd Brazilian Symposium on Multimedia and the Web*. ACM, 271–278.

[10] Peter Forbes and Mu Zhu. 2011. Content-boosted Matrix Factorization for Recommender Systems: Experiments with Recipe Recommendation. In *Proceedings of the Fifth ACM Conference on Recommender Systems (RecSys '11)*. ACM, New York, NY, USA, 261–264. https://doi.org/10.1145/2043932.2043979

[11] Eduardo P. Fressato, Arthur F. Da Costa, and Marcelo G. Manzato. 2018. Similarity-based Matrix Factorization for Item Cold-Start in Recommender Systems. In *2018 Brazilian Conference on Intelligent Systems*.

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

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

[14] Yehuda Koren and Robert Bell. 2011. Advances in Collaborative Filtering. In *Recommender Systems Handbook*, Francesco Ricci, Lior Rokach, Bracha Shapira, and Paul B. Kantor (Eds.). Springer US, 145–186. https://doi.org/10.1007/978-0-387-85820-3_5

[15] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to Information Retrieval*. Cambridge University Press.

[16] Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP Natural Language Processing Toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*. 55–60.

[17] Marcelo Garcia Manzato. 2013. gSVD++: Supporting Implicit Feedback on Recommender Systems with Metadata Awareness. In *Proceedings of the 28th Annual ACM Symposium on Applied Computing (SAC '13)*. ACM, New York, NY, USA, 908–913. https://doi.org/10.1145/2480362.2480536

[18] Ronnie S. Marinho, Rafael M. D'Addio, and Marcelo G. Manzato. 2017. Semantic Organization of User's Reviews Applied in Recommender Systems. In *Proceedings of the 23rd Brazillian Symposium on Multimedia and the Web (WebMedia '17)*. ACM, New York, NY, USA, 277–280.

[19] Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton van den Hengel. 2015. Image-Based Recommendations on Styles and Substitutes. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '15)*. ACM, New York, NY, USA, 43–52. https://doi.org/10.1145/2766462.2767755

[20] Andrea Moro, Alessandro Raganato, and Roberto Navigli. 2014. Entity linking meets word sense disambiguation: a unified approach. *Transactions of the Association for Computational Linguistics* 2 (2014), 231–244.

[21] Fedelucio Narducci, Pierpaolo Basile, Cataldo Musto, Pasquale Lops, Annalina Caputo, Marco de Gemmis, Leo Iaquinta, and Giovanni Semeraro. 2016. Concept-based Item Representations for a Cross-lingual Content-based Recommendation Process. *Inf. Sci.* 374, C (Dec. 2016), 15–31. https://doi.org/10.1016/j.ins.2016.09.022

[22] Roberto Navigli and Simone Paolo Ponzetto. 2012. BabelNet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network. *Artificial Intelligence* 193 (2012), 217–250.

[23] Sergio Oramas, Vito Claudio Ostuni, Tommaso Di Noia, Xavier Serra, and Eugenio Di Sciascio. 2016. Sound and Music Recommendation with Knowledge Graphs. *ACM Trans. Intell. Syst. Technol.* 8, 2, Article 21 (Oct. 2016), 21 pages. https://doi.org/10.1145/2926718

[24] Mohammad Taher Pilehvar, Jose Camacho-Collados, Roberto Navigli, and Nigel Collier. 2017. Towards a Seamless Integration of Word Senses into Downstream NLP Applications. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, 1857–1869. https://doi.org/10.18653/v1/P17-1170

[25] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng, and Christopher Potts. 2013. Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP '13)*. 1631–1642.

[26] Maria Terzi, Matthew Rowe, Maria-Angela Ferrario, and Jon Whittle. 2014. *Text-Based User-kNN: Measuring User Similarity Based on Text Reviews*. Springer International Publishing, Cham, 195–206.

[27] Frank Wilcoxon. 1945. Individual comparisons by ranking methods. *Biometrics bulletin* 1, 6 (1945), 80–83.

4.2 Final Remarks

This work showed the results of four different item representations applied into two recommender systems on two largely different datasets, in a simulated cold start scenario. The results show sufficient evidence that indeed applying content-based item representations on recommender systems can alleviate the cold start problem.

Even though this study does not answer our research questions, it presents strong evidence that allows us to have some insights towards Research Questions 2 and 3: "Does increasing the semantics of item representations improve recommendation accuracy?" and "Is a kind of item representation more relevant to use in determined recommender system or scenario than another?".

Regarding Question 2, one can see that, especially when regarding the Item-MSMF results, there is indeed an improvement when we refine the semantics related to the representations based on concept embeddings. *Item embeddings*, while producing interesting results, is still inferior to *Mentioned similarity*, which can be seen as a semantic refinement of it. In order to produce *Mentioned similarity*, we need to first produce *Item embeddings*, which will then be used to calculate the similarity between the item and its concepts.

Regarding Question 3, we can see that, for the neighborhood approach, all results are very close, leaving the decision of which approach is best to use to other aspects such as data availability and/or space and resources constraints. On the other hand, for Item-MSMF, only two approaches present the best results (*Sentiment concepts* and *Mentioned similarity*) and both of them present similar distribution, i.e., they both have the same size and the same assigned concepts to each item, varying only in the weighting score associated with them. This highlights the notion presented in Chapter 2, that some recommender algorithms and scenarios are highly dependent on the item representation quality.

So far, we have been focused on evaluating the impact of item representations in the task of rating prediction, i.e., how much semantic item representations can benefit a recommender system in guessing what score a certain user would give to a certain item. In the following, we will turn our focus towards the task of top-N recommendation, in which the recommender aims to produce a relevant ranking of suggestions to the user, regardless of its predicted score. We also focus on Question 2, trying to further increase the semantics employed in describing items.

CHAPTER

5

COMBINING DIFFERENT METADATA VIEWS FOR BETTER RECOMMENDATION ACCURACY

5.1 Contextualization

In order to further increase the semantics aggregated by the item representations, we present in this Chapter a study in which we combine two representations: the one with SF-IDF weighting (MARINHO; D'ADDIO; MANZATO, 2017) (presented in Chapter 3) and the one based on sentiment-analysis (D'ADDIO *et al.*, 2018) (presented in Chapter 4). The reason why we decided to combine those two representations among the five that we produced during the entirety of the research is that those are the ones that have the most clear semantics and, in a sense, they complement each other: while SF-IDF is focused in describing items by a statistical view of intra and inter feature frequency, sentiment analysis is focused on the quality of the feature perceived by users. We try to combine them in a plethora of strategies, which we divide into three categories: i) pre combination, ii) neighborhood combination, and iii) post combination. The most successful combination is based on Bayesian Personalized Ranking (BPR) (RENDLE *et al.*, 2009), in which we define an adaptation of the work of Fortes and Manzato (2014).

We have performed an extensive amount of experiments, and the most expressive results are depicted in the following published article (D'ADDIO; MARINHO; MANZATO, 2019). In this work, we also move away from the rating prediction task and focus on the top-*N* recommendation task, evaluating if our representations are able to deliver relevant suggestions in the form of rankings.

Combining different metadata views for better recommendation accuracy

Rafael M. D'Addio[a,*], Ronnie S. Marinho[a], Marcelo G. Manzato[a]

[a]*Institute of Mathematics and Computer Science*
University of São Paulo
São Carlos, SP, Brazil

Abstract

Recommender systems emerged as means to help users deal with information overload by filtering content based on their preferences. Regardless of the recommendation method, there has been a recent interest in using user reviews as source of information, since they contain both detailed items' descriptions as well as users' opinions. Even though several works have been done in the subject, very few of them consider different views that the items may have towards their features, selecting only a method of weighting their features. In this work, we propose a system that combines two item representations that represent different views of the same feature set: one based on its statistics and the other based on its quality. Features are disambiguated concepts extracted from users' reviews. We propose several strategies divided into three categories: pre combination, neighborhood combination and post combination. We evaluate our strategies in two data sets, comparing them with each other and against the isolated item representations, as well as a representation baseline based on terms and sentiment analysis. Results are promising showing that some combinations are capable of producing better rankings than their isolated versions.

Keywords: Recommender systems, Item representation, Unstructured metadata

1. Introduction

Recommender systems nowadays are largely used as tools that aid users in their decision-making processes regarding what to use or consume, by filtering content according to their preferences. They can be classified into two important paradigms: content-based filtering and collaborative filtering, which determine how to evaluate data and make suggestions [1]. Content-based filtering recommends items to a user based on information relevant to the items. In this approach, items are represented based on their available content: metadata, synopses, reviews, among others. Users, in turn, have a profile that is inferred from the items they have evaluated, and the recommendation occurs by combining the profiles with the item representations. In collaborative filtering, in turn, only user evaluations and interactions are used to define relationships between themselves and/or between items. Hybrid methods, in turn, have gained prominence by combining aspects of both paradigms, minimizing recurrent problems in both [1].

Regardless of the method, there has been a growing effort to exploit user reviews for better descriptions of items or users in recommender systems [2]. Reviews are a great source of information since they contain relevant descriptions about items, as well as personal opinions regarding the item and its features.

Works that use reviews direct their efforts to better describe items [3, 4], users [5, 6] or both [7], but they propose simple feature extraction approaches that do not consider text ambiguity problems such as synonymy and polysemy. Given that, other works focus on solving those issues, but apply their efforts

*Corresponding author
Email addresses: rdaddio@icmc.usp.br (Rafael M. D'Addio), ronnieshida@usp.br (Ronnie S. Marinho), mmanzato@icmc.usp.br (Marcelo G. Manzato)

[1]**Publication Information:** Published in Information Systems. Volume 83, July 2019, Pages 1-12. DOI: https://doi.org/10.1016/j.is.2019.01.008

on impersonal texts such as synopses and Wikipedia[2] descriptions [8, 9, 10, 11, 12, 13, 14], discarding information about users' opinions which is important in recommender systems.

Furthermore, such works do not consider the different views that each item may have towards their features. In other words, they use a single method to give weights for their characteristics, based on some statistics, such as the frequency of the term, or whether certain feature is present or not in the item; or based on any quality quantification, such as sentiment analysis. A system would benefit greatly if it could describe its items in more than one way such that each representation's strength could be explored towards producing better recommendations.

In response to these limitations, we propose in this paper a strategy that combines two different item representations in a recommender system. The representations, based on user reviews, are constructed from features directly extracted from text by employing a technique that extracts disambiguated concepts. They are constructed with the same set of concepts, varying in the weighting process: i) a statistical view, with a TF-IDF-based weighting [15] and ii) a quality-based view, using sentiment analysis [16] to assess whether the items' features are good (positive) or bad (negative). While the first view captures similarities within the content of items, the second computes items' similarities based on opinions of other users. For instance, the statistical view may regard items as similar if they have the same specific and/or rare features mentioned with similar frequency on their reviews, while the quality-based view may regard them as similar if their features have closer levels of appreciation/depreciation by the reviewers (e.g. two movies can be similar if their suspense is praised while the lightning is not).

Finally, in order to explore the strengths of both representations, we propose and analyze several combination strategies, which are divided into three categories: pre combination, neighborhood combination and post combination. We compare them against each other and against the isolated representations, as well as a baseline representation based on simple terms weighted by sentiment analysis [3].

The remainder of the paper is structured as follows. Section 2 gives a comprehensive overview towards related works; Section 3 presents preliminary concepts required to understand our proposal; Section 4 details the construction of the item representations used in this study; Section 5 details the combination strategies; Section 6 details the experiments conducted in this study; and, finally, Section 7 presents the conclusions and future work.

2. Related works

In this section, we present some related works. We begin presenting some approaches that deal with the ranking problem, then describe works that use information from unstructured data to aid in recommendation.

Correctly ranking suggestions is an ongoing problem in recommender systems. Perhaps the most classic approach in learn-to-rank is the Bayesian Personalized Ranking (BPR) strategy, which was proposed by Rendle et al. [17]. It uses pairs of items, where one is known to the user and the other is not, as the basis for a Bayesian optimization criterion. More details on this method can be seen in Section 3.2. On the same note, Yagci et al. [18] proposed two shared memory lock-free parallelized pairwise learn-to-rank approaches based on BPR. In this work, among the item representation combinations, we propose two BPR-based learn-to-rank approaches as a post-processing step, in order to calculate weights to combine the rankings produced by the representations applied in isolation to the recommender system.

Beyond learn-to-rank, some works try to aggregate additional information into their models. For instance, Liang et al. [19] proposed CoFactor, a co-factorization model that simultaneously factorizes the user-item implicit feedback and item-item co-occurrence matrices, producing item embeddings that further aid the recommendation. The co-ocurrence matrix is obtained from the implicit feedback itself, thus not categorizing as an external information.

Several works aggregate information from unstructured data (reviews, synopsis, comments, among others) into recommender systems. Chen et al. [2] conducted a survey related to recommender systems that make use of user reviews. In the article, authors divide the works into two main categories: those

[2]https://www.wikipedia.org/

that use reviews to generate item representations, and those that use them to generate and/or increase user profiles.

Recent works that use reviews to describe items tend to extract features to characterize them according to their characteristics. For example, in previous work [3], different features extraction techniques were used to define feature and aspect-based item representations, which were applied in a neighborhood-based algorithm. The weights of the characteristics for each item were calculated considering the average sentiment observed in their reviews. In a different approach, McAuley et al. [4] used user reviews to establish substitution and/or complementation relationships between items. To do this, they propose a model that extracts latent topics and uses them to predict the two types of relationships in directed graphs.

On the other hand, works that make use of reviews for the construction of user profiles can construct profiles of finer granularity which explain the preferences of the users in relation to the diverse characteristics of the items. An example is the work of Ganu et al. [5] which uses opinion mining in a restaurant recommendation system based on soft clustering. Terzi et al. [6], in turn, proposed a user-based neighborhood algorithm, whose similarities are calculated based on user reviews. The authors explore six alternatives for similarity metrics, all based on the WordNet[3] taxonomy. Regardless of the choice of the metric, the similarity between two reviews is calculated from the average similarity among all possible pairs of terms. However, the authors do not use lexical disambiguation techniques, but instead adopt the best combination of all the possible meanings of the terms.

Some papers aim to describe both users and items. For example, in the work of Chen et al. [7], a matrix factorization model is proposed that makes use of reviews to rank the features of products that are most relevant to the users preferences. The recommendation is based on this ranking, which is combined with the quality of the characteristics in the items, measured through opinion mining, and latent factors based on ratings.

None of the aforementioned approaches deal with text ambiguity problems such as polysemy and synonymy. On the other hand, there are efforts to minimize these problems on recommender systems, but they focus majorly on impersonal texts such as synopses and Wikipedia descriptions.

The WordNet taxonomy and its word-sense disambiguation (WSD) algorithms have been actively explored in the last decade by content-based and hybrid recommender systems. For example, De Gemmis, Lops, and colleagues extensively explored a strategy of constructing item representations and user profiles based on WSD of item synopses, and applied it in a variety of contexts, such as user-based hybrid recommender systems [10], content-based systems intended for multilingual recommendation [11], among others. Such a strategy divides texts into compartments, e.g. title, authors, body text, etc., which are processed by a WordNet-based WSD algorithm to produce different representations, whose weights are the synsets frequencies. Capelle et al. [9] proposed a content-based news recommender system that makes use of WSD based on the WordNet taxonomy for the construction of item representations and user profiles. They also extend their SF-IDF metric, introduced in [8], incorporating in the calculation a similarity based on page count returned by the Bing search engine[4] when querying for named entities present in the text. The SF-IDF weighing is an extension of the traditional TF-IDF [15], with the difference that features are synsets rather than terms. Systems that use the WordNet senses repository are limited by the low coverage of named entities. Although WordNet contains some instances of people, places and companies, the taxonomy is not very broad.

On the other hand, some works exploit entity linking (EL) techniques to perform feature extraction. For example, Musto et al. [12] propose a content-based system that uses EL to produce item descriptions to support the system to generate context-sensitive recommendations. Several different entity linking techniques are used to extract concepts directly from the Wikipedia, which are expanded by extracting their immediate ancestors in the Wikipedia category tree.

Finally, efforts have been made to incorporate techniques that accomplish both WSD and EL in content-based recommendation systems, such as the work of Narducci et al. [13] and Oramas et al. [14].

[3]https://wordnet.princeton.edu/
[4]https://www.bing.com/

In [13], the authors evaluated four different types of representations in the multilingual recommendation task, two based on Wikipedia and two based on BabelNet[5] [20]. In [14], the BabelFy[6] [21] tool is used to extract concepts related to music and sounds, which are used to automatically produce a knowledge base in the form of a graph. The item representations are constructed by mapping the graph in a vector space from two strategies: entity-based and path-based.

All the aforementioned works deal with only one type of item representation, built using a single feature extraction technique and weighting scheme. In a previous study [22], we found evidence that using more than one item representation may be benefical for recommender systems. In that work, we extracted features from four different techniques (two based on term extraction, one based on named entity recognition and one based on topic hierarchy) and constructed binary item representations (an item has or not a feature), and tested them into two recommender algorithms. Combination of the results were performed by a ranking ensemble based on BPR.

Our work differs from the aforementioned since we analyze a vast amount of approaches for correctly combining two item representations that characterize items in two different views of the same set of characteristics: one statistical and one regarding the item's quality. The characteristics used in this study also provide more semantics to the representations, since they are disambiguated concepts instead of simple terms.

3. Preliminary concepts

In this section we present some preliminary concepts for understanding our approach. First, we present the recommender algorithm used in our experiments (Section 3.1). Then, we briefly describe the Bayesian Personalized Ranking [17] optimization strategy (Section 3.2), which is extended in two of our ranking combination strategies.

3.1. Item k-NN

The recommendation algorithm in which we employ our items representations is the well-known Item k-NN, a collaborative filtering approach which calculates a neighborhood for each item and is used to derive a score for an unknown user-item pair [23].

This algorithm uses a similarity measure to obtain the neighborhood, whose size is set through a parameter k. In traditional Item k-NN, the similarity is calculated through items representations based on user rating and/or interaction patterns. In previous works, we have replaced the rating-based representations with feature-based ones derived from text [3].

Several similarity measures can be applied, such as cosine and Pearson correlation coefficient. In our work, we decided to use the latter based on results of preliminary experiments. The Pearson correlation coefficient can be calculated as [23]:

$$p_{ij} = \frac{\sum_{n=1}^{k}(w_n^i - \overline{w_i})(w_n^j - \overline{w_j})}{\sqrt{\sum_{n=1}^{k}(w_n^i - \overline{w_i})^2}\sqrt{\sum_{n=1}^{k}(w_n^j - \overline{w_j})^2}}, \tag{1}$$

where $\overline{w_i}$ and $\overline{w_j}$ are the average values of the features of items i and j.

In the item recommendation scenario, the score $\hat{r}_{ui}$ of an item i to a user u is simply the sum of the similarities of all other items j in the neighborhood $N_u(i)$, which is an intersection between the k most similar items and the set of items that u has interacted with [17]:

$$\hat{r}_{ui} = \sum_{j \in N_u(i)} p_{ij} \tag{2}$$

[5] http://babelnet.org/
[6] http://babelfy.org/

This score does not represent a rating that a user may provide to an item; instead, it is a score that represents the likelihood that a user may enjoy such item. The collection of scores of unknown items are used then to construct a ranking of suggestions that is provided to the user.

3.2. BPR Learning

The Bayesian Personalized Ranking strategy proposed by Rendle et al. [17] addresses the problem of ranking optimization by considering in its calculation the relationship between a known item i and an unknown item j. In order to do that, a set D_k is constructed containing all possible triples (u, i, j), where i is the item known to user u and j is the unknown. With this set at hand, a generic BPR optimization criterion is formulated:

$$\text{BPR-Opt} := \sum_{(u,i,j) \in D_K} \ln \sigma(\hat{x}_{uij}) - \lambda_\Theta ||\Theta||^2 \quad , \tag{3}$$

where $\hat{x}_{uij}$ is a real value that captures the relationship between a triple $(u, i, j) \in D_k$ in a given model (e.g. matrix factorization) with Θ as parameters, and λ_Θ the regularization values for the model parameters. The $\hat{x}_{uij}$ value can be decomposed in:

$$\hat{x}_{uij} = \hat{r}_{ui} - \hat{r}_{uj} \tag{4}$$

The goal of the BPR Learning algorithm is to maximize BPR-Opt through stochastic gradient descent. In this sense, the authors suggest to use a bootstrap sampling approach with replacement to select triples on which the following update formula is performed:

$$\Theta \leftarrow \Theta + \eta \left(\frac{1}{1 + e^{\hat{x}_{uij}}} \cdot \frac{\partial}{\partial \Theta} \hat{x}_{uij} + \lambda_\Theta \Theta \right), \tag{5}$$

where η is a learning rate constant.

4. Item representations

In this paper we focus our efforts into combining two item representations created from user reviews. They differ mainly in the way that the features are viewed, since both representations use the same feature set in their construction but are weighted differently. In this sense, one representation focuses on the statistics of an item's features, while the other focuses on their quality. We use two external natural language processing (NLP) tools to extract features from user reviews: i) Stanford CoreNLP[7] [24] and ii) BabelFy [21]. In the following subsections we detail the steps taken in constructing the representations.

4.1. Text pre-processing and feature extraction

First, we process the items' reviews with the well-known Stanford CoreNLP [24], a natural language processing toolkit that contains several NLP routines. There, we perform routines such as tokenization, lemmatization, stemming, parsing, part-of-speech (POS) tagging and sentence splitting. We also execute a sentiment analysis algorithm in this toolkit, whose results are used in one of our item representations (as it can be seen in Section 4.3). The output of this process are XML files structuring the texts into sentences and their components, as well as their parser and sentiment.

From there, we reconstruct the documents into plain text so they can be processed by the other NLP tool, BabelFy [21]. BabelFy combines the tasks of word sense disambiguation and entity linking, extracting from raw text babel synsets, which are disambiguated concepts from the BabelNet [20] knowledge database. BabelNet condenses into one big database entries from Wikipedia and WordNet, and provides links between them. The reason for reconstructing the documents is because we preserve the Stanford

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

CoreNLP sentence splitting, which is necessary to match BabelFy's output with the sentiments produced by the previous tool, thus allowing us to build one of our item representations.

The documents processed by BabelFy are text files where words are replaced by disambiguated babel synsets. From there, we extract candidate synsets that will compose our feature set, or vocabulary. First, we maintain in a list only the synsets that are nouns. Since this set is still very large, we use a filter based on the item frequency (IF) [25], which is an adaptation from the traditional document frequency (DF) [15] where a set of item's reviews is viewed as a single document. Let F be the synsets (features) set and I the items' set, the item frequency IF_f of a feature f is:

$$IF_f = \sum_{i}^{|I|} k_{if}, \tag{6}$$

where k_{if} is equal to 1 if an item i contains the synset in at least on of its reviews, and 0 otherwise.

Having calculated the IF_f of each synset, the filter consists of removing those whose value is inferior to a given threshold. In a previous study [26], we determined that a good threshold is the value of 10. We adopt this value in this study. The remaining synsets after this step will constitute the vocabulary, which will be used to produce the items representations that are detailed in the next subsections. Both representations are modeled in a vector space, where each feature corresponds to a dimension in that space.

4.2. SF-IDF

The first item representation strategy used in this work aims to characterize items according to a statistical view of their features. As in our previous work [26], we use the strategy proposed by Capelle et al. [8], where features are weighted by the SF-IDF, an extension of the well known TF-IDF metric [15].

This technique analyzes the synset frequency in a single item (SF) and its rarity among the whole item collection (IDF). Formally, we denote as $|I|$ the number of items of the system, and a synset s may appear in $|i : s \in I|$ items. We also denote as $n_{s,i}$ the frequency of a synset s in an item i. The synset frequency $SF(s, i)$ can be defined as [8]:

$$SF(s, i) = \frac{n_{s,i}}{\sum_k n_{k,i}}, \tag{7}$$

where $\sum_k n_{k,i}$ corresponds to the total frequency of the k other synsets present in i.

The inverse document frequency, IDF_i, is:

$$IDF(s) = \log \frac{|I|}{|i : t \in i|}. \tag{8}$$

Finally, the SF-IDF of a synset-item pair can be given as:

$$SF - IDF(s, i) = SF(s, i) \times IDF(s). \tag{9}$$

4.3. Sentiment

The second item representation strategy focuses on measuring the quality of the features of an item. In this sense, each synset is weighted according to the overall sentiment (e.g. positive, negative or neutral) that users assign in their reviews toward them. Thus, an item is represented by the average sentiment of many users' reviews towards each of its characteristics.

In order to do that, we use the sentiment analysis algorithm available at the Stanford CoreNLP toolkit [16], obtaining the sentiment for each sentence of each review. The main reason for using a sentence-level sentiment analysis is that since all of the features extracted from the reviews are nouns, which in most cases have neutral sentiment, a natural approach to obtain their sentiment is to match with sentiments obtained from the context they were used in the text. Moreover, since features are extracted from a

different tool than CoreNLP, and they are concepts instead of words, it is justifiable to use the sentiment of the sentences they occur.

The sentiment analysis algorithm [16] uses recursive neural networks models to build representations that capture the structure of the sentences, obtaining their sentiment based on the meaning of each of words. It classifies sentences in five sentiment levels: "Very Negative", "Negative", "Neutral", "Positive" and "Very Positive". We convert this classification into a $[1, 5]$ rating system, being 1 equals to "Very Negative" and 5 equals to "Very Positive".

We use the following strategy to obtain the sentiments of the synsets, adapted from previous works [3]. First, the system analyzes the feature set, then, for each item, reads the BabelFy output texts, finding and storing the sentences IDs that are related to each of the synsets. Then, it matches the IDs with those of the CoreNLP files, storing their corresponding sentiments.

After obtaining the sentences related to each feature, the next step is the sentiment attribution. For each feature of each item it is calculated the average sentiment of the related sentences. Thus, this value represents the collective level of appreciation or depreciation of a certain attribute of an item. This approach does not consider the level of confidence that a final sentiment score may have: a score calculated from several mentions to a feature from several different reviews is more reliable than a score calculated from a single mention. In order to address this, we devised a heuristic where we dampen the sentiment value, i.e., approximate it to 3 (the neutral value), if the number of times it was mentioned is lower than the global average number of mentions. Formally, a sentiment $S(s, i)$ of a synset s in relation to an item i can be dampen by the following formula, if its mentions are lower than the global average:

$$S(i,j) = \begin{cases} S(s,i) + log_{10}(n_s/N) & \text{if S(s,i)} > 3 \\ S(s,i) - log_{10}(n_s/N) & \text{if S(s,i)} < 3 \end{cases}, \tag{10}$$

where n_s is the number of times the synset was mentioned in the item's reviews, and N is the global average number of mentions. With this heuristic, we are able to dampen the sentiment strength to a maximum of 1, still preserving its original orientation.

Finally, a zero value indicates that an item simply does not have that feature.

5. Recommendation strategy: combining representations

As mentioned earlier, this work explores strategies to accurately combine the two item representations described in the previous section. Those strategies were elaborated to be applied in one of the steps taken during the recommendation, which is performed by the item-kNN algorithm.

We explore several strategies which are grouped into three main categories: i) pre combination, which combines the item representations before they are processed by the algorithm; ii) neighborhood combination, which combines the k-NN produced by the two representations; and iii) post combination, which combines the rankings produced by each item-kNN execution with each representation. Figure 1 illustrates our proposal.

In the following subsections we detail every strategy implemented in this study.

5.1. Pre combination

The pre combination comprises two heuristics to combine directly the items representations, i.e., combine the two item's vectors. This is done before the recommendation process begins, and the new item representation feed the recommender algorithm, which runs normally. The two heuristics are:

- *Multiplication:* In this strategy, the scores of the two representations are multiplied. In this sense, for each feature f in the vocabulary, its final score w_{fi} for a determined item i is:

$$w_{fi} = w^S_{fi} * w^{SF}_{fi}, \tag{11}$$

  where w^S_{fi} and w^{SF}_{fi} are the scores provided by the sentiment and the SF-IDF representations, respectively. Since both representations have the same structure, i.e., they only differ in the weight that each feature has, there is no possibility that scores are nullified by zero multiplications. Thus, the resulting item representation cannot be more or less sparse than the originals.

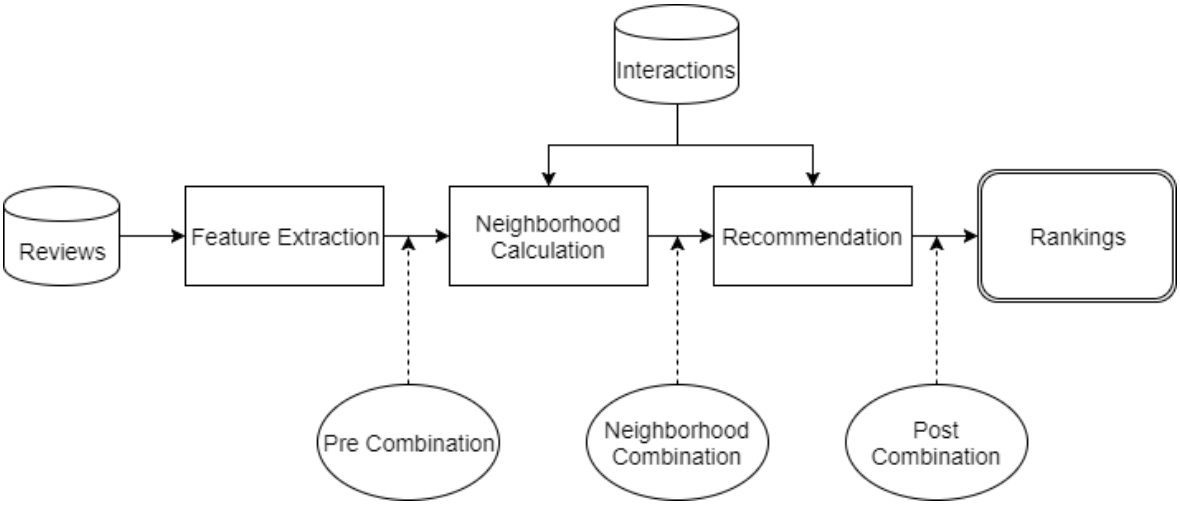


Figure 1: The proposed system's architecture.

- *Concatenation:* In this strategy, we produce a final representation by concatenating both representations. The concatenation is performed at each feature, i.e., each feature is designed to comprise two dimensions of the feature space. In this sense, the final representation will have twice the number of features of the original, thus aggravating its sparsity.

5.2. Neighborhood combination

The neighborhood combination comprises four strategies to combine the neighborhoods produced by each of the representations. In this sense, the item-kNN is executed separately for each item representation up to the point where it generates the neighborhood for an unknown user-item pair. Then, one of the strategies is employed to generate a final neighborhood, which will then be used to produce the pair's recommending score. The strategies are based on either one of the two set of operations:

- *Intersection:* the strategies based on this operation aim to maintain only the items which are present on both neighborhoods. Thus, given the sentiment neighborhood $N_u(i)^S$ and the SF-IDF neighborhood $N_u(i)^{SF}$, the final set is:

$$N_u(i) = N_u(i)^S \cap N_u(i)^{SF}. \tag{12}$$

  The two strategies based on this operation differ only in the score calculation of the neighbors. One of them maintains the highest score among the two neighborhoods, while the other calculates their average. A sorting algorithm can also be applied as a final step. No further cuts to respect the k size of the neighborhood is needed, since at the worst case (both sets are the same), they still have their sizes limited previously by k.

- *Union:* the two remaining strategies are based on this operation, aiming to provide a combined ranking of neighbors. Thus, the final set of neighbors can be viewed as:

$$N_u(i) = N_u(i)^S \cup N_u(i)^{SF}. \tag{13}$$

  In order to produce the union of the two neighborhoods, a constraint must be fulfilled: the resulting set must not be larger than the k limit. To guarantee that, we follow some steps. First, both neighborhood are sorted from highest to lowest similarity. Next, we combine both sets into a single one. Similarly as in the intersection approach, the two strategies based on union differ only in the calculation of the similarity of neighbors that are present in both sets: one maintains the highest value, while the other calculates an average score. Since we employ those strategies in situations

where there may be items appearing in both sets, the output of the combination is not necessarily sorted. Thus, we sort it, producing a final ranking of neighbors. Finally, we make a cut in the k-th position of the ranking, maintaining only the most similar neighbors for both representations.

5.3. Post combination

In this category, we explore strategies that aim to combine user rankings generated by item-kNN for each item representation. In this sense, first we run each representation separately in the recommender and, as a post-processing step, we combine the resulting rankings.

We explore four strategies that can be divided into two very different approaches. Two of them are based on heuristics, while the other two are based on machine learning. In the next subsections they are described in details.

5.3.1. Heuristic strategies

The heuristic strategies, similarly to the neighborhood combination ones, base themselves on set operations. Specifically, those strategies use the union operation. Given a ranking $R_S(u)$ for the sentiment representation execution of the item-kNN, and a ranking $R_{SF}(u)$ for the SF-IDF version, the final ranking $R(u)$ for a user u is:

$$R(u) = R_S(u) \cup R_{SF}(u). \tag{14}$$

In order to perform the union of two rankings, we follow the same steps performed in the neighborhood strategy. As rankings are already sorted, we start by combining both rankings. As before, we treat item repetitions in two different manners: i) we consider only the highest value of ranking score; ii) we sum the two values, since we follow the premise that if an item is present in both rankings, it probably will be more relevant to the user. With that, the output from the merge-sort algorithm may not be sorted, requiring us to perform it after its completion. Finally, we cut the ranking in a predetermined size (e.g. 10), since a large ranking may be overwhelming for the user.

We don't use intersection operations since there is a probability that the resulting ranking is very small, or even empty. In a worst case scenario, each ranking will contain a different set of items, resulting in an empty final ranking.

5.3.2. BPR Learning strategies

The machine learning strategies are based on BPR Learning, a concept we introduced in Section 3.2. Here, we use BPR to learn weights for a correct combination of two rankings. The idea, originally proposed in [27], modified the $\hat{x}_{uij}$ decomposition (Equation 4) into:

$$\hat{x}_{uij} = \alpha_a(\hat{r}^a_{ui} - \hat{r}^a_{uj}) + \alpha_b(\hat{r}^b_{ui} - \hat{r}^b_{uj}), \tag{15}$$

where α_a and α_b are weights assigned to predictions from recommender a and recommender b. In that work, in fact, the main goal was to use this strategy to combine multiple user interactions (such as ratings, purchase history and tag assignment), thus the equation above could be incremented with any amounts of α weights and their related predictions. The α weights were trained simultaneously with the recommender system, which was based on matrix factorization, resulting in the algorithm with full potential when using several different training data.

In our work, in turn, we restrict the calculation to only two sets of predictions. With this, a and b are not recommenders or types of interactions, but in fact two executions of the same recommender with different item representations. Moreover, we do not train any parameter of the recommender system, delegating the BPR training to a solely post-processing step. Thus, $\hat{r}^S_{ul}$ and $\hat{r}^{SF}_{ul}$ represent the predictions generated by sentiment and SF-IDF item representations (where l can be either i or j), as well as α_S and α_{SF}.

Applying the stochastic gradient descent, the update equation (5) can be employed, with the $\frac{\partial}{\partial\Theta}\hat{x}_{uij}$ component substituted by its corresponding derivative:

$$\frac{\partial}{\partial\Theta}\hat{x}_{uij} = \begin{cases} \hat{r}^S_{ui} - \hat{r}^S_{uj}, & \text{if } \Theta = \alpha_S \\ \hat{r}^{SF}_{ui} - \hat{r}^{SF}_{uj}, & \text{if } \Theta = \alpha_{SF} \end{cases}. \tag{16}$$

Another approach we experimented is to regard the known pairs $\hat{r}_{ui}$ as 1, since they are indeed always relevant for the user. With that, the Equations 15 and 16 can be rewritten as:

$$\hat{x}_{uij} = \alpha_S(1 - \hat{r}^S_{uj}) + \alpha_{SF}(1 - \hat{r}^{SF}_{uj}), \tag{17}$$

$$\frac{\partial}{\partial\Theta}\hat{x}_{uij} = \begin{cases} 1 - \hat{r}^S_{uj}, & \text{if } \Theta = \alpha_S \\ 1 - \hat{r}^{SF}_{uj}, & \text{if } \Theta = \alpha_{SF} \end{cases}. \tag{18}$$

Regardless of the approach, the process of learning the alphas is performed similarly as the original BPR Learning algorithm: we perform bootstrap sampling to select random triples from the vast amount of possible candidates and employ them to learn the *alpha* weights. After learning them, we multiply each ranking scores by their respective weights and then merge them both. In case of repetitions, i.e., both rankings have the same item, we sum their scores. It is important to note that the weights are not applied to all items, since the rankings are significantly smaller than the total number of items. Nevertheless, we experimented with several ranking sizes and determined that rankings of 100 items are optimal, since items beyond that position may not be very interesting for the final user. Finally, we sort the resulting ranking and perform a cut in a predetermined size (e.g. 10).

6. Evaluation

In this section we detail the experiments conducted in this study. In the following subsections we present the datasets used, the experimental setup and the results we obtained.

6.1. Data sets

We used two largely different data sets to evaluate our proposal.

The first is the well-known MovieLens 100k (ML-100k)[8] data set, which consists of $100,000$ ratings (from 1 to 5) performed by 943 users for $1,682$ movies. Each user of the data set has rated at least 20 movies. In order to perform our experiments, we collected up to 10 reviews per item from the IMDb[9] website, resulting in a total of $15,863$ documents. The reviews were selected as the website's top-10, ordered by their helpfulness.

The second is one of the Amazon[10] data sets extracted by McAuley et al. [28][11]. They extracted many large datasets based on review data from Amazon, each representing a single domain of items: movies, books, electronics, among others. Given that most of them are extremely large, we selected the Apps for Android data set, which is the fifth biggest data set from the collection. The original dataset has $2,638,172$ ratings and $752,937$ reviews from $1,323,884$ users for $61,275$ items, where users and items have at least 5 interactions. Given that it is a fairly large and sparse data set, we filtered it by maintaining only the reviews interactions and eliminated the items and users that had less than 10 interactions, resulting in $16,201$ users and $4,869$ items, totaling $264,047$ interactions.

As one can see, the two data sets are very different. To begin with, the user-item proportion is different, with the first data set having 1.78 times more items than users, while the second has 3.33 times more users. Furthermore, the second data set is significantly more sparse than the first: while it has 2.64 times more interactions than the previous, its user-item matrix is significantly larger: 17.18 times more rows (users) and 2.89 times more columns (items).

[8] https://grouplens.org/datasets/movielens/100k/
[9] http://www.imdb.com/
[10] https://www.amazon.com/
[11] http://jmcauley.ucsd.edu/data/amazon/

6.2. Experimental setup and evaluation metrics

In order to check the strength of our proposal, we compare our work with another baseline item representation proposed in a previous work [3]. We also regard as baselines the isolated item representations based on BabelFy described in Section 4.

As for the baseline representation from [3], called *heuristic terms*, it is based on lematized terms extracted from the same reviews used in this study. These terms go through heuristics that filter by the part-of-speech tag, maintaining only nouns, and eliminate the less frequent terms, using IF (Equation 6) with threshold 30. Weighting of the terms is based on average sentiment, using the Stanford CoreNLP algorithm. Table 1 presents the sizes of the vocabularies of the item representations in both data sets, as well as those produced in the pre combination.

Table 1: Vocabularies sizes for the item representations regarded in this study.

	ML-100k	Amazon Apps
Heuristic terms	3085	3089
BabelFy	6238	8978
Pre comb. multiplication	6238	8978
Pre comb. concatenation	12,476	17,956

The baseline representations, as well as the combinations, were all executed in the item k-NN algorithm (Section 3.1), with $k = \{20, 40, 60, 80, 100\}$ and Pearson correlation coefficient (Equation 1) as the similarity measure. All rankings produced were of maximum size of 100 per user.

Regarding the post combination techniques, the cuts performed in the final rankings were also of size 100. The techniques based on BPR learning used learning rate $\eta = 0.05$ and regularization constants λ_S and $\lambda_{SF} = 0.0025$, defined experimentally.

We also compare our findings against the classical BPR-MF learn-to-rank approach [17], and two state-of-the-art ranking approaches. The first one, called CoFactor [19], perform co-factorization on both user-item implicity feedback and item co-occurence matrices. The second, proposed by Yagci et al. [18], presents two parallel pairwise learn-to-rank approaches, called PLTRN and PLTRB. We apply those baselines in our experimental setting, using their own provided source code. We save their generated rankings and evaluate them in our evaluation protocol.

All baselines require hyperparameter tuning. For BPR-MF, PLTRN and PLTRB, we ran their implementations provided by Yagci et al. [18], with a fixed seed for random samplings. As for the parameters, we ran a grid search for learning rate η and number of factors F, obtaining $\eta = 0.005$ and $F = 20$ for ML-100k, and $\eta = 0.01$ and $F = 20$ for Amazon Apps. Regularization parameters were defined as $\lambda_{\theta_u} = \lambda_{\theta_i} = 0.0025$ and $\lambda_{\theta_F} = 0.00025$, as reported by the author. As for CoFactor [19], the authors define their hyperparameters by modifying them according to a scale ℓ, while also grid searching for the negative sampling value k. We follow this approach and grid search for both values, obtaining $\ell = 0.1$ and $k = 1$ for ML-100k and $\ell = 1$ and $k = 10$ for Amazon Apps.

We evaluated the quality of the rankings with the mean average precision at N (MAP@N) measure. It evaluates a ranking of size N, giving a greater weight for occurrences of relevant items in early positions of the ranking. It is a measure that produces a value which corresponds to the average of j queries, where each query produces a ranking and a score that is the average of different n precision levels. Formally, let $\{i_1, \ldots, i_{m_j}\}$ be the set of relevant items for a query $q_j \in Q$, and R_{jk} be the set of results returned from the first item until the i_k item, then the MAP can be measured as [15]:

$$MAP = \frac{1}{|Q|} \sum_{j=1}^{|Q|} \frac{1}{m_j} \sum_{k=1}^{m_j} \frac{\#(\text{relevant items in } R_{jk})}{R_{jk}}. \tag{19}$$

We evaluate our results in four different N values of MAP: 1, 5 and 10. We do this in order to evaluate if our solution is capable of returning relevant items as the ranking increases in size, up to 10.

All experiments were carried out in a 10-fold cross validation setting. The data set was randomly divided into 10 folds, where each fold contained at least one interaction of each user. We define as

training set 9 out of the 10 folds, and use the remaining as test. We iterate 10 times this procedure, varying the fold used on the test set. We report as results the average values of the iterations. In order to check their significance, we applied the two-tailed Wilcoxon signed-ranks test [29].

6.3. Results and discussion

In this section we present and discuss the results obtained by our approaches. Table 2 presents all results obtained in the experiments. All approaches and corresponding descriptions are summarized as follows:

- Pre comb 1: The pre combination approach detailed in Section 5.1 with the concatenated representations.
- Pre comb 2: The pre combination approach detailed in Section 5.1 with the multiplied scores of the representations.
- Neighbor comb 1: The neighborhood combination approach detailed in Section 5.2 with the intersection between the neighbors, using average values of similarity.
- Neighbor comb 2: The neighborhood combination approach detailed in Section 5.2 with the intersection between the neighbors, using the highest value of similarity.
- Neighbor comb 3: The neighborhood combination approach detailed in Section 5.2 with the union between the neighbors, using average values of similarity in case of repetitions.
- Neighbor comb 4: The neighborhood combination approach detailed in Section 5.2 with the union between the neighbors, using the highest value of similarity in case of repetitions.
- Post comb 1: The post combination approach detailed in Section 5.3 with the union between the rankings, considering the highest ranking score in case of repetitions.
- Post comb 2: The post combination approach detailed in Section 5.3 with the union between the rankings, summing ranking scores in case of repetitions.
- Post comb BPR 1: The post combination approach detailed in Section 5.3 with the BPR Learning algorithm considering the known pairs' predictions as 1.
- Post comb BPR 2: The post combination approach detailed in Section 5.3 with the BPR Learning algorithm using the known pairs' predictions.

In Table 2, numbers in bold represent the best results for each column, i.e. the best result for determined metric in determined number of k. Values with the symbol $*$ represent statistical significance with $p \leq 0.05$, and with the symbol $**$ represent significance with $p \leq 0.01$, both over all other approaches and baseline representations. In the following subsections we detail the results of each data set separately, and, finally, provide a discussion summarizing them.

6.3.1. MovieLens 100k

Here we detail the results related to the MovieLens 100k data set. We first compare the results between each category of combination approaches, and then compare the best of each of them against each other.

Regarding the pre combination approaches, the item representations concatenation (i.e. Pre comb 1) showed the category's best results, being statistically superior with $p \leq 0.01$ against the multiplication approach. It also provided statistically superior results against two of the baseline representations: Heuristic terms and BabelFy SF-IDF. In relation to BabelFy Sentiment, it is statistically superior with $p \leq 0.05$ only for $k = 20$.

Concerning the neighborhood approaches, the neighbor union with highest similarity in repetitions (i.e. Neighbor comb 4) provided the best results, being statistically superior with $p \leq 0.01$ against all other approaches of this category, and the Heuristic terms and Babelfy SF-IDF baseline representations.

MovieLens 100k

	k = 20			k = 40			k = 60			k = 80			k = 100		
	MAP@1	MAP@5	MAP@10	MAP@1	MAP@5	MAP@10	MAP@1	MAP@5	MAP@10	MAP@1	MAP@5	MAP@10	MAP@1	MAP@5	MAP@10
Isolated Representations															
Heuristic Terms	0,15917	0,22473	0,22001	0,14613	0,21481	0,20968	0,15631	0,22197	0,21587	0,15854	0,22068	0,21387	0,15684	0,21839	0,21391
Babelfy SF-IDF	0,15037	0,22472	0,22366	0,14793	0,22092	0,22226	0,14666	0,22010	0,22046	0,14783	0,22052	0,22019	0,14952	0,22131	0,22035
Babelfy Sentiment	0,20117	0,26343	0,25457	0,19576	0,25911	0,25052	0,19427	0,26097	0,25233	0,18961	0,25742	0,24979	0,18335	0,25168	0,24390
Pre combinations															
Pre comb 1	0,20530	0,26765	0,25685	0,19067	0,25575	0,24777	0,19300	0,26027	0,25146	0,19162	0,25869	0,25036	0,18441	0,25196	0,24385
Pre comb 2	0,14995	0,22065	0,22100	0,14719	0,21769	0,21769	0,15143	0,21975	0,21877	0,15483	0,22230	0,22048	0,15589	0,22300	0,22043
Neighborhood combinations															
Neighbor comb 1	0,16253	0,18043	0,17910	0,14467	0,19467	0,19316	0,15899	0,21362	0,21114	0,16939	0,22140	0,21683	0,15878	0,21100	0,20773
Neighbor comb 2	0,16497	0,18345	0,18332	0,13460	0,18687	0,18550	0,15189	0,20913	0,20724	0,16939	0,21998	0,21574	0,15581	0,20866	0,20635
Neighbor comb 3	0,15663	0,23323	0,22969	0,17614	0,24891	0,24292	0,17847	0,25078	0,24577	0,17678	0,24743	0,24204	0,17200	0,24397	0,23925
Neighbor comb 4	0,17413	0,25175	0,24566	0,18844	0,26326	0,25555	0,19332	0,26642	0,25943	0,19046	0,26289	0,25622	0,19024	0,26072	0,25467
Post combinations															
Post comb 1	0,15546	0,23437	0,23086	0,17381	0,24430	0,23955	0,18505	0,25440	0,24838	0,18632	0,25438	0,24795	0,18282	0,25187	0,24470
Post comb 2	0,14539	0,21861	0,21710	0,14422	0,21264	0,21118	0,13563	0,20343	0,20254	0,12057	0,18699	0,18883	0,11453	0,17882	0,18093
Post comb BPR 1	**0,21644**	0,28924	0,27681	**0,20933***	**0,28196****	**0,27239****	**0,21029****	**0,28421****	**0,27437****	**0,21039****	**0,28187****	**0,27154****	**0,20265***	**0,27389****	**0,26603****
Post comb BPR 2	0,21304	**0,28960**	**0,27950**	0,19618	0,27273	0,26619	0,19135	0,27143	0,26415	0,18706	0,26137	0,25616	0,17847	0,25281	0,24785

Amazon Apps

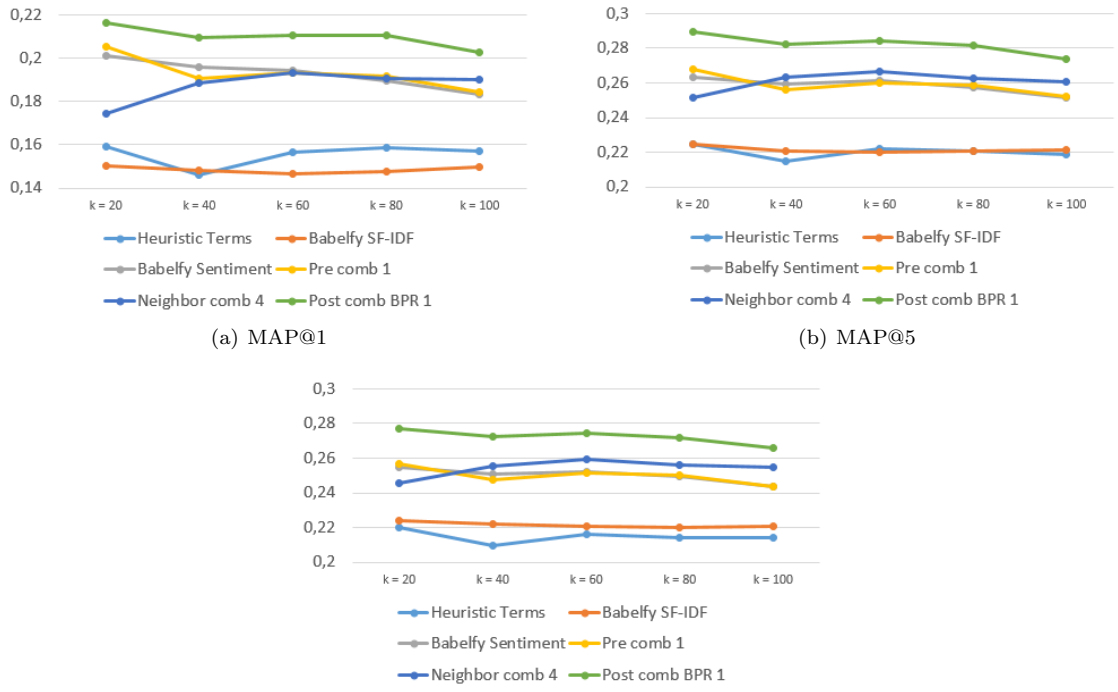
	k = 20			k = 40			k = 60			k = 80			k = 100		
	MAP@1	MAP@5	MAP@10	MAP@1	MAP@5	MAP@10	MAP@1	MAP@5	MAP@10	MAP@1	MAP@5	MAP@10	MAP@1	MAP@5	MAP@10
Isolated Representations															
Heuristic Terms	0,01865	**0,03461****	**0,03868****	0,01964	0,03488	0,03936	0,01816	0,03416	0,03882	0,01852	0,03444	0,03919	0,01876	0,03445	0,03929
Babelfy SF-IDF	0,01736	0,02817	0,03085	0,01831	0,03043	0,03333	0,01978	0,03248	0,03562	0,02070	0,03387	0,03717	0,02112	0,03456	0,03792
Babelfy Sentiment	0,01366	0,01979	0,02114	0,01489	0,02357	0,02558	0,01626	0,02767	0,02969	0,01564	0,02820	0,03162	0,01604	0,02921	0,03259
Pre combinations															
Pre comb 1	0,01475	0,02485	0,02782	0,01479	0,02715	0,03061	0,01512	0,02815	0,03199	0,01499	0,02800	0,03197	0,01497	0,02817	0,03222
Pre comb 2	0,01731	0,02808	0,03079	0,01850	0,03052	0,03350	0,02003	0,03249	0,03571	0,02042	0,03332	0,03661	0,02079	0,03413	0,03749
Neighborhood combinations															
Neighbor comb 1	0,00770	0,00773	0,00773	0,00636	0,00641	0,00641	0,00611	0,00627	0,00627	0,00610	0,00622	0,00622	0,00591	0,00606	0,00607
Neighbor comb 2	0,00671	0,00677	0,00677	0,00583	0,00587	0,00587	0,00579	0,00591	0,00592	0,00561	0,00567	0,00568	0,00624	0,00638	0,00638
Neighbor comb 3	0,01797	0,03067	0,03416	0,01899	0,03308	0,03693	0,01992	0,03458	0,03867	0,02000	0,03504	0,03936	0,02004	0,03525	0,03964
Neighbor comb 4	0,01923	0,03221	0,03559	0,02006	0,03434	0,03816	0,02108	0,03600	0,04004	0,02160	0,03689	0,04111	0,02153	0,03716	0,04156
Post combinations															
Post comb 1	0,01787	0,03011	0,03340	0,01913	0,03277	0,03634	0,02007	0,03434	0,03827	0,01984	0,03473	0,03888	0,02035	0,03515	0,03932
Post comb 2	0,01308	0,02352	0,02668	0,01376	0,02524	0,02870	0,01462	0,02655	0,03019	0,01466	0,02697	0,03086	0,01460	0,02699	0,03079
Post comb BPR 1	0,01857	0,03254	0,03600	0,02030	0,03560	0,03926	0,02189	0,03764	0,04166	0,02310	0,03948	0,04356	0,02308	0,03987	0,04407
Post comb BPR 2	**0,01992***	0,03329	0,03683	**0,02120***	**0,03609**	**0,03988**	**0,02277****	**0,03838****	**0,04224***	**0,02407***	**0,04013***	**0,04424****	**0,02435****	**0,04073***	**0,04490***

Table 2: Results obtained for both data sets. Values with the symbol $*$ represent statistical significance with $p \leq 0.05$, and with the symbol $**$ represent significance with $p \leq 0.01$ over all other approaches and baselines.

In relation to BabelFy Sentiment, it has worse results with $k = 20$, it is not statistically different with $k = 40$ and it is superior with $p \leq 0.05$ with the remaining k.

Regarding the post combination results, both BPR Learning-based approaches provide the best results (being both statistically superior with $p \leq 0.01$ against the others of this category), but the version where known pairs' predictions are considered as 1 (i.e. Post comb BPR 1) is the best between them. This approach has no statistical significance against the other BPR Learning approach for $k = 20$, but it is superior with $p \leq 0.01$ for the rest of the results. In relation to the baseline representations, it is statistically superior than all of them with $p \leq 0.01$ in all cases, except for MAP@1 with $k = \{20, 40, 100\}$, where it is superior with $p \leq 0.05$.

Comparing each category's best approach, the post combination BPR Learning approach has the best results for this data set, being statistically superior with $p \leq 0.01$ in the majority of the cases, and $p \leq 0.05$ in the remaining. A graphical comparison between them and the baseline representations can be seen in Figure 2.



(a) MAP@1

(b) MAP@5

(c) MAP@10

Figure 2: Comparison between the best combination approaches and the baseline representations for the MovieLens 100k dataset.

Finally, we compare our representations with some state of the art recommender baselines. We elect as the best neighborhood configuration $k = 20$, and compare its results against the baselines in Table 3. Numbers with bold typeset represent the best results for each MAP value. Also, those with * are statistically superior with $p - value < 0.01$. As it can be seen, the Post comb BPR 1 approach produced the best results and it is statistically superior against all baselines. Pre comb 1 produced statistically superior results against the baselines for MAP@1 and MAP@5, while Neighbor comb 4 is statistically superior only on MAP@1.

6.3.2. Amazon Apps

We detail in this section the results from the Amazon Apps data set in the same way as the previous data set.

Table 3: Comparison between the best combination approaches (with $k = 20$) and the baselines for the MovieLens 100k dataset.

	MAP@1	MAP@5	MAP@10
Pre comb 1	*0.20530	*0.26764	0.25685
Neighbor comb 4	*0.17412	0.25175	0.24566
Post comb BPR 1	***0.21643**	***0.28924**	***0.27680**
BPR-MF	0.13648	0.25237	0.26250
PLTRN	0.13669	0.25113	0.26202
PLTRB	0.14390	0.25584	0.26405
CoFactor	0.05429	0.10841	0.12288

In relation to the pre combination category, the results of the multiplication approach (i.e. Pre comb 2) are the best, being statistically superior with $p \leq 0.01$ against the concatenation approach. Despite being the best result for this approach, it does not have better results than the baselines, being inferior to Heuristic terms, and having very close results with BabelFy SF-IDF.

Regarding the neighborhood strategies, again the best approach is the neighbor union with highest similarity in repetitions (i.e. Neighbor comb 4), having statistically superior results with $p \leq 0.01$ against other approaches from this category. This approach has statistically better results (with $p \leq 0.01$) than all baseline representations with $k \geq 60$, except for MAP@1. With $k = 40$, the results are close to those from Heuristic terms, while $k = 20$ provide inferior results.

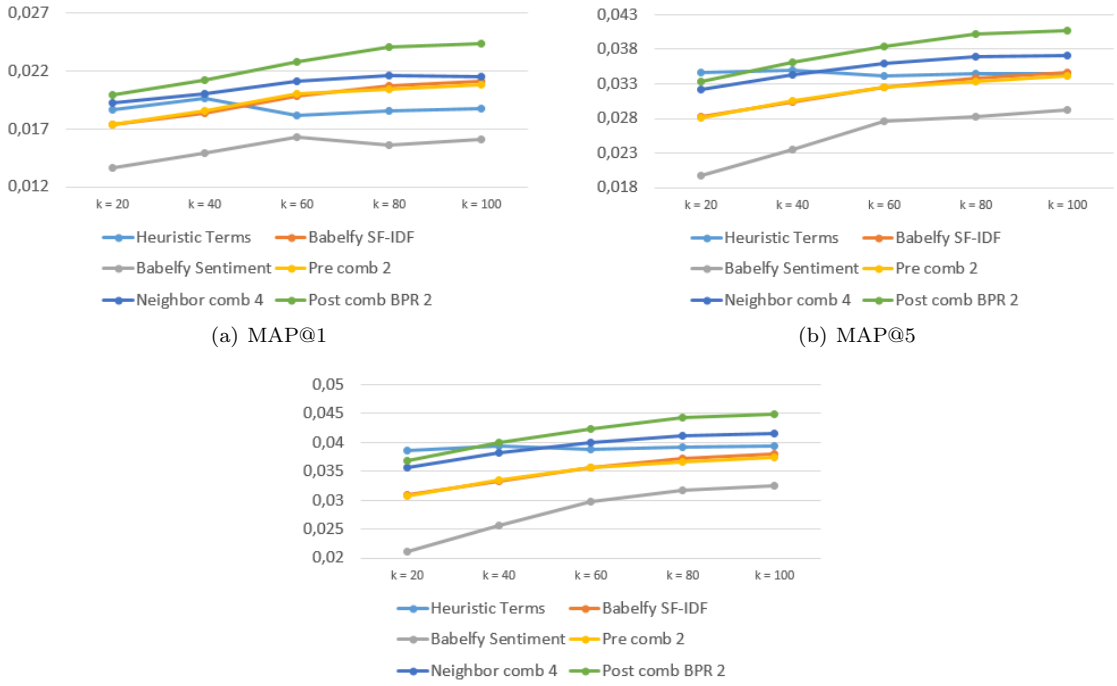
Concerning the post combination results, again both BPR Learning-based approaches provide the best results, being both statistically superior with $p \leq 0.01$ against the others of this category. Between them, the version where we use known pairs' predictions (i.e. Post comb BPR 2) can be considered the best, being statistically superior than the other BPR Learning approach with $p \leq 0.01$ in half the cases, and $p \leq 0.05$ in the rest. In relation to the baseline representations, it is inferior than Heuristic terms for $k = 20$, in metrics MAP@5 and MAP@10, and it has no statistical difference for $k = 40$ at MAP@5 and MAP@10. For the remaining k, as well as the remaining baseline representations, it is statistically superior with $p \leq 0.01$.

The post combination BPR Learning approach has the best results between the best approaches of each category, being statistically superior with $p \leq 0.01$ except for $k = 20$, where it is superior with $p \leq 0.05$ against the neighborhood approach. A graphical comparison between them and the baseline representations can be seen in Figure 3.

Finally, we again compare our representations with the recommender baselines. We elect as the best neighborhood configuration $k = 100$, and compare its results against the baselines in Table 4. Numbers with bold typeset represent the best results for each MAP value. Also, those with * are statistically superior with $p - value < 0.01$. The Post comb BPR 2 approach produced the best results and it is statistically superior against all baselines. Pre comb 1 has superior, but not statistically different results against the baselines for MAP@1, while Neighbor comb 4 is statistically superior only on MAP@1, while nonetheless presenting better results for the remaining MAP values.

Table 4: Comparison between the best combination approaches (with $k = 100$) and the baselines for the Amazon Apps dataset.

	MAP@1	MAP@5	MAP@10
Pre combination 2	0.02079	0.03413	0.03749
Neighbor combination 4	*0.02152	0.03716	0.04156
Post combination BPR 2	***0.02435**	***0.04072**	***0.04489**
BPR-MF	0.02027	0.03589	0.04057
PLTRN	0.02028	0.03578	0.04059
PLTRB	0.02057	0.03584	0.04058
CoFactor	0.01620	0.02793	0.02916



(a) MAP@1

(b) MAP@5

(c) MAP@10

Figure 3: Comparison between the best combination approaches and the baseline representations for the Amazon Apps data set.

6.3.3. Discussion

As it can be seen, both BPR Learning strategies provide the best results in both data sets, but in each data set one has performed better than the other. With that, further experimentation in other data sets is required in order to define which is better. Nevertheless, a great advantage that Post comb BPR 1 has in relation to the other is that it requires less computational resource and is faster, since it does not require to compute predictions of the training pairs, considering them as 1. Even so, both proved to be good approaches, especially since their strength resides in using a learn-to-rank approach to correctly weight the relevance of each ranking.

The heuristics ranking approaches, in turn, did not provide good results. We argue that using direct combinations may have dragged relevant items to lower positions of the ranking. Since those approaches do not have a learning mechanism to weight what is or is not relevant, the resulting ranking has a poorer composition than the isolated rankings. Even so, the Post comb 1 approach was able to provide better results against baseline representations in some cases, especially with larger rankings (MAP@5 and MAP@10) and larger neighborhoods ($k = \{80, 100\}$).

The pre combination strategies also did not have good results. As well as the BPR strategies, further experimenting is required to define which strategy is the best, since in each data set one performed better than the other. Furthermore, even though those strategies' results did not have statistical significance against the baseline representations, in several cases they performed slightly better than the isolated BabelFy representations that they are based on.

The neighborhood combination category results, in turn, were more consistent: in both datasets, the Neighbor comb 4 approach provided the best results. It also performed better than the baseline representations for $k \geq 60$. We argue that since we are combining neighbors, the larger the set, the better the combination can be performed. Since the sets are an intersection between the most similar items and the set of items evaluated by the user, as we raise k we have more chances of constructing a neighborhood. This can be clearly seen with the intersection neighborhood combination approaches:

they have very poor results since their final neighborhood sets are potentially very small. As for the strategy for combining repeated items' similarities, the best approach was to maintain only the highest value. We argue that the reason it performed better is that, in this sense, good neighbors are not dragged down by lower scores in the other neighborhood, thus maintaining their high similarity score, and, finally, not having a chance of being cut in the final sorting step.

Nevertheless, in most of the cases, each category produced better results than both the baseline representations and recommenders for at least one of their approaches. By comparing against the baseline recommenders, it can be noticed that in most cases both pre and neighborhood combinations produce better results, and at some cases with statistical difference, for MAP@1 and MAP@5. This highlights the ability of our approaches to produce better suggestions early on the rankings. Our BPR-based combination approaches also produce far superior results against baselines, highlighting the importance of combining two different views of the items.

This thorough evaluation shows the strength of our approaches, and allows us to select which of them to use according to our needs. Since the pre combination approaches require little computational resources and are only processed once before the recommendation, they can be used in any settings and still provide a small gain on ranking accuracy for the system. They can also be used on other content-based recommender systems, but further experiments are required to assess their advantages on these scenarios. The neighborhood combination approaches in turn revealed good results, specially with larger sets of k, being strongly recommended to be used in such scenarios. They also have the advantage of being simpler and faster alternatives than the BPR learning-based post combination approaches. These, in turn, have the best of the results, but are the ones that require more of computational resources, being suggested to be used on top notch servers. Moreover, as well as the pre combination approaches, these are not bound to the item-kNN algorithm, being capable of being used in any recommender system that can produce rankings.

Finally, the Amazon Apps results, in general, are very low. This can be attributed to the large size of the data set and the evaluation method itself. Since items can be considered relevant only if they are in the test set, and the total number of items is very large, it becomes difficult to correctly match items in such small ranking with those present in the test set.

7. Conclusions and future work

In this paper we have explored several strategies for correctly combining two item representations based on user reviews in a recommender system. Experiments showed promising results, with machine learning-based ranking combination strategies presenting the best results of all approaches. Bellow we summarize the main contributions of this paper:

- The item representations based on user reviews. These representations carry a large amount of semantics, since they use concepts instead of simple terms as features. The main implication is that concepts minimize problems of ambiguity that terms can have. Moreover, since the item representations are based on texts provided by users, they can be employed in any data domain, i.e., movies, music, products, among others.

- Two different views of the same feature set. This way, items can be detailed in two different manners in relation to their features: one that scores features according to their statistics in relation to the items, and the other that scores features according to the sentiment users have towards them in the items.

- The combination of these representations to improve recommendation accuracy. By combining the representations, the system is analyzing the items in two manners, and the strength of both can be exploited. We explore an extensive array of combination approaches and analyze their contributions to produce rankings.

- The generality of pre and post combination strategies. In this paper we explored these approaches in an item-kNN algorithm, but they are not bound to this algorithm. They can be employed with

other content-based recommender, but further experimentation must be carried out to assess their advantages.

As future work, we intent to perform more experiments in data sets of other domains and sizes in order to define which techniques are the best among those that could not be defined in this study. We also aim to explore machine learning to give weights for pre and neighborhood combinations. Finally, the semantics of the BabelNet synsets extracted from the texts can also be explored to further enrich the representations, since the synsets are connected in a very large network with semantic relations that are derived from both WordNet and Wikipedia.

Acknowledgements

The authors would like to thank the financial support of: CAPES, CNPq and FAPESP (process 2016/20280-6).

References

[1] J. Bobadilla, F. Ortega, A. Hernando, A. GutiéRrez, Recommender systems survey, Know.-Based Syst. 46 (2013) 109–132. doi:10.1016/j.knosys.2013.03.012.
URL http://dx.doi.org/10.1016/j.knosys.2013.03.012

[2] L. Chen, G. Chen, F. Wang, Recommender systems based on user reviews: The state of the art, User Modeling and User-Adapted Interaction 25 (2) (2015) 99–154. doi:10.1007/s11257-015-9155-5.
URL http://dx.doi.org/10.1007/s11257-015-9155-5

[3] R. M. D'Addio, M. A. Domingues, M. G. Manzato, Exploiting feature extraction techniques on users' reviews for movies recommendation, Journal of the Brazilian Computer Society 23 (1) (2017) 7. doi:10.1186/s13173-017-0057-8.
URL https://doi.org/10.1186/s13173-017-0057-8

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

[5] G. Ganu, Y. Kakodkar, A. Marian, Improving the quality of predictions using textual information in online user reviews, Information Systems 38 (1) (2013) 1–15. doi:10.1016/j.is.2012.03.001.

[6] M. Terzi, M. Rowe, M.-A. Ferrario, J. Whittle, Text-Based User-kNN: Measuring User Similarity Based on Text Reviews, Springer International Publishing, Cham, 2014, pp. 195–206. doi:10.1007/978-3-319-08786-3_17.

[7] X. Chen, Z. Qin, Y. Zhang, T. Xu, Learning to rank features for recommendation over multiple categories, in: Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '16, ACM, New York, NY, USA, 2016, pp. 305–314. doi:10.1145/2911451.2911549.
URL http://doi.acm.org/10.1145/2911451.2911549

[8] M. Capelle, F. Frasincar, M. Moerland, F. Hogenboom, Semantics-based news recommendation, in: Proceedings of the 2Nd International Conference on Web Intelligence, Mining and Semantics, WIMS '12, ACM, New York, NY, USA, 2012, pp. 27:1–27:9. doi:10.1145/2254129.2254163.
URL http://doi.acm.org/10.1145/2254129.2254163

[9] M. Capelle, M. Moerland, F. Hogenboom, F. Frasincar, D. Vandic, Bing-sf-idf+: A hybrid semantics-driven news recommender, in: Proceedings of the 30th Annual ACM Symposium on Applied Computing, SAC '15, ACM, New York, NY, USA, 2015, pp. 732–739. doi:10.1145/2695664.2695700.
URL http://doi.acm.org/10.1145/2695664.2695700

[10] M. de Gemmis, P. Lops, G. Semeraro, A content-collaborative recommender that exploits wordnet-based user profiles for neighborhood formation, User Modeling and User-Adapted Interaction 17 (3) (2007) 217–255. doi:10.1007/s11257-006-9023-4.
URL http://dx.doi.org/10.1007/s11257-006-9023-4

[11] P. Lops, C. Musto, F. Narducci, M. De Gemmis, P. Basile, G. Semeraro, Mars: A multilanguage recommender system, in: Proceedings of the 1st International Workshop on Information Heterogeneity and Fusion in Recommender Systems, HetRec '10, ACM, New York, NY, USA, 2010, pp. 24–31. doi:10.1145/1869446.1869450.
URL http://doi.acm.org/10.1145/1869446.1869450

[12] C. Musto, G. Semeraro, P. Lops, M. de Gemmis, Combining Distributional Semantics and Entity Linking for Context-Aware Content-Based Recommendation, Springer International Publishing, Cham, 2014, pp. 381–392. doi:10.1007/978-3-319-08786-3_34.

[13] F. Narducci, P. Basile, C. Musto, P. Lops, A. Caputo, M. de Gemmis, L. Iaquinta, G. Semeraro, Concept-based item representations for a cross-lingual content-based recommendation process, Inf. Sci. 374 (C) (2016) 15–31. doi:10.1016/j.ins.2016.09.022.
URL https://doi.org/10.1016/j.ins.2016.09.022

[14] S. Oramas, V. C. Ostuni, T. D. Noia, X. Serra, E. D. Sciascio, Sound and music recommendation with knowledge graphs, ACM Trans. Intell. Syst. Technol. 8 (2) (2016) 21:1–21:21. doi:10.1145/2926718.
URL http://doi.acm.org/10.1145/2926718

[15] C. D. Manning, P. Raghavan, H. Schütze, Introduction to Information Retrieval, Cambridge University Press, 2008.

[16] R. Socher, A. Perelygin, J. Wu, J. Chuang, C. D. Manning, A. Y. Ng, C. Potts, Recursive deep models for semantic compositionality over a sentiment treebank, in: Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP '13), 2013, pp. 1631–1642.

[17] S. Rendle, C. Freudenthaler, Z. Gantner, L. Schmidt-Thieme, Bpr: Bayesian personalized ranking from implicit feedback, in: Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence, UAI '09, AUAI Press, Arlington, Virginia, United States, 2009, pp. 452–461.
URL http://dl.acm.org/citation.cfm?id=1795114.1795167

[18] M. Yagci, T. Aytekin, F. Gurgen, On parallelizing sgd for pairwise learning to rank in collaborative filtering recommender systems, in: Proceedings of the Eleventh ACM Conference on Recommender Systems, RecSys '17, ACM, New York, NY, USA, 2017, pp. 37–41. doi:10.1145/3109859.3109906.
URL http://doi.acm.org/10.1145/3109859.3109906

[19] D. Liang, J. Altosaar, L. Charlin, D. M. Blei, Factorization meets the item embedding: Regularizing matrix factorization with item co-occurrence, in: Proceedings of the 10th ACM Conference on Recommender Systems, RecSys '16, ACM, New York, NY, USA, 2016, pp. 59–66. doi:10.1145/2959100.2959182.
URL http://doi.acm.org/10.1145/2959100.2959182

[20] R. Navigli, S. P. Ponzetto, Babelnet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network, Artif. Intell. 193 (2012) 217–250. doi:10.1016/j.artint.2012.07.001.
URL http://dx.doi.org/10.1016/j.artint.2012.07.001

[21] A. Moro, A. Raganato, R. Navigli, Entity linking meets word sense disambiguation: a unified approach, Transactions of the Association for Computational Linguistics 2 (2014) 231–244.
URL http://www.aclweb.org/anthology/Q14-1019

[22] M. G. Manzato, M. A. Domingues, A. C. Fortes, C. V. Sundermann, R. M. D'Addio, M. S. Conrado, S. O. Rezende, M. G. C. Pimentel, Mining unstructured content for recommender systems: an ensemble approach, Information Retrieval Journal 19 (4) (2016) 378–415. doi:10.1007/s10791-016-9280-8.
URL https://doi.org/10.1007/s10791-016-9280-8

[23] X. Ning, C. Desrosiers, G. Karypis, A Comprehensive Survey of Neighborhood-Based Recommendation Methods, Springer US, Boston, MA, 2015, pp. 37–76. doi:10.1007/978-1-4899-7637-6_2.
URL https://doi.org/10.1007/978-1-4899-7637-6_2

[24] C. D. Manning, M. Surdeanu, J. Bauer, J. Finkel, S. J. Bethard, D. McClosky, The stanford corenlp natural language processing toolkit, in: Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations, 2014, pp. 55–60.

[25] R. M. D'Addio, M. G. Manzato, A collaborative filtering approach based on user's reviews, in: Proceedings of the 2014 Brazilian Conference on Intelligent Systems, BRACIS '14, IEEE Computer Society, Washington, DC, USA, 2014, pp. 204–209. doi:10.1109/BRACIS.2014.45.
URL http://dx.doi.org/10.1109/BRACIS.2014.45

[26] R. S. Marinho, R. M. D'Addio, M. G. Manzato, Semantic organization of user's reviews applied in recommender systems, in: Proceedings of the 23rd Brazillian Symposium on Multimedia and the Web, WebMedia '17, ACM, New York, NY, USA, 2017, pp. 277–280. doi:10.1145/3126858.3131600.
URL http://doi.acm.org/10.1145/3126858.3131600

[27] A. da Costa Fortes, M. G. Manzato, Ensemble learning in recommender systems: Combining multiple user interactions for ranking personalization, in: Proceedings of the 20th Brazilian Symposium on Multimedia and the Web, WebMedia '14, ACM, New York, NY, USA, 2014, pp. 47–54. doi:10.1145/2664551.2664556.
URL http://doi.acm.org/10.1145/2664551.2664556

[28] J. McAuley, C. Targett, Q. Shi, A. van den Hengel, Image-based recommendations on styles and substitutes, in: Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '15, ACM, New York, NY, USA, 2015, pp. 43–52. doi:10.1145/2766462.2767755.
URL http://doi.acm.org/10.1145/2766462.2767755

[29] F. Wilcoxon, Individual Comparisons by Ranking Methods, Springer New York, New York, NY, 1992, pp. 196–202.

5.2 Final Remarks

The combination between two different item representations provides the next step into further enhancing the semantics of item descriptions, thus relating directly with Research Question 2: "Does increasing the semantics of item representations improve recommendation accuracy?". By combining those two representations, we guarantee that the item is described with two different views: one that regards the rarity of its features and how much related those features are to the item; and the other that regards the quality of those features, i.e., whether they are viewed by users as positive or negative aspects of the item. Those two interpretations are fairly different, but they share the same composition: in both representations the item has the same features (non-zero values), but they differ in their scores.

We combine them by using ten different strategies, which are divided in pre, neighborhood and post combination. The majority of them are based on heuristics, while two of them are based on machine learning. These two approaches provide the best results, being statistically superior than baselines in almost every situation. The other approaches also produce good results, being superior than baselines in the majority of the cases. This highlights the importance of the combination, and reinforces the notion that increasing the semantics involved in item descriptions do help improve the recommender's suggestion.

One highlight of the pre and post combination approaches is that they are independent of the recommender system, being able to be applied into other content-based algorithms. Nevertheless, further experimentation is required to ascertain their benefits with other recommenders.

One drawback of this study is that the top-N evaluation performed in the area is flawed, especially if we consider very large datasets. This kind of evaluation assumes that relevant items are those that appear on the test set, i.e., items that the user have previously evaluated but are regarded as unknown pairs. This is no issue in a rating prediction scenario, since we are aiming to evaluate if the system is able to correctly predict which rating the user would give to an item. But, when considering top-N evaluation, we are concerned in ranking items by relevance. When considering large datasets, the chance that an test item is ranked high is very low, but the other items that are indeed top ranked could be potentially relevant to the user. With that, the most correct way to evaluate these scenarios would be with user trials.

Another issue presented so far is that we were not able to directly evaluate the item representations and their features' descriptive power. The features and their weights themselves could be used to other ends than to improve a recommender's accuracy. For instance, some of them have such intuitive semantics, such as the sentiment-based representation, that could be employed to aid the recommender provide explanations about its suggestions.

In the next chapter, we take those issues into consideration and present, along with a state-of-the-art recommendation system, a user trial in which our features are used to aid the construction of suggestions' explanations.

CHAPTER

6

SENTIMENT-AWARE EXPLANATION CHAINS

6.1 Contextualization

So far, we have focused our advances towards maximizing the accuracy of content-based recommender systems by designing semantically rich item representations. We have followed the assumption that these systems' capability of producing relevant suggestions is directly related to the quality of their descriptions.

In this next study, we shift our focus to better analyse the descriptive power of one of our representations, the sentiment-based representation used in two of our works (D'ADDIO *et al.*, 2018; D'ADDIO; MARINHO; MANZATO, 2019). Here, we do not focus solely on accuracy metrics such as precision, but evaluate, with a wider spectrum of metrics, the application of our representation in a state-of-the-art recommender system called *Recommendation-by-Explanation* (*r-by-e*) (RANA; BRIDGE, 2018). This recommender produces suggestions based on the quality of the explanations that it would provide for those items, and those explanations, in turn, are solely based on the features of the items. We make extensive offline experiments, evaluating the recommender system's performance in relation to precision, novelty, surprise, diversity and coverage (KAMINSKAS; BRIDGE, 2016). We also perform a user trial that evaluates both recommendation and explanation relevance.

This work was conducted in collaboration with *r-by-e*'s author, Arpit Rana, and his supervisor, Derek Bridge, from University College Cork, Ireland. Arpit has been working on several extensions to *r-by-e*'s formulation, but some of them required semantically enhanced item features, which were the main objective of this research. The following article, which is currently in preparation to be submitted to Springer's "User Modeling and User-Adapted Interaction" and was first described in Rana's thesis (RANA, 2020), details the results of our collaboration, which analyzes *r-by-e*'s extensions applied together with sentiment-based item descriptions.

Extended Recommendation-by-Explanation

ARPIT RANA*, Department of Mechanical & Industrial Engineering, University of Toronto, Canada
RAFAEL M. D'ADDIO, Institute of Mathematics and Computer Science, University of São Paulo, São Carlos, SP, Brazil
MARCELO M. MANZATO, Institute of Mathematics and Computer Science, University of São Paulo, São Carlos, SP, Brazil
DEREK BRIDGE, Insight Centre for Data Analytics, University College Cork, Ireland

Studies have shown that a user of a recommender system is more likely to be satisfied by the recommendations if the system provides explanations that allow the user to understand the rationale that lies behind the recommendations. In current recommender systems, explanation is a step that comes after recommendation, i.e., computing recommendations and generating corresponding explanations are two separate and sequential processes. Divorcing recommendation and explanation in this way affords the system the freedom to include in the explanation information different from that which it used to compute the recommendation. Such differences are one cause of low fidelity between the recommender and its explanations.

Previously, we have proposed Recommendation-by-Explanation (*r-by-e*), an approach that unifies both worlds. *r-by-e* enables the system to find relevant recommendations through explanations that have a high degree of fidelity and interpretability, called *explanation chains*. To generate suggestions, the system first constructs an explanation for each candidate item (*chain generation*); then it recommends those candidate items that have the best explanations (*chain selection*). While there were promising results, *r-by-e* used item features in a limited perspective: those were unweighted, binary features that presented limited semantics in which explanations could be built. The system was designed to handle only this kind of information, thus not benefiting from more semantic item descriptions.

To address those limitations, we extend *r-by-e*'s formulation so it can explore different feature weighting schemes to give more refined versions of chain generations. In this sense, the system is able to further capture the relationships between items and their features. We also give ways of generating chains that use different types of item representations: one that use the item features explicitly and the other that makes no explicit use of features. Finally, we generalize *r-by-e*'s approach to chain selection, allowing the system to choose whether to cover more aspects of the candidate item or the user profile.

We present a comprehensive empirical comparison of all the extended versions (99 configurations of each version) with corresponding classic content-based methods on two datasets that mainly differ on their item feature set. One dataset deals with keyword-based features, weighted by their frequency, while the other uses concepts extracted from user reviews, weighted by the sentiments of the users towards them. The versions of *r-by-e* that make explicit use of item features have several advantages over the others, and the empirical comparison shows that one of these versions —the one that assigns weights to the item features based on their importance to that item (designated as *wfb*)— is also the best in terms of recommendation accuracy, diversity, and surprise, while still generating chains whose lengths are manageable enough to be interpretable by users.

We conduct a user trial on the dataset which uses concepts from user reviews and their sentiments, dubbing the recommendations as sentiment-aware explanation chains. We found on this study that *wfb* provides more relevant recommendations and that they are also diverse and serendipitous; and we found that its

*The research was conducted when the author was at Insight Centre for Data Analytics, University College Cork, Ireland

Authors' addresses: Arpit Rana, Department of Mechanical & Industrial Engineering, University of Toronto, Canada, arana@mie.utoronto.ca; Rafael M. D'Addio, Institute of Mathematics and Computer Science, University of São Paulo, São Carlos, SP, Brazil, rdaddio@icmc.usp.br; Marcelo M. Manzato, Institute of Mathematics and Computer Science, University of São Paulo, São Carlos, SP, Brazil, mmanzato@icmc.usp.br; Derek Bridge, Insight Centre for Data Analytics, University College Cork, Ireland, derek.bridge@insight-centre.org.

sentiment-aware chains are more helpful to users when judging recommendation quality than the competitor explanations.

Additional Key Words and Phrases: Explanation, Recommendation, Sentiments, User trials

PUBLICATION INFORMATION:
Arpit Rana, Rafael M. D'Addio, Marcelo M. Manzato, and Derek Bridge. 2020. Extended Recommendation-by-Explanation. Submitted to *User Modeling and User-Adapted Interaction*. 33 pages.

1 INTRODUCTION

Automated personalization is one way of tuning systems to the needs and preferences of individual users and, consequently, alleviating the problems caused by choice overload. Recommender systems, in particular, attempt to filter and rank a set of items and suggest to the user a smaller ordered set – one which contains items that it thinks are most likely to satisfy the user [3].

Some recommender systems can also provide explanations. Explanations can serve a multiplicity of aims: they give credibility to recommendations [33, 41], help users make better choices [4], positively contribute to a better user experience [19], and so on. Due to the wide adoption of recommender systems in many aspects of our lives, explaining recommendations has attracted considerable attention [46, 49].

Most recommendation explanations are post-hoc rationalizations. In other words, in current recommender systems, computing recommendations and generating corresponding explanations are considered as two separate, sequential processes. This affords the recommender the freedom to include in the explanation information different from that which it used to compute the recommendation [1]. For example, in [38], a recommendation generated by factorization of a ratings matrix is explained using topic models mined from textual data associated with items — data that was not used when when building the recommendation model. Such differences are one cause of low fidelity [20] (also called objective transparency [15]): the extent to which the explanation reveals the logic of the underlying recommender. In an experiment with a music recommender, Kulesza et al. found that the more that explanations were both sound and complete with respect to the recommender, the greater the users' trust in the recommender and the better their understanding [20]. This finding indicates that there is an intimate connection between the process of computing recommendations and generating corresponding explanations; and that this close relationship may lead to better recommendations for the user. In this paper, we investigate the role of explanations in the process of computing recommendations, aiming to achieve a higher degree of fidelity between the explanations and the operation of the recommender system, without compromising the interpretability of the explanations and the quality of the recommendations. Our approach, *Recommendation-by-Explanation* (*r-by-e*), unifies recommendation and explanation to a greater degree than has been achieved hitherto. *r-by-e* employs two main steps: *chain generation* and *chain selection*. In the former, for each candidate item, the system constructs and scores an explanation, a chain of items from the user's profile, referred to as an *explanation chain*, based on *overlap* between the representations of items in the chain; then, in the later step, it recommends those candidate items that have the best explanations based on their scores. By unifying recommendation and explanation, *r-by-e* finds relevant recommendations with explanations that have a high degree of fidelity.

We have previously introduced *r-by-e* in [35, 36]. Although we observed promising results, the way we built the chains was relatively simple. In particular, we considered only one way of representing items — in terms of their features, which were keywords that items may or may not have. Explanation chains could be refined and present more information about the item and its features. For instance, item features could have weights that represent their worth towards the

items, thus adding a new layer of semantics to the recommendation and the explanation. But, the current *r-by-e* formulae is based on feature sets, and do not take into account these weights. Moreover, its formulae can be extended in several ways to allow more flexibility in choosing item representations, feature weighting schemes, and the influence of item or user coverage in the chains.

This paper, drawn from the first author's PhD research [34], significantly extends *r-by-e* in three ways (see Figure 1). First, we consider a scheme for assigning weights to an item's features based on their informativeness. We define *weighted overlap* to take advantage of these weights. Second, we propose an alternative item representation which makes no explicit reference to features. We refer to it as a *neighbour-based* item representation. For this new item representation, we define both an unweighted and weighted overlap. Finally, we also generalize *r-by-e*'s *chain selection*. In place of simply adding the average overlap and average profile overlap (as in [36]), we define the score to be a linear combination of the two but controlled by a parameter α.

We explore these variants using two versions of a movie recommendation dataset. In the first version of the dataset, we describe each item using a set of keywords. These descriptions are simpler in the sense that they convey information regarding the frequency or rarity of the keywords, not regarding more interesting, semantic aspects such as qualities and flaws of the movies. We hypothesize that explanations would benefit greatly if items were described by their good and bad aspects. Therefore, the second version of the dataset uses sentiment data extracted from user reviews and describes each item using a set of terms associated with sentiment values. The complete details of these datasets are covered in Section 6.1.

In a set of offline experiments on the first version of the dataset, we compare all four versions of *chain generation*: i) unweighted feature-based (*fb*), ii) weighted feature-based (*wfb*), iii) unweighted neighbour-based (*nb*), and iv) weighted neighbour-based (*wnb*). Notice that unweighted feature-based is what we covered already in our previous work [36]. We include it here again for better understanding the difference among all four approaches.[1] In our experiments, we found out that the approaches that explicitly use item features perform better than their counterpart, and weight-based approaches are able to produce more accurate recommendations while being competitive in terms of diversity and surprise. We also show in the results that by varying the balancing parameter (α), *r-by-e* selects longer chains that subsequently increase the surprise and the diversity of the recommendations.

Further offline experiments, this time on the second version of the dataset, compare the weighted feature-based and weighted neighbour-based forms of *chain generation*. (As we will explain, the unweighted forms of *chain generation* do not apply to the sentiment version of the dataset.) In this experiment, we can see that, again, the feature-based approach attain higher levels of precision, with comparative results on diversity and somewhat lower results of surprise and novelty, when compared to its counterpart.

From the offline experiments, we choose the best version of *r-by-e* to compare with a baseline in a user trial, which is split into two parts: a recommendation trial and an explanation trial. We conduct them on the sentiment version of the dataset. The recommendation trial reveals that *r-by-e*'s suggestions are more relevant than baseline, being also diverse and serendipitous. The explanation trial reveals that sentiment-aware chains help users make better informed judgements towards the quality of recommended movies.

The paper is organized as follows. After a review of related work (Section 2), Section 3 briefly summarises the notion of *Recmmendation-by-Explanation* and the explanation chains, while Section

[1] In fact, the results we present here are not identical to the ones in [36] because in this work we have included some extra normalization of the scores its uses to give a fair comparison with the three other variants.

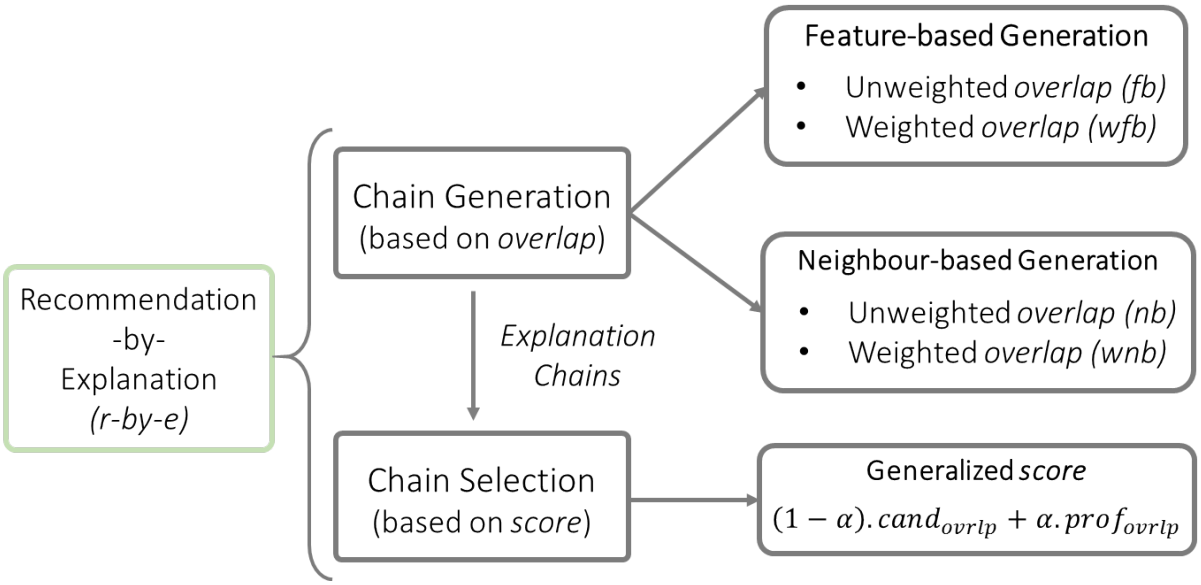


Fig. 1. Extended Recommendation-by-Explanation

4 describes the extensions and refinements over the original *r-by-e* formulation. In Section 5, we evaluate those extensions on offline experiments with keywords as features, while in Section 6 we detail our Sentiment-Aware Explanation Chains and present both online and offline experiments to evaluate them. Finally, in Section 7 we offer some final remarks.

2 RELATED WORK

An explanation of a recommendation is any content additional to the recommendation itself that justifies the recommended item to the user. For instance, a textual explanation, "We recommend you the movie *A Beautiful Mind* because it has features *drama* and *biography* that you liked before" justifies the movie 'A Beautiful Mind' to the user by means of its features 'drama' and 'biography' which she liked before.

Explanations are especially important in high-risk domains where the cost of making a wrong decision is higher (e.g. buying a laptop, planning a holiday, etc.) than in low-risk domains (e.g. selecting a song to play) [16]. The level of detail in an explanation may also vary with the level of risk associated with the decision-making process [7]. In general, it also makes more sense if these explanations are personalized to the end-users so that the explanations are sensitive to the users' level of understanding [45].

Explanations of recommendations vary in many ways. They may vary in their goals: they may be intended to help the user make a better decision (effectiveness), change the user's behaviour (persuasion), make a system more correctable (scrutability), and so on [44]. In our work, we are interested in *effectiveness*, which is why one of our user trials is a *re-ranking* task.

Explanations often relate the recommended item to the user through *intermediary entities* [47]. These intermediary entities may be other users, other items, item features, or context. Based on these intermediary entities, explanations can be described as either *user-based*, *item-based*, *feature-based*, *context-based* or, in the case of combinations, *hybrid* [4, 31].

User-based explanations say that an item is recommended because users who are similar to the active user liked it. For example, social networks such as Facebook[2] often use user-based

[2]https://www.facebook.com

explanations when recommending a person to add as a friend or to follow. However, these methods do not scale up to user-based collaborative filtering systems, where: the number of neighbours is usually larger; most, if not all, of the neighbours are not known to the active user; and the number of co-rated items between the active user and any neighbour can be too large to be readily comprehended [5].

Item-based explanations say that the item is being recommended because the user liked similar items. Famously, Amazon[3] uses item-based explanations for its recommendations [21]. Studies show that item-based approaches present the relationship between the user and recommended items in an easily interpretable way which helps users to make accurate decisions [4]. Accordingly, in [5], the authors showed how even *user-based* collaborative recommendations can be explained using *item-based* explanations. They mined (item-based) rules from the neighbours' ratings. However, item-based explanations may have a shortcoming, which is that users may not understand the relationship between the items in the explanation and the recommended item [43].

Feature-based explanations say that the recommended item has features that the user likes. For instance, Pandora uses altogether 450 musical attributes for representing each music track[4] and provides explanations such as: *Based on what you've told us so far, we're playing this track because it features a leisurely tempo, a sparse piano solo, a lazy swing groove, major tonality and many other similarities identified* [46]. In the literature, features take numerous different forms, e.g. attribute-value pairs [40, 42], item content [4], user-generated tags [14, 15, 47], opinions mined from user reviews [6, 10, 26], and linked data [28, 29].

Most recently, contextual information has been used in explanations too. In [39], Sato et al. proposed explanations that include contexts suitable for consuming the recommended item.

Hybrid explanations are also possible. For example, in the case of item-based explanations, we often want to show why the items in the explanation are similar to the recommended item, and this is typically done by showing the features that they have in common. Since these item-based explanations combine items and features, they are hybrid explanations [31]. Explanation Chains are of this kind: they are item-based but they expose item relationships through features.

In Artificial Intelligence in general, explanations are sometimes categorized as white-box (also sometimes called model-based) or black-box (sometimes called model-agnostic) [13, 16]. The distinction typically reflects on the fidelity of the explanations to the underlying reasoning done by the AI system.

White-box explanations are built from traces of the system's reasoning. These explanations disclose something of the underlying model in order to reveal 'how' the system has reached its conclusions. For example, if we have a user-based nearest-neighbours recommender that makes recommendations by finding items liked by the active user's nearest neighbours, then a histogram of the neighbours' ratings [16] is a white-box explanation.

Black-box explanations, by contrast, make no use of knowledge of how the system produced its decision. Black-box explanations are post-hoc rationalizations. For example, the LIME system explains classification decisions by interrogating the classifier to obtain a dataset from which LIME builds a distinct explanation model [37]. Since they make no use of traces of the system's reasoning, black-box explanations may make use of other sources of information that were not used in the decision-making. In [38], for example, recommendations are made by matrix factorization on a ratings matrix but the recommendations are explained using topic models that are mined from textual data associated with the items but not used by the recommender.

[3] https://www.amazon.com/

[4] https://www.pandora.com/about/mgp

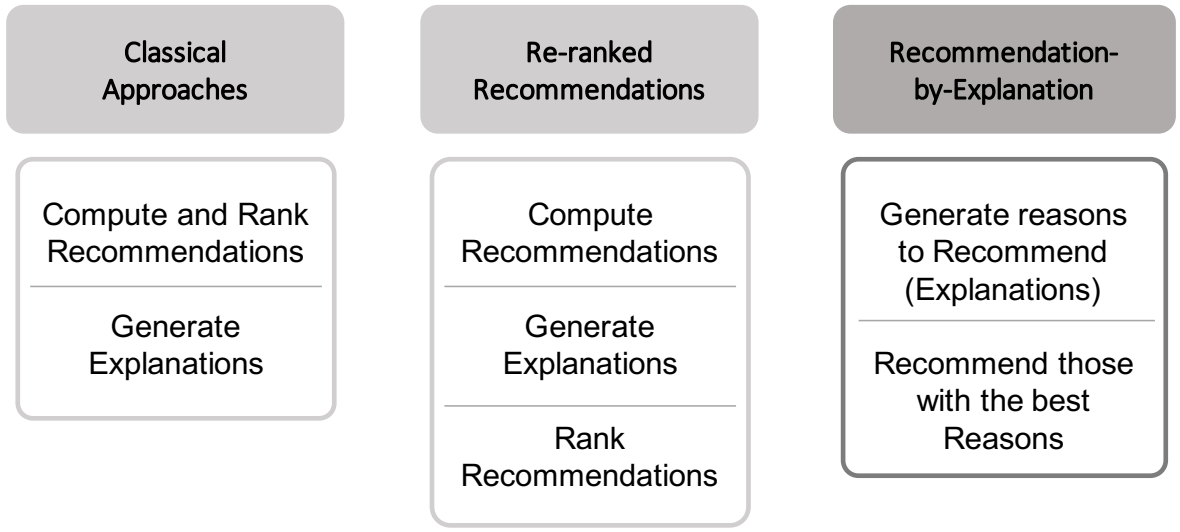


Fig. 2. Role of explanations in producing recommendations.

Black-box explanations raise the issue of fidelity [20] (also called objective transparency [15]): the extent to which the explanation reveals the logic of the underlying recommender. Kulesza et al. considered two dimensions of explanation fidelity: *soundness* and *completeness*. They defined the former as the extent to which each component of an explanation's content is truthful in describing the underlying system; and the latter as the extent to which all of the underlying system is described by the explanation. For example, a recommender system that explains its reasoning with a simpler model than it actually uses (e.g. a set of rules instead of additive feature weights) reduces soundness, whereas a system that explains only some of its reasoning (e.g. only a subset of a user neighbourhood) reduces completeness.

In an experiment with a music recommender, Kulesza et al. found that the more that explanations were both sound and complete with respect to the recommender, the greater the users' trust in the recommender and the better their understanding [20]. Recommendation-by-Explanation seeks to achieve quite high fidelity since, in *r-by-e*, explanation is intrinsic to recommendation. Indeed, *r-by-e*'s explanations are not black-box: the suggestions are exclusively guided by explanations' quality, and the recommender's model is the explanations themselves. This leads to sound and complete explanations, i.e., they do not simplify the model nor expose a limited part of its reasoning.

In current recommender systems, computing a recommendation and generating corresponding explanation are two separate, sequential processes. Below we will look at some work that challenges this assumption.

It seems obvious that a recommender should first produce its recommendations and then seek to build explanations for them. This is the classic approach depicted leftmost in Figure 2. Almost all of the systems that we have cited previously work in this way.

There have been a few efforts that modify the classical approach a little. These are shown in the middle of Figure 2. In Re-ranked Recommendations, for example, the system finds some recommendations; it generates explanations for the recommendations; it scores the explanations; and it re-ranks the recommendations based on their explanation scores before showing them to the user [26, 27]. In [48], Yu et al. use this strategy to increase the recommendation diversity.

Another strategy that falls within the Re-ranked Recommendations category is to use explanations as means for the users to analyse recommendation quality and, if appropriate, propose

a change to the predicted value [9]. For example, Cleger et al. [8] uses explanations to learn a regression model that can predict the error and thus "correct" the rating of a target item.

Our new approach, Recommendation-by-Explanation, is shown rightmost in Figure 2. Uniquely, as far as we are aware, it reverses the process. First, it finds explanations for all the candidate items. Then, it recommends the candidates that have the best explanations. Hence, Recommendation-by-Explanation is an approach that unifies the two processes: computing recommendations and generating corresponding explanations. This, we believe, gives it high fidelity. We describe it at much greater length in the remainder of this paper.

3 *RECOMMENDATION-BY-EXPLANATION*

Recommendation-by-Explanation is a novel, unified approach for recommendation and explanation: the system constructs an explanation (a chain of items from the user's profile) for each candidate item (we call this step *chain generation*); then it recommends those candidate items that have the best explanations (which we call *chain selection*). By unifying recommendation and explanation, *r-by-e* finds relevant recommendations with explanations that have a high degree of fidelity [35, 36]. *r-by-e*'s explanations take the form of what we call *Explanation Chains*. Figure 3 shows an example of an Explanation Chain in the movie domain. The rightmost item (in this case, *The Notebook*) is the candidate for recommendation to the user, and will typically not already be in the user's profile. The other items (*Big Fish*, *Pearl Harbour* and *The Illusionist*) form the chain. They are drawn from positively-rated items in the user's profile and are intended to support recommendation of the candidate item. Pairs of successive items in a chain satisfy a local constraint in the form of a similarity threshold; additionally, each item in the chain satisfies a global constraint in the form of a threshold on the level of coverage it contributes towards features of the candidate item. For example, *Big Fish* has the keywords: `secret-mission` and `parachute` in common with *Pearl Harbour*, as well as the keyword `romantic-rivalry` in common with *The Notebook*.

r-by-e treats each item as a set of elements. In our original work, elements are feature, but in one of our extensions (next section), an item' elements are other items — ones that its neighbours. *r-by-e* constructs an explanation chain by iteratively adding items from the user profile in an effort to cover potentially different elements of the candidate item. Hence, in the chain, the item closest to the candidate shares more of its elements with the candidate and the item farthest from the candidate shares the least with the candidate. Consecutive items in the chain must also share elements. Thus, the explanation chain enables a user to understand the mutual relationships between adjacent items as well as relationships between items from her profile and the candidate item in an incremental manner. In some sense, the chain "leads" the user through ever more relevant items from her profile towards the candidate. This, we believe, has the potential to explain recommendations in an effective manner that is sensitive to user understanding.

3.1 *r-by-e* top-*n* Recommendation

Let $\mathbb{I}$ be the set of all items. *r-by-e* works in a scenario of implicit ratings, where the user's profile $P \subseteq \mathbb{I}$ is the set of items that she likes. *r-by-e* will recommend up to n items from a set of candidate items, $I \subseteq \mathbb{I}$. Candidates I can be defined in whatever way is suited to the task in hand. Typically, for example, they will be items not already in P. But they could be further constrained by contextual factors such as time or location, e.g. recently-released movies, TV shows to be broadcast in the next few hours, or restaurants in the vicinity of the user. In our experiments, we define I to be items that are not in the user's profile but which do have at least a certain degree of similarity to the user's profile, $I = \left\{i \in I \setminus P \mid \text{sim}(F_i, F_p) > \theta, \exists\, p \in P\right\}$. Here F_i and F_p denote the features of items i and p, and sim is a similarity measure.

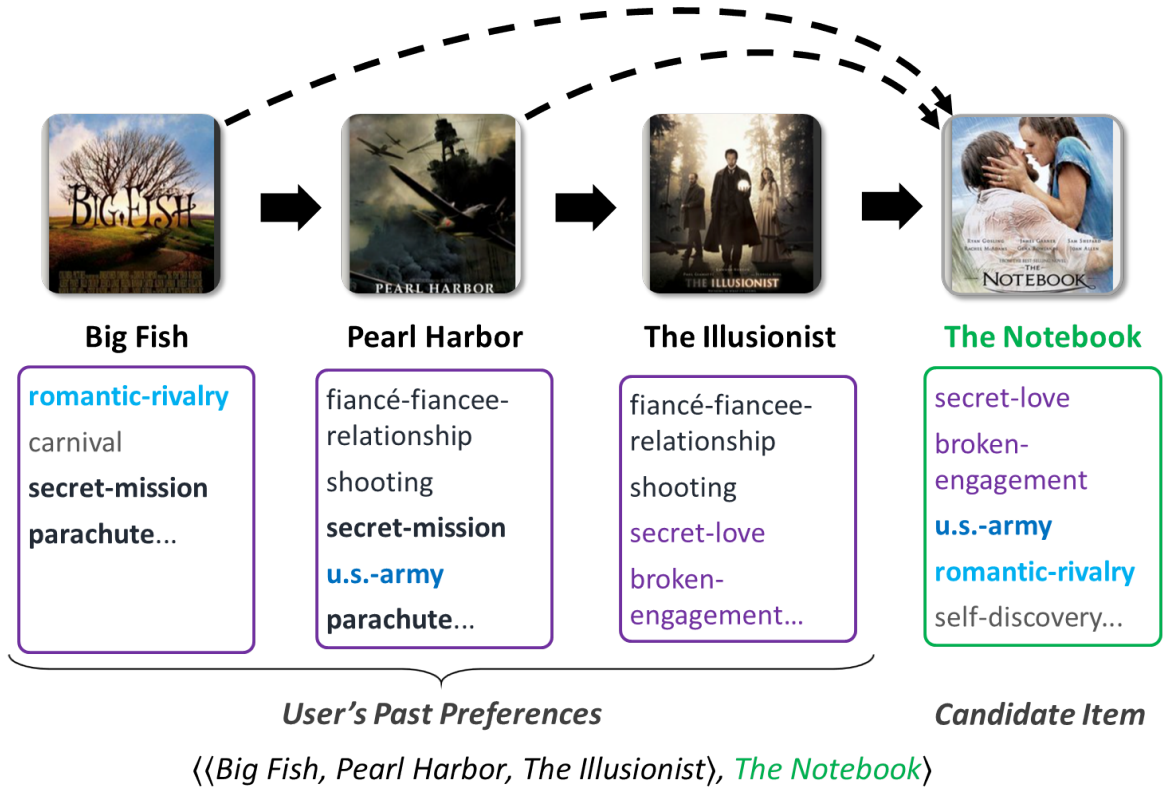


Fig. 3. An Explanation Chain in the Movie domain

Algorithm 1 *r-by-e* top-n recommendation.

Input: n, number of recommendations
I, set of candidate items
P, user's profile
θ, similarity threshold
ϵ, marginal gain threshold
Output: L^*, ranked list of top-n Explanation Chains.

```
1: function RECOMMEND(n, I, P, θ, ε)
2:     L ← [ ]
3:     for each i ∈ I do
4:         C ← GENERATECHAIN(i, P, θ, ε)
5:         if |C| > 0 then
6:             append ⟨C, i⟩ to L
7:     return SELECTCHAINS(L, n)
```

For each candidate item, *r-by-e* generates an Explanation Chain and then it selects the top n of those chains to recommend to the user; see Algorithm 1.

3.2 Chain generation

Given a candidate item, *r-by-e* works backwards to construct a chain: starting with the candidate item, it finds predecessors, greedily selects one, finds its predecessors, selects one; and so on; see Algorithm 2.

Algorithm 2 Chain generation.

Input: i, a candidate item
P, user's-profile
θ, similarity threshold
ϵ, marginal gain threshold
Output: C, an Explanation Chain C for candidate i.

function GenerateChain(i, P, θ, ϵ)
  $C \leftarrow [\,]$
  $sum\_ovrlps = 0$
  $j \leftarrow i$
  while True **do**
    $J \leftarrow \{p \in P \setminus C \mid \text{sim}(F_j, F_p) > \theta \ \wedge \ \text{ovrlp}(p, i, C) > \epsilon\}$
    if $|J| = 0$ **then**
      return C
    $j = \arg\max_{p \in J} \ \text{ovrlp}(p, i, C)$
    append j to C
    $sum\_ovrlps = sum\_ovrlps + \ \text{ovrlp}(j, i, C)$

The predecessors of an item are all its neighbours in the item-item similarity graph that satisfy four conditions: (a) they are members of the user's profile P; (b) they are not already in this chain; (c) their similarity to the subsequent item in the chain exceeds a similarity threshold θ; and (d) their overlap (see below) exceeds a marginal gain threshold ϵ. When there are no further predecessors, the chain is complete.

At each step, the predecessor that gets selected is the one with the highest *overlap*. The overlap $\text{ovrlp}(p, i, C)$ of adding predecessor p to partial chain C that explains candidate item i is given by:

$$\text{ovrlp}(p, i, C) = \frac{\left|(F_p \setminus \text{covered}(i, C)) \cap F_i\right|}{|F_i|} + \frac{\left|(F_p \setminus \text{covered}(i, C)) \cap F_i\right|}{\left|F_p\right|} \tag{1}$$

Here F_i and F_p denote the features of items i and p. $\text{covered}(i, C)$ is the set of features of candidate i that are already covered by members of the chain C, i.e. $\text{covered}(i, C) = \bigcup_{j \in C} F_j \cap F_i$. For more details on this, please refer to [36].

3.3 Chain selection

After constructing a chain C for each candidate item i, we must select the top-n chains so that we can recommend n items to the user, along with their explanations. This is done iteratively based on a chain's total coverage of the candidate item's features and the chain's dissimilarity to other chains already included in the top-n; see Algorithm 3.

Specifically, we score $\langle C, i\rangle$ relative to a list of all the items that appear in already-selected chains L^*. using the following:

$$\text{score}(\langle C, i\rangle, L^*) = \frac{sum\_ovrlps}{|C| \ + 1} + \frac{\left|C \setminus \bigcup_{j \in L^*} j\right|}{|C| \ + 1} \tag{2}$$

Here, *sum_ovrlps* is the sum of the overlaps of the items in the chain calculated while chain generation. For further details, please refer to [36].

Algorithm 3 Chain selection.

Input: L, list of Explanation Chains for different candidate items
n, number of recommendations
Output: L^*, ranked list of top-n Explanation Chains.

```
1: function SelectChains(L, n)
2:     if |L| ≤ n then
3:         sort L using score
4:         return L
5:     L* ← [ ]
6:     while |L*| < n do
7:         ⟨C, i⟩* = arg max_{⟨C,i⟩∈L} score(⟨C, i⟩, L*)
8:         append ⟨C, i⟩* to L*
9:         remove ⟨C, i⟩* from L
10:    return L*
```

4 EXTENDED *RECOMMENDATION-BY-EXPLANATION*

We extend *r-by-e* in three ways (see Figure 1). First, we consider a scheme for assigning weights to an item's features based on their informativeness. We define *weighted overlap* to take advantage of these weights. Second, we propose an alternative item representation which makes no explicit reference to features. We refer to it as a *neighbour-based* item representation. We explain it in more detail below. As we will see in our experiments, representing an item in terms of its neighbors (in place of its features) and generating chains in an effort to cover the candidate item's neighbours may result in more surprising recommendations. For this new item representation, we define both an unweighted and weighted overlap. Finally, we also generalize *r-by-e*'s *chain selection* (Equation 2). In place of simply adding the average overlap and average profile overlap, we define the score to be a linear combination of the two but controlled by a parameter α.

We model the *chain generation* step as a set-cover problem. In our previous work [35, 36], we formulated it in a feature-oriented way but, in this paper, we show that we have two ways of formulating this. In one approach, *feature-based* generation, our aim is to cover the candidate item's set of features; in the other approach, *neighbour-based* generation, we aim to cover the candidate's set of neighbours (i.e. similar items). Hence, the core of *chain generation* is computing overlap (ovrlp) between a potential predecessor and the candidate item for which the chain is being built. In the simplest case, overlap can be computed by *counting* the number of elements (either *features* or *neighbours*) that are covered. We call this *unweighted overlap*. An alternative way of computing overlap (wovrlp) is to assign weights to the elements based on their informativeness. We call this *weighted overlap*.

In *r-by-e*, *chain selection* takes the explanation chains (one per candidate item) and selects the top-n to recommend to the user. This is based on the chains' scores. The score function was originally a sum of the average overlap of the chain members and a diversification term. The diversification term measures the number of items (not features) in the user's profile uniquely covered by the items in a chain members relative to the chain length. We refer to this as *neighbour-based* selection. Analogous to the way we have both feature-based and neighbour-based *chain generation*, it seem obvious to design a feature-based version of *chain selection* to complement the neighbour-based representation. We did indeed design and experiment with a feature-based version of *chain selection*. However, we found that, in our dataset, the neighbour-based diversification term had a negligible

effect on a chain's total score. This is because, for a manageable size of chain (2–4 members), the features of the items in the chain cover only a small part of the large number of features of the items in the user profile. Consequently, a chain's score ends up being nearly equal to just the average overlap of the items. Therefore, in this paper we only show results for neighbour-based *chain selection*, irrespective of whether the *chain generation* is feature-based or neighbour-based. However, we do make one change to the score function: we generalize it so that it is no longer a simple sum of the average overlap and the diversification term; it is now a weighted sum, using a parameter α, allowing us to vary the importance of the two components of the definition.

We apply these ideas to a dataset in which we have user sentiments towards items' features, referring to this as *Sentiment-Aware Explanation Chains*. But, in order to do that, we first examine the effect of our extensions to *r-by-e* on a dataset that we used in our previous work, where items are described only by keyword (and not sentiments) [35, 36]. The complete experiment can be viewed in Section 5. After that, we extract concepts from user reviews and their respective sentiments towards each items, thus producing polarity-oriented item representations. We then evaluate how *r-by-e*'s extensions can benefit from these descriptions. The experiments details are covered in Section 6.

The following subsections give the formulae for the extensions that we have just outlined.

4.1 Feature-based generation

In *feature-based* settings, an item is represented as the set of its features. At each step of the *chain generation*, the predecessor that gets selected is the one that most covers the candidate' features.

4.1.1 Unweighted overlap. The unweighted overlap $\text{ovrlp}(p, i, C)$ of adding predecessor p to partial chain C that explains candidate item i is given by:

$$\text{ovrlp}(p, i, C) = \frac{2 \cdot \left|(F_p \setminus \text{covered}(i, C)) \cap F_i\right|}{|F_i| \ + \ \left|F_p\right|} \tag{3}$$

Here F_i and F_p denote the features of items i and p. $\text{covered}(i, C)$ is the set of features of candidate i that are already covered by members of the chain C, i.e. $\text{covered}(i, C) = \bigcup_{j \in C} F_j \cap F_i$. This equation returns a value of ovrlp in the range of $[0, 1]$.

4.1.2 Weighted overlap. Features associated with an item can be assigned weights based on how representative or informative they are to that item. In the information retrieval domain, for example, there are many ways to weight the terms of a corpus of documents [23].

We design an approach that deals with weighted features, called weighted overlap $\text{wovrlp}(p, i, C)$. For adding predecessor p to partial chain C that explains candidate item i, it is defined as follows:

$$\text{wovrlp}(p, i, C) = \frac{2 \cdot \left(\sum_{f \in (F_p \setminus \text{covered}(i,C)) \cap F_i)} w_{max} - \left|w_{fp} - w_{fi}\right|\right)}{|F_i| \ + \ \left|F_p\right|} \tag{4}$$

The numerator in the definition of $\text{wovrlp}(p, i, C)$ measures p's weighted coverage of those features of i that are not yet covered by the chain. Specifically, it penalizes the number of these features by subtracting the difference between their weights w_{fi} and w_{fp}. Since weights of features can vary depending on the type of representation, w_{max} represents the maximum value that $|w_{fp} - w_{fi}|$ can assume, thus allowing an increase in $\text{wovrlp}(p, i, C)$ based on the closeness of w_{fp} and w_{fi}. For instance, if two items possess the feature `cinematography` and their weights are close, $\text{wovrlp}(p, i, C)$ would have a great increase in its score, whereas if they have very different weights, the difference factor would have lower score thus contributing less to $\text{wovrlp}(p, i, C)$. The weights of the features are defined depending on the approach being used. In this paper, we evaluate with a

weighted keyword approach (detailed in Section 5.1.1) and our sentiment-based weighted approach (detailed in Section 6.1.3).

4.2 Neighbour-based generation

In *neighbour-based* settings, an item i is represented as a set of its neighbours N_i. Its neighbourhood contains items whose similarity to i exceeds a threshold θ: $N_i = \{j \in \mathbb{I} \setminus i : \text{sim}(F_i, F_j) > \theta\}$. At each step, the aim is to cover neighbours of the candidate item instead of its features. Notice that here item features are still used, but they are used *implicitly*, i.e., they provide item-item similarity but they (and their weights, where appropriate) are not used directly in the formulae.

We define *neighbour-based* overlap slightly differently from *feature-based* overlap. In this approach, we foud that covering a candidate item's neighbours may result in relatively loosely connected chains — more loosely connected than those built by covering its contents. In *r-by-e*, loosely connected chains may have lower interpretability. To 'tighten' the chains, in the definition of *neighbour-based* overlap, we remove already covered elements (covered(i, C)) from the size of neighbours (e.g. N_i and N_p) in the denominator. This assures that chain members have relatively more neighbours in common with the candidate's neighbours.

4.2.1 Unweighted overlap. We will denote the unweighted overlap of adding predecessor p to partial chain C that explains candidate item i in the neighbour-based setting by ovrlp(p, i, C), which is the same as we used in the feature-based setting. The context will make clear which version is intended at any point. The definition is:

$$\text{ovrlp}(p, i, C) = \frac{2 \cdot \left|(N_p \setminus \text{covered}(i, C)) \cap N_i\right|}{|N_i \setminus \text{covered}(i, C)| \; + \; \left|N_p \setminus \text{covered}(i, C)\right|} \tag{5}$$

Here N_i and N_p are the neighbours of items i and p. covered(i, C) is the set of neighbours of candidate i that are already covered by members of the chain C, i.e. covered$(i, C) = \bigcup_{j \in C} N_j \cap N_i$. The denominator means that coverage is relative to the size of N_i and N_p after removing already covered neighbours. Including N_p in the denominator ensures that p's fitness to explain the candidate is not inflated simply by virtue of having more neighbours.

4.2.2 Weighted overlap. Neighbours of an item can be assigned weights based on their closeness to the item. In this approach, we simply define closeness between two items as the similarity between their sets of features. So, the weight (w_{ji}) of a neighbour j of a candidate item i equals the similarity between them: $w_{ji} = \text{sim}(j, i)$.

The weighted neighbour-based reward wovrlp(p, i, C) of adding predecessor p to partial chain C that explains candidate item i is given by:

$$\text{wovrlp}(p, i, C) = \frac{2 \cdot (\sum_{j \in (N_p \setminus \text{covered}(i,C)) \cap N_i)} 1 - \left|w_{jp} - w_{ji}\right|)}{|N_i \setminus \text{covered}(i, C)| \; + \; \left|N_p \setminus \text{covered}(i, C)\right|} \tag{6}$$

This is analogous to Eq. 4. Notice that neighbour-based overlap makes no explicit reference to features. Features are being used, but only implicitly: the set N_i contains items that have high feature similarity with the candidate item i; and, in the weighted case, weights are defined in term of feature similarity.

4.3 Generalized chain selection

We generalize *chain selection* so that we score a chain $\langle C, i \rangle$ relative to a list of all the items that appear in already-selected chains L^* using the following:

$$\text{score}(\langle C, i\rangle, L^*) = (1-\alpha) \cdot \frac{sum\_ovrlps}{|C| \ + 1} \ + \alpha \cdot \frac{\left|C \setminus \bigcup_{j \in L^*} j\right|}{|C| \ + 1} \tag{7}$$

Here, the first term is the average of the overlaps of the candidate features (or neighbours, depending upon the item representation) in the chain and the second term is the coverage of items in the user profile with respect to the length of the chain. α is a parameter that balances the two. We have found increasing α to have the effect of increasing the length of the chains. On the whole, as we shall see, this has the effect of increasing the surprise as well as the diversity of recommendations.

5 EXTENDED CHAINS ON KEYWORDS

In this section, we evaluate the extensions we have made to the formulation of *r-by-e* on a movie dataset in which item features are keywords.

5.1 Dataset

We used the hetrec2011-movielens-2k dataset[5] but, in place of the tags given in that dataset, we assigned each movie its keywords from IMDb[6]. From the original dataset, only those movies for which IMDb has keyword information are used in our experiments. Thus, the dataset comprises 2113 users, 5992 movies, 80639 keywords, and over half a million ratings.

On average, a typical movie has 107 keywords, ranging from 2 to 626, which shows a very high variance in the number of keywords. Each movie has non-zero similarity with, on average, 77% of the other movies in the dataset. This suggests that the item-item similarity graph is highly dense with an average out-degree of a typical node being around 4600.

5.1.1 Keyword extraction and weighting. For the feature-weighted approaches, we use the well-known term frequency-inverse document frequency (TF-IDF) weighting scheme [23]. In effect, we treat an item as a document and its features as terms.

As shown in Eq. 8, the weight of a feature f of an item i with respect to the set of all items $\mathbb{I}$ is proportional to the frequency of occurrence of f in i (denoted as o_{fi}), but inversely proportional to the frequency of occurrence of f in $\mathbb{I}$ overall, thus giving preference to the features that help to discriminate each item $i \in \mathbb{I}$ from the other items in the collection. The set of items consisting of the feature f is denoted as $\mathbb{I}_f$.

$$w_{fi} = \frac{\left(1 \ + \ \log(o_{fi})\right) \cdot \left(\log \frac{|\mathbb{I}|}{|\mathbb{I}_f|}\right)}{\sqrt{\sum_{f' \in i} \left(\left(1 \ + \ \log(o_{f'i})\right) \cdot \left(\log \frac{|\mathbb{I}|}{|\mathbb{I}_{f'}|}\right)\right)^2}} \tag{8}$$

The equation above is a variant of TF-IDF modeling with cosine normalization in feature–item space. Intuitively, it measures the informativeness of a feature f for an item i with respect to the informativeness of all other features in the item. The set of item features are of different sizes. In general, larger set of features have higher feature frequencies because many features are repeated. The cosine normalization helps lessen the impact of the size of the item description in the modeling [17].

5.2 Offline evaluation

We ran an offline experiment to evaluate the different versions of *r-by-e* on this dataset. We wanted the experiment to reveal the effect of the differences between the following:

[5]https://grouplens.org/datasets/hetrec-2011/
[6]http://www.imdb.com

- *feature-based* versus *neighbour-based*: The former represents an item as a set of its keywords, while the latter represents an item as a set of its neighbours (similar items) in which the features are used only indirectly.
- *unweighted* versus *weighted*: The former takes into account the features or neighbours in a binary way (1 indicates that the feature or neighbour is present and 0 shows it is not), while the latter assigns weights in $[0, 1]$ to the features or neighbours.
- *CB* and *CB*-$|C|$ versus versions of *r-by-e*: *CB* is the classic content-based system [32] and *CB*-$|C|$ is the dynamic content-based system [36]. The *CB* systems rely on similarity relationships between members of the user profile and the candidate item, whereas versions of *r-by-e*, by requiring consecutive members of chains to be similar to each other, additionally take into account similarity relationships between members of the user profile themselves.
- The influence of α: When selecting the top-n chains, α balances the overlap of candidate features and the overlap of items in the user profile (see Eq. 7). We vary α from 0 (overlap of candidate features or neighbours only) to 1 (overlap with the user profile only) in steps of 0.1.

For conciseness, we will refer to the four versions of *r-by-e* using just *fb* for unweighted feature-based, *wfb* for weighted feature-based, *nb* for unweighted neighbour-based, and *wnb* for weighted neighbour-based.

5.2.1 Experiment settings. In *r-by-e*, user profiles simply contain items the user likes. We treated ratings of 4 and 5 as 'likes', so user u's profile is given by $\{i \mid r_{u,i} \geq 4\}$. We split each user's ratings into training, validation and test sets in the ratio 60 : 20 : 20, repeated five times.

We vary the α parameter from Eq. 7 in a $[0, 1]$ interval in steps of 0.1. We consider the values (0.03, 0.06, 0.09) for the similarity threshold (θ) in the definition of N_i and different sets of values for the marginal gain threshold (ϵ): for the feature-based representation, we experimented with (0.03, 0.06, 0.09); and for neighbour-based representation, we experimented with (0.05, 0.10, 0.15).

The reason behind using different values for the marginal gain threshold ϵ for the two representations is the difference in the size of an item's set of keywords (for the feature-based representation) and the size of its set of neighbours (for the neighbour-based representation). A typical item has on average 107 keywords while it is connected to 77% of the available items (i.e. over 4600). Hence, the contribution each chain member makes when covering a candidate's features is generally lower than when covering its neighbours. Consequently, we experiment with higher values of the marginal gain threshold for the neighbour-based representation.

Three values of each of θ and ϵ with eleven values of α gives 99 configurations for each of the four versions of *r-by-e*. We use this offline experiment to decide which version performs the best and how it works on different values of α.

5.2.2 Experiment results. Table 1 and Table 2 summarize the results of the feature-based and neighbour-based approaches. The columns of the table are the different evaluation measures. We use the definitions of those measures found in [18]. The rows are divided into blocks, one block per optimization criteria for which all hyperparameters are tuned. Rows within blocks are for different recommendation approaches.

Feature-based chain generation. We looked at the differences in the results between: i) *fb* and *fb-CB*-$|C|$; ii) *wfb* and *wfb-CB*-$|C|$; and iii) *fb* and *wfb*. The results for the first two comparisons are statistically significant except for *Diversity* when optimized for precision. For the most part, differences in the results for (iii) are small but, since standard deviations are low, in all but one case they are statistically significant. They are not significantly different for precision when optimized for % of explanations of size 2–4. In comparison to the *fb* and *wfb* recommenders, *CB* attains higher diversity, surprise, novelty, and coverage, but around five times lower precision. Similarly, the

Table 1. Results of the offline experiment that uses *feature-based* representations, where features are keywords. All of the *fb* and *wfb* results are statistically significant with respect to *fb-CB-|C|* and *wfb-CB-|C|* respectively (t-test with $p < 0.05$) except the one shown in italics.

Recommender	θ, ϵ & α optimized for	Precision	Diversity	Surprise	Novelty	Coverage	% of explanations of size 2–4
CB		0.0209	0.9803	0.9444	0.5791	0.7606	NA
fb		0.1076	*0.9274*	0.7682	0.3715	0.2026	0.2899
fb-CB-|C|	Precision	0.0124	0.9251	0.8894	0.4391	0.0585	0.2085
wfb		0.1093	*0.9256*	0.7687	0.3640	0.1990	0.3115
wfb-CB-|C|		0.0124	0.9251	0.8893	0.4391	0.0585	0.2048
CB		0.0203	0.9825	0.9498	0.6062	0.6727	NA
fb		0.0694	0.9498	0.7932	0.4429	0.2556	0.4598
fb-CB-|C|	Diversity	0.0063	0.9613	0.9255	0.5198	0.0391	0.5726
wfb		0.0730	0.9489	0.7915	0.4312	0.2521	0.4851
wfb-CB-|C|		0.0063	0.9613	0.9255	0.5201	0.0391	0.5724
fb		0.0146	0.9307	0.8906	0.4390	0.2697	0.9882
fb-CB-|C|	% of explanations	0.0050	0.9747	0.9346	0.4930	0.0288	0.0074
wfb	of size 2–4	0.0152	0.9302	0.8901	0.4381	0.2660	0.9878
wfb-CB-|C|		0.0053	0.9742	0.9338	0.4915	0.0296	0.0057

CB-|*C*| recommenders have higher diversity, surprise, and novelty but lower precision and coverage. We discard the *CB* recommender when optimizing for % of explanations of size 2–4 as this criteria is not applicable to *CB*.

As stated before, feature-based *r-by-e* attempts to cover the features of the candidate item as well as the items in the user profile. This enables *r-by-e* to generate more relevant recommendations while remaining competitive in its diversity and serendipity. On the other hand, the content-based approaches only consider item-item similarities and are unable to provide a balance between recommendation accuracy, diversity and surprise. A set of randomly-chosen items, for example, is likely to be diverse, but the individual recommendations are less likely to be relevant to the user. Or, to give another example, recommending a set of popular items will, in many cases, result in high accuracy but may lead to lower diversity [2]. In our experiments, we see that *CB*-|*C*| approaches generate more diverse and serendipitous but less accurate recommendations. Such recommendations are concentrated on only a small fraction of the catalog that results in lower catalog coverage than *r-by-e*.

Neighbour-based chain generation. Differences in the results between: i) *nb* and *nb-CB-|C|*; and ii) *wnb* and *wnb-CB-|C|* are statistically significant in all cases. On the other hand, differences in the results for *nb* and *wnb* are generally very low and in no case are they statistically significant. Overall, the *CB*-|*C*| recommenders attain higher values of diversity, surprise, and novelty, but have lower precision and coverage. Moreover, one can see that both *nb* and *wnb* have produced more explanations (in their case, chains) of size 2–4 size than the *CB*-|*C*| systems in any optimization criteria.

The results here indicate that a comparison between neighborhood-based *r-by-e* and content-based baselines follow roughly the same pattern of the comparison as with the feature-based approaches. This highlights *r-by-e*'s capability of understanding user's preferences better than the content-based approaches even when the item representations make no explicit use of item features.

Feature-based vs. neighbour-based chain generation. We will now compare the two types of item representation by looking at results from both Table 1 and Table 2 together. When optimizing hyperparameters for precision, feature-based approaches attain three times higher precision with

Table 2. Results of the offline experiment that uses *neighbour-based* representations, where features are keywords. All of the *nb* and *wnb* results are statistically significant with respect to *nb-CB-|C|* and *wnb-CB-|C|* respectively (t-test with $p < 0.05$).

Recommender	θ, ϵ & α optimized for	Precision	Diversity	Surprise	Novelty	Coverage	% of explanations of size 2–4
CB		0.0209	0.9803	0.9444	0.5791	0.7606	NA
nb		0.0361	0.9129	0.8410	0.4130	0.1771	0.6518
nb-CB-|C|	Precision	0.0125	0.9240	0.8914	0.4467	0.0641	0.4266
wnb		0.0357	0.9128	0.8410	0.4104	0.1772	0.7409
wnb-CB-|C|		0.0124	0.9240	0.8913	0.4464	0.0642	0.4245
CB		0.0203	0.9825	0.9498	0.6062	0.6727	NA
nb		0.0138	0.9456	0.8896	0.4896	0.2904	0.8174
nb-CB-|C|	Diversity	0.0041	0.9885	0.9524	0.5596	0.0264	0.4102
wnb		0.0177	0.9463	0.8895	0.4664	0.2907	0.8131
wnb-CB-|C|		0.0040	0.9885	0.9524	0.5598	0.0265	0.4078
nb		0.0157	0.9121	0.8775	0.4101	0.1602	0.9756
nb-CB-|C|	% of explanations	0.0041	0.9837	0.9469	0.5483	0.0215	0.0870
wnb	of size 2–4	0.0159	0.9124	0.8774	0.4087	0.1599	0.9750
wnb-CB-|C|		0.0041	0.9837	0.9468	0.5490	0.0627	0.0868

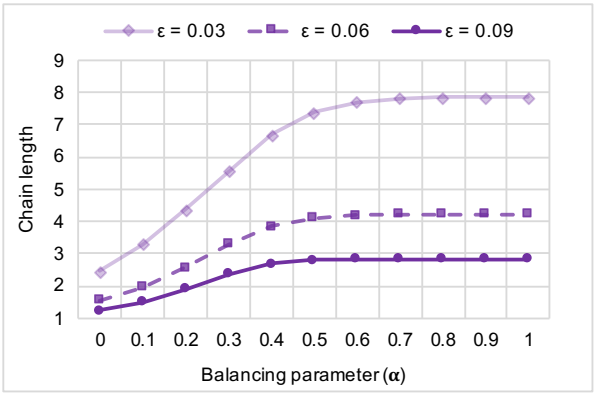
almost similar levels of diversity and % of explanations of size 2–4. The neighbour-based approaches result in more surprising and novel recommendations. In the case of optimizing for diversity, feature-based approaches give five times more relevant recommendations with a nearly equal level of diversity. Again, neighbour-based approaches attain higher levels of surprise and novelty with over 81% of chains of size 2–4. Finally, when hyperparameters are optimized for % of explanations of size 2–4, the two types of approaches recommend items with almost equal precision. However, feature-based approaches produce recommendations with greater variety, higher levels of surprise and novelty and with a somewhat greater percentage of chains of size 2–4.

Neighbour-based approaches try to cover neighbours of the candidate items instead of their features. Intuitively, this may lead to higher levels of diversity and surprise than the feature-based approaches. Our offline results confirm that the neighbour-based recommendations are more diverse and surprising but there is a trade-off with their relevance. However, they still perform better than the content-based baselines.

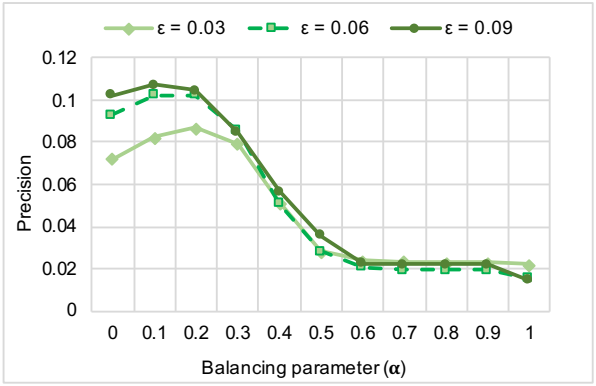
Let us select one of the four approaches for further study. To pick one, let us assume that optimizing for the % of explanations of size 2–4 is best, since it generally gives explanations that are not so long as to be uninterpretable. In this setting, *wfb* performs better than other versions of *r-by-e*, and so this is the version that we will explore further.

We will study the effect of hyperparameters θ, ϵ, and α on the performance of *wfb*. Although, we considered different values (0.03, 0.06, 0.09) for the similarity threshold θ in the definition of N_i, we found that varying θ does not make any noticeable effect on the evaluation measures. Therefore, we only show results for $\theta = 0.03$. Varying the marginal gain threshold ϵ affects the *chain generation* step (in particular, the chain length) and the balancing parameter α plays a role in the scoring function of the *chain selection* step (hence it affects the top-*n* recommendations).

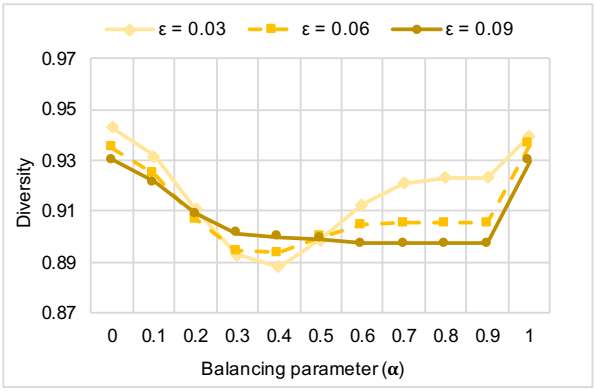
Figure 4 has six sub-figures — one for each evaluation measure. Each line that we plot in a sub-figure is for one of the three different values of ϵ. In most cases, increasing ϵ does not change the trend of the measure; it only 'shifts' the values of the measure because higher values of ϵ impose a stricter constraint. It can also be seen that in almost all the plots, values of the evaluation measures remain constant for $\alpha \in [0.06 - 0.09]$. This means that almost the same chains are selected in the top-*n* for this range of values for α. We look in the detail at the results for each evaluation measure individually. We explain results by referring to Eq. 7: for conciseness, we refer to its first term,



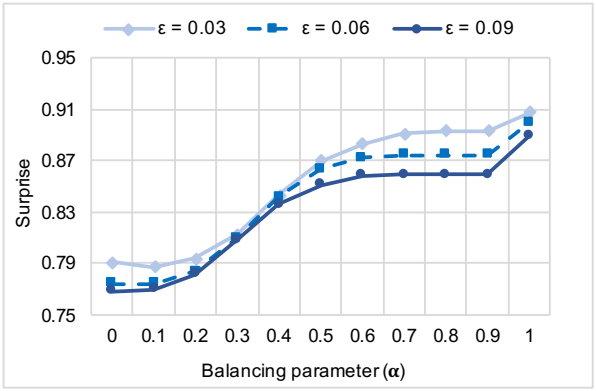
(a) Chain length



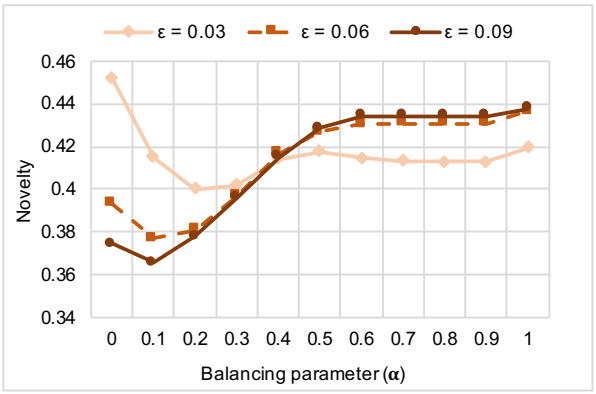
(b) Precision



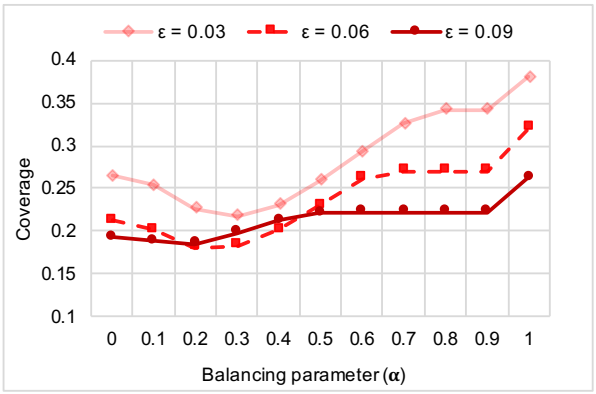
(c) Diversity



(d) Surprise



(e) Novelty

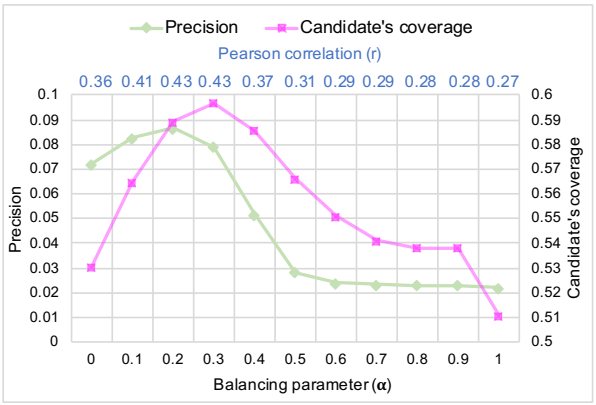


(f) Coverage

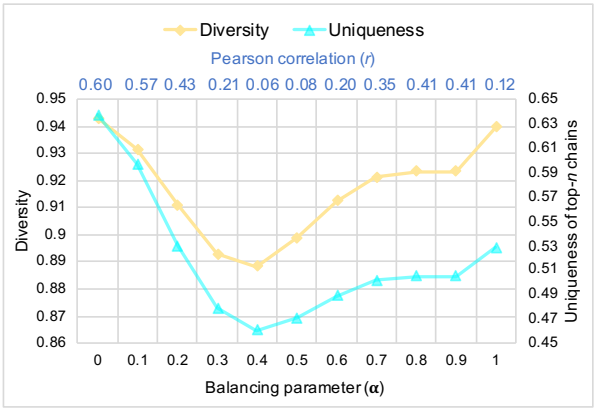
Fig. 4. Results for *wfb* with $\theta = 0.03$, $\epsilon \in \{0.03, 0.06, 0.09\}$, and α ranges in $[0.0 - 1.0]$ for extended chains on keywords.

which indicates the average amount of overlap of candidate features, as the *overlap term*; and we refer to its second term, which indicates the overlap of items in the user profile with respect to the length of the chain, as the *profile term*.

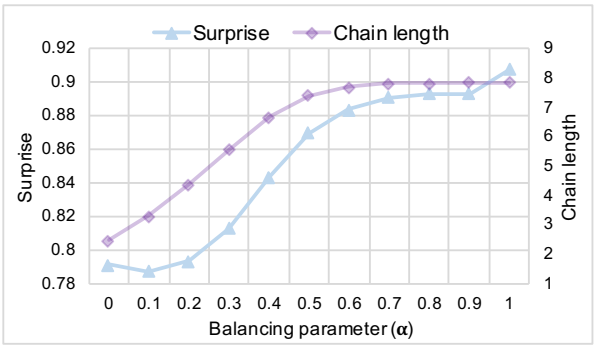
Chain length: In Figure 4(a), we see that the length of top-n chains (averaged over all users) increases up to $\alpha = 0.5$; then, further increase in α does not affect the length much. This indicates



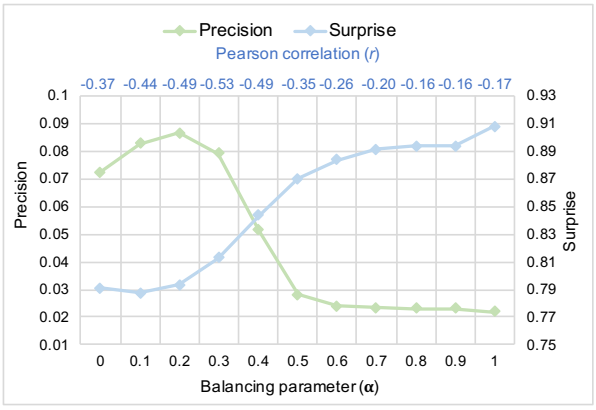
(a) Precision vs. Candidate coverage



(b) Diversity vs. Uniqueness



(c) Surprise vs. Chain length



(d) Precision vs. Surprise

Fig. 5. Relationships between precision, diversity and surprise with candidate coverage, uniqueness, chain length and surprise at $\theta = 0.03$ and $\epsilon = 0.03$. Above each sub-figure, we show Pearson correlation. All of the correlation coefficient values are statistically significant (i.e. $p < 0.05$).

that increasing α gives more weight to the profile term which enables the system to select longer chains. It is also noteworthy that increasing ϵ imposes a stricter constraint on chains such that their average length reduces substantially.

Precision: In Figure 4(b), we see that precision varies in four different ways as we increase the value of α: i) up to 0.2, it increases; ii) from 0.2 to 0.7, it decreases; iii) from 0.7 to 0.9, it remains almost constant; and iv) at 1.0, it slightly decreases. In explanation chains, precision is proportional to the candidate's coverage: the greater the candidate's coverage, the higher is the precision. We will define the candidate's coverage as the ratio of the number of candidate's features covered by the chain members to the size of the candidate's feature set. The variation in precision with respect to α indicates that the system selects those chains where: in the case of (i), adding members to the chain (as chain length increases) helps to increase the candidate's coverage; for (ii), the overlap term dominates, so the system selects chains that have a large candidate feature set that cannot be covered easily, which reduces the coverage and so also reduces the precision; for (iii), the system selects almost similar chains; and for (iv), the system totally ignores the overlap term and so the chains it selects do not try to cover the candidate, they only try to cover the profile, hence precision decreases. We plot the relationship between the precision and the candidate's coverage in Figure 5(a).

Diversity: In Figure 4(c), we see that, (i) up to $\alpha = 0.4$, diversity decreases; and (ii) it then increases up to $\alpha = 1.0$. In explanation chains, the diversity of the top-n recommendations depends upon the uniqueness of the chain members: the lower the overlap among chains, the higher is the level of diversity. We will define the uniqueness of a set of chains recommended to a user as the ratio of the number of distinct items in the union of the chains over the sum of their lengths. We find that diversity is highest at $\alpha = 0.0$ because there is least overlap among chains; increasing α, for (i), increases the chain length and so the overlap; however, for (ii), when the profile term starts to dominate, the system selects even longer chains which increases uniqueness among chains, and thus diversity. In Figure 5(b), we show that both diversity and uniqueness follow the same trend for increasing α.

Surprise: Figure 4(d) shows that up to $\alpha = 0.6$, surprise increases; then it remains almost unchanged (up to $\alpha = 0.9$); and finally, at $\alpha = 1.0$, it increases again. Increasing α gives more weight to the profile term. In effect, the system selects longer chains that increases their surprise. The intuition behind this relationship is that chains are shorter when they easily cover the candidate's features, while they are longer when the candidate's features are not easily covered.We find a correlation between the surprise and the chain length that we show in Figure 5(c). We also see in Figure 5(d) that precision and surprise exhibit almost the opposite behaviour of each other. This is confirmed by the Pearson correlation values (also shown in the Figure) that are negative for all values of α. This inverse relation between precision and surprise also indicates inverse proportionality between surprise and the candidate's coverage.

Novelty: Figure 4(e) shows that novelty decreases up to $\alpha = 0.2$; then, increases up to $\alpha = 0.6$; up to $\alpha = 0.9$, it remains almost unchanged; and finally, at $\alpha = 1.0$, it slightly increases. It can be seen that novelty varies in a way that is similar to surprise on increasing α. This indicates that for lower values of α, the system selects those chains that have high coverage of candidates: intuitively, popular items can be easily covered. As α increases, the system suggests novel items which cannot be covered easily and need more items from the user profile to support them.

Coverage: In Figure 4(f) we see that coverage varies in a way that is quite similar to diversity on increasing α. First, it decreases up to $\alpha = 0.3$, then it increases up to $\alpha = 0.6$, it becomes almost unchanged up to $\alpha = 0.9$, and finally, it increases at $\alpha = 1.0$. Shorter chains with low levels of uniqueness cannot cover much of the catalogue, while longer chains with high uniqueness cover a larger part of it.

6 EXTENDED CHAINS ON SENTIMENTS

Thus far, we have described some extensions to *r-by-e* and evaluated them in a keyword-based scenario. But those extensions, especially the weighted approaches, enabled us to develop *Sentiment-Aware Explanation Chains*.

Using keywords as item features presents some issues, mainly: i) we can only convey frequency and rarity information about them; ii) they do not necessarily convey the aspect's of the items that interest the users; iii) they may be ambiguous if taken out of context. Taking these issues in consideration, we turn to user-provided texts, i.e. user reviews, in order to further aggregate semantics to our explanation chains.

Using these texts, allied with state-of-the-art natural language processing tools, we are able to produce semantically richer item features which can both help produce better explanation chains and help users understand their recommendations. These features are 'concepts', which do not suffer from the issues of synonymy and polysemy since they convey an idea and are not merely single words. For instance, before we could have a keyword bank which could mean a land mass or a financial institution; now we would have a concept for each of those meanings. Moreover, we can extract *sentiments* towards those concepts, since user reviews are opinionated texts.

In this section we evaluate *Sentiment-aware Explanation Chains*, which are extended chains that use sentiment-aware concepts instead of keyword-based features. These chains guarantee that items are connected only if they share features with close polarity (sentiment) scores.

6.1 Dataset

In this study, we use only hetrec2011-movielens-2k movies that were released between the years 2000 and 2011 inclusive. This results in trials that use 1851 ($\approx$ 30%) of the 5992 movies in the dataset. In the offline experiments and user trial that we report in the remainder of this paper, we use sentiment data extracted from user reviews for each of these 1851 movies.

We followed the steps that are proposed in [11, 12] for preparing the sentiment-based item representation. The only difference in the way we prepared this dataset from the methods described in [11, 12] is that, after extracting concepts from the user reviews, here we filtered them using cosine normalized TF-IDF scores as we mentioned in Eq. 8, whereas in the previous work these are filtered using a different but related measure that the authors called item frequency (IF) [12]. The remainder of this section summarizes these steps.

6.1.1 User reviews to concepts. A concept in the approach given in [11, 12] describes an idea or a notion. Concepts can be seen as *synsets*, i.e. sets of word synonyms which define an idea. As stated before, using concepts instead of words reduces the problems of synonymy and polysemy.

In order to extract concepts and sentiments from user reviews, two different natural language processing resources were used: Stanford CoreNLP[7] [24] for sentence splitting, parsing and sentence-level sentiment analysis; and BabelFy[8] [25] for word sense disambiguation and entity linking.

First, the items' reviews were processed with Stanford CoreNLP using the following pipeline: tokenization, part-of-speech (POS) tagging, parsing, sentence splitting and sentiment analysis.

Next, BabelFy processes the texts, returning disambiguated concepts in the form of BabelNet synsets [30]. BabelNet[9] is a knowledge base that links several linguistic resources, such as Wikipedia, Wikidata, and WordNet, among others. Its unified ontology organizes all these resources into BabelNet synsets, which define both concepts (such as "romance", "action movie") and named entities (such as "Steven Spielberg" or "Willem DaFoe"), and provides links between them.

The synsets that are selected to compose our vocabulary (i.e., our features) are only those that come from common and proper nouns, and noun phrases. Our vocabulary is built only with these parts-of-speech because, in sentiment analysis, features most commonly come from nouns and noun phrases (such as 'action', 'plot' and 'special effects' in a movie review). Adjectives and adverbs are opinion words, i.e. words that indicate sentiment towards features, and thus are used in calculating the sentiment of an aspect, sentence or document [22]. This vocabulary, in its current state, is very large and contain many noise and useless features. Before assigning sentiments to them, first we need to filter out some of the features, reducing its size. In the following, we employ TF-IDF weights to aid in that filtering. After that, we assign sentiments to the remaining concepts.

6.1.2 Filtering concepts. In this experiment, explanation chains are built over concepts. In order to improve their quality and informativeness, concepts were filtered using the following two steps:

- Concepts that appear in only one item were removed from the vocabulary. Since our chains are constructed from links between features of the target item and items present in the user profile, it is natural that features which appear in a single item are removed because they

[7] https://stanfordnlp.github.io/CoreNLP/
[8] http://babelfy.org/
[9] https://babelnet.org/

will not influence *chain generation*. Similarly, concepts that were present in all the items in the dataset were also removed since they are too general.
- Concepts obtained from the previous step were assigned weights using cosine normalized TF-IDF scores as in Eq. 8. Concepts whose average weight across the reviews in which they appeared were less than 0.01 were removed. The remaining concepts constitute the vocabulary which was used to produce item representations.

With that filtering, the vocabulary contains 34,088 concepts. On average, a typical movie has 324 concepts ranging from 104 to 646, which shows a very high variance in the number of concepts. Each movie has non-zero similarity with 90% (over 1670) of the other movies in the dataset. This suggests that the item-item similarity graph is even denser than the similarity graph of keywords where each item was connected to 77% of the other movies. Also, the average item-item similarity is greater in comparison to the previous version of the dataset which, we will see, will affect the quality of the top-n chains recommended to the user.

6.1.3 Representing items and producing explanation chains. Finally, concepts and the items they occur in were modeled using a traditional vector space representation. Each item is a vector of sentiment scores assigned to the concepts from the vocabulary. These sentiments can be positive (>=4), negative (<3), or neutral (=3). The overall sentiment score for a feature of an item is the average of the sentiment scores according to its appearances in the reviews of the corresponding item. In case a feature is not present, a zero is assigned to it in the vector. This, of course, is still a *feature-based representation* (not a neighbour-based representation).

However, we can define the item-item similarity graph on this *feature-based representation* using cosine similarity. From this, we can also define the *neighbour-based representation*, where each item is a set of its neighbours in the graph. The values in the vectors are like the weights we used in our weighted approaches. Hence, unweighted versions of the sentiment-aware approached do not make any sense. Therefore, we run experiments only on weighted versions: weighted feature-based (*wfb*) and weighted neighbour-based (*wnb*).

6.2 Offline evaluation

We ran an offline experiment to evaluate the performance of sentiment-aware *r-by-e*. We compare i) *wfb* and *wnb*; ii) both *wfb* and *wnb* with a classic content-based recommender (*CB*); and iii) both *wfb* and *wnb* with their corresponding dynamic content-based recommenders: *wfb-CB-|C|* and *wnb-CB-|C|*.

6.2.1 Experiment settings. We use the same experiment settings as we described previously. The only difference is in the values of the similarity threshold (θ) and the marginal gain threshold (ϵ). We experimented with θ from (0.06, 0.09, 0.12) for both *wfb* and *wnb*; and we use ϵ from (0.03, 0.06, 0.09) for *wfb*, and from (0.10, 0.20, 0.30) for *wnb*.

6.2.2 Experiment results. Table 3 and Table 4 summarize the results of the weighted feature-based and weighted neighbour-based approaches.

Feature-based chain generation. We looked at the differences in the results between: i) *CB* and *wfb*; and ii) *wfb* and *wfb-CB-|C|*. The *wfb* results for both the comparisons are statistically significant. In comparison to the *wfb* recommender, the *CB* and *wfb-CB-|C|* recommenders attain higher levels of diversity, surprise, and novelty but lower values of precision and coverage. In particular, the *wfb-CB-|C|* recommender covers only around 3% of the catalogue with almost irrelevant recommendations. We discard the *CB* recommender when optimizing for % of explanations of size 2–4 as this criteria is not applicable to *CB*. One thing worth noticing is that *wfb-CB-|C|* is capable of producing a very

Table 3. Results of the offline experiment that uses *feature-based* representations on sentiments. All of the *wfb* results are statistically significant with respect to *wfb-CB-|C|* (t-test with $p < 0.05$).

Recommender	θ, ϵ & α optimized for	Precision	Diversity	Surprise	Novelty	Coverage	% of explanations of size 2–4
CB		0.0145	0.9383	0.9126	0.5048	0.3068	NA
wfb	Precision	0.1053	0.9130	0.8697	0.3468	0.3271	0.4040
wfb-CB-|C|		0.0036	0.9300	0.9094	0.5597	0.0282	0.0000
CB		0.0145	0.9383	0.9126	0.5048	0.3068	NA
wfb	Diversity	0.0223	0.9225	0.9006	0.4056	0.5185	0.9025
wfb-CB-|C|		0.0033	0.9308	0.9102	0.5619	0.0245	0.0000
wfb	% of explanations	0.0838	0.9104	0.8778	0.3752	0.4930	0.9731
wfb-CB-|C|	of size 2–4	0.0014	0.9483	0.9234	0.6102	0.0361	0.2054

Table 4. Results of the offline experiment that uses *neighbour-based* representations on sentiments. All of the *wnb* results are statistically significant with respect to *wnb-CB-|C|* (t-test with $p < 0.05$).

Recommender	θ, ϵ & α optimized for	Precision	Diversity	Surprise	Novelty	Coverage	% of explanations of size 2–4
CB		0.0145	0.9383	0.9126	0.5048	0.3068	NA
wnb	Precision	0.0342	0.9003	0.8884	0.4072	0.1305	0.8448
wnb-CB-|C|		0.0024	0.9327	0.9108	0.5926	0.0348	0.0000
CB		0.0145	0.9383	0.9126	0.5048	0.3068	NA
wnb	Diversity	0.0088	0.9276	0.9053	0.4695	0.7462	0.9857
wnb-CB-|C|		0.0020	0.9448	0.9211	0.6241	0.0739	0.7061
wnb	% of explanations	0.0117	0.9186	0.9005	0.4250	0.2700	0.9997
wnb-CB-|C|	of size 2–4	0.0018	0.9513	0.9247	0.5989	0.0976	0.1804

small number of explanations of size 2–4; in fact, while optimizing hyperparameters for precision and diversity, it selected no explanations of that size. *wfb*, on the other hand, was able to attain over 90% of explanations of that size for two out of three optimization criteria.

Neighbour-based chain generation. Now, we see the differences in the results between: i) *CB* and *wnb*; and ii) *wnb* and *wnb-CB-|C|*. They are statistically significant in all cases. Changing the item representation does not apply to *CB*, so its results are the same as before. The *CB* and *wnb-CB-|C|* recommenders attain higher values of diversity, surprise, and novelty than *wnb*, while having lower precision and % of explanations of size 2–4. In particular, the *wnb-CB-|C|* recommender has low catalog coverage with explanations satisfying the size constraint.

Feature-based vs. neighbour-based chain generation. We will now compare the two types of item representation by looking at results from both Table 1 and Table 2 together. One can see that *wfb* attained, in all configurations, a much higher level of precision, achieving, when optimizing for % of explanations with size 2–4, around eight times more relevant suggestions than *wnb*. When optimized for precision and % of explanations of size 2–4, *wfb* also presented greater coverage. On the other hand, *wnb* presented greater levels of novelty and % of explanations with size 2–4 in every configuration, with surprise being greater in two out of three configurations. As for diversity, they both present similar results in every configuration.

We cannot directly compare results from the previous section (Section 5) and this one, because they use different datasets. However, we can see that, on an average, *r-by-e* approaches perform in an almost similar fashion on both datasets. In particular, as said previously, greater similarity

among items leads to shorter chains which ultimately lowers the overall diversity and serendipity of the recommendations. We will look at this in detail below.

We now select one of the two representations for further study. As before, to pick one, we assume that optimizing for the percentage of explanations of size 2–4 is best, since it generally gives explanations that are not so long as to be uninterpretable. In this setting, *wfb* performs better than *wnb*, and so this is the version that we will explore further.

We will study the effect of hyperparameters ϵ and α on the performance of *wfb*. Figure 6 shows six sub-figures — one for each evaluation measure. It can be seen that in, all the plots, for $\epsilon \in \{0.06, 0.09\}$, values of the evaluation measures remain almost constant for values of α in the range of $[0.02 - 0.09]$. Only for $\epsilon = 0.03$ is there some variation in the evaluation measures but this variation occurs only for the initial and last values of α; the measures remain largely constant in between. This shows the effect of high similarity among items in this experiment.

We look in detail at the results for each evaluation measure individually. Again, we refer to Eq. 7 and, as before, for conciseness, we refer to its first and second terms as the *overlap term* and the *profile term* respectively.

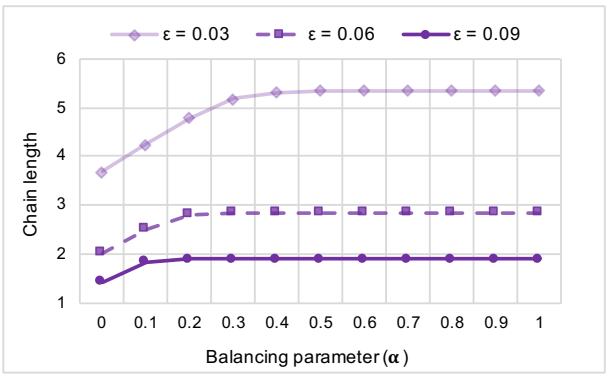
Chain length: In Figure 6(a), we see that the length of top-n chains increases up to $\alpha = 0.4$; then, increasing α does not show a noticeable effect on the length. This indicates that for lower α, the overlap term dominates, and the system selects shorter chains.For higher values of ϵ, the system imposes a stricter constraint on chains so their average length reduces substantially.

Precision: In Figure 6(b), for $\epsilon = 0.03$, we see that precision varies in three different ways when increasing the value of α: i) up to 0.5, it decreases; ii) from 0.5 to 0.9, it remains unchanged; and iii) at 1.0, it decreases again. As we mentioned before, in explanation chains, precision is proportional to the candidate's coverage. In particular, for this experiment, in the case of (i), items are very similar to each other and adding more members to the chain (i.e. increasing chain length) does not necessarily increase the candidate's coverage; for (ii), the system selects almost similar chains; and for (iii), the system totally ignores the overlap term, selects chains based only on the profile term and these chains do not try to cover the candidate, only the profile; therefore, precision decreases. For $\epsilon \in \{0.06, 0.09\}$, where there is a stricter constraint, precision becomes constant even earlier. We show the relationship between the precision and the candidate's coverage for different values of α in Figure 7(a).

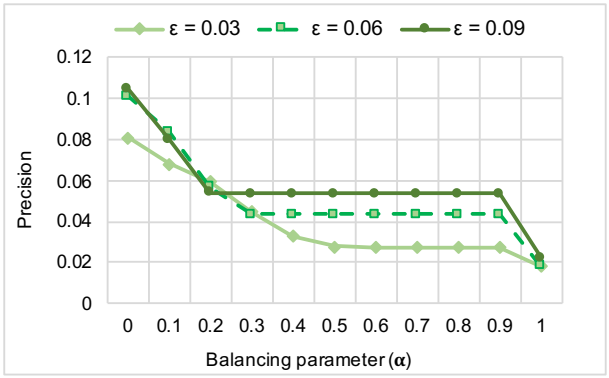
Diversity: In Figure 6(c), we see diversity remains almost unchanged up to $\alpha = 0.9$ and decreases at $\alpha = 1.0$ for $\epsilon = 0.03$ (and increases for $\epsilon \in \{0.06, 0.09\}$). In our system, as we have shown in the previous experiment, the diversity of the top-n recommendations depends upon their uniqueness: the lower the overlap among the members of top-n chains, the higher is the diversity. However, in this experiment, items are quite similar to each other so varying α does not affect the uniqueness of the top-n chains except at $\alpha = 0.0$ when the profile term is totally ignored. We show in Figure 7(b) that in all but one case uniqueness of chain members is negatively correlated with diversity.

Surprise: Figure 6(d) shows that for $\epsilon = 0.03$, (i) surprise increases up to $\alpha = 0.4$; after that (ii) it remains almost unchanged up to $\alpha = 0.9$; and (iii) it increases again at $\alpha = 1.0$. In the case of (i), the system initially selects shorter chains that easily cover the candidate's features thus giving low surprise, but, as α increases, the system selects those candidates that need more chain members (i.e. longer chains) to be covered. We show the relationship between the chain length and the surprise in Figure 7(c). It shows negative correlation at the extremes of α because, for lower values of α, variation in the values of surprise is much lower than the chain length, while the reverse applies at $\alpha = 1.0$. We also see in Figure 7(d) that precision and surprise behave almost the reverse of each other. This is confirmed by the Pearson correlation values in the Figure that are all negative.

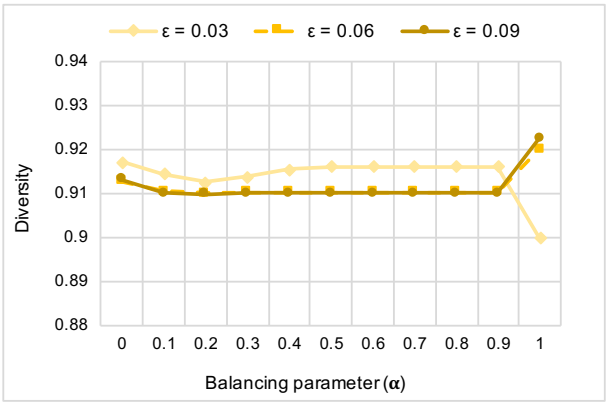
Novelty: Figure 6(e) shows that novelty increases up to $\alpha = 0.4$, then remains almost at the same level. It shows that on lower values of α, the overlap term dominates, enabling the system to



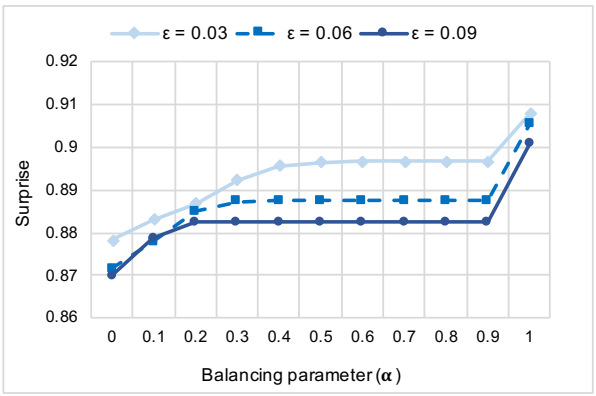
(a) Chain length



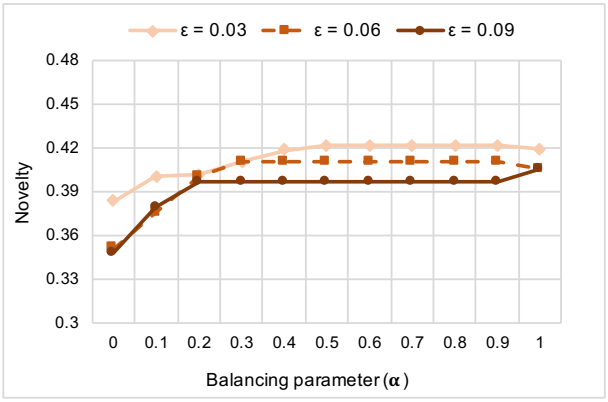
(b) Precision



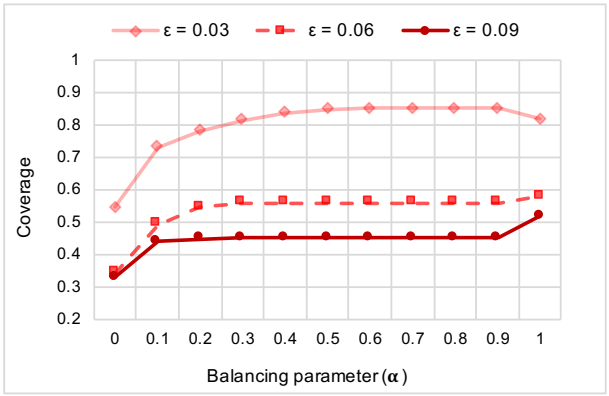
(c) Diversity



(d) Surprise



(e) Novelty

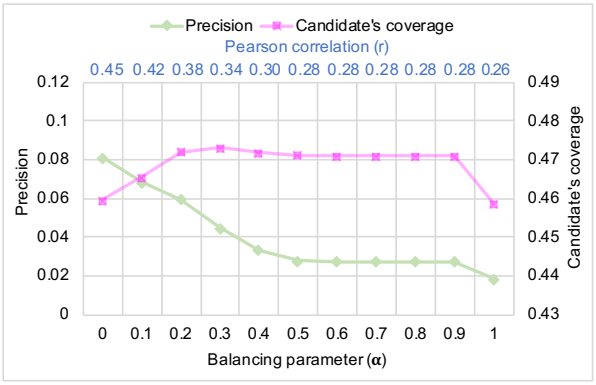


(f) Coverage

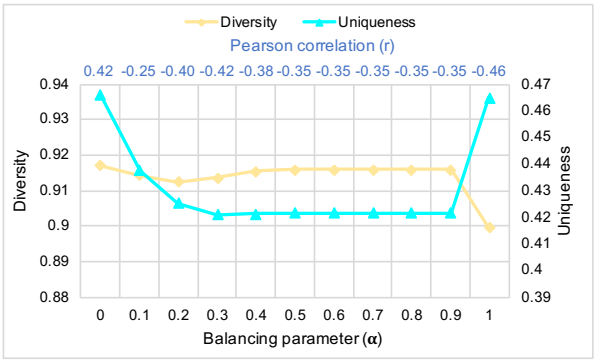
Fig. 6. Results for *wfb* with $\theta = 0.06$, $\epsilon \in \{0.03, 0.06, 0.09\}$, and α ranges in $[0.0 - 1.0]$ for extended chains on sentiments.

recommend mostly popular items which can be easily covered by shorter chains; on increasing α, the profile term dominates and the system suggests novel items that cannot be easily covered. On $\epsilon \in \{0.06, 0.09\}$, novelty almost increases with the increase in chain length.

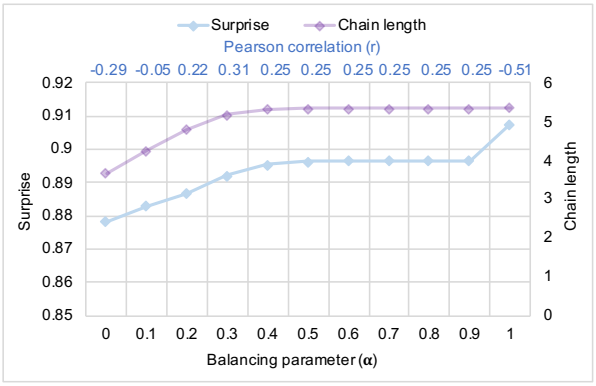
Coverage: Figure 6(f) shows that coverage increases up to $\alpha = 0.4$, then remains almost unchanged. In this experiment, this indicates that up to $\alpha = 0.4$, coverage increases with the increase in the



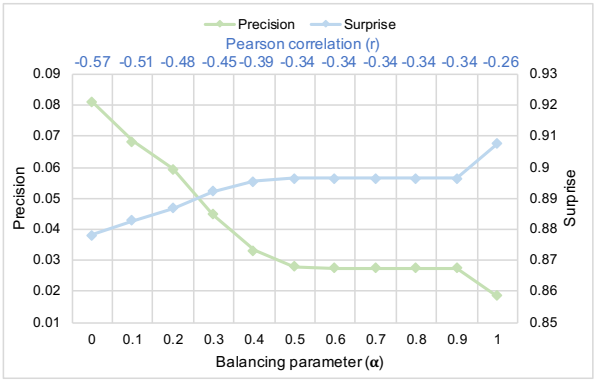
(a) Precision vs. Candidate's coverage



(b) Diversity vs. Uniqueness



(c) Surprise vs. Chain length



(d) Precision vs. Surprise

Fig. 7. Relationships between precision, diversity and surprise with candidate coverage, uniqueness, chain length and surprise at $\theta = 0.06$ and $\epsilon = 0.03$. Above each sub-figure, we show Pearson correlation. All of the correlation coefficient values are statistically significant (i.e. $p < 0.05$).

chain length; afterwards, it remains nearly at the same level as the chain length. On higher values of ϵ, the system imposes a stricter constraint that lowers the coverage.

6.3 User trials

We updated our web-based system [35, 36] in order to conduct a new user trial. In this one, we compare *wfb* with *wfb-CB-|C|*, with both using sentiment-aware concepts as features. We use the hyperparameter values (θ, ϵ) and α that optimized the percentage of explanations of size 2–4, obtained from our offline experiments.

In total, 144 people attempted the trial. The majority of them were undergraduate and postgraduate students recruited online from universities in Ireland, Brazil, and India.

We assigned half the participants to the recommendation trial and the other half to the explanation trial. Of the 144, only 100 completed all parts of the trial to which they were assigned, where 55 performed the recommendation trial, and 45 performed the explanation trial.

6.3.1 *Recommendation trial.*

Experiment settings. The recommendation trial is defined as the following.

We asked users to evaluate two different lists of recommendations produced by the two recommenders we were comparing. These lists of recommendations have length 5 and are sorted in decreasing order of recommender scores.

Table 5. Results of the Recommendation Trial for Extended *r-by-e* on Sentiments.

User's opinion	Diversity	Serendipity	Satisfaction
Much more *r-by-e*	11	8	20
More *r-by-e*	17	16	14
About the same	9	12	5
More *CB*-$\|C\|$	11	12	7
Much more *CB*-$\|C\|$	7	7	9

Before displaying the recommendations, we ensured that the two lists contained different movies. Each movie that was common to both lists was removed and the next best recommendations from the top-10 were added to the end of the lists. If it was not possible to create two different lists of length 5 from the top-10 recommendations, the user's responses to the survey were discarded. We did this to avoid skewing responses about the diversity of recommendations: shorter lists are less likely to be diverse. In our experiments, there were only two users whose responses were discarded for this reason.

For half the users, the list on the left ('List A') came from *r-by-e* and the list on the right ('List B') from *CB*-$|C|$; for the other half of the users, List A was from *CB*-$|C|$ and List B from *r-by-e*. Users were not aware of which list belonged to which recommender.

Participants were required to answer three questions on Diversity, Serendipity and Satisfaction.

- Diversity: Which list has a greater variety of movies?
- Serendipity: Which list has more pleasantly surprising recommendations?
- Satisfaction: Which list has more recommendations that you would be likely to try?

Their answers were on a 5-point: Much more List A than List B; More List A than List B; About the Same; More List B than List A; and Much more List B than List A.

Experiment results. Fifty-five participants completed this trial. Table 5 summarizes their responses.

- *Diversity question:* 50.9% of participants found *r-by-e* recommendations to be much more or more diverse than *CB*-$|C|$ recommendations, 16.4% found the recommendation lists to be equally diverse, leaving 32.7% finding *CB*-$|C|$ to be much more or more diverse.
- *Serendipity question:* 43.7% of participants found *r-by-e* recommendations to be much more or more pleasantly surprising, 21.8% found the recommendation lists to be equally surprising, leaving 34.5% finding *CB*-$|C|$ to be much more or more surprising.
- *Satisfaction question:* 61.8% of participants found *r-by-e* recommendations to be ones they would be much more or more likely to try, 9.1% found the recommendations to be equally worthy of trying, leaving 29.1% finding *CB*-$|C|$ to be much more or more worth trying.

On all criteria *r-by-e* produced the better recommendation lists. However, only in the case of the satisfaction question was this statistically significant. (We used two-tailed proportion tests with significance level $p_0 = 0.05$. The null hypothesis was that those preferring *r-by-e* was equal to those preferring *CB*-$|C|$, i.e. ignoring those who thought the two lists were about the same.) This is also in line with the results of the offline experiments where the weighted feature-based approach attains better precision than the other approaches, while remaining competitive on measures of diversity and serendipity. Since precision is directly related to user satisfaction, i.e. the former directly evaluates the recommender capability of generating relevant suggestions, *r-by-e* is able to provide in both offline and online settings more relevant (and thus, more satisfactory) suggestions than its baseline competitor.

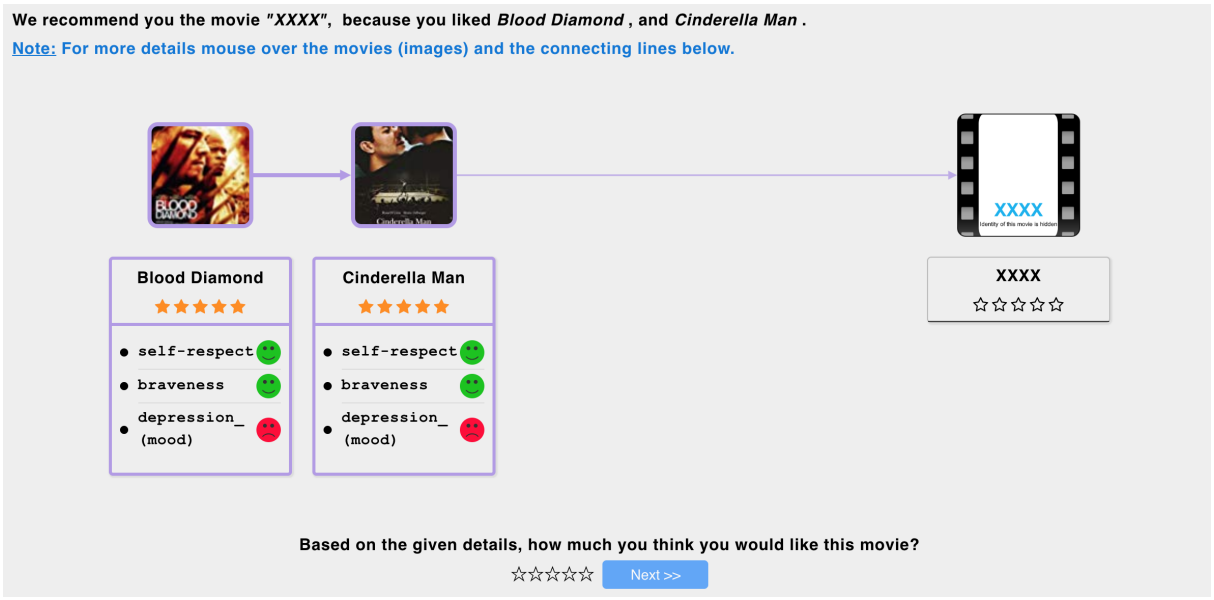


Fig. 8. A screenshot of an explanation chain. The user has moused over the arrow that connects the first two movies, which causes the system to bring up boxes of sentiments that these two movies have in common.

6.3.2 Explanation trial. Users who were directed to this trial participated in a re-rating task. Re-rating tasks are an established method of evaluating explanation quality when the goal of the explanation is effectiveness: helping users make better decisions [4, 15]. The users are initially asked to rate a recommendation in the case where they are given only the explanation and not the identity of the movie. This is called the *explanation-rating*. The users are asked later to re-rate the recommended item in the case where they are given information about the item, including its identity. This is called the *actual-rating*. An effective explanation is one where the explanation-rating is close to the actual-rating. Effective explanations will be ones for which (a) μ_d (the mean difference between explanation-ratings and corresponding actual-ratings) is close to zero; (b) σ_d (their standard deviation) is small; and (c) r (their Pearson correlation) is highest.

Experiment settings. Explanation Chains were displayed in the fashion shown in Figures 8 and 9: arrows connect a movie to its successor in the chain. While *CB*-|*C*|'s explanations (sets of neighbours, rather than chains) were displayed in the fashion shown in Figure 10: arrows connect each movie to the recommended movie. In both cases, users can mouse over parts of the explanation, which causes the system to display features that movies have in common. A maximum of three features is displayed in any box. These features are selected by their sentiment scores, i.e. if they share the same polarity. Each is also associated with an emoji indicating the sentiment of the feature. We used different colors for different sentiments: positive (green smiley face), negative (red frowny face), and neutral (yellow neutral face).

We asked the user to supply an explanation-rating (1-5 stars): how much they thought they might like the movie based only on the explanation. We do it for both *r-by-e* and *CB*-|*C*| explanations. After the users have given these $2n$ ratings, the system then shows them in a random order each of the n recommended movies again. This time, the identity of the movie is not redacted but no explanation is shown. Instead, we show genre, plot synopsis, main cast members, directors, writers, duration, and release date. Again we ask them for ratings to indicate how much they think they will like the movies. Note that, although users have rated the same movie three times, nothing in the on-screen instructions makes this apparent.

Fig. 9. A screenshot of an explanation chain. The user has moused over the icon for the first movie, which causes the system to display an arrow between that movie and the recommended movie and to bring up boxes of sentiments that these two movies have in common.

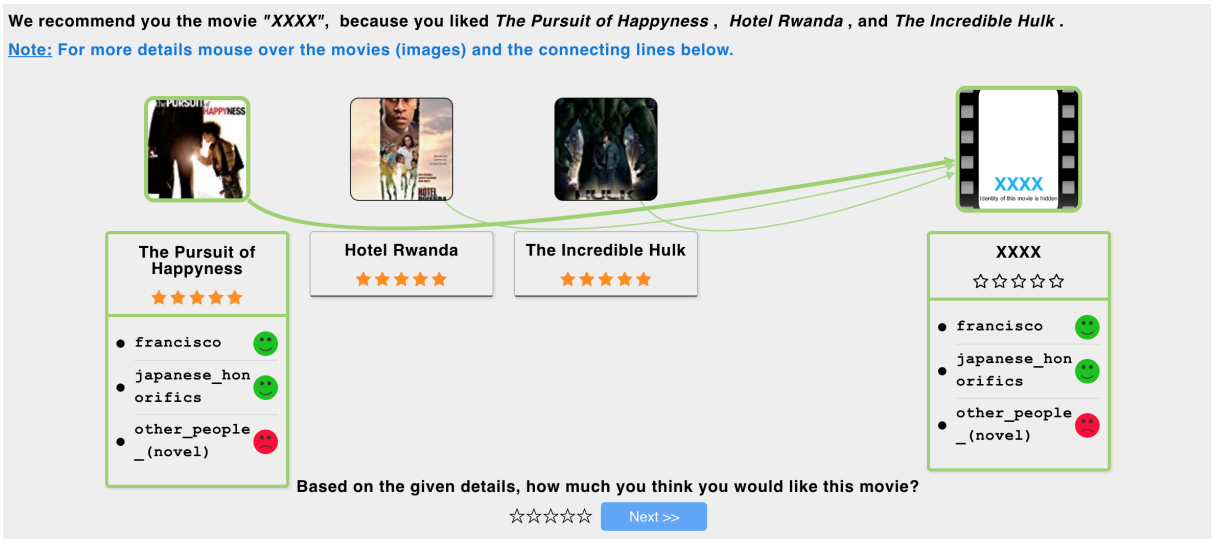


Fig. 10. A screenshot of a *CB*-|*C*| explanation. The user has moused over the icon for the first movie, which causes the system to increase the width of the arrow between that movie and the recommended movie and to bring up boxes of sentiments that these two movies have in common.

Experiment results. Forty-five participants completed this trial: it is quite onerous and more participants abandoned it partway through than did for the other trial. In total, we obtained 675 ratings, this being three ratings for 225 recommended movies. Figure 11 shows the distribution of the users' ratings; Table 6 gives summary statistics.

We can see that users mostly think they will like the movies that the system recommends, both when they see explanations only and when they see movie identities. For the differences between

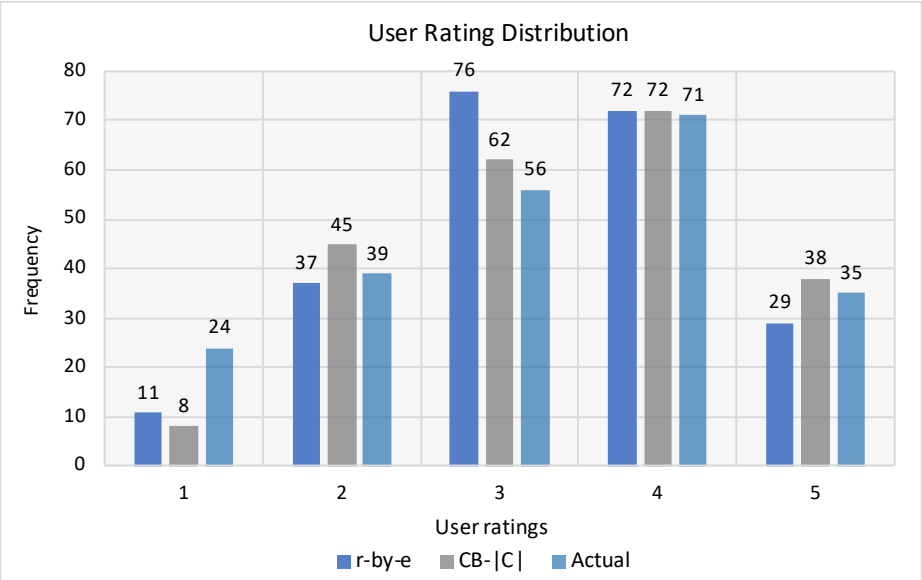


Fig. 11. Ratings from the Explanation Trial for Extended chains on Sentiments.

Table 6. Mean (μ), standard deviation (σ) and Pearson correlation (r) of ratings from the Explanation Trial for Extended *r-by-e* on Sentiments.

Rating type	μ	σ	r
Actual	3.2400	1.2193	–
r-by-e	3.3156	1.0492	0.5338
CB-|C|	3.3867	1.0925	0.1613

Table 7. Differences in ratings from the Explanation Trial for Extended *r-by-e* on Sentiments.

Explanation type	μ_d	σ_d	95% Conf. Int.
r-by-e	0.0756	1.1054	(-0.0688, 0.2199)
CB-|C|	0.1467	1.5002	(-0.0494, 0.3427)

explanation-ratings and actual-ratings, Figure 12 shows the distribution of values and Table 7 gives summary statistics.

The mean difference between *r-by-e* ratings and actual ratings is 0.0756; for *CB-|C|*, it is 0.1467. Hence, both kinds of explanations cause users to overestimate their actual-ratings. Using a two-tailed paired t-test ($p_0 = 0.05$), we observed that in this study, i) the difference between *r-by-e*-ratings and actual-ratings are not statistically different; ii) the differences between *CB-|C|*ratings and actual-ratings are also not statistically significant; and iii) *r-by-e*-ratings and *CB-|C|*-ratings are not statistically different. In terms of μ_d and σ_d, then, neither kind of explanation is better than the other. But there is still the question of correlation with the actual-ratings.

Table 6 shows r, the Pearson correlation between explanation-ratings and actual-ratings. We see that *r-by-e*-ratings are better correlated with actual-ratings. We calculated the probability of getting this correlation due to chance to be 0 in both cases.

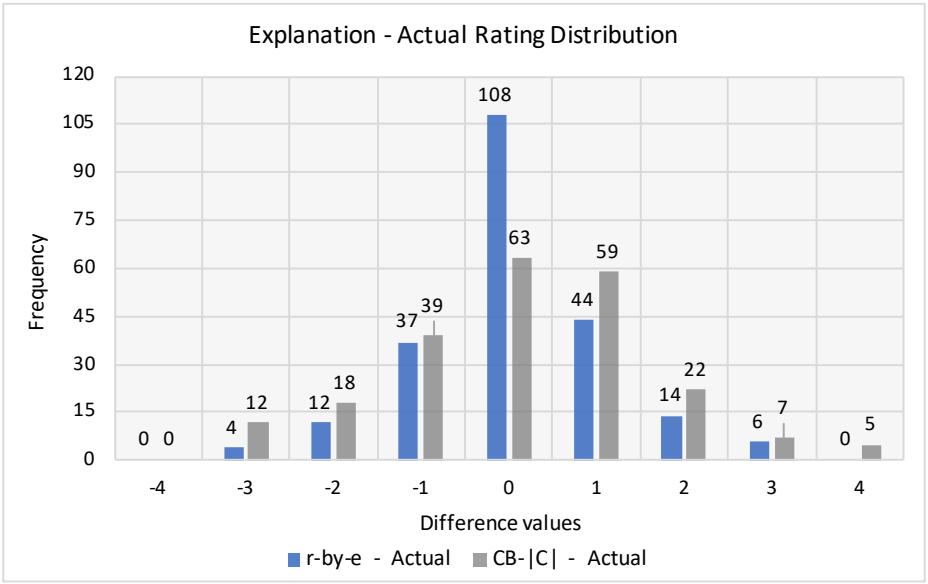


Fig. 12. Differences in ratings from the Explanation Trial for Extended *r-by-e* on Sentiments.

This is evidence that the relationships between items depicted in *r-by-e*'s explanation chains are capable of providing a more thorough explanation to users, thus helping them make accurate and informed decisions towards items. The use of sentiment-based features in the construction of the chains also adds another layer of information: users are able to perceive not only the features shared by the movies they like, as well as the reasoning behind the chain, but also whether those features are perceived as positive or negative aspects of the films. All of this semantic information, coupled with the chain itself, provides sufficient support to users to make appropriate decisions towards recommended movies.

7 CONCLUSION

In this paper, we considered various extensions to *r-by-e*. We presented two item representations: i) feature-based; and ii) neighbour-based. The former represents an item as a set of its features, and the latter makes no explicit reference to features but represents an item as a set of its neighbours instead. For each of these representations, we also explored weighting schemes to assign weights to the features (or neighbours). We formulated a normalized form of both an unweighted and weighted overlap for the two representations and thus defined four versions of *r-by-e*'s *chain generation*. We also generalized *r-by-e*'s *chain selection* by redefining the scoring function as a linear combination of the average candidate overlap and the average profile overlap balanced by a parameter α.

We performed extensive experiments to evaluate these extensions to *r-by-e*. An offline experiment on a dataset where items are described by keywords shows that the *weighted feature-based (wfb)* version gives more relevant recommendations than all other versions of *r-by-e* and the baselines and ones that are also competitive on measures of diversity and serendipity. We also found an interesting relationship between the chain length and the surprise: the higher the chain length, the more surprising is the recommendation. We ran another offline experiment, this time on a dataset where items are described by features weighted by sentiment scores. We refer to the chains that *r-by-e* generates on this kind of dataset as *sentiment-aware explanation chains*. This offline experiment confirms that *wfb* outperforms the other system, however, because of greater similarities among

the items in the dataset, *wfb* provides recommendations with a similar level of relevance but lower levels of diversity and surprise than it did when the dataset used keywords.

We also conducted user trials in the case of the sentiment-aware explanation chains to evaluate the quality of recommendations and the effectiveness of the corresponding explanations. The Recommendation Trial shows that *r-by-e* produces recommendations that are more diverse and serendipitous than those of a baseline content-based recommender (although this is not statistically significant) and with statistically significantly higher levels of user satisfaction. User responses in the Explanation Trial confirmed that the sentiment-aware explanation chains allow users to make more accurate judgements about the quality of the recommended items than do the baseline's explanations.

ACKNOWLEDGMENTS

We thank Arpit Jain for help with web design. This paper emanates from research supported by a grant from Science Foundation Ireland (SFI) under Grant Number SFI/12/RC/2289 which is co-funded under the European Regional Development Fund.

REFERENCES

[1] Behnoush Abdollahi and Olfa Nasraoui. 2016. Explainable Matrix Factorization for Collaborative Filtering. In *Proceedings of the 25th International Conference Companion on World Wide Web*. International World Wide Web Conferences Steering Committee, 5–6.

[2] Gediminas Adomavicius and YoungOk Kwon. 2008. Overcoming accuracy-diversity tradeoff in recommender systems: A variance-based approach. In *Proceedings of WITS*, Vol. 8. Citeseer.

[3] Gediminas Adomavicius and Alexander Tuzhilin. 2005. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge & Data Engineering* 17, 6 (2005), 734–749.

[4] Mustafa Bilgic and Raymond J Mooney. 2005. Explaining recommendations: Satisfaction vs. promotion. In *Beyond Personalization Workshop, IUI*, Vol. 5. 153.

[5] Derek Bridge and Kevin Dunleavy. 2014. If you liked Herlocker et al.'s explanations paper, then you might like this paper too. In *Joint Workshop on Interfaces and Human Decision Making in Recommender Systems*. 22.

[6] Shuo Chang, F Maxwell Harper, and Loren Gilbert Terveen. 2016. Crowd-based personalized natural language explanations for recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems*. ACM, 175–182.

[7] Li Chen and Pearl Pu. 2005. Trust building in recommender agents. In *Proceedings of the Workshop on Web Personalization, Recommender Systems and Intelligent User Interfaces at the 2nd International Conference on E-Business and Telecommunication Networks*. Citeseer, 135–145.

[8] Sergio Cleger, Juan M Fernández-Luna, and Juan F Huete. 2014. Learning from explanations in recommender systems. *Information Sciences* 287 (2014), 90–108.

[9] Sergio Cleger-Tamayo, Juan M Fernandez-Luna, and Juan F Huete. 2012. Explaining neighborhood-based recommendations. In *Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval*. ACM, 1063–1064.

[10] Felipe Costa, Sixun Ouyang, Peter Dolog, and Aonghus Lawlor. 2018. Automatic Generation of Natural Language Explanations. In *Proceedings of the 23rd International Conference on Intelligent User Interfaces Companion*. ACM, 57.

[11] Rafael M D'Addio, Eduardo P Fressato, Arthur F Da Costa, and Marcelo G Manzato. 2018. Incorporating Semantic Item Representations to Soften the Cold Start Problem. In *Proceedings of the 24th Brazilian Symposium on Multimedia and the Web*. ACM, 157–164.

[12] Rafael M D'Addio, Ronnie S Marinho, and Marcelo G Manzato. 2019. Combining different metadata views for better recommendation accuracy. *Information Systems* 83 (2019), 1–12.

[13] Gerhard Friedrich and Markus Zanker. 2011. A taxonomy for generating explanations in recommender systems. *AI Magazine* 32, 3 (2011), 90–98.

[14] Fatih Gedikli, Mouzhi Ge, and Dietmar Jannach. 2011. Understanding recommendations by reading the clouds. In *International Conference on Electronic Commerce and Web Technologies*. Springer, 196–208.

[15] Fatih Gedikli, Dietmar Jannach, and Mouzhi Ge. 2014. How should I explain? A comparison of different explanation types for recommender systems. *International Journal of Human-Computer Studies* 72, 4 (2014), 367–382.

[16] Jonathan L Herlocker, Joseph A Konstan, and John Riedl. 2000. Explaining collaborative filtering recommendations. In *Proceedings of the 2000 ACM conference on Computer supported cooperative work*. ACM, 241–250.

[17] M Shahriar Hossain, Joseph Gresock, Yvette Edmonds, Richard Helm, Malcolm Potts, and Naren Ramakrishnan. 2012. Connecting the dots between PubMed abstracts. *PloS one* 7, 1 (2012), e29509.

[18] Marius Kaminskas and Derek Bridge. 2016. Diversity, Serendipity, Novelty, and Coverage: A Survey and Empirical Analysis of Beyond-Accuracy Objectives in Recommender Systems. *ACM Transactions on Interactive Intelligent Systems* 7, 1 (December 2016), 2:1–2:42.

[19] Joseph A Konstan and John Riedl. 2012. Recommender systems: from algorithms to user experience. *User modeling and user-adapted interaction* 22, 1-2 (2012), 101–123.

[20] Todd Kulesza, Simone Stumpf, Margaret Burnett, Sherry Yang, Irwin Kwan, and Weng-Keen Wong. 2013. Too much, too little, or just right? Ways explanations impact end users' mental models. In *Visual Languages and Human-Centric Computing (VL/HCC), 2013 IEEE Symposium on*. IEEE, 3–10.

[21] Greg Linden, Brent Smith, and Jeremy York. 2003. Amazon. com recommendations: Item-to-item collaborative filtering. *IEEE Internet computing* 7, 1 (2003), 76–80.

[22] Bing Liu and Lei Zhang. 2012. A Survey of Opinion Mining and Sentiment Analysis. In *Mining Text Data*, Charu C. Aggarwal and ChengXiang Zhai (Eds.). Springer US, 415–463. https://doi.org/10.1007/978-1-4614-3223-4_13

[23] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to Information Retrieval*. Cambridge University Press, USA.

[24] Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP Natural Language Processing Toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*. 55–60.

[25] Andrea Moro, Alessandro Raganato, and Roberto Navigli. 2014. Entity linking meets word sense disambiguation: a unified approach. *Transactions of the Association for Computational Linguistics* 2 (2014), 231–244.

[26] Khalil Muhammad, Aonghus Lawlor, Rachael Rafter, and Barry Smyth. 2015. Great explanations: Opinionated explanations for recommendations. In *International Conference on Case-Based Reasoning*. Springer, 244–258.

[27] Khalil Muhammad, Aonghus Lawlor, and Barry Smyth. 2016. On the Use of Opinionated Explanations to Rank and Justify Recommendations. In *Proceedings of the 28th FLAIRS Conference*. 554–559.

[28] Cataldo Musto, Fedelucio Narducci, Pasquale Lops, Marco De Gemmis, and Giovanni Semeraro. 2016. ExpLOD: A Framework for Explaining Recommendations Based on the Linked Open Data Cloud. In *Proceedings of the 10th ACM Conference on Recommender Systems*. 151–154.

[29] Cataldo Musto, Fedelucio Narducci, Pasquale Lops, Marco de Gemmis, and Giovanni Semeraro. 2019. Linked open data-based explanations for transparent recommender systems. *International Journal of Human-Computer Studies* 121 (2019), 93–107.

[30] Roberto Navigli and Simone Paolo Ponzetto. 2012. BabelNet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network. *Artificial Intelligence* 193 (2012), 217–250.

[31] Alexis Papadimitriou, Panagiotis Symeonidis, and Yannis Manolopoulos. 2012. A generalized taxonomy of explanations styles for traditional and social recommender systems. *Data Mining and Knowledge Discovery* 24, 3 (2012), 555–583.

[32] Michael J Pazzani and Daniel Billsus. 2007. Content-based recommendation systems. In *The adaptive web*. Springer, 325–341.

[33] Pearl Pu and Li Chen. 2007. Trust-inspiring explanation interfaces for recommender systems. *Knowledge-Based Systems* 20, 6 (2007), 542–556.

[34] Arpit Rana. 2020. *Chain-based recommendations*. Ph.D. Dissertation. Insight Centre for Data Analytics, School of Computer Science & Information Technology, University College Cork, Ireland.

[35] Arpit Rana and Derek Bridge. 2017. Explanation Chains: Recommendations by Explanation. In *Proceedings of the Poster Track of the 11th ACM Conference on Recommender Systems*, Domonkos Tikk and Pearl Pu (Eds.). CEUR Workshop Proceedings, vol-1905.

[36] Arpit Rana and Derek Bridge. 2018. Explanations that are Intrinsic to Recommendations. In *Proceedings of the 26th Conference on User Modeling, Adaptation and Personalization*. ACM, 187–195.

[37] Marco Túlio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. "Why Should I Trust You?": Explaining the Predictions of Any Classifier. *CoRR* abs/1602.04938 (2016).

[38] Marco Rossetti, Fabio Stella, and Markus Zanker. 2013. Towards explaining latent factors with topic models in collaborative recommender systems. In *Database and Expert Systems Applications (DEXA), 2013 24th International Workshop on*. IEEE, 162–167.

[39] Masahiro Sato, Budrul Ahsan, Koki Nagatani, Takashi Sonoda, Qian Zhang, and Tomoko Ohkuma. 2018. Explaining Recommendations Using Contexts. In *23rd International Conference on Intelligent User Interfaces*. ACM, 659–664.

[40] Christian Scheel, Angel Castellanos, Thebin Lee, and Ernesto William De Luca. 2012. The reason why: A survey of explanations for recommender systems. In *International Workshop on Adaptive Multimedia Retrieval*. Springer, 67–84.

[41] Rashmi Sinha and Kirsten Swearingen. 2002. The role of transparency in recommender systems. In *CHI'02 extended abstracts on Human factors in computing systems*. ACM, 830–831.

[42] Nava Tintarev. 2007. Explanations of recommendations. In *Proceedings of the 1st ACM conference on Recommender systems*. ACM, 203–206.

[43] Nava Tintarev and Judith Masthoff. 2007. Effective explanations of recommendations: user-centered design. In *Proceedings of the 1st ACM conference on Recommender systems*. ACM, 153–156.

[44] Nava Tintarev and Judith Masthoff. 2007. A survey of explanations in recommender systems. In *Data Engineering Workshop, 2007 IEEE 23rd International Conference on*. IEEE, 801–810.

[45] Nava Tintarev and Judith Masthoff. 2008. The effectiveness of personalized movie explanations: An experiment using commercial meta-data. In *International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems*. Springer, 204–213.

[46] Nava Tintarev and Judith Masthoff. 2015. Explaining recommendations: Design and evaluation. In *Recommender systems handbook*. Springer, 353–382.

[47] Jesse Vig, Shilad Sen, and John Riedl. 2009. Tagsplanations: explaining recommendations using tags. In *Proceedings of the 14th international conference on Intelligent user interfaces*. ACM, 47–56.

[48] Cong Yu, Laks VS Lakshmanan, and Sihem Amer-Yahia. 2009. Recommendation diversification using explanations. In *2009 IEEE 25th International Conference on Data Engineering*. IEEE, 1299–1302.

[49] Yongfeng Zhang and Xu Chen. 2018. Explainable recommendation: A survey and new perspectives. *arXiv preprint arXiv:1804.11192* (2018).

6.2 Final Remarks

This paper presents extensions to *r-by-e*'s original formulation (RANA; BRIDGE, 2018), as well as the results obtained in applying one of the techniques developed in this research into the *r-by-e*'s framework. This was a year-long collaboration due to the time required to process a review dataset with BabelFy (MORO; RAGANATO; NAVIGLI, 2014), the large amount of offline experiment configurations, the reformulation of an online plataform to reflect the new, semantic item information and, finally, the execution of the user trial. The main results of this collaboration can be seen especially in the article's Section 6.

Even though the experiments were designed to evaluate *r-by-e*'s performance and confirm its capabilities over classical content-based recommenders instead of assessing the item representations' descriptive power, some conclusions can be drawn towards the scope of this research. One clear aspect is that results reveal that *r-by-e* benefits more from using the sentiment-based item representation than classic content-based recommender, as both of them used these representations. This directly impacts Research Question 3, i.e. "Is a kind of item representation more relevant to use in determined recommender system or scenario than another?". The offline experiments show that *r-by-e* provide suggestions that are more relevant and that cover more of the user's interest than content-based recommenders, at the expanse of Diversity, Surprise and Novelty. The Recommendation Trial of the online experiments also confirms that *r-by-e*'s suggestions are more relevant than the baseline. As for the Explanation Trial of the online experiment, results show that *r-by-e* could make better use of the sentiment-based representations to provide explanations that allow users to make more accurate judgements towards their suggestions. All this indicates that *r-by-e* is able to use the features and their relationships of this representation better than content-based baselines.

Finally, even thought this user trial cannot be directly comparable to the one performed previously in (RANA; BRIDGE, 2018) (where they used the same item set with keyword-based representations, but different users), if we look at the results of both Explanation Trials, we can see that the sentiment-based representation, on *average*, produces more closely related explanation and actual ratings, with a greater correlation among them. While this is an indicative that sentiment-based representations do help *r-by-e* construct more descriptive explanations than keyword-based representations, more experiments are required to support this claim.

We leave for future endeavors a better comparison between those item representations in *r-by-e*'s framework, as well as other possible extensions to its formulae and the application of the other item representations we have built throughout this research.

CHAPTER

7

CONCLUSION AND FINAL REMARKS

In this doctorate research we explored alternatives to create rich and semantic item representations that can be used on content-based recommender algorithms. We have produced five concept-based representations that minimize the well-known linguistic problem of synonymy and polysemy. All those representations were built as vector space models, where in four of them each dimension corresponds to a concept or named entity, and their weighting scheme varies representing different semantics, such as their statistics, sentiments or relationships towards items. The remaining representation was built using a latent space which represents concept and item embeddings.

Beyond that, we have tested those representations in several models, domains and recommendation scenarios, and presented here the most significant experiments which resulted in publications. We have tested most of our representations in three different datasets, two based on movies and one based on phone/tablet apps. We have applied them mostly on neighborhood-based approaches, but also performed experiments on clustering and factorization models, with experiments directed towards rating prediction and top-N recommendation tasks. We explored their impact in a simulated cold-start scenario, revealing their strength in recommending items that have no previous user interactions but are nonetheless relevant. We have also explored combination techniques in order to boost even more the semantics conveyed in those representations.

Finally, we also tested one of our approaches in an explanation-based algorithm, and carried a user trial designed to provide explanations aided by our features and their corresponding sentiments. The experiments were carried with the goal of understanding whether our representations were able to aid users to make better judgements towards what to consume.

7.1 Research Findings

The main motivation for our work was the premise that as we aggregate more semantics to item representations, the system is able to discern them better, leading to better analysing user interests and, finally, producing better recommendations. This could be done especially when considering user opinionated texts, i.e. reviews, since they provide insights about items' features as well as their quality (e.g. whether they are good or bad).

To test that premise, we devised a Research Question and a Hypothesis, which needed to be verified with our experiments. The Research Question was *"How can we effectively use user-provided unstructured information such as user reviews to better describe items?"*. Our hypothesis was that using state-of-the-art NLP and IR methods to extract meaningful information from user reviews, ultimately devising rich and semantic item representations, would benefit recommender systems. This was proven throughout the experiments reported in this thesis, with our recommendations presenting better results than baselines in several recommendation tasks and domains. Even though not every experiment was successful in the entirety of this doctorate research, we have sufficient evidence that carefully crafted representations may aid recommender systems in selecting more relevant suggestions.

To reinforce our claims, we devised two more Research Questions that could be more effectively addressed in our experiments. The first question was *"Does increasing the semantics of item representations improve recommendation accuracy?"*. The hypothesis was that recommenders need to have means to differentiate their items, and thus increasing the semantics of representations would aid recommenders in this task, ultimately helping them make better predictions. We analyzed this assumption specifically with the experiments depicted in Chapters 2, 3, 4 and 5. Chapter 2 showed that, in the majority of the cases, the recommenders performed better when using a more semantic representation, i.e., based on user reviews rather than structured metadata. Chapter 3 raised more evidence on the matter by showing that a concept-based approach produced better results than our previous term-based approaches. In this experiment, it was clear that minimizing the synonymy and polysemy benefited the descriptive power of item representations. Chapter 4 introduced four types of representations and, even though some of them performed with similar results, one can see that there was an increase in recommendation accuracy when observing the increase in semantics on the embedding-based representations. Finally, Chapter 5 introduced the idea of increasing semantics by means of combining representations. In that work, sentiment and TF-IDF representations were combined due to the fact that they complement each other: the former depicts the quality aspect of a feature, while the later represents its relevance towards the item and the collection.

The last Research Question was *"Is a kind of item representation more relevant to use in determined recommender system or scenario than another?"*. We hypothesized that different semantics may benefit different recommendation settings, making it necessary to understand specificities of each scenario that a content-based recommender is being applied on. This question

was explored on Chapters 2, 4 and 6. On Chapter 2, results showed that cluster-based approaches may benefit from shorter, less sparse representations, while neighborhood approaches performed better on larger representations. Chapter 4 revealed that, in the case of concept-based item representations, neighborhood approaches perform similarly, leaving the decision to which representation to use based on other aspects, such as computational power and time constraints. Nevertheless, some representations presented better results than others when applied to the other algorithm of that study, giving evidence that some recommenders are highly dependant of item representation quality. Finally, in Chapter 6, we have shown by offline and online experiments that one recommender system may benefit more from our sentiment-based representations than other. Both *r-by-e* and classic content-based recommenders use the same representations, but *r-by-e* makes better use of the information conveyed there both on its formulation as well as on the design of its explanations. Even though those are sufficient evidence to support our claim that different algorithms, methodologies and domains require different item representations, it would require much more experimentation to correctly define which representation should be used in which setting.

7.2 Limitations and Future Work

This work presented interesting results and findings, but there are also limitations to its execution. They are presented in topics in the following:

- Dealing with content, especially unstructured information which requires large quantities of data, can be time and resource consuming. Some NLP tools, such as Babelfy (MORO; RAGANATO; NAVIGLI, 2014), are not open source, thus requiring access to their servers. This creates a limitation directly related to their availability, which, in the case of Babelfy, is defined as query access called *babelcoin*, which have a daily limit.

- Even though this research ascertained that its item representations may benefit different recommendation settings and presented evidence for some of them, many more settings need to be evaluated. Such endeavour needs massive experimentation, which cannot be performed in the course of a single doctorate research.

- Refining the semantics of item representations may lead to a decrease in recommendation diversity, especially if we are considering content-based recommendation approaches. Since the suggestions are going to be based on similarities found between the *items' features*, it is difficult for the recommender to suggest items that are different from the user profile.

We leave as future work the following endeavours:

- This work mainly focused on defining item representations. The next natural step would be defining user preference directly towards those same features. Context-based recommendation often uses the notion of tensors, which are three-dimension constructs that depicts user and item relation towards some context. The same notion could be applied here, where we could have a user-item-feature triple, showing a user's opinion towards an item's feature. Of course, user reviews would need to be devised by the user itself, which may lead to limited content analysis in some domains and/or applications.

- Concept and item embeddings, as well as similarity-based representations, could be used to aid *r-by-e*'s formulae, since the explanations are largely based on relationships between items and features. The latent space in which concept and items are inserted could be explored to strengthen the relationships between those two entities, producing a notion of distance/proximity between the elements of the chains, further increasing their visualization and explanation power to the end user.

- Each and every item representation introduced here could aid conversational recommender systems. Sentiment-based approaches would be specifically powerful, in the sense that the user could refine his/her queries based on the sentiment strength he/she is looking for in the features of an item. Moreover, the concepts extracted in our approach can provide an array of features that the user may or may not regard in their query, and suggestions of features could be captured by the similarity measures between concepts and items through their embeddings.

7.3 Additional Publications

Beyond the articles described in the previous chapters of this thesis, other publications were prepared during the execution of this doctoral research. Even though most of them relate to this research's theme, they do not fall into its scope: they do not make use of the item representations developed during the research. The 3 conference papers and 2 journal articles in which we collaborated are listed below in chronological order:

- DOS SANTOS, E. B.; DA COSTA, A. F.; D'ADDIO, R. M.; MANZATO, M. G.; GOULARTE, R.. Introducing the concept of "always-welcome recommendations". In: 2015 IEEE/ACIS 14th International Conference on Computer and Information Science (ICIS), 2015, Las Vegas. p. 197–202. - *Conference full paper.*

- DA COSTA, A. F.; D'ADDIO, R. M.; MANZATO, M. G.; CAMPELLO, R. J. G. B.. Exploiting different users' interactions for profiles enrichment in recommender systems. In: Proceedings of the 31st Annual ACM Symposium on Applied Computing - SAC '16, 2016, Piza. p. 1080–1082. - *Conference short paper.*

- MANZATO, M. G.; DOMINGUES, M. A.; COSTA, A. F.; SUNDERMANN, C. V.; D'ADDIO, R. M.; CONRADO, M. S.; REZENDE, S. O.; PIMENTEL, M. G. C.. Mining unstructured content for recommender systems: an ensemble approach. Information Retrieval (Dordrecht. Online), v. 19, p. 378–415, 2016. - *Journal article.*

- D'ADDIO, R. M.; DOMINGUES, M. A.; MANZATO, M. G.. Exploiting feature extraction techniques on users' reviews for movies recommendation. Journal of the Brazilian Computer Society (Online), v. 23, p. 7, 2017. - *Journal article.*

- DA COSTA, A. F.; D'ADDIO, R. M.; FRESSATO, E. P.; MANZATO, M. G.. A personalized clustering-based approach using open linked data for search space reduction in recommender systems. In: Proceedings of the 25th Brazillian Symposium on Multimedia and the Web - WebMedia '19, 2019, Rio de Janeiro. p. 409–416 - *Conference full paper. Awarded with the conference's Best Paper Award.*

BIBLIOGRAPHY

ADOMAVICIUS, G.; TUZHILIN, A. Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions. **IEEE Transactions on Knowledge and Data Engineering**, v. 17, n. 6, p. 734–749, 2005. Citation on page 30.

AGGARWAL, C. C. An introduction to recommender systems. In: ____. **Recommender Systems: The Textbook**. Cham: Springer International Publishing, 2016. p. 1–28. ISBN 978-3-319-29659-3. Available: <https://doi.org/10.1007/978-3-319-29659-3_1>. Citations on pages 29, 37, and 53.

____. Knowledge-based recommender systems. In: ____. **Recommender Systems: The Textbook**. Cham: Springer International Publishing, 2016. p. 167–197. ISBN 978-3-319-29659-3. Available: <https://doi.org/10.1007/978-3-319-29659-3_5>. Citations on pages 30 and 31.

BOBADILLA, J.; ORTEGA, F.; HERNANDO, A.; GUTIÉRREZ, A. Recommender systems survey. **Knowledge-Based Systems**, v. 46, n. 0, p. 109–132, 2013. Citation on page 29.

CAMACHO-COLLADOS, J.; PILEHVAR, M. T.; NAVIGLI, R. Nasari: Integrating explicit knowledge and corpus statistics for a multilingual representation of concepts and entities. **Artificial Intelligence**, v. 240, p. 36 – 64, 2016. ISSN 0004-3702. Available: <http://www.sciencedirect.com/science/article/pii/S0004370216300820>. Citation on page 53.

CAPELLE, M.; FRASINCAR, F.; MOERLAND, M.; HOGENBOOM, F. Semantics-based news recommendation. In: **Proceedings of the 2nd International Conference on Web Intelligence, Mining and Semantics**. New York, NY, USA: Association for Computing Machinery, 2012. (WIMS '12). ISBN 9781450309158. Available: <https://doi.org/10.1145/2254129.2254163>. Citations on pages 47 and 52.

CAPELLE, M.; MOERLAND, M.; HOGENBOOM, F.; FRASINCAR, F.; VANDIC, D. Bing-sf-idf+: A hybrid semantics-driven news recommender. In: **Proceedings of the 30th Annual ACM Symposium on Applied Computing**. New York, NY, USA: ACM, 2015. (SAC '15), p. 732–739. ISBN 978-1-4503-3196-8. Available: <http://doi.acm.org/10.1145/2695664.2695700>. Citation on page 31.

CHEN, L.; CHEN, G.; WANG, F. Recommender systems based on user reviews: The state of the art. **User Modeling and User-Adapted Interaction**, Kluwer Academic Publishers, Hingham, MA, USA, v. 25, n. 2, p. 99–154, Jun. 2015. ISSN 0924-1868. Available: <http://dx.doi.org/10.1007/s11257-015-9155-5>. Citation on page 31.

CHEN, L.; WANG, F. Explaining recommendations based on feature sentiments in product reviews. In: **Proceedings of the 22nd International Conference on Intelligent User Interfaces**. New York, NY, USA: Association for Computing Machinery, 2017. (IUI '17), p. 17–28. ISBN 9781450343480. Available: <https://doi.org/10.1145/3025171.3025173>. Citation on page 31.

CONTRATRES, F. G.; ALVES-SOUZA, S. N.; FILGUEIRAS, L. V. L.; DESOUZA, L. S. Sentiment analysis of social network data for cold-start relief in recommender systems. In: ROCHA,

Á.; ADELI, H.; REIS, L. P.; COSTANZO, S. (Ed.). **Trends and Advances in Information Systems and Technologies**. Cham: Springer International Publishing, 2018. p. 122–132. ISBN 978-3-319-77712-2. Citation on page 31.

D'ADDIO, R. M.; DOMINGUES, M. A.; MANZATO, M. G. Exploiting feature extraction techniques on users' reviews for movies recommendation. **Journal of the Brazilian Computer Society**, SpringerOpen, v. 23, 2017. Citations on pages 32, 37, and 52.

D'ADDIO, R. M.; MANZATO, M. G. **Filtragem baseada em conteúdo auxiliada por métodos de indexação colaborativa**. Master's Thesis (Master's Thesis) — Universidade de São Paulo, 2015. Citations on pages 32, 35, 37, and 52.

D'ADDIO, R. M.; FRESSATO, E. P.; COSTA, A. F. da; MANZATO, M. G. Incorporating semantic item representations to soften the cold start problem. In: **Proceedings of the 24th Brazilian Symposium on Multimedia and the Web**. New York, NY, USA: Association for Computing Machinery, 2018. (WebMedia '18), p. 157–164. ISBN 9781450358675. Available: <https://doi.org/10.1145/3243082.3243112>. Citations on pages 35, 53, 63, and 85.

D'ADDIO, R. M.; MANZATO, M. G. Exploiting item representations for soft clustering recommendation. In: **Proceedings of the 22nd Brazilian Symposium on Multimedia and the Web**. New York, NY, USA: Association for Computing Machinery, 2016. (Webmedia '16), p. 271–278. ISBN 9781450345125. Available: <https://doi.org/10.1145/2976796.2976858>. Citations on pages 35 and 37.

D'ADDIO, R. M.; MARINHO, R. S.; MANZATO, M. G. Combining different metadata views for better recommendation accuracy. **Information Systems**, v. 83, p. 1 – 12, 2019. ISSN 0306-4379. Available: <http://www.sciencedirect.com/science/article/pii/S0306437918300413>. Citations on pages 35, 63, and 85.

FORTES, A. da C.; MANZATO, M. G. Ensemble learning in recommender systems: Combining multiple user interactions for ranking personalization. In: **Proceedings of the 20th Brazilian Symposium on Multimedia and the Web**. New York, NY, USA: Association for Computing Machinery, 2014. (WebMedia '14), p. 47–54. ISBN 9781450332309. Available: <https://doi.org/10.1145/2664551.2664556>. Citation on page 63.

GANU, G.; KAKODKAR, Y.; MARIAN, A. Improving the quality of predictions using textual information in online user reviews. **Inf. Syst.**, Elsevier Science Ltd., GBR, v. 38, n. 1, p. 1–15, Mar. 2013. ISSN 0306-4379. Available: <https://doi.org/10.1016/j.is.2012.03.001>. Citation on page 37.

GEMMIS, M. de; LOPS, P.; MUSTO, C.; NARDUCCI, F.; SEMERARO, G. Semantics-aware content-based recommender systems. In: ____. **Recommender Systems Handbook**. Boston, MA: Springer US, 2015. p. 119–159. ISBN 978-1-4899-7637-6. Available: <https://doi.org/10.1007/978-1-4899-7637-6_4>. Citations on pages 30, 31, and 33.

HARUNA, K.; ISMAIL, M. A.; SUHENDROYONO, S.; DAMIASIH, D.; PIEREWAN, A.; CHIROMA, H.; HERAWAN, T. Context-aware recommender system: A review of recent developmental process and future research direction. **Applied Sciences**, MDPI AG, v. 7, n. 12, p. 1–25, Dec 2017. ISSN 2076-3417. Available: <http://dx.doi.org/10.3390/app7121211>. Citations on pages 30 and 31.

KAMINSKAS, M.; BRIDGE, D. Diversity, serendipity, novelty, and coverage: A survey and empirical analysis of beyond-accuracy objectives in recommender systems. **ACM Transactions on Interactive Intelligent Systems**, v. 7, n. 1, p. 2:1–2:42, December 2016. Citations on pages 30 and 85.

KOREN, Y.; BELL, R. Advances in collaborative filtering. In: ____. **Recommender Systems Handbook**. Boston, MA: Springer US, 2015. p. 77–118. ISBN 978-1-4899-7637-6. Available: <https://doi.org/10.1007/978-1-4899-7637-6_3>. Citation on page 30.

MANNING, C. D.; RAGHAVAN, P.; SCHüTZE, H. **Introduction to Information Retrieval**. USA: Cambridge University Press, 2008. ISBN 0521865719. Citations on pages 34 and 47.

MANNING, C. D.; SURDEANU, M.; BAUER, J.; FINKEL, J.; BETHARD, S. J.; MCCLOSKY, D. The Stanford CoreNLP natural language processing toolkit. In: **Association for Computational Linguistics (ACL) System Demonstrations**. [s.n.], 2014. p. 55–60. Available: <http://www.aclweb.org/anthology/P/P14/P14-5010>. Citation on page 53.

MARINHO, R. S.; D'ADDIO, R. M.; MANZATO, M. G. Semantic organization of user's reviews applied in recommender systems. In: **Proceedings of the 23rd Brazillian Symposium on Multimedia and the Web**. New York, NY, USA: Association for Computing Machinery, 2017. (WebMedia '17), p. 277–280. ISBN 9781450350969. Available: <https://doi.org/10.1145/3126858.3131600>. Citations on pages 35, 47, 63, and 133.

MCAULEY, J.; PANDEY, R.; LESKOVEC, J. Inferring networks of substitutable and complementary products. In: **Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining**. New York, NY, USA: ACM, 2015. (KDD '15), p. 785–794. ISBN 978-1-4503-3664-2. Available: <http://doi.acm.org/10.1145/2783258.2783381>. Citation on page 31.

MORO, A.; RAGANATO, A.; NAVIGLI, R. Entity Linking meets Word Sense Disambiguation: a Unified Approach. **Transactions of the Association for Computational Linguistics (TACL)**, v. 2, p. 231–244, 2014. Citations on pages 33, 47, 119, and 123.

MUSTO, C.; ROSSIELLO, G.; GEMMIS, M. de; LOPS, P.; SEMERARO, G. Combining text summarization and aspect-based sentiment analysis of users' reviews to justify recommendations. In: **Proceedings of the 13th ACM Conference on Recommender Systems**. New York, NY, USA: Association for Computing Machinery, 2019. (RecSys '19), p. 383–387. ISBN 9781450362436. Available: <https://doi.org/10.1145/3298689.3347024>. Citation on page 31.

NARDUCCI, F.; BASILE, P.; MUSTO, C.; LOPS, P.; CAPUTO, A.; GEMMIS, M. de; IAQUINTA, L.; SEMERARO, G. Concept-based item representations for a cross-lingual content-based recommendation process. **Inf. Sci.**, Elsevier Science Inc., New York, NY, USA, v. 374, n. C, p. 15–31, Dec. 2016. ISSN 0020-0255. Available: <https://doi.org/10.1016/j.ins.2016.09.022>. Citation on page 31.

NAVIGLI, R.; PONZETTO, S. P. BabelNet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network. **Artificial Intelligence**, v. 193, p. 217–250, 2012. Citation on page 47.

NING, X.; DESROSIERS, C.; KARYPIS, G. A comprehensive survey of neighborhood-based recommendation methods. In: ____. **Recommender Systems Handbook**. Boston, MA: Springer US, 2015. p. 37–76. ISBN 978-1-4899-7637-6. Available: <https://doi.org/10.1007/978-1-4899-7637-6_2>. Citations on pages 29 and 30.

O'REILLY, T. What is web 2.0? design patterns and business models for the next generation of software. 2005. Available: <http://oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html>. Citation on page 29.

PAZZANI, M. J.; BILLSUS, D. Content-based recommendation systems. In: ____. **The Adaptive Web: Methods and Strategies of Web Personalization**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007. p. 325–341. ISBN 978-3-540-72079-9. Available: <https://doi.org/10.1007/978-3-540-72079-9_10>. Citation on page 31.

PILEHVAR, M. T.; CAMACHO-COLLADOS, J.; NAVIGLI, R.; COLLIER, N. Towards a seamless integration of word senses into downstream nlp applications. In: **Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)**. Association for Computational Linguistics, 2017. p. 1857–1869. Available: <http://www.aclweb.org/anthology/P17-1170>. Citation on page 53.

QUMSIYEH, R.; NG, Y.-K. Predicting the ratings of multimedia items for making personalized recommendations. In: **Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval**. New York, NY, USA: Association for Computing Machinery, 2012. (SIGIR '12), p. 475–484. ISBN 9781450314725. Available: <https://doi.org/10.1145/2348283.2348349>. Citation on page 31.

RANA, A. **Chain-based recommendations**. Phd Thesis (PhD Thesis) — Insight Centre for Data Analytics, School of Computer Science & Information Technology, University College Cork, Ireland, 2020. Citation on page 85.

RANA, A.; BRIDGE, D. Explanations that are intrinsic to recommendations. In: ACM. **Proceedings of the 26th Conference on User Modeling, Adaptation and Personalization**. [S.l.], 2018. p. 187–195. Citations on pages 85 and 119.

RENDLE, S.; FREUDENTHALER, C.; GANTNER, Z.; SCHMIDT-THIEME, L. Bpr: Bayesian personalized ranking from implicit feedback. In: **Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence**. Arlington, Virginia, USA: AUAI Press, 2009. (UAI '09), p. 452–461. ISBN 9780974903958. Citation on page 63.

RICCI, F.; ROKACH, L.; SHAPIRA, B. Recommender systems: Introduction and challenges. In: ____. **Recommender Systems Handbook**. Boston, MA: Springer US, 2015. p. 1–34. ISBN 978-1-4899-7637-6. Available: <https://doi.org/10.1007/978-1-4899-7637-6_1>. Citations on pages 29, 30, and 37.

SOCHER, R.; PERELYGIN, A.; WU, J.; CHUANG, J.; MANNING, C.; NG, A.; POTTS, C. Recursive deep models for semantic compositionality over a sentiment treebank. **Conference on Empirical Methods in Natural Language Processing**, v. 1631, p. 1631–1642, 01 2013. Citation on page 53.

TERZI, M.; ROWE, M.; FERRARIO, M.-A.; WHITTLE, J. Text-based user-knn: Measuring user similarity based on text reviews. In: ____. **Procceedings of the 22nd International Conference on User Modeling, Adaptation, and Personalization, UMAP 2014**. Cham: Springer International Publishing, 2014. p. 195–206. ISBN 978-3-319-08786-3. Citation on page 31.

ZHANG, Y.; CHEN, X. Explainable recommendation: A survey and new perspectives. **Foundations and Trends® in Information Retrieval**, v. 14, n. 1, p. 1–101, 2020. ISSN 1554-0669. Available: <http://dx.doi.org/10.1561/1500000066>. Citations on pages 30 and 31.

ZHAO, G.; LEI, X.; QIAN, X.; MEI, T. Exploring users' internal influence from reviews for social recommendation. **IEEE Transactions on Multimedia**, v. 21, n. 3, p. 771–781, 2019. Citation on page 31.

APPENDIX

A

SEMANTIC ORGANIZATION OF USER'S REVIEWS APPLIED IN RECOMMENDER SYSTEMS - ORIGINAL PAPER

In the following we provide the original paper (MARINHO; D'ADDIO; MANZATO, 2017) from Chapter 3, written in Brazilian Portuguese.

Semantic organization of user's reviews applied in recommender systems

Ronnie S. Marinho, Rafael M. D'Addio e Marcelo G. Manzato
Instituto de Ciências Matemáticas e de Computação
Universidade de São Paulo
São Carlos, São Paulo - Brasil
ronnieshida@usp.br,{rdaddio,mmanzato}@icmc.usp.br

ABSTRACT

Recommender systems are widely used to minimize the information overload problem. A great source of information is users' reviews, since they provide both item descriptions and users' opinions. Recent works that process reviews often neglect problems such as polysemy and sinonimy. On the other hand, systems that rely on word sense disambiguation focus their efforts on items's static descriptions. In this paper, we propose a hybrid recommender system that uses word sense disambiguation and entity linking to produce concept-based item representations extracted from users' reviews. Our findings suggest that adding such semantics to items' representations have a positive impact on recommendations.

CCS CONCEPTS

• **Information systems → Content analysis and feature selection**; **Personalization**;

KEYWORDS

Recommender systems, item representation, word sense disambiguation

PUBLICATION INFORMATION

Published in proceedings of the 23rd Brazilian Symposium on Multimedia and the Web (Webmedia '17). 2017. Pages 271–278. DOI:10.1145/2976796.2976858

1 INTRODUÇÃO

Devido ao crescimento da Internet e a vasta gama de informações disponíveis na Web, surgiram os sistemas de recomendação (SR). Tais sistemas, auxiliam usuários na procura por itens de seu interesse.

De modo geral, SRs podem ser classificados em dois importantes paradigmas: i) filtragem baseada em conteúdo, onde os itens são recomendados a um usuário com base na similaridade de seu conteúdo com o perfil do usuário; e ii) filtragem colaborativa, onde as recomendações são inferidas a partir de relacionamentos criados através das interações dos usuários com o sistema [1]. Métodos híbridos têm ganhado destaque por combinarem aspectos de ambas filtragens, minimizando seus problemas [1].

Um problema recorrente nesses sistemas é a falta de informações para descrever os itens do acervo e as preferências de seus usuários. Recentemente, trabalhos têm estudado a possibilidade de utilizar revisões de usuários [4–6, 8, 12, 18], entretanto, tais textos apresentam diversos problemas, como ruídos e palavras escritas de forma errada. Apesar de haverem esforços relacionados à extração correta de informação [5–7], tais sistemas sofrem o agravante chamado ambiguidade lexical. Muitas palavras extraídas dos textos podem possuir diferentes significados e contextos (polissemia), do mesmo modo que diversas palavras podem referenciar o mesmo conceito e serem tratadas como características diferentes (sinonímia).

Diversos esforços têm sido realizados para agregar técnicas que realizam a extração de conceitos ao invés de palavras em sistemas de recomendação [2, 10, 14, 15]. Porém, a maioria desses trabalhos utilizam como fonte de informação textos de natureza estática, como sinopses e páginas da Wikipedia, negligenciando a fonte de informação detalhada das anotações de usuário. Em especial, revisões do usuário podem fornecer, ao mesmo tempo, uma descrição detalhada do item e as opiniões dos usuários sobre o produto, porém, sua utilização depende do correto processamento de informações.

Deste modo, este trabalho propõe um sistema que extrai conceitos de revisões de usuários para construir representações de itens. Para tal, utilizou-se uma ferramenta que realiza ambas operações de desambiguação lexical de sentido (DLS) e ligação de entidades (LE) chamada BabelFy[1] [13], a qual obtém conceitos na forma de *synsets* da base de conhecimento BabelNet[2] [16]. Tais *synsets* compõem as representações de itens, que são processadas por um algoritmo híbrido de recomendação. Resultados mostram a eficiência dessa abordagem, que supera as abordagens baseadas em termos com os quais o trabalho foi comparado.

Este artigo está organizado na seguinte maneira: a Seção 2 relaciona trabalhos que seguem a mesma linha de pesquisa deste projeto; na Seção 3 é apresentada a arquitetura do sistema proposto; na Seção 4 apresentam-se os experimentos realizados; por fim, a Seção 5 expõe conclusões e sugestões para trabalhos futuros.

2 TRABALHOS RELACIONADOS

Diversos esforços têm sido realizados com intuito de utilizar anotações não-estruturadas de usuários em sistemas de recomendação. Chen et al. [3] realizaram uma revisão da literatura relacionada a sistemas de recomendação que fazem uso de revisões de usuário.

Alguns trabalhos recentes fazem uso de revisões para caracterizar os itens do sistema, produzindo representações de itens [5, 6] ou construindo grafos relacionais [12]. Outros trabalhos também fazem uso de revisões para descrever usuários, construindo perfis que explicitam as suas preferências em relação à características [8], ou usando seus textos para encontrar similaridades entre eles [18]. Por fim, alguns trabalhos descrevem ambos usuários e itens [4], fazendo uso de revisões para organizar em ranques as características que são

[1] http://babelfy.org/
[2] http://babelnet.org/

mais relevantes para a preferência do usuário. A maior limitação dessas abordagens é que nenhuma delas lida com problemas de ambiguidade nos textos, como polissemia e sinonímia.

A taxonomia do WordNet[3] e os algoritmos de DLS baseados nela têm sido ativamente explorados na última década por sistemas de recomendação baseados em conteúdo [2, 10]. Por outro lado, alguns trabalhos de recomendação fazem uso de técnicas de LE para extrair características [14]. Finalmente, esforços têm sido realizados na incorporação de técnicas que façam ambos em SRs baseados em conteúdo, facilitando a criação de representações de itens [15, 17].

Este trabalho se diferencia dos demais apresentados nessa seção nos seguintes pontos. Primeiramente, trabalhos que usam DLS e/ou LE se baseiam em descrições pré-definidas, como sinopses ou páginas da Wikipedia. Por outro lado, trabalhos que usam revisões de usuários não realizam a extração de conceitos, focando apenas no uso de palavras-chave. O trabalho aqui apresentado, por sua vez, concentra-se na extração de conceitos discutidos pelos usuários em suas revisões. Em segundo lugar, as pesquisas apresentadas focam-se em recomendadores baseados em conteúdo. Este trabalho, por sua vez, produz descrições de itens que serão usadas em um sistema híbrido baseado em vizinhança de itens.

3 SISTEMA PROPOSTO

Conforme mencionado anteriormente, o sistema proposto neste trabalho utiliza DLS e LE com o objetivo de extrair conceitos dos itens. A partir dessa estruturação, é possível caracterizar os itens de maneira mais eficiente, favorecendo o cálculo da recomendação. Assim, foram elaborados módulos que realizam funções específicas, como ilustrado na Figura 1. Em linhas gerais, revisões de usuários são utilizadas para alimentar o módulo de extração de conceitos, o qual produz uma lista, ou vocabulário, que será utilizado pelo módulo de construção de representações de itens. Tendo produzido as representações, elas serão processadas pelo módulo de recomendação, o qual faz uso de um algoritmo baseado em vizinhança de itens para gerar sugestões para os usuários. Tais módulos serão melhor detalhados nas subseções seguintes.

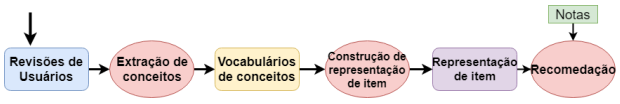


Figura 1: Arquitetura Geral do sistema proposto.

3.1 Extração de conceitos

O módulo de extração de conceitos é responsável pela realização de diversas tarefas que visam a estruturação de revisões de usuário. Nesse módulo, com o auxílio da ferramenta Babelfy, as seguintes tarefas são realizadas:

- Remoção de ruído: removem-se palavras com pouca carga semântica, ou seja, as *stop-words*, além de palavras com caracteres especiais, datas e caracteres numéricos.
- Desambiguação e Ligação de Entidades: realiza-se a identificação de conceitos e entidades nomeadas. Assim, é atribuído um *synset* presente na BabelNet para cada termo desambiguado. Além disso, obtém-se a classe gramatical de cada *synset*.
- Filtragem morfossintática: selecionam-se apenas os *synsets* com classe gramatical de substantivo, devido ao fato que características de itens na maioria dos domínios são substantivos.

O processo descrito anteriormente é aplicado a todas as revisões dos itens, produzindo um conjunto de *synsets* candidatos. Como o conjunto gerado é muito grande, realiza-se uma filtragem de características, removendo-se aqueles pouco frequentes. Como um item possui diversas revisões, utilizou-se a métrica de frequência do item (IF, do Inglês, *item frequency* [6]. A métrica IF é uma adaptação da tradicional frequência do documento (DF, do Inglês, *document frequency*) [11], porém considera todas as revisões do item como um único documento.

Calculado o IF de cada *synset*, o filtro consiste em remover aqueles cujo valor de IF é inferior a um determinado limiar. Neste trabalho, diversos valores de limiar foram testados, e seus resultados em relação à recomendação final são reportados na Seção 4.2.1. O resultado dessa filtragem produz um conjunto de *synsets*, o vocabulário, o qual será utilizado no módulo de construção de representações, detalhado a seguir.

3.2 Construção de representações

Esse módulo é responsável por utilizar o vocabulário de *synsets* para construir representações dos itens. Tais representações são modeladas em um espaço vetorial, onde cada *synset* corresponde a uma dimensão desse modelo, e cada vetor representa um item.

Assim como [2], este trabalho adota como esquema de pesagem dos vetores a técnica SF-IDF, uma extensão da técnica TF-IDF [11] aplicada a *synsets* ao invés de termos.

Essa técnica analisa quanto um *synset* é frequente em um item (SF) e quanto ele é raro dentre todos os itens (IDF). Formalmente, assume-se que $|I|$ é o número total de itens do sistema, e que um *synset* s aparece em $|i : t \in i|$ itens. Assume-se, também, que $n_{s,i}$ é a frequência do *synset* s nas revisões do item i. A frequência de um *synset* $SF(s, i)$ pode ser definida como [2]:

$$SF(s,i) = \frac{n_{s,i}}{\sum_k n_{k,i}}, \tag{1}$$

onde $\sum_k n_{k,i}$ corresponde à frequência total dos k demais *synsets* presentes nas revisões do item i.

A inversa frequência do documento, IDF_i, é computada como:

$$IDF(s) = \log \frac{|I|}{|i : t \in i|}. \tag{2}$$

Finalmente, o SF-IDF de um *synset* s em um item i pode ser dado pelo produto:

$$SF - IDF(s,i) = SF(s,i) \times IDF(s). \tag{3}$$

3.3 Recomendação

Como algoritmo de recomendação, optou-se pelo uso de um método de vizinhança baseado em item (item k-NN) [9], e este foi ajustado para utilizar no processo de obtenção de vizinhos as representações de itens geradas anteriormente.

O algoritmo item-kNN tem como objetivo prever notas de itens desconhecidos pelo usuário, baseando-se nas notas de outros itens

[3] https://wordnet.princeton.edu/

similares já avaliados por ele. Para tal, utiliza-se uma medida de similaridade entre as representações de itens. A medida de similaridade escolhida é o coeficiente de correlação de Pearson s_{ij} [9].

Com os valores de similaridade, o algoritmo identifica os k itens avaliados por u que são mais similares a i, ou seja, os k-vizinhos mais próximos. Denota-se este conjunto como $S^k(i;u)$. Usando este conjunto, a nota predita final é uma média das notas dos k itens mais similares, ajustadas às suas estimativas de referência [9]:

$$\hat{r}_{ui} = b_{ui} \quad \frac{\sum_{j \in S^k(i;u)} s_{ij}(r_{uj} - b_{uj})}{\sum_{j \in S^k(i;u)} s_{ij}}, \tag{4}$$

onde b_{ui} representa as estimativas de referência, um valor que encapsula os vieses de ambos usuários e itens.

4 METODOLOGIA E AVALIAÇÃO

Nessa seção, apresentam-se os experimentos executados para validar esta proposta. Primeiramente, apresentam-se algumas informações sobre a base de dados utilizada, em seguida, descrevem-se as configurações dos experimentos e, por fim, apresentam-se os resultados.

4.1 Configuração dos experimentos

No intuito de avaliar o sistema proposto, realizaram-se experimentos na base de dados MovieLens 100k[4]. Esta foi expandida ao se agregar revisões de usuários do IMDb[5]. Foram recolhidas em média as 10 primeiras revisões de cada item, ordenadas por utilidade, resultando em um total de 15963 documentos.

A proposta foi comparada com outras abordagens, também baseadas em vizinhança no cenário de predição de notas. Comparou-se com as seguintes representações basais:

- *Gêneros:* essa representação faz uso dos metadados de gêneros dos filmes, onde cada posição dos vetores se refere a um gênero que o item contém (valor 1) ou não (valor 0);
- *Termos Heurísticos:* essa representação, proposta em [6], faz uso de heurísticas para a extração de termos a partir das revisões de usuários. Tal técnica seleciona termos lematizados da classe morfossintática de substantivos, e aplica também o IF (com limiar 30) para filtragem de termos menos frequentes. Consideram-se aqui dois pesos diferentes para os termos: TF-IDF e, conforme proposto no trabalho original, análise de sentimento. O sentimento representa a opinião geral dos revisores com respeito a cada característica do item. Adota-se a terminologia TH-TFIDF e TH-Sentimento para cada versão das representações.
- *Termos Classificados:* nessa representação, proposta em [5], faz-se uso de um algoritmo de classificação transdutivo para a extração de termos radicalizados, os quais também possuem as duas pesagens descritas na representação anterior. Adota-se a nomenclatura TC-TFIDF e TC-Sentimento para cada versão de representações.

Para o desenvolvimento, execução e comparação das representações, utilizou-se o algoritmo de recomendação descrito na Seção 3.3, com o número de vizinhos configurado para k=(20, 40, 60, 80, 100), e a similaridade entre os itens calculada pelo coeficiente de correlação de Pearson retraído de acordo com [9].

Os sistemas foram submetidos ao método de validação cruzada com 10 partições, e os resultados são avaliados no cenário de predição de notas, cuja métrica utilizada é o erro médio quadrático (RMSE, do Inglês *Root Mean Square Error*). Finalmente, foi utilizado o teste Wilcoxon para verificar se os resultados obtidos pela abordagem são estatisticamente diferentes daqueles obtidos pelas representações basais.

4.2 Resultados

Para validar a proposta, foram realizados dois experimentos. No primeiro, analisaram-se diferentes tamanhos de vocabulários de *synsets*, definidos por cortes baseados no IF das características. No segundo experimento, comparou-se a qualidade da técnica proposta em relação a outras representações basais, descritas na seção anterior. Tais experimentos são detalhados nas seguintes subseções:

4.2.1 Experimento 1: encontrando um limiar ideal. Neste experimento, diversos limiares de corte baseados no IF foram verificados no intuito de encontrar o conjunto mais adequado de *synsets*, eliminando-se as características menos frequentes e que, consequentemente, agregam pouca ou nenhuma informação para as representações. Assim, o vocabulário extraído pelo BabelFy foi filtrado com limiares definidos como l_1= 1, l_2= 10, l_3= 30 e l_4= 50. Os algoritmos foram chamados de: *i*) Babelfy-limiar 1, *ii*) Babelfy-limiar 10, *iii*) Babelfy-limiar 30, e *iv*) Babelfy-limiar 50, respectivamente. Além disso, comparam-se as representações filtradas com a representação sem aplicar a filtragem (Babelfy-completo). A Tabela 1 apresenta os resultados obtidos.

Tabela 1: Valores obtidos de RMSE para as representações com diferentes limiares. Valores em negrito indicam melhores resultados.

Algoritmo	Termos	k=20	k=40	k=60	k=80	k=100
Babelfy-completo	60466	0,9197	0,9200	0,9208	0,9216	0,9224
Babelfy- limiar 1	28182	0,9189	0,9192	**0,9201**	**0,9209**	**0,9217**
Babelfy- limiar 10	6238	**0,9183**	**0,9190**	0,9203	0,9214	0,9224
Babelfy-limiar 30	2747	0,9212	0,9217	0,9230	0,9243	0,9254
Babelfy-limiar 50	1797	0,9225	0,9231	0,9246	0,9260	0,9271

Nota-se que as representações com limiares menores apresentaram melhores resultados, ou seja, os vocabulários produzidos por l_1 = 1 e l_2 = 10. Ambos vocabulários apresentam resultados similares, sendo que o primeiro é estatisticamente superior apenas para k = 100 com valor $p < 0.05$. Além disso, comparado ao segundo, o tamanho do primeiro vocabulário é significativamente maior (29000 características em relação à pouco mais de 6000). Desse modo, dá-se preferência pela utilização da abordagem Babelfy-limiar 10, dado que uma vez que ter uma dimensionalidade menor reduz recursos computacionais, além de gerar recomendações mais rápidas. Utiliza-se, portanto, essa abordagem para o próximo experimento.

4.2.2 Experimento 2: comparação com representações basais. Nesse experimento, a abordagem Babelfy-limiar 10 foi comparada com as representações descritas na Seção 4.1. Ambas representações TH possuem 3085 termos, enquanto as representações TC possuem 8433 termos. A representação baseada em gêneros possui, ao todo, 18 características.

[4] https://grouplens.org/datasets/movielens/100k/
[5] http://www.imdb.com/

Segundo trabalhos anteriores [6, 7], o tamanho da representação é um fator importante no cálculo da similaridade e, consequentemente, recomendação do algoritmo item k-NN. Para se avaliar a qualidade da representação independentemente do tamanho do vocabulário dela, foram realizados experimentos com o mesmo número de características de ambas representações basais. Os resultados são reportados na Tabela 2.

Tabela 2: Valores obtidos de RMSE para as representações propostas e basais. Valores em negrito indicam melhores resultados.

Algoritmo	Termos	k=20	k=40	k=60	k=80	k=100
Babelfy- limiar 10	6238	0,9183	0,9190	0,9203	0,9214	0,9224
Gêneros	18	0,9404	0,9401	0,9401	0,9401	0,9401
Babelfy-3085		0,9202	0,9208	0,9222	0,9235	0,9246
TH-TFIDF	3085	0,9435	0,9407	0,9402	0,9403	0,9404
TH-Sentimento		0,9310	0,9314	0,9330	0,9347	0,9361
Babelfy-8433		**0,9180**	**0,9186**	**0,9199**	**0,9210**	**0,9219**
TC-TFIDF	8433	0,944	0,9436	0,9434	0,9432	0,9431
TC-Sentimento		0,9301	0,9305	0,9327	0,9345	0,936

A princípio, foi comparado Babelfy-limiar 10 com as representações TH-TFIDF, TH-Sentimento, TC-TFIDF, TC-Sentimento e Gêneros. Nota-se um desempenho estatisticamente superior para valor $p < 0.01$ contra todas elas.

Além disso, foram construídas versões com o mesmo tamanho de vocabulário das representações basais, produzindo-se a representação Babelfy-3085 para ser comparada com ambas representações TH, e Babelfy- 8433 para ser avaliada contra as representações TC. Observa-se que ambas abordagens baseadas em Babelfy apresentam resultados estatísticamente superiores com valor $p < 0.01$. Isso reforça a premissa que, ao agregar maior semântica às representações, os itens são melhores caracterizados e, consequentemente, o sistema é capaz de produzir recomendações mais apuradas.

Finalmente, ao se analisar os resultados entre as versões baseadas em sentimento e TF-IDF das representações basais, pode-se observar que há uma melhora nos valores de RMSE. Isso serve de indicativo que a análise de sentimento fornece uma carga semântica a mais para as representações, o que pode ser incorporado à estratégia proposta neste artigo em trabalhos futuros.

5 CONCLUSÃO

Este artigo apresentou um sistema de recomendação baseado em conceitos desambiguados extraídos de revisões de usuários. Os experimentos foram focados no domínio de filmes, no entanto, o sistema é generalizável a outros domínios.

Os resultados mostram que a utilização de conceitos desambiguados, associados a entidades nomeadas, pode melhorar a qualidade das representações de itens, consequentemente aprimorando a eficácia de um sistema de recomendação.

Como trabalhos futuros, espera-se integrar a análise de sentimento à proposta, tendo em vista que ela agrega uma carga semântica maior às representações. Pretende-se também estender a aplicabilidade da técnica a outros algoritmos de recomendação que façam uso de representações de itens baseadas em conteúdo.

6 AGRADECIMENTOS

Os autores gostariam de agradecer o suporte financeiro de: CAPES, CNPq e FAPESP (processo 2016/20280-6).

REFERÊNCIAS

[1] J. Bobadilla, F. Ortega, A. Hernando, and A. Gutiérrez. 2013. Recommender systems survey. *Knowledge-Based Systems* 46, 0 (2013), 109–132.

[2] Michel Capelle, Flavius Frasincar, Marnix Moerland, and Frederik Hogenboom. 2012. Semantics-based News Recommendation. In *Proceedings of the 2Nd International Conference on Web Intelligence, Mining and Semantics (WIMS '12)*. ACM, New York, NY, USA, Article 27, 9 pages.

[3] Li Chen, Guanliang Chen, and Feng Wang. 2015. Recommender systems based on user reviews: the state of the art. *User Modeling and User-Adapted Interaction* 25, 2 (2015), 99–154.

[4] Xu Chen, Zheng Qin, Yongfeng Zhang, and Tao Xu. 2016. Learning to Rank Features for Recommendation over Multiple Categories. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '16)*. ACM, New York, NY, USA, 305–314. https://doi.org/10.1145/2911451.2911549

[5] Rafael D'Addio, Merley Conrado, Solange Resende, and Marcelo Manzato. 2014. Generating recommendations based on robust term extraction from users' reviews. In *Proceedings of the 20th Brazilian Symposium on Multimedia and the Web*. ACM, 55–58.

[6] Rafael Martins D'Addio and Marcelo Garcia Manzato. 2014. A collaborative filtering approach based on user's reviews. In *Intelligent Systems (BRACIS), 2014 Brazilian Conference on*. IEEE, 204–209.

[7] Rafael M D'Addio and Marcelo G Manzato. 2016. Exploiting Item Representations for Soft Clustering Recommendation. In *Proceedings of the 22nd Brazilian Symposium on Multimedia and the Web*. ACM, 271–278.

[8] Gayatree Ganu, Yogesh Kakodkar, and AméLie Marian. 2013. Improving the Quality of Predictions Using Textual Information in Online User Reviews. *Information Systems* 38, 1 (March 2013), 1–15. https://doi.org/10.1016/j.is.2012.03.001

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

[10] Pasquale Lops, Cataldo Musto, Fedelucio Narducci, Marco De Gemmis, Pierpaolo Basile, and Giovanni Semeraro. 2010. MARS: A MultilAnguage Recommender System. In *Proceedings of the 1st International Workshop on Information Heterogeneity and Fusion in Recommender Systems (HetRec '10)*. ACM, New York, NY, USA, 24–31. https://doi.org/10.1145/1869446.1869450

[11] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to Information Retrieval*. Cambridge University Press.

[12] J. McAuley, R. Pandey, and J. Leskovec. 2015. Inferring Networks of Substitutable and Complementary Products. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '15)*. ACM, New York, NY, USA, 785–794.

[13] Andrea Moro, Alessandro Raganato, and Roberto Navigli. 2014. Entity linking meets word sense disambiguation: a unified approach. *Transactions of the Association for Computational Linguistics* 2 (2014), 231–244.

[14] Cataldo Musto, Giovanni Semeraro, Pasquale Lops, and Marco de Gemmis. 2014. *Combining Distributional Semantics and Entity Linking for Context-Aware Content-Based Recommendation*. Springer International Publishing, Cham, 381–392. https://doi.org/10.1007/978-3-319-08786-3_34

[15] Fedelucio Narducci, Pierpaolo Basile, Cataldo Musto, Pasquale Lops, Annalina Caputo, Marco de Gemmis, Leo Iaquinta, and Giovanni Semeraro. 2016. Concept-based Item Representations for a Cross-lingual Content-based Recommendation Process. *Inf. Sci.* 374, C (Dec. 2016), 15–31. https://doi.org/10.1016/j.ins.2016.09.022

[16] Roberto Navigli and Simone Paolo Ponzetto. 2012. BabelNet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network. *Artificial Intelligence* 193 (2012), 217–250.

[17] Sergio Oramas, Vito Claudio Ostuni, Tommaso Di Noia, Xavier Serra, and Eugenio Di Sciascio. 2016. Sound and Music Recommendation with Knowledge Graphs. *ACM Trans. Intell. Syst. Technol.* 8, 2, Article 21 (Oct. 2016), 21 pages. https://doi.org/10.1145/2926718

[18] Maria Terzi, Matthew Rowe, Maria-Angela Ferrario, and Jon Whittle. 2014. *Text-Based User-kNN: Measuring User Similarity Based on Text Reviews*. Springer International Publishing, Cham, 195–206. https://doi.org/10.1007/978-3-319-08786-3_17


ICMC
USP
SÃO CARLOS