Chain-Based Recommendations

Arpit Rana
B.Eng., M.Tech.
115220377

Thesis submitted for the degree of Doctor of Philosophy

NATIONAL UNIVERSITY OF IRELAND, CORK

College of Science, Engineering and Food Science

School of Computer Science & Information Technology
Insight Centre for Data Analytics

October 2019

Head of Department:	Prof. Cormac J. Sreenan
Supervisors:	Dr. Derek G. Bridge, Prof. Gregory Provan

Research supported by Science Foundation Ireland

Contents

List of Figures

List of Tables

I, Arpit Rana, certify that this dissertation is my own work and I have not obtained a degree in this university or elsewhere on the basis of the work submitted in this dissertation.

Arpit Rana

To my parents

Acknowledgements

I would like to thank:

- My supervisor, Dr. Derek G. Bridge, for teaching me: how to do research, how to streamline my thoughts, and how to attain clarity of expression; for sharing his wisdom and in-depth knowledge with me; for his patience in the face of my mistakes and, sometimes, my stubborn insistence; and for constantly pushing me towards greater things.
- My colleagues: Francisco J. Peña, Mesut Kaya, and Diego Carraro for their support and suggestions.
- Staff of the Insight Centre for Data Analytics: Caitriona Walsh and Eleanor O'Riordan, for their support right from the submission of the PhD application; Linda O'Sullivan, for her assistance, especially in preparing the terms and conditions for the user trials; and Peter J. MacHale, for his assistance in setting up the server and making sure that it ran smoothly at the time of the user trials.
- My collaborator: Rafael Martins D'Addio for his contribution in collecting, preparing, and sharing one of the datasets on which we carried out our experiments.
- My friends: Kapil Bhagchandani and Arpit Jain, for their assistance in the design and deployment of the web-based system, without which none of the user trials would have been possible. Shrikant Govind, Sanjeev Kumar, Ghanshyam Verma, and Vaibhav Sinha for giving me the best reviewers' comments on my work; for all the great things we have done together, especially in Ireland; and overall, for their support.
- My parents, for constantly inspiring me, believing in me, and encouraging me with their positive attitude.

This research is 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.

Abstract

Recommender systems are discovery tools. Typically, they infer a user's preferences from her behaviour and make personalized suggestions. They are one response to the overwhelming choices that the Web affords its users.

Recent 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 their rationale, and if the system allows the user to provide feedback on the recommendations to improve the next round of recommendations so that they take account of the user's ephemeral needs.

The goal of this dissertation is to introduce a new recommendation framework that offers a better user experience, while giving quality recommendations. It works on content-based principles and addresses both the issues identified in the previous paragraph, i.e. explanations and recommendation feedback. We instantiate our framework to produce two recommendation engines, each focusing on one of the themes: (i) the role of explanations in producing recommendations, and (ii) helping users to articulate their ephemeral needs.

For the first theme, we show how to unify recommendation and explanation to a greater degree than has been achieved hitherto. This results in an approach that enables the system to find relevant recommendations with explanations that have a high degree of both fidelity and interpretability. For the second theme, we show how to allow users to steer the recommendation process using a conversational recommender system. Our approach allows the user to reveal her short-term preferences and have them taken into account by the system and thus assists her in making a good decision efficiently. Early work on conversational recommender systems considers the case where the candidate items have structured descriptions (e.g. sets of attribute-value pairs). Our new approach works in the case where items have unstructured descriptions (e.g. sets of genres or tags).

For each of the two themes, we describe the problem settings, the state-of-the-art, our system design and our experiment design. We evaluate each system using both offline analyses as well as user trials in a movie recommendation domain. We find that the proposed systems provide relevant recommendations that also have a high degree of serendipity, low popularity-bias and high diversity.

Part I

Introduction and Background

Chapter 1

Introduction

In this dissertation, we introduce *chain-based recommendation*, a new framework for content-based recommendation in which recommender engines construct *chains of items*. We instantiate this framework in two concrete ways. Firstly, we use chains of items to *explain* recommendations. In this case, chains contain items from the user's profile, apart from the final item in the chain, which is a recommended item. Secondly, we use chains to *explore* item space. In this case, chains are sequences of recommended items, and the chains are constructed iteratively in response to user feedback, allowing the user to reveal her ephemeral needs, tastes, and interests.

1.1 Motivation

The Web affords its users an abundance of choice among items that include products, services and sources of information. However, having more choice is not necessarily better than having less. For example, Iyengar & Lepper found in their studies on choice overload that participants actually reported greater subsequent satisfaction with their selections when their original set of options had been limited [IL00]. People say that they like more choices but they often experience greater satisfaction when the range of choices matches their ability to manage them [OHS09].

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 a set of items and suggest to the user a smaller set – ones which it thinks are most likely to satisfy the user [AT05]. While a recommender system can make non-personalized recommendations (e.g. based on item popularity), most often they use personalization techniques: they infer the user's preferences from her behaviour and suggest items that they think will match these inferred preferences.

Some recommender systems can also provide explanations. Explanations can serve a multiplicity of aims: they give credibility to recommendations [SS02, PC07], help users make better choices [BM05], positively contribute to a better user experience [KR12] and so on. Due to the wide adoption of recommender systems in many aspects of our lives, explaining recommendations has attracted considerable attention [ZC18, TM15]. Despite this attention, most recommendation explanations are posthoc rationalizations.

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 [AN16]. For example, in [RSZ13], a recommendation generated by matrix factorization is explained using topic models mined from textual data associated with items. Such differences are one cause of low fidelity [KSB$^+$13] (also called objective transparency [GJG14]): the extent to which the explanation reveals the logic of the underlying recommender. In an experiment with a music recommender, Kulesza et al. found that the more that explanations were both sound and complete with respect to the recommender, the greater the users' trust in the recommender and the better their understanding [KSB$^+$13]. 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 Part II of this dissertation, 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.

It can also be advantageous to incorporate user feedback into the recommendation process [JJ17]. He et al. claimed that recommender systems achieve higher levels of trust and transparency, and greater acceptance if they enable users to steer the recommendation process [HPV16]. This is not possible in early so-called single-shot recommender systems, where, on receipt of a set of recommendations, the user can only choose to consume the recommended items or not. It was soon realized that the process by which a user selects an item to consume (e.g. a movie to watch) should be an iterative one: the user's requirements may be uncertain, or even erroneous, and may be refined by exposure to the items a system presents to her [PC08]. Single-shot recommendation fails to handle the case where the user has an ephemeral goal, in which case her short-term preferences may differ from her long-term ones. For example, a user might usually watch documentaries but this evening she is not in the mood for something so serious. Or, perhaps, this evening she wants something to watch with her mother, so she should accommodate her mother's tastes as well as her own. These factors motivate the use of conversational recommender systems that allow repeated interactions between the user and the system. Part III of this dissertation investigates how to integrating a user's feedback on a set of recommendations into the process of computing the next round of recommendations, in order to help the user reveal her ephemeral needs.

1.2 Contributions

This dissertation presents a new recommendation framework that offers a better user experience, while giving quality recommendations. We focus on two main issues that can improve the user experience: *explanation* and *incorporation of recommendation feedback*. Both exploit a new form of content-based recommendations that constructs *chains of items*.

We highlight the main contributions that we make to address the aforementioned issues below.

1.2.1 Recommendation-by-Explanation

We introduce Recommendation-by-Explanation (*r-by-e*), a novel unified approach for recommendation explanation: the system constructs an explanation, a chain of items from the user's profile, for each candidate item; then it recommends those candidate items that have the best explanations. By unifying recommendation and explanation, *r-by-e* finds relevant recommendations with explanations that have a high degree of fidelity.

Unlike classic content-based systems, *r-by-e* treats each item as a set of elements (e.g. its features or neighbours) and 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 the neighbouring 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 more diverse and serendipitous recommendations in an effective manner that is sensitive to user understanding.

1.2.1.1 Basic version of ***r-by-e***

In Chapter 4, we present a basic form of *r-by-e* where the chains of items (explanations) try to cover the features of the candidate items; top-n candidates are selected for recommendation based on this feature coverage and their capacity to cover the user's profile.

We compare *r-by-e* with a customized version of a classic content-based method on a movie recommendation dataset. Results from offline experiments show that *r-by-e* attains better precision, while remaining competitive on measures of diversity and surprise.

We also evaluate *r-by-e* through two user trials involving over a hundred participants. In one trial, we evaluate the quality of recommendations. Participants in this trial found r-by-e's recommendations to be more diverse, serendipitous and relevant than those of the competitor system. In the other trial, we evaluate the effectiveness of its explanations. We analyse users' response and found that *r-by-e*'s explanations allow users to make more accurate judgments about the quality of recommended items.

1.2.1.2 Extensions to ***r-by-e***

In Chapter 5, we extend *r-by-e*. We give ways of generating chains that use different item representations: i) as a set of its features (e.g. keywords or tags); and ii) as a set of its neighbours (i.e. similar items). We also explore weighting schemes to give more refined versions of the proposed algorithms. Thus, we define four versions of *chain generation*.

We also generalize *chain selection*: *r-by-e*'s approach to selecting the top-n chains to recommend. In Chapter 4, *r-by-e* assigns a score to each generated chain that is simply a sum of the coverage of the candidate's features and coverage of the user's profile. In Chapter 5, we redefine the score to be a linear combination of the two controlled by a balancing parameter α.

We give a comprehensive empirical comparison of all four versions (99 configurations of each) with their corresponding customized classic content-based methods on two different datasets. The versions of *r-by-e* that represent an item as a set of its features has several advantages over the others, and the empirical comparison shows that the one of these versions —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 analyse the influence of α on the measures of quality such as precision, diversity, and surprise for *wfb*. We plot graphs of these measures against α to better understand their relationships. We also find correlation among these measures and other factors; for example, we show how precision correlates to the candidate's coverage.

We conduct another user trial using a new version of the dataset in which item features are nouns extracted from user-generated reviews and where these nouns are assigned weights that reflect user sentiment to generate sentiment-aware explanation chains. We found on this version of the dataset that *wfb* provides more relevant recommendations and that they are also diverse and serendipitous; and we found that its sentiment-aware chains are more helpful to users when judging recommendation quality than the competitor explanations.

1.2.2 Navigation-by-Preference

We propose Navigation-by-Preference (*n-by-p*), a new conversational recommender that uses preference-based feedback (where a user simply selects one of the current round of recommendations) to help users navigate through item space to find an item of interest, especially when it is different from her long-term tastes. *n-by-p* works on unstructured item representations (such as sets of keywords or tags), thus it extends preference-based feedback to these representations. *n-by-p* can be configured to also take into account long-term preferences, or feedback from earlier cycles in the dialog, or both, in order to minimize the effort of reaching an item of interest.

We define twelve techniques to update the set of candidate items available for the next round of recommendation for each of the neighbour-based and feature-based representations. We group the update techniques into two: *immediate* and *cumulative*. The former includes five techniques that select or discard candidates based on the item that the user selects, while the latter includes seven techniques that update weights for all available candidates.

We give a comprehensive empirical comparison of all 120 configurations (60 configurations for each representation) for their performance (in terms of hit-rate) on a movie recommendation dataset with simulated users. The techniques that use the neighbour-based representation have several advantages over the others, and the empirical comparison shows that one of these methods —one that includes the user's long-term preferences; uses both positive and negative feedback, uses previous rounds of feedback— is the one with highest hit-rate.

We also evaluate the best performing version of *n-by-p* in a user trial. We compare this version of *n-by-p* with one that ignores long-term preferences for both the neighbour-based and feature-based representations. This version of *n-by-p* obtains the best survey responses and lowest measures of effort in a between-subject trial involving over 200 participants.

We give an improvement metric to analyze the difference between the user's seed movie (her initial choice from her profile) and the final accepted movie (from the candidate items). We also found that *n-by-p* reduces the popularity bias and suggests more surprising recommendations as the dialog length increases.

It is also noteworthy that we design a novel protocol for exploring short-term preferences in recommender system user trial of this kind.

1.3 Structure of the Dissertation

This dissertation comprises four parts. Part I (Chapters 1–2) contains this introduction and the conceptual background to the proposed chain-based recommendation framework. Part II (Chapters 3–5) concerns recommendation-by-explanation, covering a literature review, the design and experimental evaluation of a basic technique, and the design and evaluation of extensions to that technique. Part III (Chapters 6–7) concerns navigation-by-preference, again covering a literature review design and experimental evaluation. Part IV (Chapter 8) presents conclusions and ideas for future work.

Part I

Chapter 1 – Introduction

In this chapter, we present the contributions that we have made in this work.

Chapter 2 – Background

In this chapter, we present the main types of recommender system, including their strengths and weaknesses. We also introduce our recommendation framework: chain-based recommendation. We also give material that is common to the whole dissertation. This includes an overview of the dataset we use, the evaluation measures that we use in offline experiments and the main notation.

Part II

Chapter 3 – Recommendation Explanation: State of the Art

In this chapter, we review existing work on explanations of recommendations. We begin by looking at explanation goals and end with their evaluation with real users. In between, we review previous work according to various characterizations of explanations, such as based on their information content, based on their assumptions about the recommender, and based on their role in the recommendation process.

Chapter 4 – Explanation Chains

In this chapter, we explain Recommendation-by-Explanation (*r-by-e*) in detail. We report the results of an offline experiment on a publicly available movie dataset. Using the best performing configuration of *r-by-e* that we find in these offline experiments, we also evaluate our system with real users. As *r-by-e* unifies recommendation and explanation, we conduct two user-trials: one to evaluate the quality of its recommendations and the other to measure the effectiveness of the corresponding explanations.

Chapter 5–Extensions to Recommendation-by-Explanation

Where Chapter 4 presents and evaluates a basic form of r-by-e, in Chapter 5 we consider various extensions. We present two different ways of representing items: the first, feature-based, was used in Chapter 4, and we contrast it with an alternative, neighbour-based. We also explore weighting schemes that can be used to give a more refined form of r-by-e. Finally, we formulate versions of r-by-e that work with item representations that include user sentiment. In the case of these sentiment-aware explanation chains, we carry out another user trial.

Part III

Chapter 6 – Conversational Recommendation: State of the Art

In this chapter, we review conversational recommender systems, in which user iteratively reveal their preferences. We describe the goals of these systems, and how they are evaluated. Between these two, we review existing work according to various characterizations, such as who is taking the initiative in the interaction, what kind of feedback the user provides during the interaction, and the role of long- and short-term preferences.

Chapter 7 – Preference Chains

In this chapter, we explain Navigation-by-Preference (*n-by-p*) in detail. We present various versions, including ones that only take the most recently revealed preferences into account and ones that accumulate preferences across the dialog, and versions that use the feature-based and neighbour-based item representations that we covered previously in Chapter 5.

We present an offline experiment, with simulated users, that selects the best of 60 different configurations of two versions of n-by-p. Then we used a web-based system to conduct a user trial. The trial has a novel protocol for assigning short-term preferences to participants.

Part IV

Chapter 8 – Conclusions and Future Work

This concluding chapter summarizes the contributions we have made, discusses the implications of the work, and points to directions for future research.

1.4 Publications

- Arpit Rana and Derek Bridge: *Navigation-by-Preference: A New Conversational Recommender with Preference-based Feedback*, submitted, 2019.
- Arpit Rana and Derek Bridge: *Explanations that are Intrinsic to Recommendations*, Proceedings of the Twenty-Sixth Conference on User Modeling, Adaptation and Personalization, ACM, pp.187-195, 2018.
- Arpit Rana and Derek Bridge*Explanation Chains: Recommendations by Explanation*, in Domonkos Tikk and Pearl Pu (eds.), Proceedings of the Poster Track of the Eleventh ACM Conference on Recommender Systems, CEUR Workshop Proceedings, vol-1905, 2017.

Chapter 2

Background

In this chapter, we present the background concepts used in this dissertation. We begin with the fundamentals of recommender systems, and the primary approaches to recommendation along with some of their benefits and short-comings. We then introduce the notion of *chain-based recommendations*. We also cover some material that is common to the rest of the dissertation: the dataset that we use, the evaluation measures that we use in offline experiments, and finally the main notation that we use.

2.1 Recommender Systems

Recommender systems are prominent for services with a large number of items, where users are faced with an overwhelming choice of which items to consume. They are tools for item discovery. Typically, they use data that they have collected about user behaviour to algorithmically infer user preferences, so as to select and present relevant and novel content to the users [KR12]. News [DDGR07], blogs [EAVSG09], search results [GLK$^+$09] and streamed videos [GUH16] are a few among the many domains in which recommendation can be valuable.

Let $\mathbb{U}$ be a set of *users* and $\mathbb{I}$ be a set of *items*. Then one simple formulation of a recommender system is to define it as a system that attempts to find an item $i^* \in \mathbb{I}$ for user $u \in \mathbb{U}$ such that the *utility* of item i^* for user u, $\mathbf{u}(u, i^*)$, is maximized:

$$i^* = \arg\max_{i \in \mathbb{I}} \mathbf{u}(u, i) \tag{2.1}$$

The recommender may use models learned from demographic or social network data about users, or from descriptions of the items (such as the genres of a movie). But most common is to use data about user-item *interactions*. These might be in the form of actions such as clicks, downloads or purchases, or *feedback* in the form of ratings or reviews. In some cases, interactions are accompanied by contextual information, which affords the possibility of building a context-aware recommender, in which recommendations are contextualized as well as personalized. Since the majority of recommender systems so far are built from user feedback interactions, we consider this in more detail.

A user can reveal her opinions in either an *explicit* or *implicit* manner. Directly stated opinions are explicit feedback. Most often this takes the form of a rating from an ordinal scale, e.g. a star rating between 1 and 5 stars. But it could be binary: positive/negative or like/dislike or thumbs-up/thumbs-down. Less usually, it could take the form of a binary comparison, e.g. item A is preferred over item B. User reviews also offer explicit feedback, although it may take a lot of processing to recover opinions about items and their features from the text contained in these reviews.

On the other hand, if a user's opinion is derived from her other interactions with the system, then the feedback is said to be implicit. Typically, this type of feedback does not contain negative observations. Examples include inferring preference from purchase actions, from clicks, from dwell-time, from consumption frequency (such as the number of times a song is listened to), and so on.

2.2 Recommendation Approaches

Based on the information that a recommender uses to compute the utility of items to users, recommendation systems can be classified into at least the following three types:

- *Collaborative methods* recommend items that either users with similar tastes liked in the past or that, according to the other users, are similar to items that are liked by the active user.
- *Content-based methods* recommend items which, according to the item descriptions, are similar to items that are liked by the active user.
- *Hybrid methods* combine collaborative and content-based methods.

2.2.1 Collaborative methods

Collaborative methods rely on the notion that similar users tend to like items similarly. In order to recommend items to an active user, these methods learn the inter-dependencies among users and items from the observed set of user-item interactions, such as ratings.

We can visualize user-item ratings as a *ratings matrix* whose rows designate users and whose columns designate items. Thus an entry in the matrix denotes a user's rating for an item. Typical collaborative algorithms use the available information in the rating matrix to predict missing values, i.e. to predict ratings for items that the active user has not rated yet. Imputing missing values is thus a matrix completion task and is commonly accomplished by means of either *memory-based methods* or *model-based methods* [Agg16].

Memory-based methods (also known as neighbourhood-based collaborative methods) are usually heuristic approaches that predict the missing ratings either based on the ratings given by peers to that item (referred to as *user-based collaborative filtering*) or based on the ratings of items that have similar rating patterns to that item (referred to as *item-based collaborative filtering*). The discovery of these neighbourhoods of ratings and their aggregation is usually deferred to recommendation time.

By contrast, model-based methods use machine learning and data mining algorithms in advance of recommendation time to learn a model that can predict the missing ratings for an active user. Common model-building techniques in this context include Matrix Factorization [Kor08] and Sparse Linear Methods [NK11]. More recent methods even eschew the prediction of ratings, building instead models that can predict item rankings, from which top-n recommendation lists can be taken and presented to the user. Bayesian Personalized Ranking is one such method [RFGST09].

Limitations of collaborative methods

- *Sparsity:* In any recommender system, the number of observed ratings is usually very small compared to the number of user-item pairs. It is challenging to find similar users, similar items or other patterns that are non-spurious in such sparse data [AT05, Agg16].
- *Cold-start items and users:* Collaborative systems rely solely on ratings to

make recommendations. Therefore, until a new item is rated by a substantial number of users, and a new user rates a substantial number of items, this kind of recommender system would not be able to recommend the new item or recommend to the new user [RRS11, Agg16].

- *Popularity bias:* Collaborative methods recommend items based on ratings and hence on prior consumption. Hence, they tend not to recommend products with limited historical data. For example, in the movie domain, there may be many movies that have been rated by only few people and these movies would be recommended very rarely, even if those few users gave high ratings to them. Therefore, recommenders can create a rich-get-richer effect for popular items. This is known as concentration- or popularity-bias [AT14].
- *Shilling attacks:* In collaborative settings, malicious users and/or competing vendors may insert fake profiles in an effort to effect the rating predictions for their own advantages [GKBP14].

2.2.2 Content-based methods

Content-based methods try to predict the utility of items for an active user based on item descriptions and her past preferences. In content-based systems, there are choices to be made about the following [LDGS11]: how items are represented, how user preferences are modeled, and how items are matched with the user preferences.

Item representation: Items are typically represented in one of the following ways:

- *Structured:* There is a finite and typically small set of attributes, each with a corresponding domain of values. Each item is described by a set of attribute-value pairs. For example, in a restaurant recommender, the attributes might include the type of cuisine and the average cost of a main course.
- *Unstructured:* Each item is described by free text as, for example, in the case of news articles. However, the text is often processed to obtain, for example, a set of the most important terms or vectors of weights (such as TF-IDF weights) for terms. This processing presents challenges in handling

polysemy (one word having several meanings) and synonymy (several words having the same meaning), among others. In some cases, we may be given sets of terms directly, without having to recover them from text. For example, we might be given keywords that describe a movie or tags that users have assigned to images. Media other than text (such as images, audio or video) can also be thought of as unstructured but again it is common to try to derive simpler representations from the raw media.

- *Semi-structured:* Each item is described by a mixture of structured and unstructured information. For example a movie has attributes such as its running time but also free text that gives a synopsis of the plot.

User profile: A user's profile represents her preferences. Preferences may be persistent, indicating a user's long-term tastes and interests, or ephemeral, reflecting her transient requirements [SHY04]. The user profile is most typically a representation of the persistent preferences.

A user profile might simply consist of a history of the user's interactions with the recommender system, e.g. items she has viewed or purchased, or ratings that she has given to the items. Sometimes instead a user profile comprises features that describe the user's tastes and interests. These might have been obtained from a form on sign-up to the system or they might be aggregated from the items the user interacts with.

Filtering technique: A filtering technique suggests relevant items from a set of candidate items. These techniques are also split into memory-based and model-based. The former employ similarity measures to match the representations of candidate items against the profile [PB07, RRS11]. The latter learn from the profile a model that can predict item relevance. One example of the latter are Naïve Bayes recommenders [PB07, RRS11].

Limitations of content-based methods The main advantage of content-based methods is that they are easy to explain at feature-level. Their most significant challenges include the following:

- *They are limited by the degree of content analysis:* Their ability to discriminate between items depends on the granularity of the item representations. If two different items are represented by the same set of features, they are indistinguishable and equally likely to be recommended. But it might be that the representations were derived in one case from a well-written article and in the other case from a badly written one, if they happen to use the same terms [AT05, Agg16].

- *Over-specialization:* Content-based methods tend to recommend items that are similar to items the user has liked in the past. For instance, if a user has never seen or rated a “reality-show”, a content-based system will probably not recommend one to her. Thus, content-based recommendations are often among the least serendipitous recommendations [AT05, PB07, Agg16].
- *Cold-start user:* The user needs to build enough of a profile (e.g. she must rate a sufficient number of items) before a content-based recommender system can really understand her preferences and present her with reliable recommendations. A new user, with an immature profile, is less likely to get accurate recommendations [AT05, RRS11].

2.2.3 Hybrid methods

Hybrid recommender systems combine one or more techniques, e.g. collaborative and content-based, to improve performance [Bur02]. The intention is that the combination reduces the limitations of the individual recommendation algorithms. For instance, collaborative recommenders are susceptible to the item cold-start problem (where recommendations cannot be generated for new items) [LKH14, FTBE+16]. By including content-based methods in a hybrid, a new item that has few ratings may still be recommended in a content-based way on the basis of its description. There are many ways to combine different recommender techniques in a hybrid system, and they are surveyed in [Bur02].

In this dissertation, we propose a new framework that we call *chain-based recommendation*. It works on content-based principles; it represents items with unstructured descriptions; it requires only implicit user feedback; and it makes use of a coverage-based approach to find the top-n potentially interesting items for the user. We describe our framework in the next section.

2.3 Chain-Based Recommendations

Over the years, the recommendation task has variously been formulated as predicting the ratings for an unseen user-item pair (e.g. [NK11]), learning to rank the *top-k* recommendations for a user (e.g. [RFGST09]), and predicting a link between a user and an item in the (bipartite) user-item interaction graph (e.g. [LC13]). In our chain-based recommendation framework, we reduce the recommendation task to one of constructing a chain in the item-item similarity graph that connects items in the user's observed space of items (e.g. her past preferences) to items in the unobserved space (e.g. candidate items that can be recommended to the user).

2.3.1 Definitions and concepts

A *chain* in chain-based recommendation is a sequence of items $\langle i_1, i_2, \cdots, i_n \rangle$ such that in every neighbouring pair (i_r, i_{r+1}), item i_r reinforces its successor i_{r+1}.

Since chain-based recommendation is a form of content-based recommendation, each item has a description. We assume that this takes the form of a set of features (e.g. a set of keywords or a set of user-assigned tags). However, we also define an alternative item representation that makes use of these features indirectly. In this alternative, an item is represented by its set of neighbours, i.e. by similar items, where similarity is defined in term of the item features used in the other representation. In both cases, we assume a symmetric similarity measure between item descriptions, based on feature/neighbour overlap. Hence, we can define an item-item similarity graph: nodes are items, and there is an edge between a pair of nodes if the similarity of the items represented by those nodes is non-zero.

The problem of constructing a chain can then be thought of as a path search in the item-item similarity graph. In adding the next item to a chain, candidates come from its neighbours in the item-item graph.

The exact details of the chain construction method will depend on the purpose. This will determine where the start node comes from. For example, in Chapter 4, the start node is an item that is a candidate for recommendation (i.e. it belongs to the user's unobserved space); in Chapter 7, it is an item that the user chooses from her observed space. The purpose will also determine how the process terminates. For example, in Chapter 4, we stop adding nodes to a chain when no candidate node improves the chain sufficiently; in Chapter 7, we stop when the user chooses an item to consume (or abandons the recommender).

We formulate the path search problem as a set-cover problem with constraints. Pairs of successive items in a chain must satisfy a local constraint in the form of a 'similarity threshold' (θ) since there is a desire for 'smooth' transitions between consecutive chain members. Additionally, each item in the chain may have to satisfy a global constraint in the form of a threshold on the level of coverage it contributes towards elements of the goal item (referred to as the 'gain threshold' ϵ). The similarity threshold refers to the minimum acceptable similarity between two neighbouring items in the chain. A lower similarity threshold gives a larger set of potential chain members at each step but may lead to less interpretable transitions. The gain threshold refers to the minimum acceptable coverage of the goal item by a chain member. Higher values of the gain threshold impose stricter similarity requirements that leads to shorter chains. Again, the purpose for which the chain is built determines which constraints to apply. For instance, in Chapter 4, we impose both constraints while constructing a chain; in Chapter 7, we omit the global constraint, giving freedom to subsequent chain members.

2.3.2 Types of chains

Let $\mathbb{I}$ be the set of all items. Let's assume a recommender does not recommend items that have already been consumed. We define two disjoint subsets on $\mathbb{I}$. First, are the items P in the user's *profile*. These items form the user's observed space. Since we are assuming unary, implicit feedback, these are items that the user has interacted with and we are inferring (possibly inaccurately) that they are items she likes. Second are the items I, which are candidate items — ones that can be recommended to the user. These items form the user's unobserved space.[1]

Chains are constructed in such a way that they form a path between the two disjoint sets. Given P and I, we realize the notion of chains of items in the following ways:

[1]In the simplest case, $I = \mathbb{I} \setminus P$. However, suppose we also know the items that the user has interacted with but that she does not like. We make no use of these ourselves in this work. But, in that case, the candidates I comprises $\mathbb{I}$ less all the items she has interacted with.

Explanation Chains In this case, chain-based recommendation begins with the goal $i_g \in I$ and works backwards. The other items in the chain are drawn from items in the user's profile P and are intended to support the recommendation (i.e. the goal). In this way, the chain relates the user's past preferences to the candidate item. The strength of their connections is assured by means of the local and global constraints: each chain member shares a part of its description with its predecessor and also contributes to covering the description of the goal. Based on the strength of the chains, top-n chains are chosen to recommend to the user. We distilled this idea into a system which we call 'recommendation-by-explanation' and the chains are designated as 'explanation chains'. We cover the details of this in Part II of the dissertation.

Preference Chains In contrast to the previous case, this type of chain begins with a seed $i_s \in P$ and moves forwards. The other chain members are recommendations that belong to the candidate set I. Instead of automatically generating the chain, at each step the user is recommended a certain number of items and is asked to choose from among them, thus extending the chain. This process is repeated until the user finds her item of interest. We use this idea in a system which we call 'navigation-by-preference' and the chains generated by this system are called 'preference chains'. In navigation-by-preference, the user has control over the direction in which to proceed, how much the chain members have affinity with the seed, and where to stop the chain. In this type of chain, the global constraint is omitted. The details of navigation-by-preference and preference chains are covered in Part III of this dissertation.

2.3.3 Advantages of chains

Chains are personalized paths that connect items in the user's observed space to items in her unobserved space. These chains are found either automatically (in the case of explanation chains) or by taking user feedback into account (in the case of preference chains). Chains are built on the same principles as other content-based recommender systems. However, they have several advantages over classic content-based recommender systems.

- Chain-based systems use coverage-based heuristics. The heuristics penalize chains that cover features of the candidates or the seed that have already been covered. By thus encouraging feature-level uniqueness, they ultimately lead to an increase in the diversity of the recommendation list. We discuss this in detail in Chapter 5.

- In our experiments, we have found a positive correlation between the length of the chain and the surprise of the recommended items. Shorter chains result in more obvious recommendations while longer chains may lead to more surprising recommendations. We discuss this in detail in Chapter 5.
- Chain-based systems can be built on different item representations and can also be extended to collaborative settings, though in the latter case explaining the connections may present some challenges.

This dissertation contains many offline experiments and some user trials. In the next two sections, we summarize some of what is common to these experiments and user trials.

2.4 Overview of Dataset

We used the hetrec2011-movielens-2k dataset[2] but, in place of the tags given in that dataset, we assigned each movie its keywords from IMDb[3]. From the original dataset, only those movies for which IMDb has keyword information are used in our experiments performed. 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.

In our user-trials, users interact with web-based recommender systems. In order to increase the chances of user familiarity with the movies, these trials 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 Chapter 5, we use some sentiment data from user reviews. We postpone a description of this until Chapter 5, where it is needed.

[2]https://grouplens.org/datasets/hetrec-2011/
[3]http://www.imdb.com

2.5 Evaluation Measures

In the offline experiments, for each user u, we generate a list of top-n recommendations, R_u. We evaluate this list for accuracy (against T_u, the set of items in the test set that are known to be relevant for user u) and beyond accuracy-measures. Here we describe the evaluation metrics that we will use in this dissertation. All of these metrics are calculated as an average of all users in the test set (denoted $\mathbb{U}_T$) using definitions given in Section 7 of [KB16].

Precision@n. This is the fraction of relevant items in the recommended list R_u for each test user u.

$$\frac{1}{|\mathbb{U}_T|}\sum_{u\in\mathbb{U}_T}\frac{1}{|R_u|}\,|R_u\cap T_u| \tag{2.2}$$

Diversity (*Div*). This measures the diversity of the recommendation list R_u as the average pairwise distance among its elements. In content-based settings, we calculate the distance between two items (i, j) as the complement of their Jaccard similarity computed on their features $sim(F_i, F_j)$.

$$\frac{1}{|\mathbb{U}_T|}\sum_{u\in\mathbb{U}_T}\frac{1}{|R_u|(|R_u|-1)}\sum_{i\in R_u}\sum_{j\in R_u\setminus i}1-sim(F_i,F_j) \tag{2.3}$$

Surprise (*Sur*). This measures the surprise of a recommended item as the minimum distance between the item and items in the user's profile P_u. This is averaged over the recommended items $i \in R_u$.

$$\frac{1}{|\mathbb{U}_T|}\sum_{u\in\mathbb{U}_T}\frac{1}{|R_u|}\sum_{i\in R_u}\min_{j\in P_u}1-sim(F_i,F_j) \tag{2.4}$$

Novelty (*Nov*). This is based on the fraction of users in the dataset who rated the item i. The logarithm is used to emphasize the novelty of the most rare items.

$$\frac{1}{|\mathbb{U}_T|}\sum_{u\in\mathbb{U}_T}\frac{1}{novelty_{max}\cdot|R_u|}\sum_{i\in R_u}-\log_2\frac{|u\in\mathbb{U},\, r(u,i)\neq 0|}{|\mathbb{U}|} \tag{2.5}$$

Here $novelty_{max} = -\log_2 \frac{1}{|\mathbb{U}_T|}$ is the maximum possible novelty value which is used to normalize the novelty score of each individual item into $[0, 1]$.

Coverage. This is the fraction of the items which are recommended at least once, across all users. Higher values of *coverage* indicate that the algorithm counterbalances the popularity bias by covering a large portion of the catalogue.

$$\frac{|\cup_{u \in \mathbb{U}_T} R_u|}{|\mathbb{I}|} \tag{2.6}$$

There are a few other metrics that are used in the dissertation. But we describe them in the chapters in which they are needed.

2.6 Table of Symbols

In Table 2.1, we list symbols used throughout the dissertation. Each chapter then also introduces chapter-specific notation.

In the next two parts of the dissertation we describe the two forms of chain-based recommendation that we have explored in our work. Specifically, the next part describes *Recommendation-by-Explanation*, which is a novel approach to recommendation and explanation that unifies the process of computing recommendations and generating corresponding explanations.

Table 2.1: Table of the most common symbols used in the dissertation.

Symbol	Description
$\mathbb{U}$	Set of users, $\mathbb{U} = \{u_1, u_2, \cdots, u_m\}$
$\mathbb{I}$	Set of items, $\mathbb{I} = \{i_1, i_2, \cdots, i_l\}$
$r(u,i)$	A rating assigned by user u to item i.
P	A user's profile, $P = \{i \in \mathbb{I} \mid r(u,i) \geq 4\}$. Only where we need to be explicit, do we write P_u
I	Set of candidate items, $I \subseteq \mathbb{I}$. Only where we need to be explicit, do we write I_u
R	Set of recommended items, $R \subseteq I$. Only where we need to be explicit, do we write R_u
$\lvert \cdot \rvert$	Cardinality of a set
n	Number of recommendations required
θ	Similarity threshold
ϵ	Gain threshold
η or α	Balancing factor
F_i	Set of features of an item i
N_i	Candidate neighbours of item i, $N_i = \{j \in I \mid sim(F_i, F_j) > \theta\}$

Part II

Recommendation-by-Explanation

Chapter 3

Recommendation Explanation: State of the Art

Recommender systems provide explanations to help users make better decisions [BM05], increase user trust in the system [SS02], and improve user acceptance of recommendations [HKR00].

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.

In addition to supporting end-users, explanations of recommendations may have a role in issues such as: detecting shilling attacks [GKBP14]; detecting bias and discrimination [LOL$^+$18]; and contesting algorithmic decisions on personal data as allowed for in government legislation such as GDPR[1] [GF17].[2] These concerns related to user safety, system fairness, and government policies implicitly or explicitly demand that choices recommended to the user need to be perceived as reliable, fair, and transparent by those who rely on them. Recommender systems might achieve higher levels of trust and greater acceptance by means of explanations that allow users to understand and scrutinize automated decisions, and to provide feedback to improve the systems [AN18].

[1]https://eugdpr.org/

[2]Some people believe that GDPR gives rise to the *right* to an explanation [GF17], while other people disagree [WMF17].

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) [HKR00]. The level of detail in an explanation may also vary with the level of risk associated with the decision-making process [CP05]. 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 [TM08a].

It is important to understand the goals of an explanation when designing an explanation facility for a recommender system [TM07b, FZ11]. The goals help to determine what information should be given to the user, to what extent the system's logic needs to be revealed, and how the system will be evaluated [NJ17]. We describe these goals in detail in the next section.

3.1 Goals of Explanations of Recommendations

Tintarev & Masthoff [TM07b] identified seven explanation goals (*effectiveness, persuasion, scrutability, etc.*) applicable to single item recommendations. These goals are often presented in a list as if they were independent. However, they are related in different ways [TM07a]. Nunes & Jannach [NJ17] argued that the choice and the design of an explanation mechanism must be guided by the overarching goals that should be achieved with the explanations. They distinguish between three levels of explanation objectives: i) stakeholder goals, ii) user-perceived quality factors, and iii) explanation purposes. The service provider has an objective (e.g. long-term customer relations) which should be related to a user-perceived service quality factors (e.g. confidence), and that is ultimately implemented with a corresponding explanation purpose (e.g. effectiveness).

Beside end-users and service-providers, bodies who audit or who police intelligent systems also have (or will have in the future) a perspective on investigating algorithmic decision-making [SHKL14]. Automated decision making systems are (or will be) audited to detect bias and other unwanted algorithmic behaviours. Often this will be done from the outside without the need to know about, or the ability to inspect, specific design or implementation details. This is performed by means of scripts and tools emulating real data and real users in the auditing process. Explanations may play a role in scaling auditing to an ever larger number of algorithms [MZR18]. This calls for explainable systems that take into account fidelity (e.g. [RB18]), safety (e.g. [RMTM10]) and fairness (e.g. [AN18]).

In this section, we describe Tintarev & Masthoff's well-accepted seven explanation goals.

Transparency. An explanation may reveal how a recommender system has reached its conclusions. Sinha & Swearingen highlighted the importance of transparency in recommender systems [SS02]. They found in a user-study that users are not just looking for blind recommendations from a system, but are also looking for a justification of the system's choice. Thus, users like and feel more confident about recommendations that they perceive as transparent.

Gedikli et al. [GJG14] distinguish between *objective* and *perceived* transparency. While objective transparency is the extent to which the explanation reveals the true logic of underlying recommender (also referred to as fidelity [KSB$^+$13]), perceived transparency is based on a user's perception about how well the system explains its inner logic. Of course, this perception may be mistaken when there is a perception of high transparency but the explanation is not faithful to the underlying logic. Where explanations are designed for perceived transparency with no or little regard to objective transparency, they are sometimes referred to as *justifications* instead of explanations [VSR09].

Scrutability. Explanations may help users to modify or correct the assumptions (or steps) that the system made while collecting and interpreting information about the user. Scrutability can give more control to the user by allowing her to modify incorrect assumptions.

There have been efforts that let users take control over recommendations including: enabling them to modify data which is stored about them [CK02]; allowing them to adjust the weights of the items they liked and friends they have [KBOK12]; critiquing recommended items by rating them [WWP$^+$13]; and allowing them to specify which features are relevant or changing their weights [LAW14].

Scrutability is closely related to transparency but deserves to be distinguished from it. A system can offer transparency but transparency alone does not allow users to influence the system's reasoning.

Trust. An explanation may help increase users' confidence in the recommender system's competence. A study of users' trust suggests that users are more likely to report an intention to return to systems which they find trustworthy [CP05].

Trust in the recommender system is found to be positively correlated with its competence [MLKR03]. A study shows that transparency and the possibility of interacting with the system can increase user trust [SS02]. Also, in the case of a bad recommendation, an explanation though does not compensate; however may reduce its effect on user trust by unfolding the reasoning behind and may allow user to interact with the system to prevent it from occurring again [TM15].

In contrast, there are also some cases where transparency and trust are not found to be related [CER+08].

Satisfaction. Explanations may increase enjoyment in the use of a recommender system and ultimately acceptance of the overall system. The effect of other explanation goals on satisfaction is reported in [GJG14]. They showed that perceived transparency had a positive effect on overall satisfaction with the explanation interface but they did not find any effect of efficiency or effectiveness on satisfaction.

Nunes & Jannach [NJ17] split satisfaction into ease of use, enjoyment, and usefulness. Explanations may be provided to increase the perceived usefulness of the system; however, this may require large cognitive effort on the part of the user.

Effectiveness. Effective explanations help users to evaluate the quality of a recommendation with respect to their own preferences. Thus, effective explanations may help users accept relevant items and discard irrelevant ones [SRS+13, NDZ18].

Apart from accurate decision making, effective explanations can also be used to introduce a new domain or product range to novice users and help them understand the range of alternatives [CP05].

Efficiency. Explanations may help users make decisions more quickly. For example, explanations of trade-offs may help users to more rapidly understand the set of alternatives and to reach a quicker decision [McS05, MRMS04, CW17].

Efficiency is often used in the evaluation of conversational recommender systems where users repeatedly interact with the system and refine their preferences. These systems are called efficient if the task completion time or the number of cycles needed to find the item of interest are low [TGL04].

Persuasiveness. Explanations may change a user's behaviour, i.e. they may increase the likelihood that a user accepts one of the given recommendations [HKR00]. Influencing user behaviour has been investigated in low- [CLA$^+$03, SRS$^+$13] and high-cost domains [TM08b].

Cosley et al. [CLA$^+$03] show that users can be manipulated to rate close to the predicted value whether the prediction is accurate or not. It has also been found that users trust a recommendation more if the system displays high confidence in its recommendation, even steering the user towards less relevant items [SRS$^+$13].

In the next section, we describe different ways of characterizing explanations of recommendations.

3.2 Characterizing Explanations of Recommendations

Explanations of recommendations can be characterized in a variety of different ways such as: the type of knowledge they use (e.g. user demographics, item descriptions, etc.); their fidelity to the recommender (i.e. white-box vs. black-box); and their role in producing recommendations. We explore these characterizations in detail below with the help of twenty-six systems proposed in the literature from the year 2000 to the year 2018. These systems span the breadth and depth of research into explanations in recommender systems.

3.2.1 Based on their type of knowledge

Explanations of recommendations often relate the recommended item to the user through *intermediary entities*. These intermediary entities may be other users, other items, item features, or context; see Figure 3.1. This way of thinking about explanations was introduced in [VSR09] but we have extended it to also include 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* [BM05, PSM12].

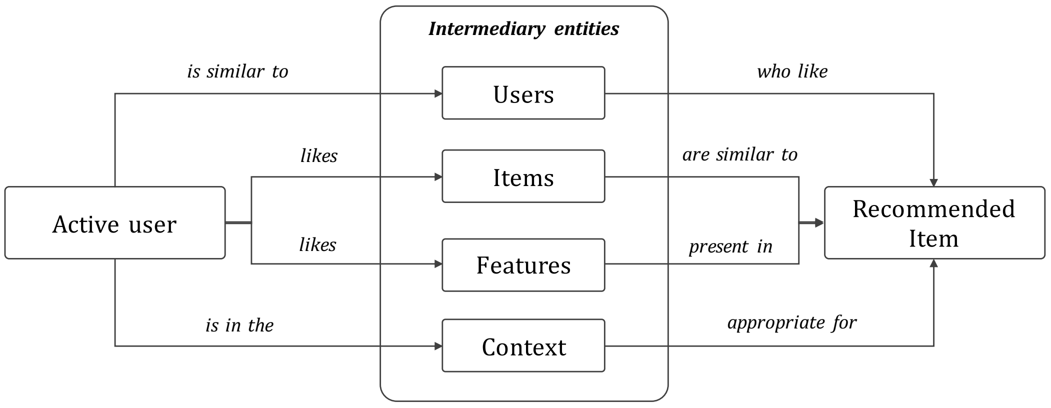


Figure 3.1: Intermediary entities relate the active user to the recommended item (adapted from [VSR09]).

3.2.1.1 User-based

User-based explanations say that an item is recommended because users who are similar to the active user liked it. More specifically, at one end, these users are related to the active user as they have similar tastes; at the other end, these users have rated the recommended item positively. For example, social networks such as Facebook[3] often use user-based explanations when recommending to the active user a person to add as a friend or to follow. The explanations show a list of the active user's friends who are already a friend of the recommended person; see the examples in Figure 3.2. 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 them, 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 readily comprehended [BD14]. Even in professional networks such as LinkedIn[4], user-based explanations sometimes take the form of a statistic that summarizes the neighbours' behaviour or tastes. Figure 3.2 shows examples in which recommendations to join a group or follow an organization take the form of a count of connections.

In [HKR00], twenty-one different explanations were proposed. In a user survey, the most persuasive of these was an explanation of user-based collaborative filtering that comprised a histogram of the active user's neighbours' ratings of the recommended item: it showed the number of neighbours who had assigned high ratings (4s and 5s), neutral ratings (3s) and low ratings (1s and 2s).

[3]https://www.facebook.com
[4]https://www.linkedin.com/

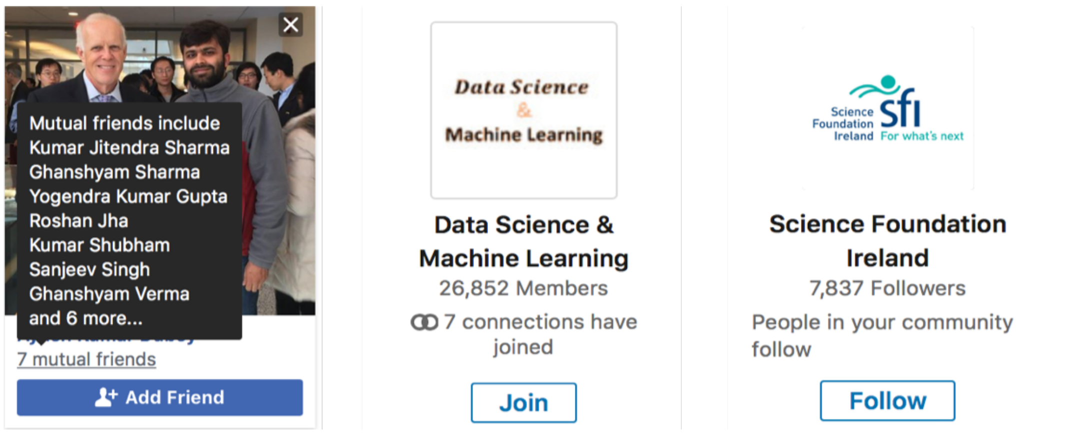


Figure 3.2: Examples of *user-based* explanations in social networks

Another example, this time from personalized information retrieval, is in [CS05], where the search histories of a community of online users are used to generate explanations in the form of a relevance percentage, a set of related queries, and recency information. These highlight how other users have interacted with search results under similar search conditions in the past. It was found that such explanations help searchers to better understand the relevance of search results.

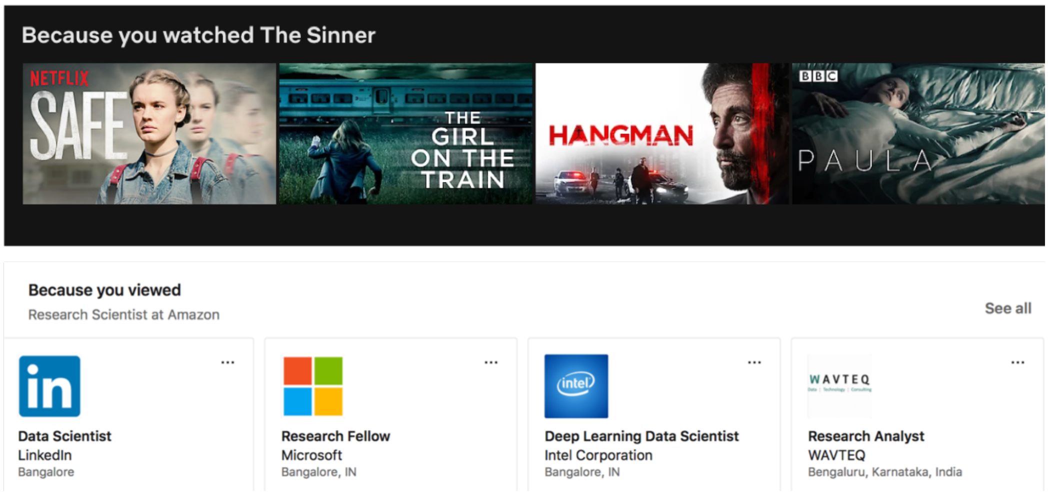
3.2.1.2 Item-based

Item-based explanations say that the item is being recommended because the user liked similar items, i.e. the similar items that the user liked before relate the recommended item to the user. Figure 3.3 shows examples of item-based explanations from Netflix[5] and LinkedIn. Famously, Amazon[6] also uses item-based explanations for its recommendations [LSY03].

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 [BM05]. Accordingly, in [BD14], 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 [TM07a]. This problem can be resolved by hybrid explanations (Subsection 3.2.1.5) by showing the features that the explicands have in

[5]https://www.netflix.com/
[6]https://www.amazon.com/


Because you watched The Sinner

Because you viewed
Research Scientist at Amazon
See all

Data Scientist
LinkedIn
Bangalore

Research Fellow
Microsoft
Bangalore, IN

Deep Learning Data Scientist
Intel Corporation
Bangalore, IN

Research Analyst
WAVTEQ
Bengaluru, Karnataka, India

Figure 3.3: Examples of *item-based* explanations.

common with recommended item.

3.2.1.3 Feature-based

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[7] 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* [TM15]. In the literature, features take numerous different forms, e.g. attribute-value pairs, item content, user-generated tags, opinions mined from user reviews, and linked data. We will look at each in turn.

Attribute-value pairs. One common way to represent an item is in terms of values for predefined attributes. For example, Tintarev generated explanations for a recommended movie based on the user's most preferred actor, genres and director [Tin07]. In another approach [SCLDL12], each item is represented as an entity and described by its attributes retrieved from the Freebase semantic knowledge base. A dataset was obtained from a user study in which users were asked to rate self-selected movies and to annotate the movies' attribute-value pairs: 'good' for attribute-values that predisposed them towards the movie and 'bad' for those that predisposed them against the movie. Finally, explanations of recommended movies were provided similarly, i.e. by showing 'good' and 'bad'

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

annotations against their attribute-values.

Although attribute-value pairs have the potential to explain recommended items, they have problems too, which include: designing the set of attributes and the permissible values requires domain experts' knowledge; they might not be sufficiently descriptive; and they may not accommodate changing needs [Shi07].

Item content. Items such as news, books, articles or blogs take the form of textual content. This content can be mined for keywords and the keywords can be used to describe the items. Bilgic & Mooney showed that these keywords can be used in explanations that help users to make more accurate decisions [BM05].

Explanations that are based on item content have some limitations, which include: there are items which may not have easily available textual content (e.g. music or images); and keywords extracted from textual descriptions give a quite low-level representation, rather than conveying high-level meta-data that represent the quality of the item [VSR09].

User-generated tags. Many platforms allow users to assign tags to items. Tags can be characterized as objective (where they convey factual information about an item) or subjective (where they express the user's opinion about an item). Tags have a possible advantage over other forms of item description: they are created, maintained, and applied by users themselves without any dependency on domain experts [GH06]. Hence, they may capture what it is that users care about. The downside, however, is that, being the responsibility of end-users, tags may be of lower quality or more inconsistently applied than domain expert descriptions.

A user may assign a tag for the purpose of item identification, organization, semantic search and so on. But tags may also be useful in recommendation and in explanation of recommendations. There have, for example, been several efforts to exploit user-generated tags to explain recommendations [VSR09, GGJ11, GJG14]. Vig et al. [VSR09] proposed tag-based explanations, which they call 'tagsplanations'. Their system explained a recommended item (in their case, a movie) by means of a filtered and ranked list of tags with their relevance scores. For filtering and ranking, they use two measures:: tag preference and tag relevance. Tag preference is user-specific; it measures the relationship between a tag and a user as the degree to which user likes that tag (based on frequency of use). Tag relevance is item-specific; it captures the relationship between a tag and an item as the degree to which a tag describes an item and not other items.

Tag clouds are a popular way to visualize an item's tags. A tag cloud is a set of tags where properties related to their appearance such as font size, font weight, colour, and position within the cloud may have a semantic meaning such as relevance, sentiment score, or the frequency with which users assign that tag to that item [GGJ11].

Gedikli et al. [GGJ11] used tag clouds to explain movie recommendations. They experimented with non-personalized tag clouds, where font size showed the tag's importance for the recommended movie, which they measured by the number of times the tag was assigned to the movie. They also experimented with personalized tag clouds, where font colour was based on the average rating for that movie by the user's nearest-neighbours: blue in the case where on average the neighbours like the item, red where they dislike it, and black where they are neutral. Gedikli et al. conducted a user trial that showed the advantages of their tag clouds over more traditional keyword-based explanations [GJG14].

User reviews. Online marketplaces (such as Amazon and AirBnB) allow their customers to share their experiences or opinions about a product or service in the form of reviews. User reviews benefit both businesses and customers because of the valuable information that they contain. Hu et al. [HLZ08] argue that customers can identify, understand and respond to favourable opinions. Users are often influenced by reviews when making decisions, e.g. customers are more likely to buy items with more reviews [PLH07].

User reviews are typically expressed in free text. They are often accompanied by a score (e.g. a count of votes) indicating their helpfulness. There may be information about the context in which the user consumed the item, e.g. in the case of a hotel, the review may be accompanied by information about whether the user was traveling for business or pleasure, whether she was traveling alone or with others, etc.

User reviews can be mined for features of the item and sentiment towards the item or its features, e.g. [HL04, HEZ$^+$12, DSOS13]. Recently, an amount of work has proposed to use features and sentiments mined from user reviews in recommendation explanations [CHT16, CODL18, BZIL18, CW17, MLRS15].

In [CHT16], for example, Chang et al. propose a process that combines crowd-sourcing and computation to produce personalized natural language explanations. They first generate tag clusters for a movie and select a representative tag for each cluster and leverage crowd-wisdom to refine the output. For these representative tags which serve as topical dimensions, they ask crowdworkers to write explanations based on quotes mined from online movie reviews. Finally, they compute users interest in the topical dimension based on her activities and present users explanation that are most interesting to them. Also, Costa et al. [CODL18] propose a method for the automatic generation of personalized natural language explanations. They train a character-level attention-enhanced long short-term memory (LSTM) model on user-rated review texts for items. In the offline experiments, by means of natural language processing metrics, they find that the quality of the generated texts is close to that of text written by real users.

User reviews have also been processed to extract so-called aspects: relevant factors of interest. Baral et al. [BZIL18] proposed an approach in which they extracted aspects from user reviews posted on location-based social networks. They modeled users and point-of-interests (POIs) as review-aspects, and user preferences on aspects as a location-aspect bipartite graph. They used these bipartite graphs to generate the explanation by extracting the most dense sub-graphs. They found that their explainable recommendation model outperformed the ones without explanations.

In domains in which items have a high purchase cost (e.g. laptops) or a high consumption cost (e.g. tours of historic sites), trade-off explanations have been found to be effective. For example, Mohammad et al. describe a case-based recommender system that exploits opinions mined from user-generated reviews in the domain of holiday tour planning [MLRS15]. They produce explanations that shows an item's pros and cons in terms of the item's features. The features are found in the reviews using one of two patterns such as a noun followed by a noun or an adjective followed by a noun or a noun on its own (after eliminating nouns that are rarely associated with sentiment words in reviews). In this case, the trade-offs are for a single item. But trade-offs can also explain sets of recommendations by showing how members of the set compare to each other. Chen & Wang [CW17], for example, create such explanations in the domain of digital cameras and laptops. They produced trade-off-oriented explanations by incorporating products' specifications and feature sentiments extracted from product reviews.

Linked data. Linked data is inter-connected data that is published in a way that complies with the Linked Data principles. For example, DBpedia[8] is the result of an effort to extract structured content from the information available in various Wikipedia projects. DBpedia is one part of the Linked Open Data (LOD) cloud, which contains publicly-available linked data resources.

Musto et al. present EXPLOD [MNL+16], in which data available in the LOD cloud (especially DBpedia) is used to fill templates to generate natural language explanations of recommendations. EXPLOD exploits the properties encoded in DBpedia to link the user's previously liked items with the recommended ones. Musto et al. extend their idea by incorporating properties not directly connected to the item to be recommended in order to build explanations containing more interesting and unexpected patterns [MNL+19].

3.2.1.4 Context-based

It has been argued that contextual information, such as time, location, weather, or companions, can influence how a person perceives a product or service. Hence, recommender systems should take context into account when providing recommendations [AMRT11]. For example, it is important to know the user's companions before recommending a movie to watch.

Most recently, contextual information has been used in explanations too. In [SAN+18], Sato et al. proposed explanations that include contexts suitable for consuming the recommended item. For this, Sato et al. prepared visit logs by asking crowdworkers to describe the restaurants they have visited and the context of their visits. Then, a model was trained on these visit logs to produce recommendations as well as context-based explanations: the item is presented to the user as a recommendation and the context is used as a corresponding explanation.

3.2.1.5 Hybrid

Recall that in user-based explanations, we show similar users who like the recommended item. However, it may not be clear why the users in the explanation are similar to the active user. Assuming that similarity between the active user and the users in the explanation is based on having co-rated items and, indeed,

[8]https://wiki.dbpedia.org/about

Table 3.1: Summary of past research based on the type of knowledge they use in their explanations. In the table, ‘Features’ is denoted as Feats. and ‘Context’ as Ctxt.

System	Users	Items	Feats.	Ctxt.	Style
Herlocker et al. [HKR00]	✓				neighbours
Coyle & Smyth [CS05]	✓				neighbours
Bilgic & Mooney [BM05]		✓	✓		hybrid
Cramer et al. [CER$^+$08]			✓		feats. list
Symeonidis et al. [SNM08]		✓	✓		hybrid
Vig et al. [VSR09]			✓		tag-list
Yu et al. [YLAY09]	✓				neighbours
Zanker & Ninaus [ZN10]			✓		nat. lang.
Gedikli et al. [GGJ11]			✓		tag-cloud
Scheel et al. [SCLDL12]			✓		trade-offs
Blanco et al. [BCL$^+$12]			✓		feats. list
Chen et al. [CHL13]			✓		tag-cloud
Rossetti et al. [RSZ13]			✓		topics
Bridge & Dunleavy [BD14]		✓			item rules
Zhang et al. [ZLZ$^+$14]			✓		word-cloud
Cleger et al. [CFLH14]	✓				neighbours
Muhammad et al. [MLRS15]			✓		trade-offs
Musto et al. [MNL$^+$16]			✓		nat. lang.
Chang et al. [CHT16]			✓		nat. lang.
Chen & Wang [CW17]			✓		trade-offs
Costa et al. [CODL18]			✓		nat. lang.
Baral et al. [BZIL18]			✓	✓	graph
Sato et al. [SAN$^+$18]				✓	context
Rana & Bridge [RB18]		✓	✓		chains
Naveed et al. [NDZ18]	✓		✓	✓	arguments
Wang et al. [WCY$^+$18]			✓		nat. lang.

similar rating values for these co-rated items, then explanations might additionally show these co-rated items. Similarly, in item-based explanations, we show items that the active user liked and that are similar to the recommended item. This time, let’s assume that similarity between the items in the explanation and the recommended item is based on sharing features. Then, explanations might additionally show these features.

As defined in [PSM12], any combination of the aforementioned types of explanation (user-based, item-based, feature-based and context-based) are called ‘hybrid’ explanations. Our own Explanation Chains are hybrids: they are item-based but they expose item relationships through the shared features.

3.2.1.6 Summary

In this section, we have reviewed work on explanations in recommender systems based on the type of knowledge that they contain. This can be similar users, similar items, item features, context or combinations of these. We summarize the papers in Table 3.1. We have found that most explanations so far are provided using item features but this encompasses a wide range of types of knowledge including user-generated tags, features extracted from user reviews and linked data.

These explanations are presented in various styles: from simple list of neighbours through to special structures such as tag-clouds, graphs, and chains.

In the next section we characterize explanations in terms of their fidelity.

3.2.2 Based on their fidelity

In Artificial Intelligence in general, explanations are sometimes categorized as white-box (also sometimes called model-based) or black-box (sometimes called model-agnostic) [HKR00, FZ11]. The distinction typically reflects on the fidelity of the explanations to the underlying reasoning done by the AI system.

Black-box explanations raise the issue of fidelity [KSB$^+$13] (also called objective transparency [GJG14]): 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 [KSB$^+$13]. Arguably, black-box systems cannot achieve fidelity. (The LIME system [RSG16], which is model-agnostic, claims to achieve 'local

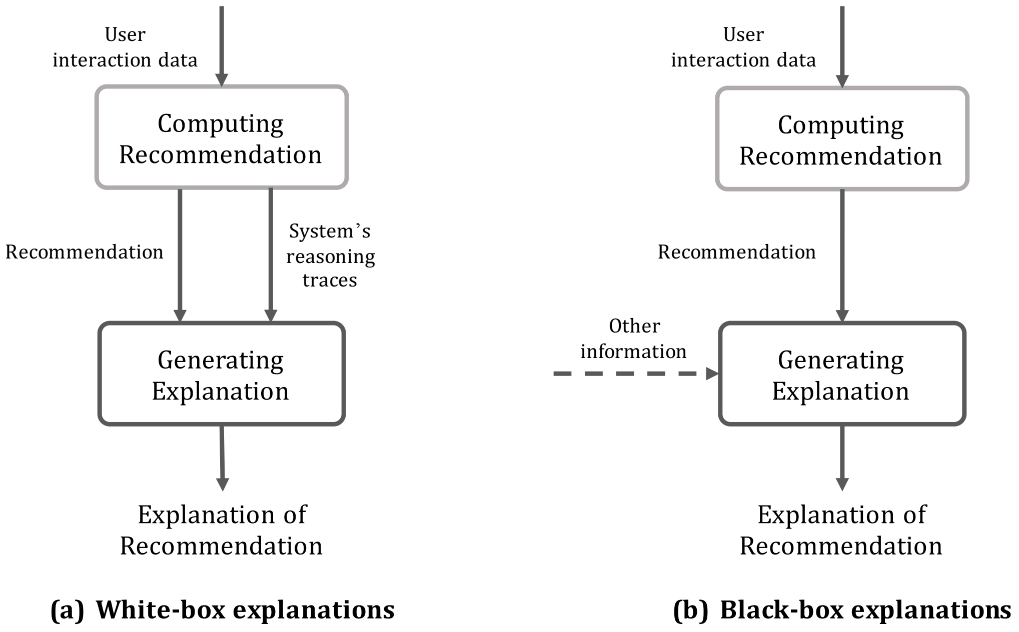


Figure 3.4: Generation of *white-box* and *black-box* explanations.

fidelity', but this is not the same concept.) Recommendation-by-Explanation seeks to achieve quite high fidelity since, in *r-by-e*, explanation is intrinsic to recommendation.

We now describe white-box and black-box explanations in further detail.

3.2.2.1 White-box explanations

White-box explanations are built from traces of the system's reasoning; see Figure 3.4a. 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 [HKR00] is a white-box explanation. Different recommender systems employ different types of knowledge and inference methods: simple content-based systems calculate similarities between the descriptions of candidate items and the descriptions of items the user has liked in the past; nearest-neighbour collaborative systems exploit similarities between users and items; and knowledge-based systems match item features against the user's requirements. However, the conceptual models employed in these systems are simple and can be easily conveyed to the user by means of white-box explanations.

Simple white-box explanations are provided, for example, in an artwork recommender [CER+08, CES+08], where features of the recommended artwork are listed that are in common to the artworks previously liked by the user. In [BM05], information about the neighbourhood (items that the user has liked before and that are similar to the recommended item, and the keywords that these items share with the recommended item) are presented in different explanation styles to explain the recommendations generated by their hybrid content-boosted collaborative system. In a similar vein, Symeonidis et al. [SNM08, SNM09] proposed a novel approach to provide relevant and justifiable recommendations. They create biclusters — group of users exhibiting similar rating behaviour (to a target user) on group of items. They also assign weights to the target user's feature profile (item features that better describe the target user tastes and distinguish her from others). They find nearest neighbours using items and features common between the user's profile and the biclusters. Finally, they identify items in the neighbouring biclusters which are highly preferred by other users and contain significant features from user's feature profile. In their approach, they use the common features to justify each of the recommended item.

For other kinds of recommender systems, white-box explanations are much more challenging. Explaining the recommendations of latent factor models, for example, is not straightforward because it is difficult to attach semantic meaning to the latent factors. However, there have been efforts to learn these models not just from user-item ratings but also jointly from item feature knowledge. The hope is that the latent factors will reflect patterns in the item feature data to improve their explainability. Chen et al. [WHL11] proposed a four-order tensor that models the quaternary relationship among users, items, tags and ratings, and builds a unified framework for user, item, tag, and rating prediction. In order to provide intuitive explanations, they represent users, tags, and items in a common latent space. In this latent space, they find the tag whose representation is closest to the representation of the active user and the recommended item in order to use it as a dominant feature for explanation. In this way, a tag-cloud is generated to explain the recommendation [CHL13]. Zhang et al. [ZLZ+14] improve the explainability of matrix factorization by incorporating both user-feature and item-feature relations along with user-item ratings into a new unified hybrid matrix factorization framework. They extracted explicit product features and user opinions from user reviews and used them while explaining a product.

As we have seen, white-box methods are aware of the underlying recommender system's conceptual model and they exploit information to generate explanations. As they use the same information to generate explanations which they use for computing recommendations, they are sound (to some degree) and can achieve a high degree of fidelity.

3.2.2.2 Black-box explanations

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 [RSG16]. In general, since they make no use of traces of the system's reasoning, black-box explanations must make use of other sources of information that were not used in the decision-making; see Figure 3.4. In [RSZ13], 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. Black-box explanations are model-agnostic, since the explanation component is independent of the reasoning component — in our case, the recommender. Black-box models have the additional advantage that they protect intellectual property: a white-box model, by contrast, discloses something of the algorithm by which decisions are made. But the main motivation for resorting to black-box explanations is when the reasoning model is too complex or unintuitive to be explained itself, e.g. as is probably the case with many deep-learning-based models. The separation between the recommender and the explanation component affords greater freedom in designing explanations as they are not restricted by the recommendation model [VSR09].

Vig et al.'s tagsplanations, whch we described earlier, were in fact an example of black-box explanations [VSR09]. They explained the recommended item (movie) by means of a list of tags with their corresponding relevance scores. But their recommender was an item-based nearest-neighbours recommender: it made no use of the tags and no trace of its reasoning was passed to the explanations component. This made the explanations model-agnostic: the item-based recommender could be replaced with any other, e.g. a matrix factorization model, and the system could still produce tagsplanatons for its recommendations. This means low, or no, fidelity. Indeed, Vig et al. recognized this and referred to their tagsplanations as justifications, not explanations, to emphasize their lack of fidelity.

Similarly, Zanker & Ninaus [ZN10] proposed a framework for generating explanations that exploits domain knowledge to show why a recommended item matches the user's preferences. They employed a variant of Predicate-based Finite State Automata (PFSA) to represent an explanation model such that transitions are represented by constraints formulating restrictions on a finite set of variables. Their knowledge-based explanation model is independent of the operation of their recommender system and therefore generates black-box explanations. The explanation frameworks reported in [MNL+16, MNL+19] that exploit the information available in the Linked Open Data (LOD) cloud (DBpedia) are also examples of black-box explanations as they use linked data to generate explanations which is not used while computing the recommendations.

Recently, a reinforcement learning framework for explaining recommendations has been proposed [WCY+18]. The framework is model-agnostic: it can be used to explain any recommendation model including complex deep-learning-based ones. Given user-item interaction data and the recommendation model, their approach represents each item in terms of interpretable components (e.g. sentences extracted from user reviews). From a set of all interpretable components, it aims to select those components that are concise (e.g. shorter sentences), consistent with the user's rating, and sufficient for predicting the user's preference for the recommended item.

3.2.2.3 Summary

In this section, we have reviewed past work on explanations in recommender systems based on their fidelity. Table 3.2 summarizes this work. Fourteen of the systems that we have reviewed are *white-box* and the other twelve are *black-box*. Content-based systems are often easy to explain but are found to be less accurate than collaborative systems. Therefore, in many *white-box* systems, hybridization of content-based with collaborative systems or joint modeling of item features with user-item ratings were proposed to achieve higher accuracy without compromising their interpretability.

3.2.3 Based on their role in producing recommendations

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.

Table 3.2: Summary of past research based on fidelity. In this table, K-type means knowledge type, UCF is user-based collaborative filtering, CB is a content-based system, CBCF is content-boosted collaborative filtering, LFM is a latent factor model, CBR is case-based reasoning, FM is a factorization machine, PBRS is a preference-based recommender system, and ANY means that the explanation component is independent of the recommender system.

System	White-box	Black-box	K-type	Model
Herlocker et al. [HKR00]	✓		users	UCF
Coyle & Smyth [CS05]	✓		users	UCF
Bilgic & Mooney [BM05]	✓		hybrid	CBCF
Cramer et al. [CER$^+$08]	✓		features	CB
Symeonidis et al. [SNM08]	✓		features	CBCF
Vig et al. [VSR09]		✓	features	ANY
Yu et al. [YLAY09]	✓		users	UCF
Zanker & Ninaus [ZN10]		✓	features	ANY
Gedikli et al. [GGJ11]		✓	features	ANY
Scheel et al. [SCLDL12]	✓		features	CBCF
Blanco et al. [BCL$^+$12]		✓	features	ANY
Chen et al. [CHL13]	✓		features	LFM
Rossetti et al. [RSZ13]		✓	features	LFM
Bridge & Dunleavy [BD14]		✓	items	UCF
Zhang et al. [ZLZ$^+$14]	✓		features	LFM
Cleger et al. [CFLH14]	✓		users	UCF
Muhammad et al. [MLRS15]	✓		features	CBR
Musto et al. [MNL$^+$16]		✓	features	ANY
Chang et al. [CHT16]		✓	features	ANY
Chen & Wang [CW17]	✓		features	PBRS
Costa et al. [CODL18]		✓	features	ANY
Baral et al. [BZIL18]		✓	hybrid	FM
Sato et al. [SAN$^+$18]	✓		hybrid	CB
Rana & Bridge [RB18]	✓		hybrid	CB
Naveed et al. [NDZ18]		✓	hybrid	UCF
Wang et al. [WCY$^+$18]		✓	features	ANY

3.2.3.1 Classical approaches

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 3.5. Almost all of the systems that we have cited in the previous subsections work in this way.

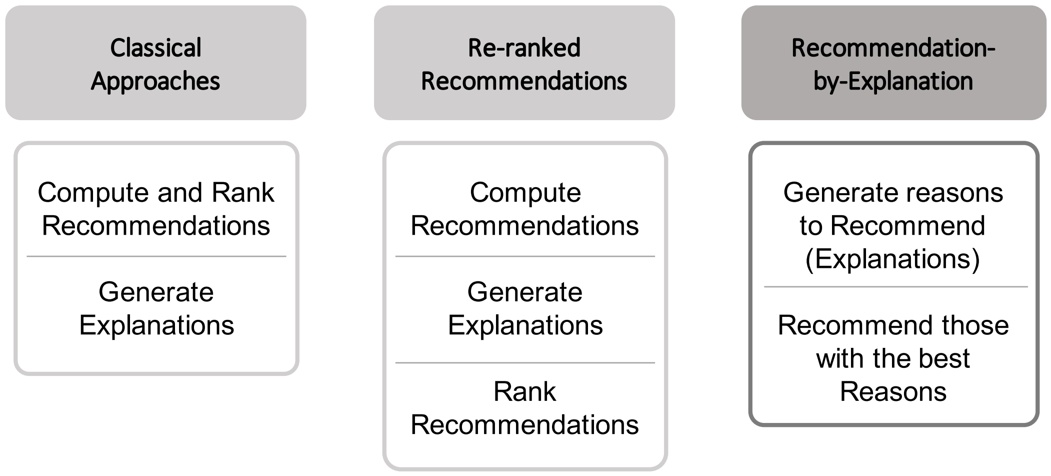

Classical Approaches
Compute and Rank Recommendations
Generate Explanations
Re-ranked Recommendations
Compute Recommendations
Generate Explanations
Rank Recommendations
Recommendation-by-Explanation
Generate reasons to Recommend (Explanations)
Recommend those with the best Reasons

Figure 3.5: Role of explanations in producing recommendations.

3.2.3.2 Re-ranked recommendations

There have been a few efforts that modify the classical approach a little. These are shown in the middle of Figure 3.5. 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 [MLRS15, MLS16].

Yu et al. [YLAY09] propose a similar approach. Their goal is to diversify recommendations. This is almost always done in recommender systems by re-ranking the recommendations so that the top-n are more dissimilar to each other. In Yu et al.'s work the re-ranking seeks to minimize the overlap between the recommendations' explanations.

One of the goals of providing explanations is that by analyzing why the system recommends a particular item or proposes a certain rating, the user might also consider the quality of the recommendation and, if appropriate, propose a change to the predicted value [CTFLH12]. On this basis, Cleger et al. [CFLH14] propose their idea to learn from explanations. For user-user collaborative filtering, they gathered knowledge from explanations showing neighbours' opinions on user's previously rated items (e.g. from histograms showing how neighbours have rated an item) in the form of a set of numerical features, e.g. predicted rating, number of items supporting the recommendations, entropy, etc. A regression model is learned on this knowledge and, where appropriate, use this model to predict the rating prediction error and change the predicted rating for a target item.

Table 3.3: Summary of past research based on the role of explanations in the recommendation process.

System	Classical	Re-ranked	r-by-e
Herlocker et al. [HKR00]	✓		
Coyle & Smyth [CS05]	✓		
Bilgic & Mooney [BM05]	✓		
Cramer et al. [CER$^+$08]	✓		
Symeonidis et al. [SNM08]	✓		
Vig et al. [VSR09]	✓		
Yu et al. [YLAY09]		✓	
Zanker & Ninaus [ZN10]	✓		
Gedikli et al. [GGJ11]	✓		
Scheel et al. [SCLDL12]	✓		
Blanco et al. [BCL$^+$12]	✓		
Chen et al. [CHL13]	✓		
Rossetti et al. [RSZ13]	✓		
Bridge & Dunleavy [BD14]	✓		
Zhang et al. [ZLZ$^+$14]	✓		
Cleger et al. [CFLH14]		✓	
Muhammad et al. [MLRS15]		✓	
Musto et al. [MNL$^+$16]	✓		
Chang et al. [CHT16]	✓		
Chen & Wang [CW17]	✓		
Costa et al. [CODL18]	✓		
Baral et al. [BZIL18]	✓		
Sato et al. [SAN$^+$18]	✓		
Rana & Bridge [RB18]			✓
Naveed et al. [NDZ18]	✓		
Wang et al. [WCY$^+$18]	✓		

3.2.3.3 Recommendation-by-Explanation

Our new approach, Recommendation-by-Explanation, is shown rightmost in Figure 3.5. 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 next two chapters of this dissertation.

3.2.3.4 Summary

Table 3.3 summarizes the systems we have reviewed in this section. We can see that most of these systems follow the classical approach, where recommendation and explanation are separate and sequential processes. Only a handful of systems are different.

In the next section, we describe how to evaluate explanations of recommendations.

3.3 Evaluating Explanations of Recommendations

As we have seen, explanations of recommendations vary in many different ways. Defining generic criteria to judge the quality of an explanation is difficult, if not impossible. Furthermore, there is no consensus on what constitutes a good explanation [NMLDL12]. While the literature offers a few principles, e.g. succinctness and authenticity [FZ11, Fri04], most definitions or methods for measuring explanation quality should take into account the goal of the explanations (Section 3.1). Evaluating explanations for their persuasiveness may not be the same as evaluating them for their effectiveness in decision-support.

Explanations are user-centric. Offline experiments are of very limited use; for example, we can measure the size of a system's explanation (e.g. how many items or features they contain). But to evaluate the subjective perception of the users and their impact on user behaviour really requires either user trials or online evaluation with a deployed system.

Table 3.4 summarizes the evaluation of recommendation explanation. In this table, we included only those systems where it was explicitly mentioned that the proposed system was evaluated for a particular explanation goal or to find an effect of one goal on another.

Bilgic & Mooney introduce a protocol for evaluating explanation effectiveness [BM05]. This protocol is now well-established, having been adopted by many others, e.g. [BD14, RB18, GJG14]. In this protocol, a user is initially asked to rate a recommendation in the case where she is given only the explanation and not the identity of the item. This is called the explanation-rating. The user is asked later to re-rate the recommended item in the case where she is not

Table 3.4: Summary of past research based on how explanations are evaluated. In this table, Transparency is denoted as Trns, Scrutability as Scrt, Trust as Trst, Satisfaction as Sats, Effectiveness as Efec, Efficiency as Effi and Persuasiveness as Pers.

System	Trns	Scrt	Trst	Sats	Efec	Effi	Pers
Herlocker et al. [HKR00]				✓			✓
Sinha & Swearingen [SS02]	✓		✓	✓			
Coyle & Smyth [CS05]	✓				✓		
Thompson et al. [TGL04]						✓	
Bilgic & Mooney [BM05]				✓	✓		✓
McSherry [McS05]					✓		
Chen & Pu [CP05]	✓		✓				
Pu & Chen [PC07]			✓		✓	✓	
Tintarev & Masthoff [TM08b]				✓	✓		✓
Cramer et al. [CES^{+}08]	✓		✓	✓	✓		
Vig et al. [VSR09]	✓				✓		
Gedikli et al. [GGJ11]	✓			✓	✓	✓	✓
Shani et al. [SRS^{+}13]					✓		✓
Kulesza et al. [KSB^{+}13]	✓				✓		
Bridge & Dunleavy [BD14]					✓		
Muhammad et al. [MLRS15]					✓		
Musto et al. 2016 [MNL^{+}16]	✓		✓		✓		
Chang et al. [CHT16]			✓	✓	✓		
Chen & Wang [CW17]	✓			✓	✓	✓	
Costa et al. [CODL18]					✓		
Sato et al. [SAN^{+}18]					✓		✓
Rana & Bridge [RB18]					✓		✓
Naveed et al. [NDZ18]				✓	✓		
Balog et al. [BRA19]	✓	✓					

given the explanation but she is given information about the item, including its identity. This is called the actual-rating. Effective explanations are ones where the explanation-ratings are close to the actual-ratings.

Persuasiveness is also often measured using a difference in the explanation-rating and actual-rating. When the explanation-rating is greater than the corresponding actual-rating, it indicates that the explanation made the user to overrate the recommendation and thus it is characterized as a persuasive explanation [CLA^{+}03, TM08b].

A recent study on transparent, scrutable and explainable user models measures the scrutability in terms of impact of user's feedback on the performance of the recommender system [BRA19]. However, not all of the seven explanation goals have standard evaluation protocols. More generally, explanations are evaluated by directly asking users about their perception of explanation quality. For instance, Vig et al. [VSR09] determined perceived effectiveness by asking user about the helpfulness of the explanation in identifying the recommended item. Transparency of an explanation is usually evaluated by asking question about the perceived understanding of the recommendation to the participants of a user-study [VSR09, CES$^+$08, GGJ11, KSB$^+$13]. Satisfaction can be evaluated informally by asking users about enjoyment, usefulness, and ease of use [GGJ11, MNL$^+$16, NDZ18, WCY$^+$18]. In the literature, trust is sometimes related to an intention to return to the system to use it again in the future [CP05, PC07, CHT16].

In our work, we are interested in effectiveness, which is why one of our user trials uses the re-rating protocol.

3.4 Conclusion

In this chapter, we have reviewed explanations in recommendations, from their goals to their evaluation. We characterized them based on various criteria including what knowledge they contain, their fidelity, and the role they play in the recommendation process.

From this literature review, we have identified a lack of work in which explanations are intrinsic to recommendations, and therefore issues with explanation fidelity.

Classic content-based systems are often inherently interpretable while collaborative systems attain higher accuracy but lower interpretability. In general, the two types of systems are integrated to overcome their shortcomings, i.e. hybrid systems have been proposed where collaborative recommendations are explained in a content-based manner. As we will see in the following chapters, by only using a content-based approach and unifying the processes of computing recommendations and generating corresponding explanations, our new approach attains greater accuracy than some other content-based systems, and also a higher degree of fidelity, diversity, and surprise.

Chapter 4

Explanation Chains

4.1 Introduction

Recommender systems provide explanations to the user to achieve one or more of the seven goals that we discussed in the previous chapter. Conventionally, computing recommendations and generating corresponding explanations are two separate, sequential processes. This separation is one possible cause of low fidelity between the explanations and the operation of the recommender.

In this chapter, we present Recommendation-by-Explanation (*r-by-e*), in which explanation is intrinsic to recommendation. In *r-by-e*, the system constructs a reason, or explanation, for recommending each candidate item; then it recommends those candidates that have the best explanations. *r-by-e*'s explanations take the form of what we call *Explanation Chains*. Figure 4.1 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*.

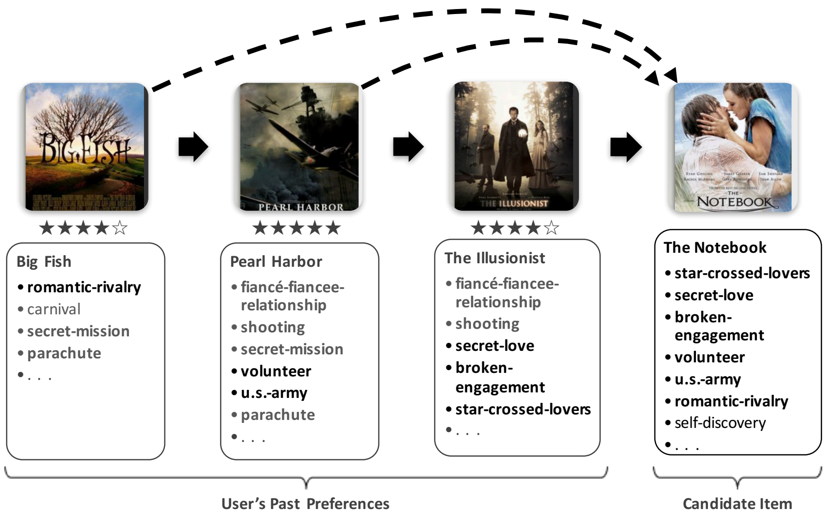


Figure 4.1: An explanation chain.

We believe that *r-by-e* has the following characteristics:

- *Unified approach:* It is a unified approach that combines the processes of computing recommendations and generating corresponding explanations.
- *Fidelity:* By unifying recommendation and explanation, there is a guaranteed level of fidelity between explanations and the operation of the recommender.
- *Diversity and serendipity:* The approach uses hyperparameters whose values can be adjusted to loosen or tighten constraints between items in the chain and thus increase or decrease the diversity and serendipity of the recommendations.

4.2 Recommendation by Explanation

Recommendation by Explanation (*r-by-e*) is a novel approach that unifies recommendation and explanation: it computes recommendations by generating and ranking corresponding personalized explanations in the form of Explanation Chains. Here we explain in detail how *r-by-e* constructs the chains for candidate items and selects the n that it will recommend.

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

```
Input: $n$, number of recommendations
       $I$, set of candidate items
       $P$, user's profile
       $\theta$, similarity threshold
       $\epsilon$, marginal gain threshold
Output: $L^*$, ranked list of top-$n$ Explanation Chains.
1: function Recommend($n, I, P, \theta, \epsilon$)
2:     $L \leftarrow [\ ]$
3:     for each $i \in I$ do
4:         $C \leftarrow$ GenerateChain($i, P, \theta, \epsilon$)
5:         if $|C| > 0$ then
6:             append $\langle C, i \rangle$ to $L$
7:     return SelectChains($L, n$)
```

4.2.1 *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. Another way to obtain candidates is to take the top-n' recommendations of another recommender system ($n' >> n$); in this case, *r-by-e* will filter and re-rank the other system's recommendations. In our experiments later in this chapter, 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, $I = \{i \in I \setminus P \mid \text{sim}(F_i, F_p) > \theta, \exists\, p \in P\}$. Here F_i and F_p denote the features of items i and p, and we define sim as Jaccard similarity.

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.

4.2.2 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
       $\theta$, similarity threshold
       $\epsilon$, marginal gain threshold
Output: $C$, an Explanation Chain $C$ for candidate $i$.
 1: function GenerateChain($i, P, \theta, \epsilon$)
 2:     $C \leftarrow [\,]$
 3:     $sum\_ovrlps = 0$
 4:     $j \leftarrow i$
 5:     while True do
 6:         $J \leftarrow \{p \in P \setminus C \mid \text{sim}(F_j, F_p) > \theta \wedge \text{ovrlp}(p, i, C) > \epsilon\}$
 7:         if $|J| = 0$ then
 8:             return $C$
 9:         $j = \arg\max_{p \in J} \text{ovrlp}(p, i, C)$
10:         append $j$ to $C$
11:         $sum\_ovrlps = sum\_ovrlps + \text{ovrlp}(j, i, C)$
```

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:

$$\text{ovrlp}(p, i, C) = \frac{|(F_p \setminus \text{covered}(i, C)) \cap F_i|}{|F_i|} + \frac{|(F_p \setminus \text{covered}(i, C)) \cap F_i|}{|F_p|} \tag{4.1}$$

Here again 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. $\text{covered}(i, C) = \bigcup_{j \in C} F_j \cap F_i$. Then the first term in the definition of $\text{ovrlp}(p, i, C)$ measures p's coverage of those features of i that are not yet covered by the chain. The second term in the definition measures the same but with respect to the size of F_p rather than the size of F_i and therefore assures p's fitness to explain the candidate by penalizing items that have high coverage simply by virtue of having more features.

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 $|L| \leq n$ then
 3:         sort $L$ using score
 4:         return $L$
 5:     $L^* \leftarrow [\,]$
 6:     while $|L^*| < n$ do
 7:         $\langle C, i\rangle^* = \arg\max_{\langle C,i\rangle \in L} \text{score}(\langle C, i\rangle, L^*)$
 8:         append $\langle C, i\rangle^*$ to $L^*$
 9:         remove $\langle C, i\rangle^*$ from $L$
10:     return $L^*$
```

4.2.3 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:

$$\text{score}(\langle C, i\rangle, L^*) = \frac{\mathit{sum\_ovrlps}}{|C| + 1} + \frac{\left|C \setminus \bigcup_{j \in L^*} j\right|}{|C| + 1} \tag{4.2}$$

Here, the first term is the sum of the overlaps of the items in the chain divided by its length plus 1 (so as to include candidate item i). It gives higher scores to chains that do a better job of covering the features of their candidate item. The second term gives higher scores to a chain if its members are not also members of already-selected chains and hence encourages the final recommendation list to cover as many items in the user's profile as possible. (Note that the second term is about coverage of *items* that appear in already-selected chains, not their features.) We have found this term to have the effect of diversifying the recommendation list.

4.3 Offline Evaluation

We ran an offline experiment to evaluate *r-by-e*'s performance. We compare it with a content-based recommender, which works as follows. Given candidate item i, it finds the items in P whose similarity to i exceeds θ; it takes the k of these neighbours with highest similarity; it scores the candidate by taking a similarity-weighted average of their ratings. It recommends the n candidates with highest scores.

The main difference between the two recommenders is that the content-based recommender relies on similarity relationships between members of P and the candidate item, whereas *r-by-e*, by requiring consecutive members of chains to be similar to each other, additionally takes into account similarity relationships between members of P themselves. We wanted this experiment to reveal the effect of this difference. So we otherwise tried to ensure that the two systems were as similar as possible. They both use the same item features (keywords, see below), and they both use the same similarity measure (Jaccard). For the content-based system, we chose to set k in a dynamic fashion, as follows. If, for candidate item i, *r-by-e* generates a chain of length $|C|$, then the content-based system uses $k = |C|$ when it scores that candidate item. It follows that k is set dynamically: different candidates may have different values for k. We designate this system *CB*-$|C|$, using a name that emphasizes that, dynamically, $k = |C|$.[1]

4.3.1 Experiment settings

We used the hetrec2011-movielens-2k dataset augmented by keywords from IMDb as described in section 2.4. 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 experimented with five different values $[0.03, 0.06, 0.09, 0.12, 0.15]$ of each of the similarity threshold (θ) and the marginal gain threshold (ϵ), giving 25 configurations of *r-by-e*. When choosing the best configuration, there is an issue about what to optimize. It makes sense, for example, to choose the configuration that

[1]Although we only report results for the dynamic version of this content-based recommender, we did perform preliminary experiments with a version whose value for k was fixed by a hyperparameter optimization process. In these preliminary experiments, the two versions of this content-based recommender had quite similar results.

optimizes precision on the validation sets. But it could be interesting to choose configurations that optimize other criteria. Therefore, we also show results for the case where we choose the configuration that optimizes for diversity on the validation set, and for the case where we choose the configuration that maximizes the percentage of explanations of size 2–4.

We also suspect that users will find an explanation to be easily intelligible only if it is fairly small (chains or sets of neighbours of size 2–4 items), so we recorded the percentage of explanations that were of this size.

Table 4.1: Results of the Offline Experiment. All of the *r-by-e* results are statistically significant with respect to *CB-|C|* (t-test with $p < 0.05$) except the one shown in italics.

Recommender	θ & ϵ optimized for	Precision	Diversity	Surprise	Novelty	Coverage	% of explanations of size 2–4
r-by-e	Precision	0.1089	0.9352	0.7834	0.3771	0.1358	0.2868
CB-|C|		0.0701	0.9091	0.8135	0.4179	0.1509	0.2719
r-by-e	Diversity	0.0370	0.9760	0.8886	*0.5404*	0.6014	0.3156
CB-|C|		0.0087	0.9736	0.9370	0.5365	0.8700	0.6534
r-by-e	% of explanations of size 2–4	0.0677	0.9626	0.8635	0.4756	0.3541	0.7598
CB-|C|		0.0097	0.9711	0.9336	0.5119	0.8976	0.7506

4.3.2 Experiment results

We evaluated the performance of *r-by-e* against *CB*-$|C|$ using evaluation measures described in Section 2.5. Table 4.1 summarizes the results.

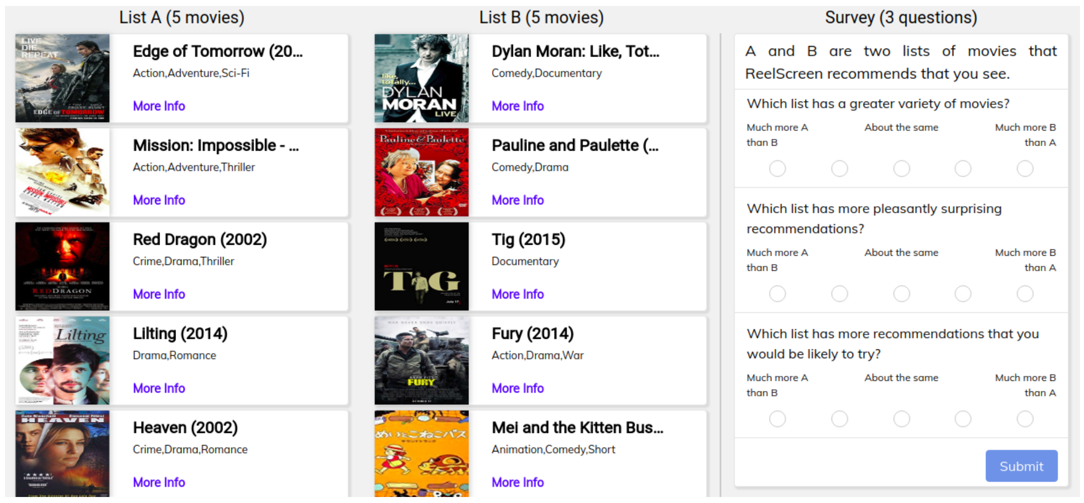
For the most part differences in the results for *r-by-e* and *CB*-$|C|$ are small but, since standard deviations are low, in all but one case they are statistically significant. In two cases, differences are larger: *r-by-e* has better precision and *CB*-$|C|$ has better catalogue coverage. It is noteworthy that *r-by-e* can produce more accurate recommendations without sacrificing diversity and surprise.

We will discuss the results further for each of the different ways of optimizing the hyperparameters:

- *Optimizing hyperparameters for precision:* In this setting, *r-by-e* performs better in terms of precision, diversity, and % of explanations of size 2 – 4, but it has lower values for surprise, novelty, and coverage. This shows that, in this setting, items recommended by *r-by-e* are more relevant and diverse, but are popular and close to the user-profile.
- *Optimizing hyperparameters for diversity*: In this setting, *r-by-e* attains four times higher precision without compromising diversity. *r-by-e*'s recommendations have a higher score for novelty but this is not statistically significant. Over 65% of *CB*-$|C|$'s explanations contain 2–4 items, but most of *r-by-e*'s chains are longer.
- *Optimizing hyperparameters for % of explanations of size 2–4:* In this setting, for both recommenders, explanations are mostly of manageable size (2–4 items). While *CB*-$|C|$'s recommendation cover 90% of the catalogue, only 1% of them are relevant; they are, however, more diverse, novel, and surprising. On the other hand, *r-by-e* gives around seven times more relevant recommendations with nearly similar diversity by covering only 35% of the catalogue.

4.4 User Trials

We also built a web-based system in order to conduct user trials, again comparing *r-by-e* with *CB*-$|C|$ using the hyperparameter values (θ and ϵ) that optimized the percentage of explanations of size 2–4. *r-by-e* is, above all, a recommender and so we designed one trial to measure recommendation quality as well as a trial


List A (5 movies)

Edge of Tomorrow (20...
Action,Adventure,Sci-Fi
More Info

Mission: Impossible - ...
Action,Adventure,Thriller
More Info

Red Dragon (2002)
Crime,Drama,Thriller
More Info

Lilting (2014)
Drama,Romance
More Info

Heaven (2002)
Crime,Drama,Romance
More Info

List B (5 movies)

Dylan Moran: Like, Tot...
Comedy,Documentary
More Info

Pauline and Paulette (...
Comedy,Drama
More Info

Tig (2015)
Documentary
More Info

Fury (2014)
Action,Drama,War
More Info

Mei and the Kitten Bus...
Animation,Comedy,Short
More Info

Survey (3 questions)

A and B are two lists of movies that ReelScreen recommends that you see.

Which list has a greater variety of movies?
Much more A than B — About the same — Much more B than A

Which list has more pleasantly surprising recommendations?
Much more A than B — About the same — Much more B than A

Which list has more recommendations that you would be likely to try?
Much more A than B — About the same — Much more B than A

Submit

Figure 4.2: A screenshot showing top-5 recommendations from the two recommenders and survey questions.

to measure explanation quality. In total, 190 people attempted the trials. The majority of them were undergraduate and postgraduate students recruited online from universities in India and Ireland. We did not collect any demographic data, but it is most likely that they were predominantly young, male Computer Science students. They were not rewarded for participation in any way. To increase the chances of user familiarity with the movies, the web-based recommenders use only movies released between the years 2000 and 2011 inclusive.: 1851 ($\approx 30\%$) of the 5992 movies in the dataset used in the offline experiment.

Each participant begins by creating a user profile containing at least 30 movies. The instructions were that the movies should be the ones the user likes. The user interface offers both a scrollable grid of movies and a search box to enable her to find these movies.

We assigned half the participants to the recommendation trial and the other half to the explanation trial. Of the 190, only 115 completed all parts of the trial to which they were assigned.

4.4.1 Recommendation trial

RQ: Does *r-by-e* generate more diverse, serendipitous, and relevant recommendations than CB-$|C|$?

4.4.1.1 Experiment settings

The recommendation trial is a within-subjects trial: users see two lists of recommendations, one list from *r-by-e* and the other from *CB*-$|C|$ and they answer questions that compare the quality of the two lists [EHWK14]; see Figure 4.2. Lists have length 5 and are sorted in decreasing order of recommender scores.

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*.

Participants were required to answer three questions:

- 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.

4.4.1.2 Experiment results

Sixty-six participants completed this trial. Table 4.2 summarizes their responses.

- *Diversity question:* 42.4% of participants found *r-by-e* recommendations to be much more or more diverse than *CB*-$|C|$ recommendations, 21.2% found the recommendation lists to be equally diverse, leaving 36.4% finding *CB*-$|C|$ to be much more or more diverse.

Table 4.2: Results of the Recommendation Trial.

User's opinion	Diversity	Serendipity	Satisfaction
Much more *r-by-e*	14	15	28
More *r-by-e*	14	11	13
About the same	14	23	8
More *CB*-$\|C\|$	11	10	7
Much more *CB*-$\|C\|$	13	7	10

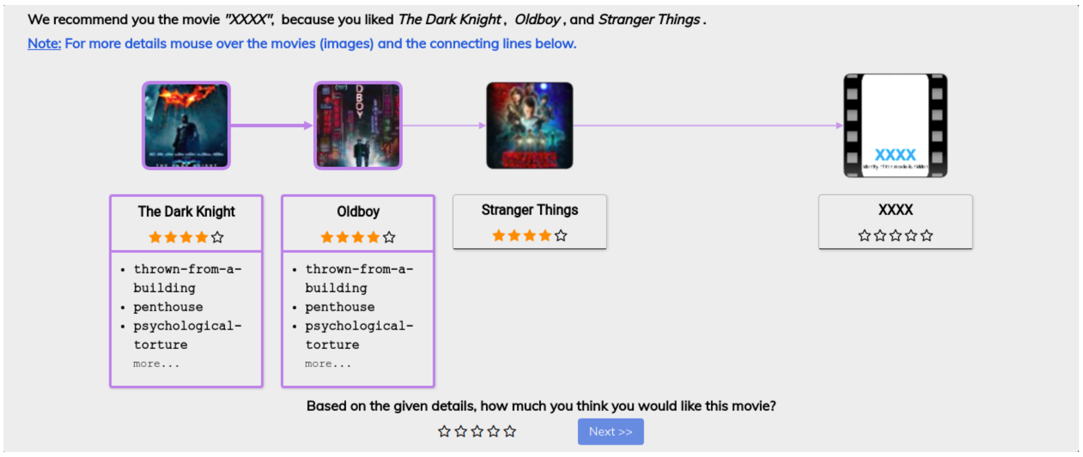
- *Serendipity question:* 39.4% of participants found *r-by-e* recommendations to be much more or more pleasantly surprising, 34.8% found the recommendation lists to be equally surprising, leaving 25.8% finding *CB*-$|C|$ to be much more or more surprising.
- *Satisfaction question:* 62.1% of participants found *r-by-e* recommendations to be ones they would be much more or more likely to try, 12.1% found the recommendations to be equally worthy of trying, leaving 25.8% 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.)

4.4.2 Explanation trial

RQ: Does *r-by-e* generate more effective explanations than *CB*-$|C|$?

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 [BM05, GJG14]. A user is initially asked to rate a recommendation in the case where she is given only the explanation and not the identity of the movie. This is called the *explanation-rating.* The user is asked later to re-rate the recommended item in the case where she is 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: the explanation allowed the user to predict how much she would like the item. Effective explanations will be ones for which (a) μ_d (the mean difference between explanation-ratings and



We recommend you the movie *"XXXX"*, because you liked *The Dark Knight*, *Oldboy*, and *Stranger Things*.

Note: For more details mouse over the movies (images) and the connecting lines below.

The Dark Knight ★★★★☆
- thrown-from-a-building
- penthouse
- psychological-torture

more...

Oldboy ★★★★☆
- thrown-from-a-building
- penthouse
- psychological-torture

more...

Stranger Things ★★★★☆

XXXX ☆☆☆☆☆

Based on the given details, how much you think you would like this movie?

☆☆☆☆☆ Next >>

Figure 4.3: 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 keywords that these two movies have in common.

corresponding actual-ratings) is close to zero; (b) σ_d (their standard deviation) is small; and (c) r (their Pearson correlation) is highest.

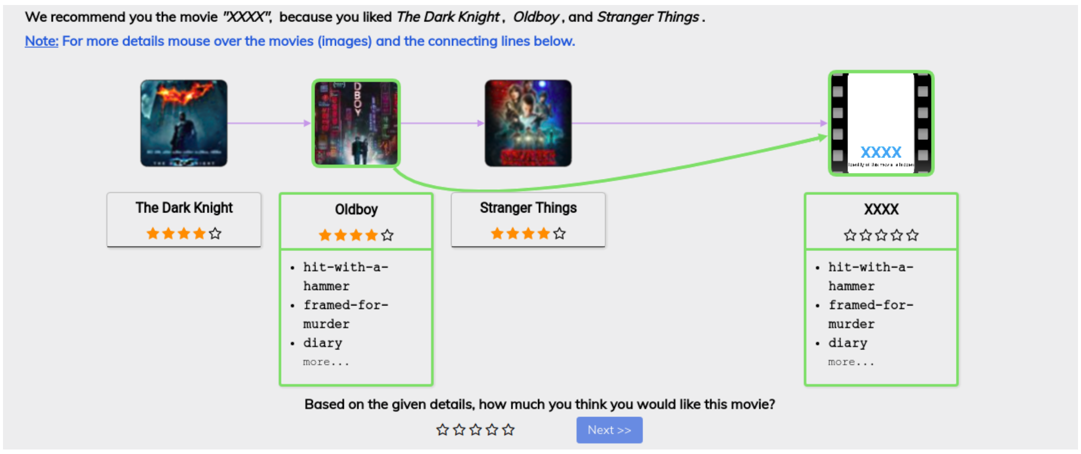
4.4.2.1 Experiment settings

In our Explanation Trial, we used *r-by-e* to generate the top-5 recommendations for the user. Each of these, of course, came with an explanation in the form of Explanation Chain, C. For the *same* movies, we then generated the explanations that the *CB*-$|C|$ system would have produced had it made these recommendations: the $k = |C|$ most similar movies in the user's profile. If the set of movies in *r-by-e*'s chain and *CB*-$|C|$'s neighbours were identical, we replaced the recommendation by the next best recommendation from *r-by-e*'s top-10 recommendations. (In contrast to the Recommendation Trail, in this trial, where we were not able to make 5 recommendations from the top-10, we did not discard the user's responses: we are comparing the effectiveness of pairs of corresponding explanations not, for example, the diversity of lists of recommendations.)

For n recommendations, we have $2n$ explanations to show to the user: two of each kind. We show them to the user in a random order and with the identity of the recommended movie redacted (shown as "XXXX").

Explanation Chains were displayed in the fashion shown in Figures 4.3 and 4.4: arrows connect a movie to its successor in the chain.

CB-$|C|$'s explanations (sets of neighbours, rather than chains) were displayed in the fashion shown in Figure 4.5: arrows connect each movie to the recommended


We recommend you the movie *"XXXX"*, because you liked *The Dark Knight*, *Oldboy*, and *Stranger Things*.

Note: For more details mouse over the movies (images) and the connecting lines below.

The Dark Knight ★★★★☆

Oldboy ★★★★☆
- hit-with-a-hammer
- framed-for-murder
- diary

more...

Stranger Things ★★★★☆

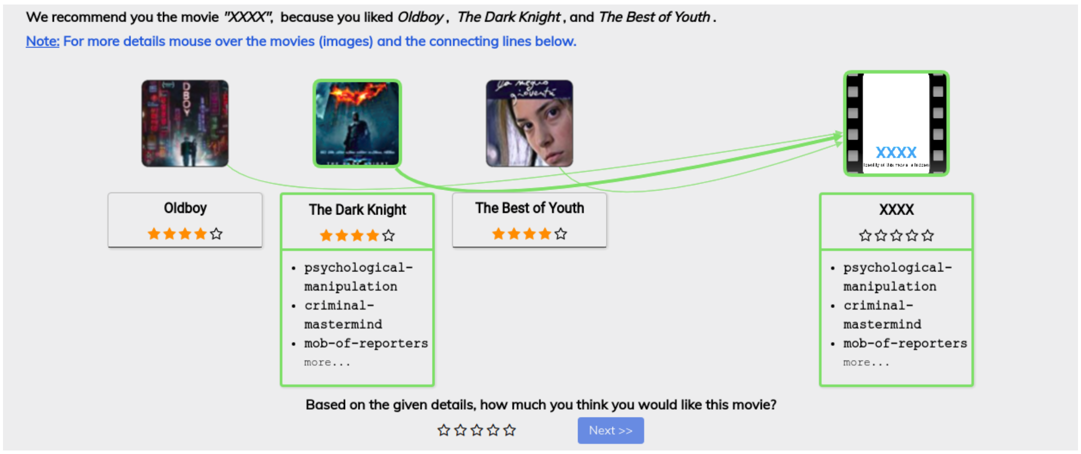
XXXX ☆☆☆☆☆
- hit-with-a-hammer
- framed-for-murder
- diary

more...

Based on the given details, how much you think you would like this movie?

☆☆☆☆☆ Next >>

Figure 4.4: A screenshot of an explanation chain. The user has moused over the icon for the second movie, which causes the system to display an arrow between that movie and the recommended movie and to bring up boxes of keywords that these two movies have in common.


We recommend you the movie *"XXXX"*, because you liked *Oldboy*, *The Dark Knight*, and *The Best of Youth*.

Note: For more details mouse over the movies (images) and the connecting lines below.

Oldboy ★★★★☆

The Dark Knight ★★★★☆
- psychological-manipulation
- criminal-mastermind
- mob-of-reporters

more...

The Best of Youth ★★★★☆

XXXX ☆☆☆☆☆
- psychological-manipulation
- criminal-mastermind
- mob-of-reporters

more...

Based on the given details, how much you think you would like this movie?

☆☆☆☆☆ Next >>

Figure 4.5: A screenshot of a *CB-*$|C|$ explanation. The user has moused over the icon for the second 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 keywords that these two movies have in common.

movie.

In both cases, the user can mouse over parts of the explanation, which causes the system to display keywords that movies have in common (see the captions of the Figures). A maximum of three keywords is displayed in any box, and they are selected by their *TF-IDF* scores.

As can be seen at the foot of Figures 4.3, 4.4 and 4.5, 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.

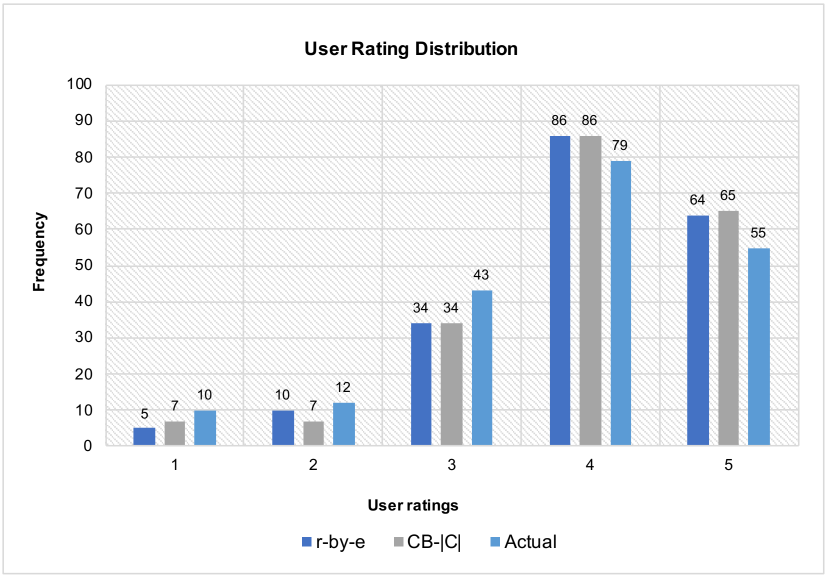


Figure 4.6: Ratings from the Explanation Trial.

Table 4.3: Ratings from the Explanation Trial.

Rating type	μ	σ	r
Actual	3.7889	1.0711	–
r-by-e	3.9749	0.9610	0.4855
CB-$\lvert C \rvert$	3.9799	0.9794	0.2367

After the user has given these $2n$ ratings, the system then shows her 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 the user for a rating (the so-called actual-rating) to indicate how much she thinks she will like the movie.

Note that, although the user has rated the same movie three times, nothing in the on-screen instructions makes this apparent.

4.4.2.2 Experiment results

Forty-nine 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 597 ratings, this being three ratings for 199 recommended movies. (As we mentioned above, we did not always have 5 recommendations per user, e.g. where explanations contained identical movies).

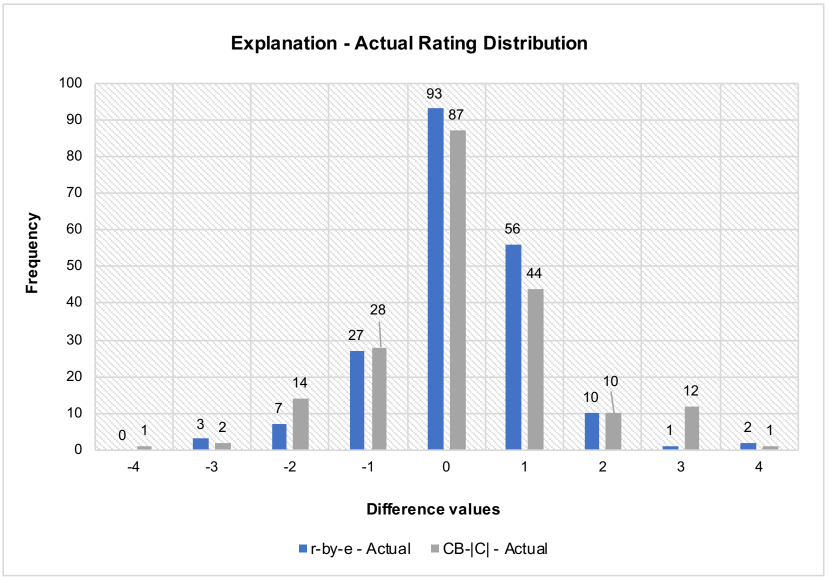


Figure 4.7: Differences in ratings from the Explanation Trial.

Table 4.4: Differences in ratings from the Explanation Trial.

Explanation type	μ_d	σ_d	95% Conf. Int.
r-by-e	0.1859	1.0350	(0.0412, 0.3306)
CB-$\|C\|$	0.1910	1.2688	(0.0136, 0.3683)

Figure 4.6 shows the distribution of the users' ratings; Table 4.3 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 identity. For the differences between explanation-ratings and actual-ratings, Figure 4.7 shows the distribution of values and Table 4.4 gives summary statistics.

The mean difference between *r-by-e* ratings and actual ratings is 0.1859; for *CB*-$|C|$, it is 0.1910. 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 statistically significant; ii) the differences between *CB*-$|C|$ratings and actual-ratings are also 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 4.3 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.

4.5 Conclusion

Recommendation-by-Explanation (*r-by-e*) unifies recommendation and explanation. It computes explanations first and then recommends the items with the best explanations. Its explanations take the form of Explanation Chains, which are sequences of items from the user's profile. There are local relationships between consecutive items in the chain: they have some similarity to each other. There are also global relationships between items in the chain and the recommended item: the items are chosen in an effort to cover the features of the recommended item. The items to be recommended are selected based both on this feature coverage relationship and the degree of coverage of items in the user's profile.

This chapter presented experiments to evaluate *r-by-e*. An offline trial shows the approach to have better precision than a closely-related content-based recommender, while remaining competitive on measures of diversity and serendipity.

We use a web-based system to conduct user trials. The Recommendation Trial shows that *r-by-e* produces recommendations that are apparently more diverse and serendipitous than those of a content-based recommender (although not statistically significantly so) and with statistically significantly higher levels of satisfaction. The Explanation Trial is a re-rating task for measuring explanation effectiveness. Users rate an item given only an explanation (with its identity hidden) and later re-rate when given the identity without the explanation. The correlation between these pairs of ratings is much greater in the case of *r-by-e* explanations.

In the next chapter, we extend *r-by-e*. We give ways of generating chains that use different item representations and weighting schemes. We compare their performances and report their evaluation results on the dataset we have used in this chapter but also a variant that includes knowledge extracted from user reviews.

Chapter 5

Extensions to Recommendation-by-Explanation

5.1 Introduction

Recommendation by Explanation (*r-by-e*) unifies recommendation and explanation: it computes recommendations by generating and ranking corresponding personalized explanations in the form of Explanation Chains. We previously explained in detail how *r-by-e* constructs the chains for candidate items and selects the top-n items that it will recommend. Although we observed promising results, the way we compute overlap (ovrlp) and score (score) were relatively simple, and we considered only one way of representing items — in terms of their features.

In this chapter, we extend *r-by-e* in two ways (see Figure 5.1). 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.

We also generalize *r-by-e*'s *chain selection*. 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 explore these variants using two versions of our movie dataset. The first version is the one we have used so far, described in Section 2.4. Using this dataset, we represent each item as a set of its keywords. In the second version

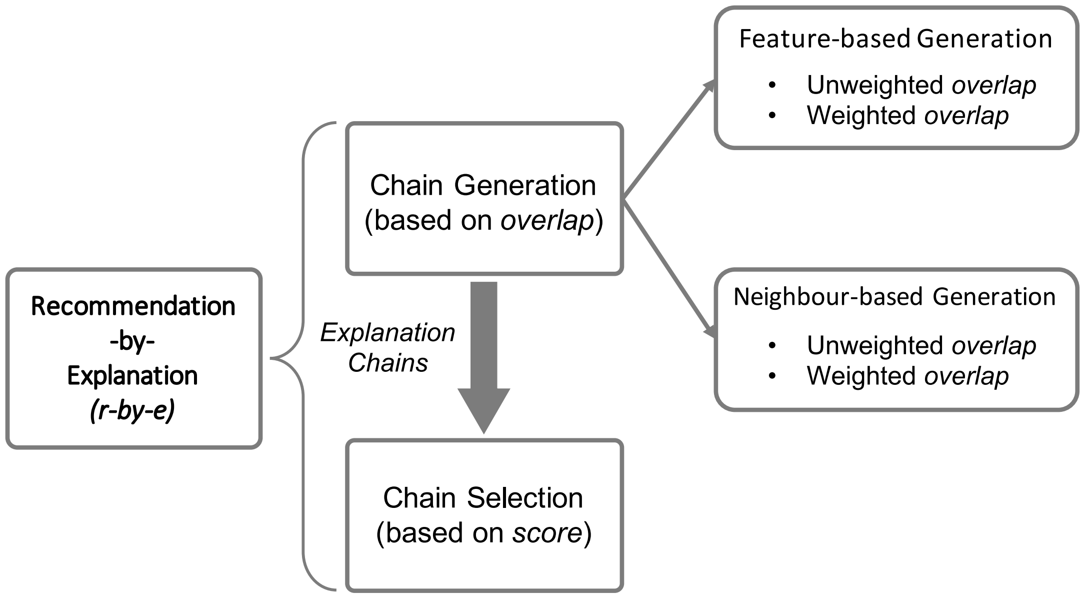

Feature-based Generation
• Unweighted *overlap*
• Weighted *overlap*
Chain Generation (based on *overlap*)
Recommendation -by- Explanation *(r-by-e)*
Explanation Chains
Neighbour-based Generation
• Unweighted *overlap*
• Weighted *overlap*
Chain Selection (based on *score*)

Figure 5.1: Extended Recommendation-by-Explanation

of the dataset, we use sentiment data extracted from user reviews and represent each item as a set of its terms associated with sentiment values. The complete details of this version of the dataset are covered in Section 5.4.

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. Notice that unweighted feature-based is what we covered already in the previous chapter. We include it here again but in a normalized form for better understanding the difference among all four approaches. We 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.)

From the offline experiments, we choose the best version of *r-by-e* to compare with a baseline in a user trial. We conduct this trial on the sentiment version of the dataset.

5.2 Extended ***r-by-e***

As previously explained, we have modeled the *chain generation* step as a set-cover problem. But, in this chapter, we show that we have two ways of formulating this. In one of our approaches, *feature-based* generation, our aim is to cover the candidate item's set of features (e.g. its keywords); in our other approach, *neighbour-based* generation, we aim to cover the candidate's set of neighbours (i.e. similar items). The core of *chain generation* is computing overlap (ovrlp) between a potential predecessor and a candidate item. This can simply be performed by *counting* the number of elements (either *features* or *neighbours*) that are covered. We call this *unweighted overlap.* An alternative way of computing overlap (ovrlp) 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 is a sum of the average overlap of the chain members and a diversification term (see Eq. 4.2). 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 dissertation we only show results for neighbour-based chain selection. However, in this chapter we do make one change to the score function 4.2: 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.

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

5.2.1 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's features.

5.2.1.1 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:

$$\text{ovrlp}(p, i, C) = \frac{2 \cdot |(F_p \setminus \text{covered}(i, C)) \cap F_i|}{|F_i| + |F_p|} \tag{5.1}$$

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. $\text{covered}(i, C) = \bigcup_{j \in C} F_j \cap F_i$. This is different from Eq. 4.1 that we defined in the previous chapter. This is the *harmonic mean* of the two terms of Eq. 4.1 and returns a value of ovrlp in the range of $[0, 1]$. We made this change to make it comparable with the other versions of overlap that we define below.

5.2.1.2 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. One such approach to term-weighting is term frequency–inverse document frequency (TF-IDF) [SB88].

As shown in Eq. 5.2, the weight of a feature f of a candidate item i with respect to the set of candidates $\mathbb{I}$ 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 item $i \in \mathbb{I}$ from the other items in the collection. The set of items consisting of the feature f is denoted as $\mathbb{I}_f$.

In our experiments, we treat an item as a document and its features as terms, and adopt the weighting scheme for a feature f of item i as:

$$w_{fi} = \frac{(1 + \log(o_{fi})) \cdot \left(\log \frac{|\mathbb{I}|}{|\mathbb{I}_f|}\right)}{\sqrt{\sum_{f' \in i} \left((1 + \log(o_{f'i})) \cdot \left(\log \frac{|\mathbb{I}|}{|\mathbb{I}_{f'}|}\right)\right)^2}} \tag{5.2}$$

where o_{fi} is the number of occurrences of a feature f in the item i, $|\mathbb{I}_f|$ is the number of items containing the feature f, and $|\mathbb{I}|$ is the total number of candidate items. 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 informativeness of all other features in the item. The set of item features (analogous to documents in term-document space) 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 [HGE+12].

Using weights assigned to features as above, we define the weighted overlap $\text{wovrlp}(p, i, C)$ of adding predecessor p to partial chain C that explains candidate item i as:

$$\text{wovrlp}(p, i, C) = \frac{2 \cdot \left(\sum_{f \in (F_p \setminus \text{covered}(i,C)) \cap F_i)} 1 - |w_{fp} - w_{fi}|\right)}{|F_i| + |F_p|} \tag{5.3}$$

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}. Here, we assume that the weights of item features are in the range of $[0, 1]$ which is guaranteed from Eq. 5.2.

5.2.2 Neighbour-based generation

In *neighbour-based* settings, an item i is represented as a set of its neighbours N_i. The set of neighbours of an item i 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.

We define *neighbour-based* overlap slightly differently from *feature-based* overlap. In this approach, 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.

5.2.2.1 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 $\text{ovrlp}(p, i, C)$, 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:

$$\text{ovrlp}(p, i, C) = \frac{2 \cdot |(N_p \setminus \text{covered}(i, C)) \cap N_i|}{|N_i \setminus \text{covered}(i, C)| + |N_p \setminus \text{covered}(i, C)|} \tag{5.4}$$

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. $\text{covered}(i, C) = \bigcup_{j \in C} N_j \cap N_i$. 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.

5.2.2.2 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 $\text{wovrlp}(p, i, C)$ of adding predecessor p to partial chain C that explains candidate item i is given by:

$$\text{wovrlp}(p, i, C) = \frac{2 \cdot \left(\sum_{j \in (N_p \setminus \text{covered}(i,C)) \cap N_i} 1 - |w_{jp} - w_{ji}|\right)}{|N_i \setminus \text{covered}(i, C)| + |N_p \setminus \text{covered}(i, C)|} \tag{5.5}$$

This is analogous to Eq. 5.3.

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.

5.2.3 Generalized chain selection

In this chapter, we generalize Eq. 4.2 so that we score $\langle C, i\rangle$ relative to a list of all the items that appear in already-selected chains L^* using the following:

$$\text{score}(\langle C, i\rangle, L^*) = (1-\alpha)\cdot\frac{sum\_ovrlps}{|C|+1} + \alpha\cdot\frac{\left|C \setminus \bigcup_{j\in L^*} j\right|}{|C|+1} \tag{5.6}$$

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.

The rest of this chapter reports the results of experiments that evaluate the extensions that we have presented in this chapter.

5.3 Extended Chains on Keywords

In this section, we evaluate extended chains on the dataset that we discussed in Section 2.4, which is a dataset in which item features are keywords.

5.3.1 Offline evaluation

We ran an offline experiment to evaluate the different versions of *r-by-e*. We wanted the experiment to reveal the effect of the difference between the following:

- *feature-based* versus *neighbour-based*: The former represents an item as a set of its keywords, while the latter represents an item as s set of its neighbours (similar items) in which the features are used only indirectly.

- *unweighted* versus *weighted*: The former takes into account the feature or neighbour 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 and *CB*-$|C|$ is the dynamic content-based system. Details of the dynamic content-based system have been covered in Section 4.3. 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. 5.6). 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.

5.3.1.1 Experiment settings

The evaluation is performed under the same experiment settings as we discussed in Section 4.3.1. However, we explain the values for the hyperparameters that we use in this experiment.

Apart from α that we vary from $[0, 1]$ in steps of 0.1, we also consider different 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)[1].

The reason behind the 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 feature-based representation) and the size of its set of neighbours (for neighbour-based representation). As we saw in Section 2.4, a typical item has on average

[1]Because of the high computational cost, in place of the five different values of each of θ and ϵ that we used in the previous chapter (Section 4.3.1), we use only three values for each of the two hyperparameters

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.

5.3.1.2 Experiment results

Table 5.1 and Table 5.2 summarize the results of the feature-based and neighbour-based approaches. The columns of the table are the different evaluation measures. 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 *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*.

If we look in more detail at the results according to the way hyperparameter values have been optimized, we see the following:

- *Optimizing hyperparameters for precision:* In this setting, *wfb* performs best in terms of precision and % of explanations of size 2–4, while giving lower values for diversity, surprise, novelty, coverage. The baseline recommender makes more diverse and surprising recommendations but at the cost of their relevance.

- *Optimizing hyperparameters for diversity:* In this setting, *CB* attains highest diversity and coverage, while giving very low precision. *CB*-$|C|$ recommenders also give higher values of diversity but they cover only around 4% of the catalogue and they make almost irrelevant recommendations.
- *Optimizing hyperparameters for % of explanations of size 2–4:* In this setting, almost all of the chains for top-n recommendations of *fb* and *wfb* satisfy the size constraint, while *CB*-$|C|$ recommenders hardly generate chains of this size. The results for *fb* and *wfb* look quite similar, but are still statistically significant except for their precision.

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 they have lower precision and coverage.

- *Optimizing hyperparameters for precision:* In this setting, *nb* and *wnb* obtain nearly the same level of precision, diversity, surprise, and coverage with no significant differences. The corresponding *CB*-$|C|$ recommenders attain higher values of diversity, surprise and novelty with lower precision, coverage, and % of explanations of size 2–4.
- *Optimizing hyperparameters for diversity*: In this setting, *CB*-$|C|$ recommenders give the most diverse, surprising, and novel recommendations but they are almost never relevant. These recommender also cover only around 3% of the catalogue. Differences in the results for *nb* and *wnb* are statistically significant in all cases except the coverage.
- *Optimizing hyperparameters for % of explanations of size 2–4:* In this setting, over 97% of the chains generated by *nb* and *wnb* recommenders are of manageable size (2–4 items). However, they cover around 16% of the catalogue with nearly 1.5% of relevant recommendations. Their diversity, surprise, and novelty are nearly equal and also lower than their corresponding *CB*-$|C|$ recommenders.

Feature-based vs. neighbour-based chain generation. We will now compare the two types of item representation by looking at results from both Table

5.1 and Table 5.2 together.

- *Optimizing hyperparameters for precision:* In this setting, feature-based approaches attain three times higher precision, greater coverage, almost similar levels of diversity and % of explanations of size 2–4, and lower surprise and novelty. This shows that items recommended by feature-based approaches are more relevant, equally diverse, but less surprising when approaches are optimized for precision.

- *Optimizing hyperparameters for diversity:* In this setting, feature-based approaches attain around five times higher precision, a nearly equal level of diversity but lower surprise, novelty, and coverage. Over 81% of neighbour-based explanation chains are of size 2–4 in comparison to around 46% of feature-based chains.

- *Optimizing hyperparameters for % of explanations of size 2–4:* In this setting, feature-based approaches attain almost equal precision but with higher levels of diversity, surprise, novelty, coverage, and explanation chains of size 2–4.

Table 5.1: Results of the offline experiment on *feature-based* approaches for extended chains on 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 of size 2–4	0.0050	0.9747	0.9346	0.4930	0.0288	0.0074
wfb		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

Table 5.2: Results of the offline experiment on *neighbour-based* approaches for extended chains on 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

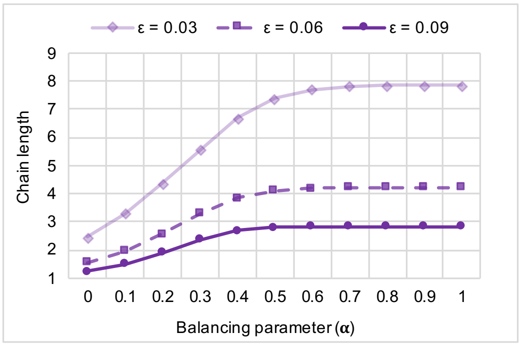
Let us select one of the four approaches for further study. To pick one, let us assume that optimizing for the % of explanations of size 2–4 is best, since it generally gives explanations that are not so long as to be uninterpretable. In this setting, *wfb* performs better than other versions of *r-by-e*, 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).

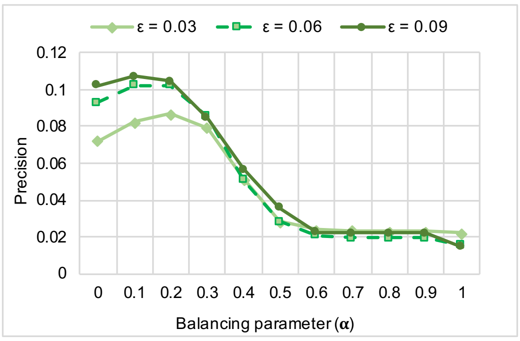
Figure 5.2 has six sub-figures — one for each evaluation measure. Each line that we plot in a sub-figure is for one of the three different values of ϵ. In most cases, increasing ϵ does not change the trend of the measure; it only 'shifts' the values of the measure because higher values of ϵ impose a 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. 5.6: for conciseness, we refer to its first term, 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 5.2(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 that increasing α gives more weight to the profile term which enables the system to select longer chains; on the other hand, the overlap term favours selecting shorter chains. It is also noteworthy that increasing ϵ imposes a stricter constraint on chains such that their average length reduces substantially.

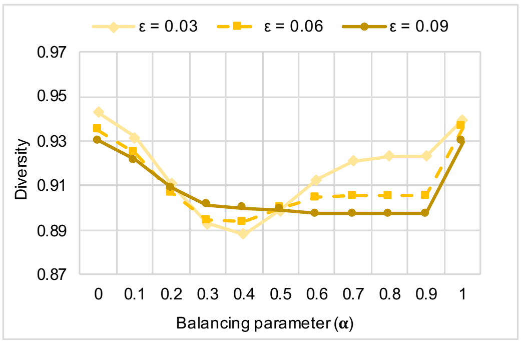
Precision: In Figure 5.2(b), we see that precision varies in four different ways as we increase the value of α: i) up to 0.2, it increases; ii) from 0.2 to 0.7, it decreases; iii) from 0.7 to 0.9, it remains almost constant; and iv) at 1.0, it slightly decreases. In explanation chains, precision is proportional to the candidate's



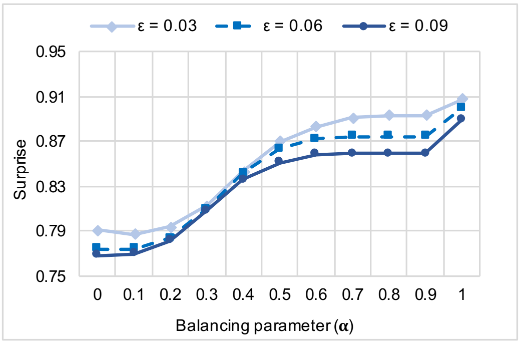
(a) Chain length



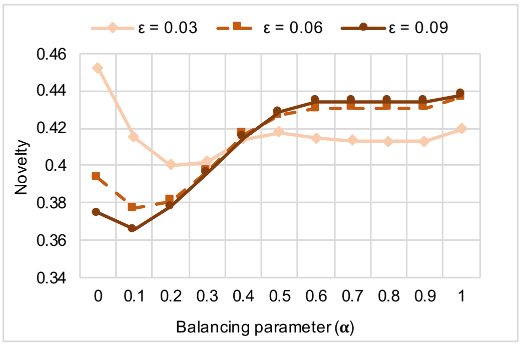
(b) Precision



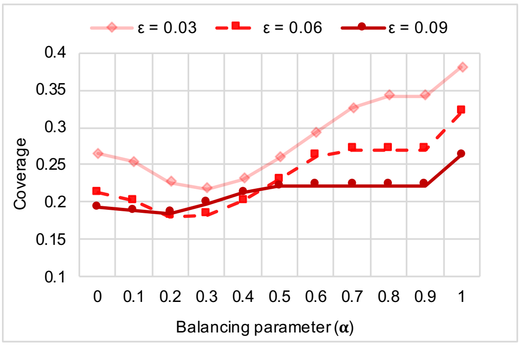
(c) Diversity



(d) Surprise



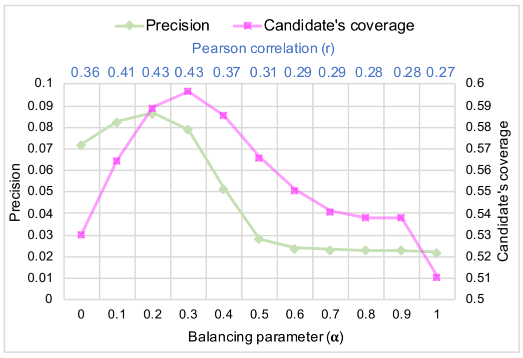
(e) Novelty



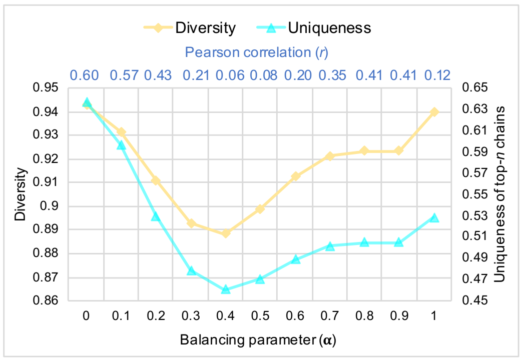
(f) Coverage

Figure 5.2: 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.

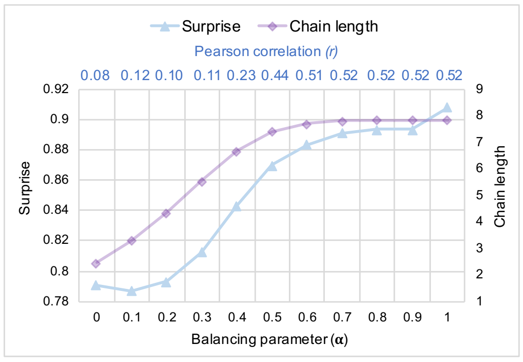
coverage: the greater the candidate's coverage, the higher is the precision. We will define the candidate's coverage as the ratio of the number of candidate's features covered by the chain members to the size of the candidate's feature set. The variation in precision with respect to α indicates that 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



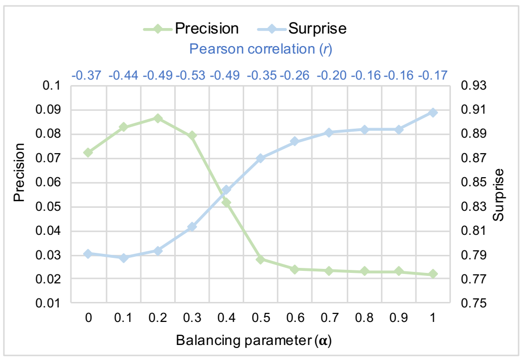
(a) Precision vs. Candidate coverage



(b) Diversity vs. Uniqueness



(c) Surprise vs. Chain length



(d) Precision vs. Surprise

Figure 5.3: 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. $p < 0.05$).

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 5.3(a).

Diversity: In Figure 5.2(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 5.3(b), we show that both diversity and uniqueness follow the same trend for increasing α.

Surprise: Figure 5.2(d) shows that up to $\alpha = 0.6$, surprise increases (though up to $\alpha = 0.2$, it increases only slightly); 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: more items from the user's profile are needed to support the candidate item. We find a correlation between the surprise and the chain length that we show in Figure 5.3(c). We also see in Figure 5.3(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 5.2(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 candidate's coverage: 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. On $\epsilon \in \{0.06, 0.09\}$, novelty almost increases with the increase in chain length.

Coverage: In Figure 5.2(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 larger part of it. On higher values of ϵ, the system imposes a stricter constraint that lowers the catalog coverage.

5.3.2 User trial

We reported the results of a user trial of *fb* on this dataset in the previous chapter. It might be interesting to see the results we would obtain were we to use *wfb* instead. However, given the cost of running user trials and the modest differences in offline results between *fb* and *wfb*, we chose not to conduct this user trial. We do, however, produce a variant of our dataset (see below) and conduct both offline experiments and another user trial on that dataset.

5.3.3 Summary

In this section, we have presented offline experiments to evaluate all four versions of *r-by-e*. They confirm that the weighted feature-based approach attains better precision than the other approaches, while remaining competitive on measures of diversity and serendipity. We have also found that, overall, feature-based approaches outperform neighbour-based ones, although they may provide relatively less serendipitous recommendations.

We have also explained the behaviour of evaluation measures for different hyper-parameter values by means of their dependency on other factors: precision and candidate coverage is one such example, among others. We will also investigate these relationships on a different version of the dataset in the next section.

5.4 Extended Chains on Sentiments

In this section, we describe evaluation of extended chains on a different version of our dataset in which items are represented as sentiment features. We will refer to chains that make use of these sentiment features as *Sentiment-aware Explanation Chains.* These chains guarantee that items are connected only if they share features with close polarity scores.

We perform experiments with the aim of revealing the effect of sentiment awareness on the quality of recommendations and on the effectiveness of their corresponding explanations.

Next, we describe how we obtain these sentiment features for our dataset.

5.4.1 Dataset

In the user trial that we described in Chapter 4, 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 chapter, we use sentiment data extracted from user reviews for each of these 1851 movies.

Here, we would like to acknowledge Rafael Martins D'Addio, who followed the steps that he proposed in [DFDCM18, DMM19] for preparing the sentiment-based item representation and sharing the dataset with us for this set of experiments. The only difference in the way he prepared this dataset from the methods he describes in [DFDCM18, DMM19] is that, after extracting concepts from the user reviews, here he filtered them using cosine normalized TF-IDF scores as we mentioned in Eq. 5.2, whereas in his previous work he filtered them using a different but related measure that he calls SF-IDF [DMM19].

5.4.1.1 User reviews to concepts

First, the top-10 most *helpful* user reviews for each of the items in the dataset were scraped from IMDb[2]. From these user reviews, concepts were extracted to be used as features. These will be weighted by the sentiments that users have towards them, and these sentiment-weighted features will be used to describe our items.

A concept in the approach given in [DFDCM18, DMM19] describes an idea or a notion. Concepts can be seen as synsets, i.e. sets of word synonyms which define an idea. By using concepts as features in place of words, we reduce issues such as polysemy (a word with multiple meanings) and synonymy (multiple words with the same meaning).

In order to extract concepts and sentiments from user reviews, two different natural language processing resources were used: Stanford CoreNLP[3] [MSB$^+$14] for sentence splitting, parsing and sentence-level sentiment analysis; and BabelFy[4] [MRN14] for word sense disambiguation and entity linking.

First, the items' reviews were processed with Stanford CoreNLP using the fol-

[2]https://www.imdb.com/
[3]https://stanfordnlp.github.io/CoreNLP/
[4]http://babelfy.org/

lowing pipeline: tokenization, part-of-speech (POS) tagging, parsing, sentence splitting and sentiment analysis. From there, the sentences of the reviews were processed by BabelFy. BabelFy process the texts, returning disambiguated concepts in the form of BabelNet synsets [NP12]. BabelNet[5] is a knowledge base that links several linguistic resources, such as Wikipedia, Wikidata, WordNet, among others. The 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 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), while adjectives and adverbs may be opinion words, i.e. words that indicate sentiment towards features [LZ12].

5.4.1.2 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 will not influence anything in the explanation chain generation. Similarly, the 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. 5.2. 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 were used to produce item representations.

Thus, the dataset comprises 2036 users, 1851 movies, 34088 concepts, and around

[5]https://babelnet.org/

300000 ratings. 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.

5.4.1.3 Representing items

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 different 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 approaches do not make any sense. Therefore, we run experiments only on weighted versions: weighted feature-based (*wfb*) and weighted neighbour-based (*wnb*).

5.4.2 Offline evaluation

We ran an offline experiment to evaluate sentiment-aware *r-by-e*'s performance. 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-*$|C|$.

5.4.2.1 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*.

5.4.2.2 Experiment results

Table 5.3 and Table 5.4 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*.

We look in more detail at the results according to the way hyperparameter values have been optimized as below:

- *Optimizing hyperparameters for precision:* In comparison to the *wfb* recommender, the *CB* recommender attains higher levels of diversity, surprise, and novelty but has around eight times lower precision and a lower coverage. Similarly, the *wfb-CB-*$|C|$ recommender has higher diversity, surprise, and novelty but covers only around 3% of the catalogue with almost no relevant recommendations. In this setting, *wfb-CB-*$|C|$ selects no explanations of size 2–4; in fact, they all contain only one item.
- *Optimizing hyperparameters for diversity:* In this setting, when compared to the *wfb* recommender, we see that *CB* and *wfb-CB-*$|C|$ suggest more diverse, surprising, and novel recommendations (though they are not much different in the case of diversity and surprise); however, they are less accurate and cover less of the catalog.
- *Optimizing hyperparameters for % of explanations of size 2–4:* In this set-

ting, around 97% of *wfb*'s top-n recommendations have explanations that satisfy the size constraint as opposed to 20% of *wfb-CB-*$|C|$'s recommendations. The *wfb* recommender attains much higher precision and catalogue coverage, but has lower levels of diversity, surprise, and novelty.

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 aply 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, while having lower precision. In particular, the *wnb-CB-*$|C|$ recommender has lower catalog coverage with explanations satisfying the size constraint.

- *Optimizing hyperparameters for precision:* In comparison to *CB* and *wnb-CB-*$|C|$, *wnb* attains higher precision but lower levels of diversity, surprise, and novelty. Over 84% of its top-n chains are of size of 2–4. *wnb-CB-*$|C|$ covers only 3% of the catalog with almost no relevant recommendations. None of its top-n explanations are of size 2–4.
- *Optimizing hyperparameters for diversity*: In this case, the *wnb-CB-*$|C|$ recommender generates the most diverse, surprising, and novel recommendations but almost none of them are relevant. Even *wnb*'s recommendations, though better, are mostly not relevant. Around 99% of *wnb*'s explanation chains are of size 2–4 as opposed to 71% of *wnb-CB-*$|C|$'s explanations.
- *Optimizing hyperparameters for % of explanations of size 2–4:* In this setting, almost all of the chains generated by the *wnb* recommender are of the size of 2–4. However, they cover only 27% of the catalogue and only nearly 1.2% of them are relevant. Its recommendation diversity, surprise, and novelty are lower than the *wnb-CB-*$|C|$ recommender.

Table 5.3: Results of the Offline Experiment using *feature-based* approaches for extended chains 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 optimized for	Precision	Diversity	Surprise	Novelty	Coverage	% of explanations of size 2–4
CB	Precision	0.0145	0.9383	0.9126	0.5048	0.3068	NA
wfb		0.1053	0.9130	0.8697	0.3468	0.3271	0.4040
wfb-*CB*-$\lvert C\rvert$		0.0036	0.9300	0.9094	0.5597	0.0282	0.0000
CB	Diversity	0.0145	0.9383	0.9126	0.5048	0.3068	NA
wfb		0.0223	0.9225	0.9006	0.4056	0.5185	0.9025
wfb-*CB*-$\lvert C\rvert$		0.0033	0.9308	0.9102	0.5619	0.0245	0.0000
wfb	% of explanations of size 2–4	0.0838	0.9104	0.8778	0.3752	0.4930	0.9731
wfb-*CB*-$\lvert C\rvert$		0.0014	0.9483	0.9234	0.6102	0.0361	0.2054

Table 5.4: Results of the Offline Experiment using *neighbour-based* approaches extended chains 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 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 of size 2–4	0.0117	0.9186	0.9005	0.4250	0.2700	0.9997
wnb-CB-$|C|$		0.0018	0.9513	0.9247	0.5989	0.0976	0.1804

Feature-based vs. neighbour-based chain generation. We will now compare the two types of item representation by looking at results from both Table 5.1 and Table 5.2 together.

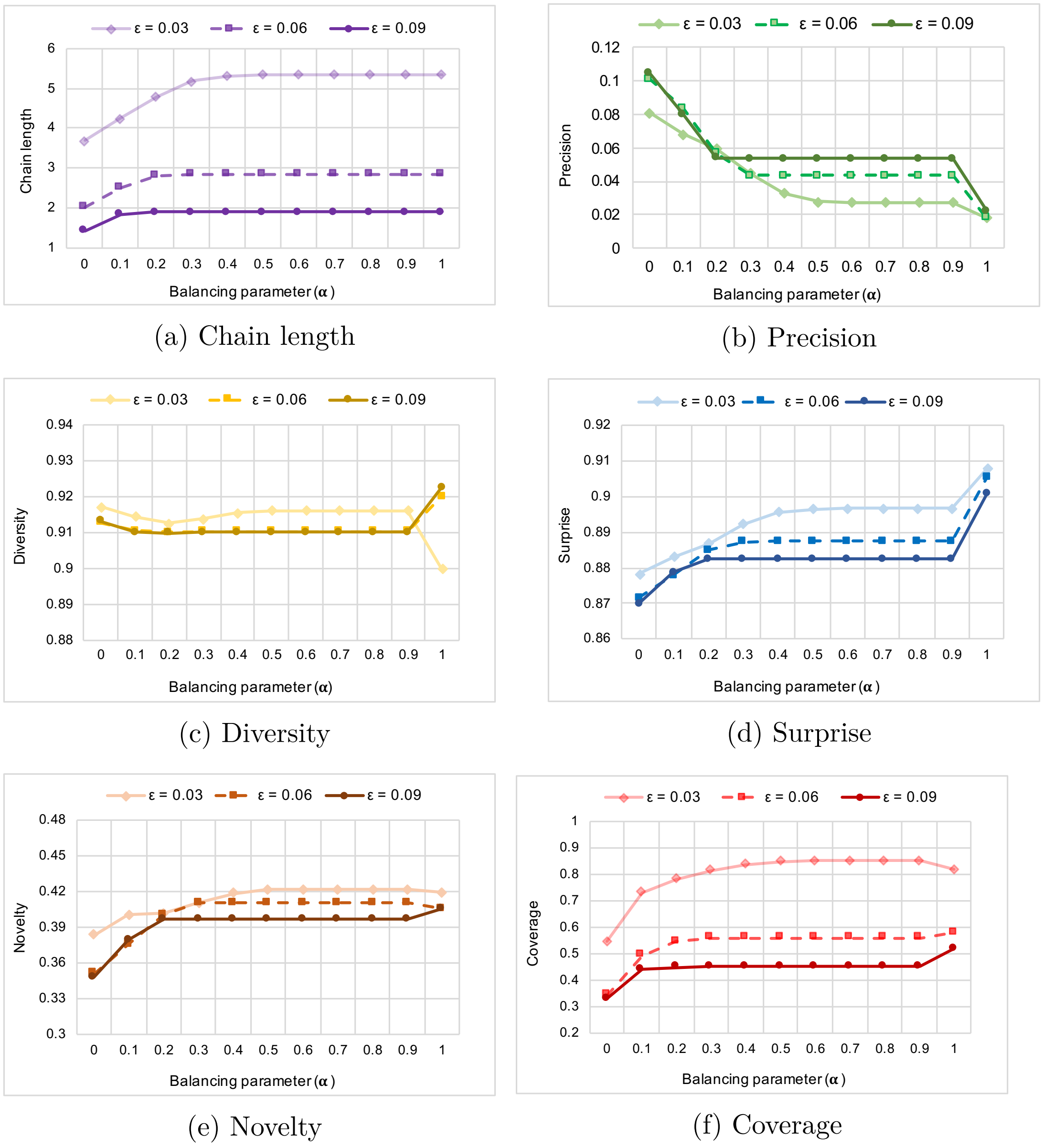
- *Optimizing hyperparameters for precision:* In this setting, the *wfb* recommender attains three times higher precision, greater coverage, nearly equal levels of diversity and % of explanations of size 2–4, but lower surprise and novelty than the *wnb* recommender.
- *Optimizing for Diversity*: In this setting, the *wfb* recommender attains around three times higher precision, nearly equal levels of diversity and surprise but lower levels of novelty and coverage than the *wnb* recommender. The *wfb* recommender generates over 90% of explanation chains of size 2–4 as opposed to 98% of *wnb*'s chains.
- *Optimizing for % of explanations of size 2–4:* In this case, the *wfb* recommender has around eight times more relevant recommendations with a nearly equal level of diversity as the *wnb* recommender. The coverage of the *wfb* recommender is greater, while its surprise and novelty are lower than the *wnb* recommender.

Overall, in comparison to the *wnb* recommender, the *wfb* recommender performs much better in terms of precision, while remaining competitive for diversity. In contrast, the *wnb* recommender gives less relevant but more surprising and novel recommendations.

As we did in the last experiment, we now select one of the four approaches for further study. 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 other versions of *r-by-e*, 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 5.4 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 dataset.

We look in detail at the results for each evaluation measure individually. Again,



(a) Chain length (b) Precision (c) Diversity (d) Surprise (e) Novelty (f) Coverage

Figure 5.4: 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.

we refer Eq. 5.6 and, as before, for conciseness, we refer to its first and second terms as the *overlap term* and the *profile term* respectively.

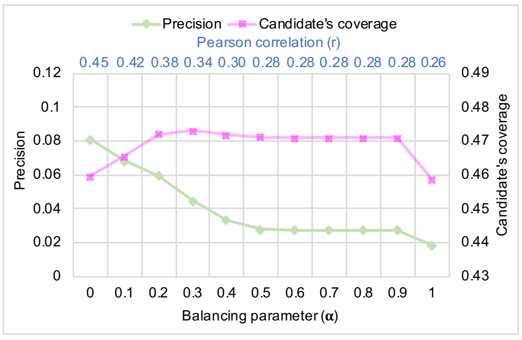
Chain length: In Figure 5.4(a), we see that the length of top-n chains increases up to $\alpha = 0.4$; then, increasing α does not show a noticeable effect on the length. This indicates that for lower α, the overlap term dominates, and the system selects shorter chains; it selects longer chains as α gives more weight to the profile term. For higher values of ϵ, the system imposes a stricter constraint on chains so their average length reduces substantially.

Precision: In Figure 5.4(b), for $\epsilon = 0.03$, we see that precision varies in three different ways on 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 dataset, 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 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 5.5(a).

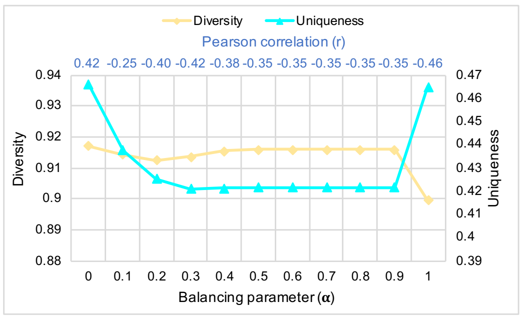
Diversity: In Figure 5.4(c), we see diversity remains almost unchanged up to $\alpha = 0.9$ and decreases at $\alpha = 1.0$ for $\epsilon = 0.03$ (and increases for $\epsilon \in \{0.06, 0.09\}$). In our system, as we have shown in the previous experiment, the diversity of the top-n recommendations depends upon their uniqueness: the lower the overlap among the members of top-n chains, the higher is the diversity. However, in this dataset, 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 5.5(b) that in all but one case uniqueness of chain members is negatively correlated with diversity.

Surprise: Figure 5.4(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 5.3(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 5.5(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.

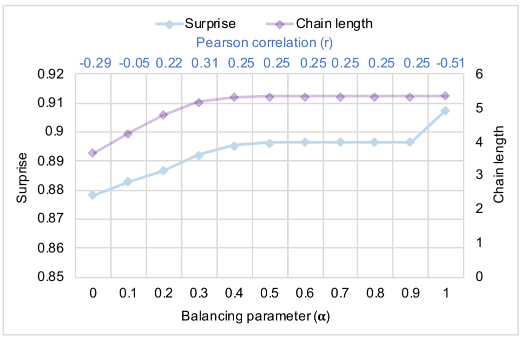
Novelty: Figure 5.4(e) shows that novelty increases up to $\alpha = 0.4$, then remains almost at the same level. It shows that on lower values of α, the overlap term



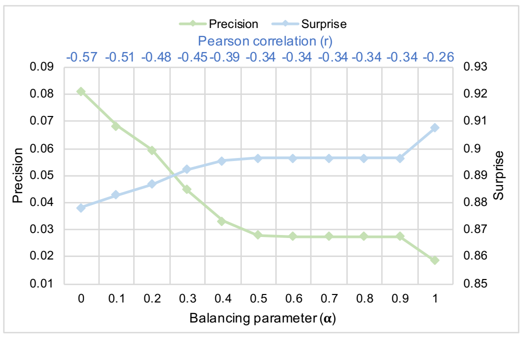
(a) Precision vs. Candidate's coverage



(b) Diversity vs. Uniqueness



(c) Surprise vs. Chain length



(d) Precision vs. Surprise

Figure 5.5: 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. $p < 0.05$).

dominates, enabling the system to 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 5.4(f) shows that coverage increases up to $\alpha = 0.4$, then remains almost unchanged. In this dataset, this indicates that up to $\alpha = 0.4$, coverage increases with the increase in the 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.

5.4.3 User trials

We used our web-based system in order to conduct a user trial, again comparing *wfb* with *wfb-CB-*$|C|$ (but this time the sentiment-aware versions) using the hyperparameter values (θ, ϵ) and α that optimized the percentage of explanations

Table 5.5: Results of the Recommendation Trial for Extended chains 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

of size 2–4. This user trial uses the same methodology as the previous one.

In total, 144 people attempted the trial. Although they were anonymous, our method of recruiting them (e.g. using personal email lists) means that the majority of them were undergraduate and postgraduate students of Computer Science recruited online from universities in Ireland, Brazil, and India. Again they were not rewarded for participation in any way. 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.

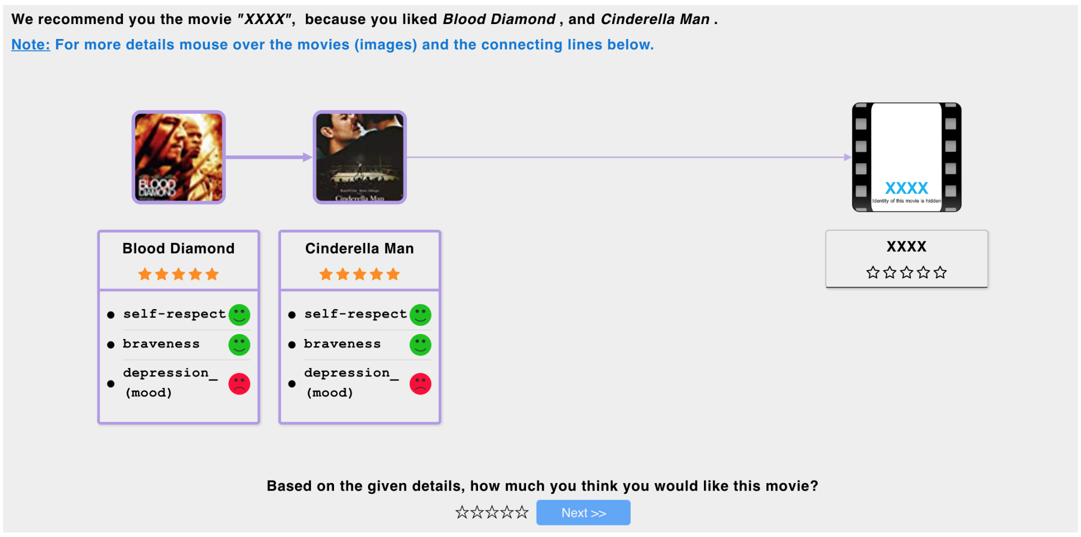
5.4.3.1 Recommendation trial

RQ: Does *wfb* generate more diverse, serendipitous, and relevant recommendations than *wfb-CB*-$|C|$?

Experiment settings. The recommendation trial is identical to the one conducted in the previous chapter (see Figure 4.2).

Experiment results Fifty-five participants completed this trial. Table 4.2 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.


We recommend you the movie *"XXXX"*, because you liked *Blood Diamond*, and *Cinderella Man*.

Note: For more details mouse over the movies (images) and the connecting lines below.

Based on the given details, how much you think you would like this movie?

Figure 5.6: 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.

- *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.)

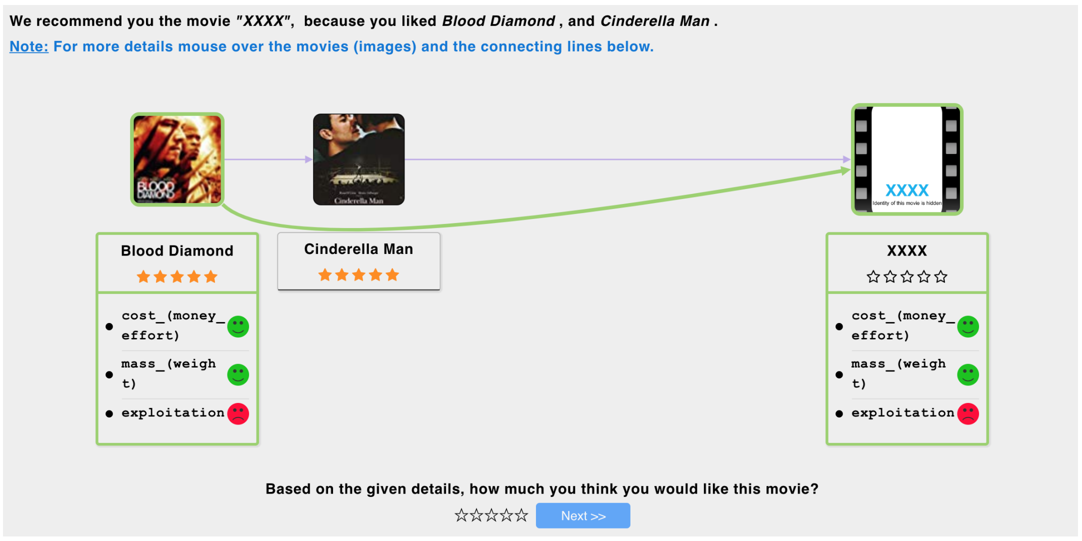
5.4.3.2 Explanation trial

RQ: Does *wfb* generate more effective explanations than *wfb-CB-*$|C|$?

Experiment settings Users who were directed to this trial participated in a re-rating task. All of the experiment settings are same as the explanation trial from the previous chapter.

CB-$|C|$'s explanations (sets of neighbours, rather than chains) were displayed in the fashion shown in Figure 4.5: arrows connect each movie to the recommended movie.

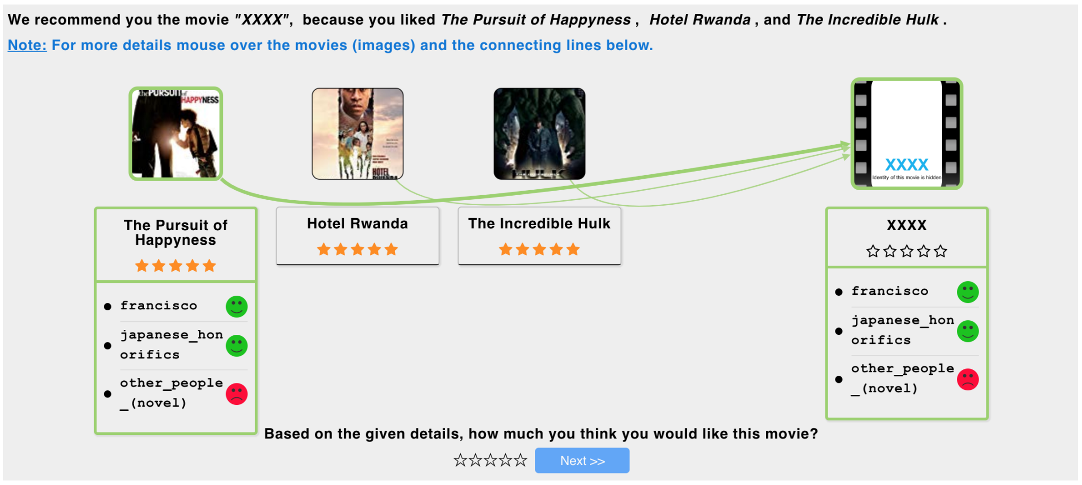
In both cases, as before, the user can mouse over parts of the explanation, which


We recommend you the movie *"XXXX"*, because you liked *Blood Diamond* , and *Cinderella Man* .

Note: For more details mouse over the movies (images) and the connecting lines below.

Based on the given details, how much you think you would like this movie?

Figure 5.7: 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.


We recommend you the movie *"XXXX"*, because you liked *The Pursuit of Happyness* , *Hotel Rwanda* , and *The Incredible Hulk* .

Note: For more details mouse over the movies (images) and the connecting lines below.

Based on the given details, how much you think you would like this movie?

Figure 5.8: 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.

causes the system to display features that movies have in common (see the captions of the Figures). A maximum of three features is displayed in any box. But now they are selected by their sentiment scores. Each is also associated with a smiley icon 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).

As before, the user gives us $2n$ ratings based on explanations with redacted movies

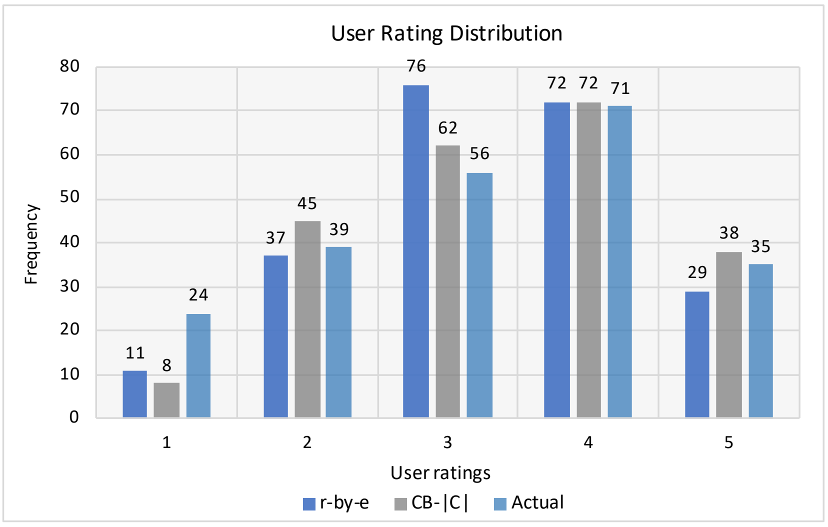


Figure 5.9: Ratings from the Explanation Trial for Extended chains on Sentiments.

Table 5.6: Ratings from the Explanation Trial for Extended chains 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

and n ratings based on movie information without any explanations.

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 4.6 shows the distribution of the users' ratings; Table 4.3 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 identity. For the differences between explanation-ratings and actual-ratings, Figure 4.7 shows the distribution of values and Table 4.4 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

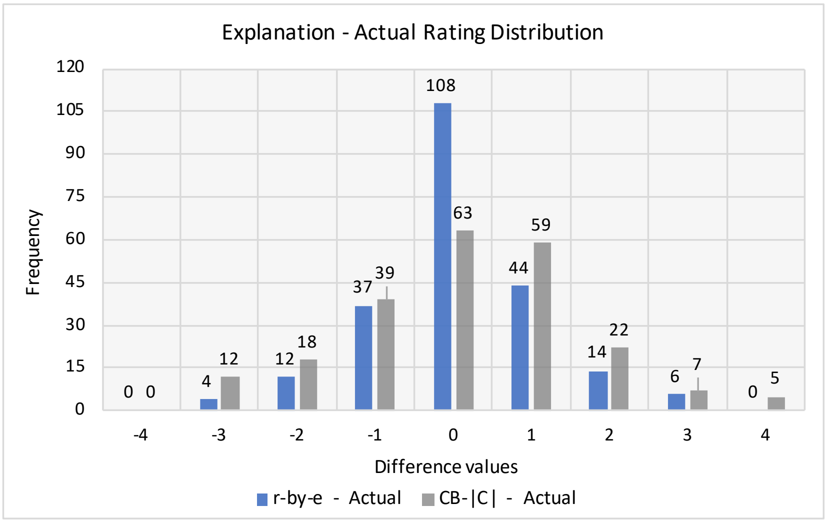


Figure 5.10: Differences in ratings from the Explanation Trial for Extended chains on Sentiments.

Table 5.7: Differences in ratings from the Explanation Trial for Extended chains 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)

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 4.3 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.

5.4.4 Summary

In this section, we have presented offline experiments on a sentiment dataset to evaluate the weighted versions of *r-by-e*. The experiments confirm that the weighted feature-based approach attains better precision than the other approaches, while remaining competitive on measures of diversity and serendipity. We have also found that, overall, feature-based approaches outperform neighbour-based ones, although they may provide less serendipitous recommendations. Because, in this dataset, item-item similarity is higher than it was in the previous one, overall diversity of the recommendations is lower. We also conducted user trials to evaluate the quality of recommendations and effectiveness of the sentiment-aware explanation chains. We have found that such chains also produce more diverse and serendipitous recommendations and generate explanations that are more helpful for users to make accurate decisions than the closely customized version of classic content-based method.

5.5 Conclusion

In this chapter, 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 (e.g. keywords), and the latter makes no explicit reference to features and represents an item as a set of its neighbours. 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.

This chapter presented 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 that are also competitive on measures of diversity and serendipity than all other versions of *r-by-e* and the baselines. We also found an interesting relationship between the chain length and the surprise: the higher the chain length, the more surprising is the recommendation. Another offline experiment, this time on a dataset where items are described by features weighted by sentiment scores, 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 case of 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 customized classic content-based recommender (although not statistically significantly so) and with statistically significantly higher levels of satisfaction. User response in the Explanation Trial confirmed that the sentiment-aware explanation chains allow them to make more accurate judgements about the quality of the recommended items than do the baseline's explanations.

Now, we move to the next part of the dissertation in which we investigate how to integrate a user's feedback on a set of recommendations into the process of computing the next round of recommendations, in order to help the user reveal her ephemeral needs.

Part III

Navigation-by-Preference

Chapter 6

Conversational Recommendation: State of the Art

As we saw in Chapter 2, chains are a unifying framework for this dissertation. They can be built backwards automatically to explain a candidate item by visiting items in the user's profile or, as we will see, they can be built forwards in partnership with the user from a seed, through a chain of recommendations (usually items that are not in the user's profile) that, it is hoped, by taking the user feedback into account, do an ever better job of satisfying the user. Incorporating the user's feedback while recommending items to the user makes the system more conversational. In this chapter, we review the state-of-the-art in conversational systems.

Recommender systems infer the user's long-term preferences from her past interactions (e.g. ratings, etc.) with the items, which are recorded in a user profile. Recommendation algorithms often assume *single-shot recommendation*: the recommender ranks the candidate items and allows the user to explore the top-n items from this ranking. The problem with single-shot recommendation comes when the user is not fully satisfied by the recommended items. Other things being equal, the recommender cannot offer her a fresh set of recommendations unless she changes her profile, e.g. adding a new interaction such as by consuming and rating an item. In *conversational recommendation*, by contrast, the user is invited to provide feedback on the current top-n recommendations, even without consuming them; for example, the user might simply indicate which of the top-n recommendations comes closest to the kind of item she wants to consume on this occasion. The recommender takes account of the feedback in generating a fresh top-n recommendations. It may take several recommendation cycles before the user finds a suitable item, which is why these systems are called conversational recommender systems and why the interaction is referred to as a dialog. This terminology dates back ten years, e.g. [BGMS05, Smy07]. More recently, the word "conversational" has implied not just a dialog, as before, but a dialog in natural language, e.g. [GGL18]. In this chapter, we continue to use the word "conversational" in its earlier, broader sense; indeed, the research we describe in this chapter uses a graphical user interface and not a natural language interface.

Conversational recommender systems (CRSs) cater for a user who is not satisfied with the initial top-n recommendations. This is particularly useful where the user has an ephemeral goal, so she has short-term preferences that differ from her long-term preferences. For example, a user might usually watch documentaries but this evening she is not in the mood for something so serious. Or, perhaps, this evening she wants something to watch with her mother, so she should accommodate her mother's tastes as well as her own. Conversational recommendation is therefore one approach to context-aware [AMRT11] and context-driven [PCL$^+$16] recommendation: the user can give feedback to steer the recommendations toward ones that best suit the context [HMB14]. However, most context-aware recommender systems are single-shot systems: they confine their contextual factors to ones that are observable at the start of the session [HMB14]. The advantage of conversational systems is that they handle the case where requirements (e.g. context, the user's mood, her ephemeral goals, etc.) are not fully observable or uncertain [PC08].

The goals of CRSs are important as they help to determine how to manage the dialog between the system and the user and they also help determine how to evaluate the performance of conversational systems. Our review starts by looking at these goals.

6.1 Goals of CRSs

A conversational system typically has two goals: effectiveness and efficiency [MGS02b, CP06]. Effectiveness is the degree to which the system helps the user to accomplish her task. In a CRS, for example, it might be whether the user finds a relevant recommendation or some broader measure of user satisfaction. Efficiency in this context is not about algorithmic run-time; rather, it is a measure of the effort involved in completing the task (perhaps in terms of time, total number of user actions with the system's user interface, number of interaction cycles, or cognitive load).

In a CRS, satisfaction is to be maximized (effectiveness) while effort is to be minimized (efficiency) [JM]. The two goals may be in conflict. For example, a system that provides explanations alongside recommendations may demand more effort from its user (due to the time taken to comprehend the explanations) but in some cases explanations have been found to increase user satisfaction. Or, to give another example, a user who takes more time exploring the item space may become more aware of the trade-offs between the items available, perhaps resulting in more confidence in her final choice, at the expense of the extra time taken.

6.2 Characterizing CRSs

Conversational recommender systems help users navigate through a complex item space to find an item of interest. During the dialog, the CRS must select items for recommendation and elicit feedback from the user, which it must take into account before making the next set of recommendations.

CRSs can be characterized in many different ways, such as: who is going to control the dialog; what form does user feedback take; and how will the user's preferences (both long-term, revealed by the user's profile, and ephemeral, revealed by the feedback) be used to produce the recommendations in each cycle of the dialog. We explore each of these in the following sections.

6.2.1 Initiative

In a conversation, e.g. with a CRS, at any time, the agent (user or system) controlling the conversation is said to have the *initiative*. In some cases, initiative always rests with the system; in other cases, it always rests with the user; and in some cases, it may flip between the two.

6.2.1.1 System-initiative

In *system-initiative* interactions, the system initiates all interactions and waits for the appropriate responses from the user before moving on to the next interaction.

At any time, both the system and the user know what the kinds of next possible actions by the user are, because they must respond to the system's request. For example, systems in which user preferences are elicited exclusively through 'question-answering' are examples of system-initiative: the system asks questions about a recommended item [LHZ14] or set of items [CRH16, CBL$^+$18] or feature values [ZCA$^+$18, SZ18], and the user must answer these questions.

These systems are usually simple to build and are applicable and well-suited to applications that are task-oriented where the task is quite specific. There is the potential for an efficient dialog, since there are no user-initiated digressions. However, users may feel over-constrained and there are challenges in making sure that the dialog has a natural 'flow' from the point of view of the user.

6.2.1.2 User-initiative

In *user-initiative* dialogs, it is the user who makes requests, and the system that must respond to the requests. A simple search system might be an example. In this case, the user takes the initiative by entering a query. The role of the system is to find sets of relevant resources and its only contribution to the dialog is to list these resources. But even allowing, for example, voice input to such a system is likely to change the dialog from user-initiative to mixed-initiative since the system may need to seize the initiative in order to seek clarification or confirmation of its interpretations of the user's input.

6.2.1.3 Mixed-initiative

In the system-initiative and user-initiative dialogs, the roles are fixed. But finding an item to consume is a two-way information exchange. The system knows the item space: which items are possible/impossible and which are available/ unavailable while the user knows (perhaps imperfectly) her preferences. It makes sense for each agent to contribute to the dialog whatever it is best suited to contribute and at the most appropriate time. Thus, the system's contributions to the dialog help users to focus on what they will reveal of their preferences, and the user's contributions help the system to focus what it reveals of the item space [Bri02]. Such a dialog is referred to as *mixed- initiative* [AGH99]. In such interactions, initiative shifts back and forth between the user and the system.

The advantage of mixed-initiative dialogs is that the system can guide the user,

Table 6.1: Summary of past research based on dialog initiative.

System proposed by	System-initiative	User-initiative	Mixed-initiative
Burke et al. [BHY97]	✓		
Smyth & Cotter [SC00]	✓		
Shimazu [Shi02]	✓		
McGinty & Smyth [MGS02a]	✓		
Thompson et al. [TGL04]			✓
Faltings et al. [FPTV04]	✓		
Nguyen & Ricci [NR04]	✓		
McCarthy et al. [MRMS05]	✓		
Reilly et al. [RMMS05]	✓		
Ricci & Nguyen [RN07]	✓		
Nguyen & Ricci [NR08]	✓		
McCarthy et al. [MSS10]	✓		
Li & Pu [CP10]	✓		
Vig et al. [VSR11]			✓
Salem & Hong [SH13]	✓		
Gupta & Chakraborti [GC13]	✓		
Loepp et al. [LHZ14]	✓		
Hariri et al. [HMB14]	✓		
Christakopoulou et al. [CRH16]	✓		
Zhang et al. [ZCA+18]	✓		
Sun & Zhang [SZ18]	✓		
Christakopoulou et al. [CBL+18]	✓		
Pan et al. [PCM18]	✓		
Nguyen & Ricci [NR18]	✓		

but the user is also free to say anything she wants, e.g. to ask questions, to introduce new 'topics', and so on. On the downside, these systems are difficult to build: the system must keep track of its own agenda, remember which parts of this agenda have been completed, interpret user actions even when they seem irrelevant to the agenda, and bring the user back to the agenda to ensure task completion.

6.2.1.4 Summary

Table 6.1 summarizes past research on CRSs in terms of dialog initiative. Most of the systems in the literature have system-initiative dialog.

Thompson et al.'s Adaptive Place Advisor [TGL04] is mixed-initiative. The system uses frames (which are sets of attribute-value pairs) and seeks to fill the frame with the user's values. The system can choose which attribute to ask about next, but the user has the flexibility to volunteer the value for an attribute different from the one suggested by the system. Similarly, Vig et al.'s Movie Tuner [VSR11] allows the user to critique tag dimensions other than the ones the system suggests.

Of the systems that we surveyed, none has a purely user-initiative dialog.

The system that we design and evaluate in the remainder of this dissertation has a system-initiative interaction strategy. In each interaction cycle, the system makes recommendations, displaying them as a tree on the screen, and the user must select the one from the tree that comes closest to what she is looking for.

6.2.2 User feedback

CRSs make use of two basic strategies to help users navigate through the item space: *navigation-by-asking* and *navigation-by-proposing* [Shi02]. Each of the two strategies depends upon different forms of user feedback. In navigation-by-asking, the user responds to questions. Usually, questions ask about specific item features, such as the maximum price she is prepared to spend on, for example, a new laptop. Hence, this form of feedback is referred to as *value elicitation*. Navigation-by-proposing collects user preferences in a less direct manner: instead of asking for specific feature details, it proposes a set of recommendations and invites the user to evaluate the recommended items. It makes use of one of the three forms of feedback: *critiquing*, *rating-based* and *preference-based* [SM03a, MS06].

The use of these different forms of feedback is influenced by four factors [MS06]: i) *cost*, i.e. the effort required by the user to provide the feedback; ii) *ambiguity*, i.e. the ability of the feedback to guide the recommender; iii) *expertise*, i.e. the level of domain knowledge needed by the user to provide the feedback; and iv) *interface*, i.e. the type of user interface required to capture the feedback. We describe the different forms of feedback with their comparative strengths and weaknesses based on these influencing factors below.

6.2.2.1 Value-elicitation

The most common and direct form of feedback is for the user to provide her preferred value or value for a specific attribute, usually in response to a question about this attribute, e.g. a user seeking a laptop and asked about her preferred

CPU might say she wants a 1.66GHz Itanium microprocessor. The challenge of building a CRS that uses value-elicitation is in equipping the CRS with a strategy for selecting an optimal set of attributes to request from the user and a logical order in which to ask about them. Attributes can be chosen for example in an heuristic manner [DC00, Sch02] or using a more sophisticated model obtained by deep reinforcement learning [SZ18], deep recurrent networks [CBL$^+$18], or multi-memory networks [ZCA$^+$18].

This form of user feedback is suitable only in domains where items are described in a structured way (e.g. a set of attribute-value pairs); strategies for selecting the next attribute to ask about may need a high level of domain knowledge; and users must be willing and able to specify their preferences as responses to direct and sometimes quite specialized questions. Demands are placed on the user interface too: it might need to allow selection from lists or the entry of values (e.g. by typing or speech). In contrast, some forms of navigation-by-proposing require only an interface that allows clicking or pointing.

6.2.2.2 Critiquing

Another form of feature-level feedback is *critiquing* where, instead of providing a specific value for an attribute, users propose 'tweaks' to attribute values that would improve a recommended item (e.g. "like this but cheaper") [BHY97].

A critique can be *unit* or *compound.* The former allows users to give feedback on a single attribute at a time [BHY97] while the latter allows users to tweak more than one attribute at a time [RMMS04]. Some people believe that compound critiquing strategies potentially lead to higher accuracy with shorter dialog length [ZP06]; others disagree [CP06].

A critique can also be classified as *system-suggested* or *user-initiated.* In system-suggested critiquing systems, the user is allowed to tweak the current recommendation by selecting one of a set of critiques offered by the system. In the early FindMe systems, these critiques were pre-designed (e.g. 'cheaper', 'quieter', 'nicer' in [BHY97]'s restuarent finder system). Because of the limitations of pre-designed critiques, Reilly et al. [RMMS04] proposed *dynamic critiquing* – the automated selection of multiple features from a range of compound critiques based on the user's current position within the item space (e.g. 'different manufacturer, lower processor speed, and cheaper' in [MRMS05]). The main advantage of these systems is that they reveal knowledge about the remaining recommendation opportunities and they accelerate the user's critiquing process if they correspond well to the user's intended feedback criteria.

In contrast, user-initiated critiquing systems do not propose pre-computed critiques, but provide a facility to users to freely create and combine critiques themselves [CP09]. Pu et al. [PCK08] presented an example critiquing agent in which a user either accepts a result, or takes a near solution by applying critiques provided for each product attribute under 'keep' (default), 'improve' and 'take any suggestion' labels. Thus, she can critique one attribute by either improving its current value (selecting 'improve') or accepting a compromised value suggested by the system (selecting 'take any suggestion'). A study has shown that user-initiated systems enable users to achieve significantly higher decision accuracy, preference certainty, and sense of control, compared to systems that provide system-suggested critiques [CP06]. This study also motivated the development of *hybrid critique-based* recommender systems that combine both the systems: *system-suggested* and *user-initiated*, in order to compensate each other. For example, Vig et al.ś Movie Tuner System [VSR11] is a hybrid system.

One characteristic of the work we have surveyed so far is that it requires structured item descriptions, usually in the form of sets of attribute-value pairs. We are aware of only one exception: in [VSR11], Vig et al. extend the critiquing idea to items whose descriptions are sets of tags. Users are free to apply unit and compound critiques, and are also allowed either to choose from the system-suggested tags or enter additional tags of their own, resulting in a mixed-initiative model of critiquing.

On the whole, critiquing provides a relatively unambiguous indication of the user's current preferences, imposes low burden on the user, may have low interface requirements (especially when critiques are pre-designed and system-suggested), and might even be usable by users with minimal understanding of the item features.

6.2.2.3 Rating-based

Rating is usually an item-level form of feedback [SM03a]. Rating is familiar in the case where a user has consumed an item, perhaps one recommended by the system, and is asked for her opinion. But here we are considering the case where a user may receive a set of recommendations and rates them without having consumed them in order to improve the next cycle of recommendations. From information about the item (e.g. a movie poster, stills from the movie, a synopsis, perhaps a trailer, and so on), the user must decide whether she is likely to enjoy the item: in a CRS, these ratings precede consumption.

In general, ratings-based feedback is a relatively low-cost form of feedback since the user only needs to express a qualitative or quantitative indication of her preference for each recommended item. However, the level of efforts increases as the number of items presented also increases [SM03a]. Since this kind of feedback has some ambiguity (e.g. exactly what it is about the item that the user thinks she will like or dislike), dialogs might be long with many recommendations and hence a lot of user feedback. This form of feedback places moderate demands on the interface: the user needs only to select her rating for each item.

6.2.2.4 Preference-based

In *preference-based* feedback, the user expresses preferences at the item level (rather than, for example at the level of features or attributes). She indicates simply which of the current recommendations she prefers. The system then may interpret this as a request for further recommendations similar to the one the user has selected ("more like this") [SM03b, GC13]. This is attractive in domains where users might struggle to articulate their preferences in more detail [SM03a]. It avoids issues about the accuracy and stability of explicit ratings [APTO09] and aligns with evidence that users prefer to compare items rather than to rate them [JBB11].

It is the lowest cost form of feedback. But if preference-based feedback is to result in efficient dialogs, account needs to be taken too of the ways in which the rejected items differ from the selected one, as is done in *comparison-based recommendation* [MGS02a]. Extensions to the work also reveal, for example, the usefulness of controlling the diversity of the recommendations in each cycle of the dialog [MGS03, SM03b, MS06].

The work on comparison-based recommendation has assumed that items have structured descriptions in the form of sets of attribute-value pairs. Differences between the attribute values of the selected and rejected items forms part of the basis for making the next set recommendations in the dialog. One of the contributions of our work described in the next chapter is that we extend the preference-based and comparison-based feedback idea to items whose descriptions are sets of features such as keywords or tags. There is a relationship here with the large body of work on *relevance feedback* in information retrieval, surveyed in, e.g., [RL03]. This work often involves modification of a query (e.g. adding terms or modifying vector weights). There is also recent work on exploratory search and information retrieval that allows users to dynamically influence document ranking by interacting with summaries of the documents' keywords or tags [dSSV16, dSBV18].

An alternative to structured representations (attribute-value pairs) and unstructured representations (sets of keywords or tags) is presented in [LHZ14], where a latent factor model is learned from user ratings, thus requiring no item descriptions at all. During a dialog, the user is repeatedly presented with, and may select between, a set of items [PCM18] that score low on a system-chosen latent factor and another set of items that score high on that factor. The user chooses between these sets of items, resulting in updates to a vector that captures the user's choices and drives the next set of recommendations. Similar work is reported in [GW15] and [CRH16] but with a focus on cold-start.

6.2.2.5 Summary

In general, there is a trade-off between the ability of a particular form of feedback to efficiently guide a CRS and the cost to the user of providing the feedback. If value-elicitation sits at one extreme, then simple preference-based feedback sits at the other: the former has high cost to the user but may focus the recommendations; the latter has low cost but is more ambiguous about what it is that the user likes about the selected item and therefore has less ability to focus the recommendations. In the end the appropriate form of feedback to use will depend largely on the characteristics of a particular application scenario.

Table 6.2 summarizes past research on CRSs based on the type of user feedback. It is noticeable that most systems using natural language for conversation use value-elicitation. A rich set of systems have been proposed that make use of critiquing. But recent years has seen an upturn in interest in preference-based user feedback.

In the next section, we characterize CRSs based on what kind of user preferences they model.

Table 6.2: Summary of past research based on user feedback.

System proposed by	Value -elicitation	Critique -based	Rating -based	Pref. -based
Burke et al. [BHY97]		✓		
Smyth & Cotter [SC00]			✓	
Shimazu [Shi02]	✓	✓		
McGinty & Smyth [MGS02a]				✓
Thompson et al. [TGL04]	✓			
Faltings et al. [FPTV04]		✓		
Nguyen & Ricci [NR04]		✓		
McCarthy et al. [MRMS05]		✓		
Reilly et al. [RMMS05]		✓		
Ricci & Nguyen [RN07]		✓		
Nguyen & Ricci [NR08]		✓		
McCarthy et al. [MSS10]		✓		
Li & Pu [CP10]		✓		
Vig et al. [VSR11]		✓		
Salem & Hong [SH13]		✓		
Gupta & Chakraborti [GC13]				✓
Loepp et al. [LHZ14]				✓
Hariri et al. [HMB14]				✓
Christakopoulou et al. [CRH16]				✓
Zhang et al. [ZCA+18]	✓			
Sun & Zhang [SZ18]	✓			
Christakopoulou et al. [CBL+18]	✓			
Pan et al. [PCM18]				✓
Nguyen & Ricci [NR18]				✓

6.2.3 User preferences

In recommender systems, and especially CRSs, a user's preferences may consist of her long-term (or persistent) preferences and her short-term (or ephemeral) preferences [SHY04, GCN+18]. We describe this in detail below.

6.2.3.1 Long-term preferences

A user's long-term preferences are ones that are relatively stable, e.g. in general, she likes comedies. In contrast, short-term preferences are more ephemeral, e.g. although she generally likes comedies, this evening she is in the mood for something more serious.

When we talk about a user profile in a recommender system, this is usually a representation of the user's long-term preferences. It might take the form of a set of features (e.g. genres) that the user likes; or it might take the form of a set of items that the user has consumed and ratings for those items. The profile might be initialized at registration time [PC08], e.g. by asking the user to select genres that she likes or by asking her to rate some movies. It grows over time as the user consumes more items.

Single-shot recommender systems generate recommendations by exploiting the user's long-term preferences [JLJ15]. No CRS solely depends upon a user's long-term preferences. If it did, it would have no basis for making an interesting set of new recommendations in each cycle of the dialog. Therefore, for a CRS, it is essential to capture short-term preferences as well as, or instead of, long-term preferences.

6.2.3.2 Short-term preferences

CRSs, in general, execute an adaptive process to personalize the current session to the user of the system [BHY97, MGS02a, TGL04]. For this, they typically exploit users' short-term preferences that may include the observable context information (e.g. time, weather, location etc.) available at the start of the session as well as the unobservable context (e.g. mood, companion, etc.) collected either explicitly by asking the user or implicitly from the feedback that the user provides within a session [HMB14].

Conversational systems exploit user feedback to improve the next cycle of recommendations. However, they usually ignore successive feedback provided within a given session. This may lead to inefficiencies on the part of the recommender system, and confusion on the part of the user [RMMS05, DLQW18]. In order to resolve this issue, Reilly et al. [RMMS05], for example, proposed an incremental approach for critiquing-based CRSs such that it maintains a critique-based user model as a set of unit critiques that the user has applied in the current session.

Another issue with CRSs is that they require some user effort while exploring the recommended items and selecting the most appropriate one. This becomes onerous specially in the case of using mobile devices with limited screen size and input modality. A few critiquing-based CRSs have been proposed to minimize user effort and improve overall recommendation efficiency by reusing the critiquing histories of other users [MSS10, SH13]. Alternatively, there have been efforts to jointly model and integrate user's long- and short- term preferences. We discuss these systems in the next sub-section.

6.2.3.3 Long- and short-term preferences

Relying solely on long-term models seems to be insufficient as these models cannot easily adapt to the user's ephemeral needs. However, long-term user models are not only suitable for generating non-contextualized recommendations but also encode valuable knowledge that can be exploited for matching immediate short-term preferences [JLJ15].

Nguyen & Ricci [NR04] proposed a preference initialization methodology integrating user's long- and short- term preference models in order to find an interesting travel product. In their approach, user's preferences are represented as a composite query consisting of three components: i) *logical query* – is a conjunction of conditions that recommended items must satisfy, e.g. price $<$ 500$; ii) *favorite pattern* – is a feature vector representing which conditions the recommender should satisfy, e.g. type of restaurant = *vegetarian*; iii) *feature importance weight vector* – is a vector whose size is the number of item features where a value between 0 and 1 represents the importance of that feature for the user. They used different knowledge sources (e.g. user's current location, her pre-travel plan, etc.) to initialize the user's preferences and update the query based on the user's feedback in the form of critiques. In their later work [NR08], they evaluated the impact on recommendation accuracy of the long- and short- term user preferences of simulated users using a critique-based mobile recommender system.

In the same line, Wu et al [WASB16] proposed inferring user intent through in-session navigation information and then updating pages of recommendations based on the inferred user intent. For this, they used a probabilistic model that incorporates current session navigation information as well as logged navigation and consumption data. The model infers interest in candidate rows of recommended items based on navigation to a certain point on the page and populates the remaining rows on the page with more relevant content that reflects the current likely predicted intent.

Du et al. [DLQW18] proposed an efficient online personalized next-item recommendation approach via long- and short-term preference learning. The approach makes use of users' long-term personalized preferences and short-term sequential actions to predict the next choices such that the model is updated in time when receiving users' new action information. They model users' past preferences by two different matrices: one for users' long-term preferences and the other matrix contributes to construct short-term preferences. They model a user's current preferences as a linear combination of her long- and short-term preferences. The long-term preference are not filtered in any way. However, they consider only a recent window of item feedback for a user's short-term preferences.

Nguyen & Ricci [NR17] proposed a method for combining users' long-term and discussion-generated preferences for group recommendations. They model the two types of preferences linearly such that when a user's preferences are not influenced by the group, the preference aggregation model should weigh more the long-term preferences. On the other hand, when discussion-generated preferences tend either to align with each other or to diverge due to the group setting, it is beneficial to take into account more the discussion-generated preferences, which helps to capture the newly arising interests of the users.

6.2.3.4 Summary

Table 6.3 summarizes past research based on the kind of preferences used. Logically, no CRSs rely solely on long-term preferences. Some systems use short-term preferences only. Others combine short-term preferences with long-term preferences. In these systems, long-term preferences might be used at the start of the dialog or they may be used jointly with short-term preferences throughout the dialog.

In the next section, we describe the evaluation of CSRs.

6.3 Evaluating CRSs

The performance of CRSs is measured in offline and online settings with simulated and real users respectively. In CRSs, a user explores the item space by navigating from one item to others, looking for better options. Most past work on CRSs assume a structured item representation (e.g. each item is represented as a set of attribute-value pairs). A user can start the navigation from a shown example item, post a critique (e.g. a cheaper camera) or select her preference (e.g. show me more like this) or rate the recommended choices (e.g. 1–5 stars). More precisely, navigating the item space involves finding items having more optimal values on

Table 6.3: Summary of past research based on user preferences.

System proposed by	Short -term	Long & Short	Feedback -type
Burke et al. [BHY97]	✓		critiquing
Smyth & Cotter [SC00]	✓		rating
Shimazu [Shi02]	✓		val elicit.
McGinty & Smyth [MGS02a]	✓		preference
Thompson et al. [TGL04]	✓		val elicit.
Faltings et al. [FPTV04]	✓		critiquing
Nguyen & Ricci [NR04]		✓	critiquing
McCarthy et al. [MRMS05]	✓		critiquing
Reilly et al. [RMMS05]	✓		critiquing
Ricci & Nguyen [RN07]	✓		critiquing
Nguyen & Ricci [NR08]		✓	critiquing
McCarthy et al. [MSS10]	✓		critiquing
Li & Pu [CP10]	✓		critiquing
Vig et al. [VSR11]	✓		critiquing
Salem & Hong [SH13]	✓		critiquing
Gupta & Chakraborti [GC13]	✓		preference
Loepp et al. [LHZ14]	✓		preference
Hariri et al. [HMB14]	✓		preference
Christakopoulou et al. [CRH16]	✓		preference
Zhang et al. [ZCA+18]	✓		val elicit.
Sun & Zhang [SZ18]		✓	val elicit.
Christakopoulou et al. [CBL+18]	✓		val elicit.
Pan et al. [PCM18]		✓	preference
Nguyen & Ricci [NR18]		✓	preference

one or several attributes, while accepting compromised values for other attributes.

In the case of evaluating with simulated users, Mc Ginty & Smyth [MGS02a, MS06] proposed an evaluation methodology that can be used for critiquing- and preference- based feedback for case-based recommendations. Each case of the case-base represents an item and is temporarily removed and used in two ways: first, each case serves as the basis for a set of random queries of varying sizes. Second, a case in the case-base which is most similar to the current target case serves as the recommendation targets during the experiments. In other words, the original target case is taken as the ideal query for a user, a generated query serves as the initial query that the user provides to the recommender, and the best case is the best available case for the user based on their ideal query. In this way, at each cycle the case most similar to the target is selected to expand in the next recommendation cycle.

Although simulations are helpful to evaluate the performance of recommendation algorithms, investigating evaluation criteria from a real-user perspectives has also been recognized as a prime concern, especially for CRSs [PCH12]. Therefore, it is valuable to conduct user trials to measure how users perceive the recommendations generated by a CRS.

In user trials, this is specifically performed *qualitatively* with real users by asking questions and *quantitatively* by analyzing their interaction data. If the task is unambiguous, it can be simply measured by absolute task success (e.g. the system books the right flight for its user). Pu & Kumar [PK04] proposed a standard protocol for evaluating critiquing-based conversational systems. According to the protocol, the user is asked to identify the most preferred item (e.g. an apartment) and then she is asked to navigate from that item to find items that achieve various trade-offs, e.g. "Can you find something closer? You can compromise on one and only one attribute". Finally, the system's performance is measured in terms of users' task completion time and error rates, i.e. the total number of wrong answers. Similarly, Chen & Pu [CP06] measured decision accuracy quantitatively by the fraction of participants that switched to a different, better option when they were asked to view all alternatives in the database. The lower is the fraction, the better is the recommendation accuracy. They also measured decision effort in terms of the number of products viewed, critiquing effort, etc.

It is often infeasible to run complete user satisfaction studies after each change in the system. For this reason, it is useful to have performance evaluation heuristics that correlate well with human satisfaction. As discussed earlier (Section 6.1), effectiveness and efficiency are the two basic factors that define a user's overall satisfaction with a conversational recommender system. Effectiveness is typically measured in terms of task completion success (i.e. hit-rate), the user's confidence in the choice she has made, the system's ability to recommend a variety of options to the user (i.e. diversity), the system's potential to suggest serendipitous recommendations (i.e. surprise), and the system's ability to minimize user effort (i.e. decision effort). The effectiveness is often measured in each cycle of the dialog [TGL04, RMMS05]. On the other hand, efficiency is typically measured in terms of task completion time in seconds or simply by the total number of recommendation cycles elapsed to reach to the item of interest.

6.4 Conclusion

In this chapter, we have reviewed work on conversational recommender systems, from their goals to their evaluation. We characterized them based on various criteria including interaction initiative, the form of feedback they use in the dialog, and how they model users' preferences.

In the next chapter we introduce Navigation-by-Preference (*n-by-p*), which extends the work on conversational recommenders by showing how we can use preference-based feedback in domains where items have unstructured descriptions. Navigation-by-Preference allows us to combine a user's long- and short-term preferences and can be configured to take short-term preferences into account in a variety of ways. We also propose a new protocol for user trials that has a novel way of giving a user some short-term preference, different from her long-term preference.

Chapter 7

Preference Chains

7.1 Introduction

As we have discussed, the process by which a user selects an item to consume (e.g. a movie to watch) is often an iterative one: the user's requirements may be uncertain, or even erroneous, and may be refined by exposure to the items a system presents to her [PC08]. This motivates the use of conversational recommender systems that allow repeated interactions between the user and the system.

In this chapter, we present a new conversational recommender system, which we call Navigation-by-Preference (*n-by-p*). As we will explain in more detail, it works in a content-based way on unstructured item descriptions such as sets of keywords or tags. It uses *preference-based feedback*, in which a user simply selects from the current n recommendations the one that comes closest to the kind of item she wants to consume [SM03b]. It combines short-term preferences (from the user feedback) with long-term preferences (from the user profile). But it can be configured in a variety of ways including: how much it weights the long-term preferences; whether it takes account of feedback given only in the most recent cycle or feedback given throughout the dialog; whether it uses only the positive feedback revealed by the item that the user selects in a cycle or whether it also takes into account the negative feedback revealed by the items that the user does not select.

Figure 7.1 shows an example *n-by-p* conversation in the form of a tree of recommendations. In the example, the user provides a *seed* movie, in this case, *A Beautiful Mind.* The system recommends three movies (*My Big Fat Greek*

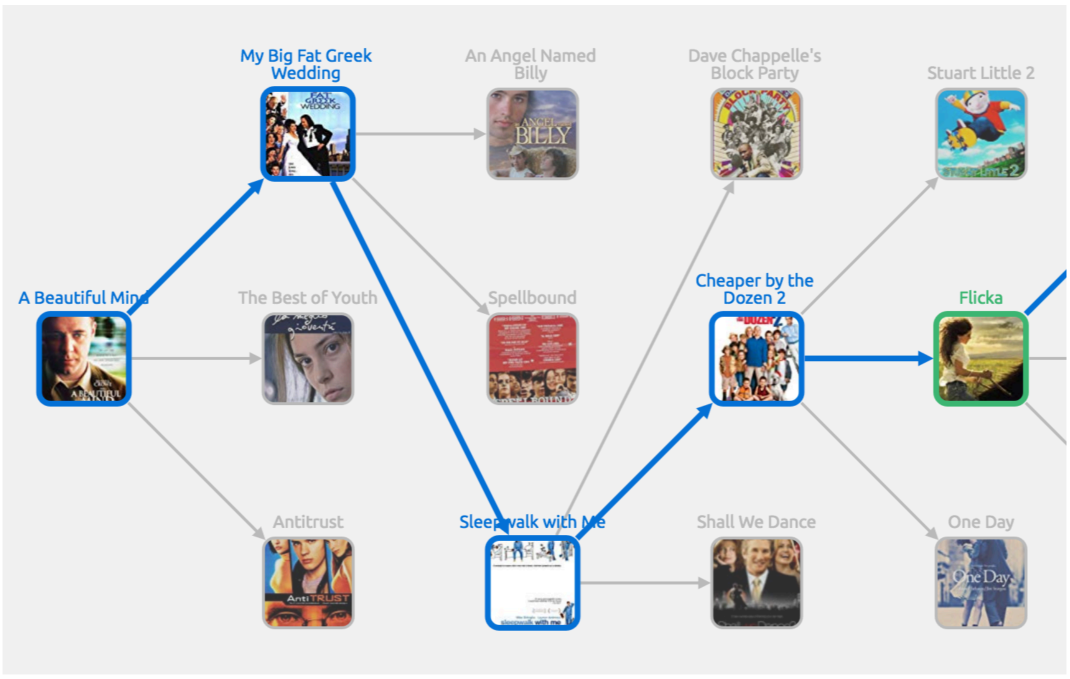


Figure 7.1: An example of navigation-by-preference, showing a *preference chain*.

Wedding, *The Best of Youth*, and *Antitrust*). The user indicates that, of these three, *My Big Fat Greek Wedding* comes closest to the kind of movie she wants to watch. From this feedback, in the next cycle, the system recommends a further three movies. This continues for several more cycles, forming a sequence of selected movies that we call a *preference chain*, which is highlighted in the diagram.

The work we report in this chapter makes the following contributions:

- Past work on preference-based feedback, and most past work on conversational recommender systems, assumes that items have structured descriptions, usually in the form of sets of attribute-value pairs. *n-by-p* works on unstructured item descriptions, such as sets of keywords or tags. Hence, the chapter extends preference-based feedback to these representations.

- Past work on preference-based feedback, and most past work on conversational recommender systems, takes into account only the most recent round of feedback: long-term preferences and feedback revealed in earlier cycles are ignored. *n-by-p* can be configured to also take into account long-term preferences, or feedback from earlier in the dialog, or both.

- Past work on preference-based feedback, and most past work on conversational recommender systems, has been evaluated only in offline experiments. In this chapter, as well as offline experiments, we report the results of a user trial. Furthermore, our trial adopts a novel protocol for exploring short-term preferences in recommender system evaluation.

Just like r-by-e, there are different versions of *n-by-p*: feature-based and neighbour-based. We will describe neighbour-based first.

7.2 Neighbour-Based *n-by-p*

Navigation-by-Preference (*n-by-p*) is a novel conversational recommender system that uses preference-based feedback on unstructured item representations. In high-level terms, *n-by-p* works as follows. The user selects a seed item, s, typically from her user profile. *n-by-p* recommends n candidate items to the user. Let's call the set of recommendations R. From R, the user selects one item — the one which comes closest to what she wants to consume on this occasion (e.g. the movie that is closest to the kind of movie she wants to watch tonight). Her choice, s, becomes the 'query' item for the next cycle. This repeats until she finds a recommendation $s \in R$ that she wants to consume.

Let $\mathbb{I}$ be the set of all items. *n-by-p* works in a scenario of implicit ratings, where the user profile of the active user P is simply a set of items that the active user likes, $P \subseteq \mathbb{I}$. Candidate items for the active user u, I, are items that might be recommended to u; these are simply the items that are not in the user's profile: $I = \{i : i \in \mathbb{I} \setminus P\}$.

Each item i has a set of features (e.g. keywords or tags), denoted F_i. The similarity of two items, $\text{sim}(i, j)$, is given by the Jaccard similarity of their features, $\frac{|F_i \cap F_j|}{|F_i \cup F_j|}$. In the version of *n-by-p* that we are describing in this section, the features are used only indirectly. When reasoning about an item i, this version of *n-by-p* uses a *set of related items*, N_i, which are candidate items that are neighbours of i, i.e. whose similarity to i exceeds a threshold θ: $N_i = \{j \in I \setminus \{i\} : \text{sim}(i, j) > \theta\}$.

Suppose, during the dialog, the user selects an item s from the latest set of recommendations R. Which items might we recommend in the next interaction cycle? The obvious answer is: candidates that are similar to s, i.e. N_s. This is the essence of preference-based feedback: recommending items that are like the one that the user selected ("more like this"). But, not every member of N_s should be a candidate for recommendation in the next cycle. We exclude any previously recommended items, for example, since we choose not to recommend an item more than once in a dialog. Furthermore, we might take into account the 'negative' feedback that we have received. We know that the user's most recent choice, $s \in R$, is preferred to the other members of R ($R \setminus \{s\}$). We might discard some members of N_s to reflect this fact. We will postpone the details of different ways of doing this to subsequent sections. But, by way of notation, let's refer to this subset of N_s as the *selection-consistent candidates* and denote it by S, $S \subseteq N_s$.

The next question is: how do we choose n items from S to recommend to the user? We want to ensure that the set of recommendations R on each cycle (a) reflect the user's long-term preferences, as revealed by the user's profile; (b) reflect her short-term preferences, as revealed by the items she has chosen (and not chosen) during the dialog, especially her most recent selection; and (c) are different from each other, to ensure diversity.

A user's long-term preferences L are represented by the candidate items that are neighbours of each item in the user's profile: $L = \cup_{i \in P} N_i$. For (a) above, for each candidate $i \in S$, we can measure how much N_i overlaps with L. Her short-term preferences are given by S itself, so for (b) above, we can measure how much N_i overlaps with S as a whole. For (c), diversity, we can make sure that N_i covers parts of S that were not covered by other recommendations. Again, we postpone the details to subsequent sections.

In fact, we propose two versions of neighbour-based *n-by-p*: *Navigation-by-Immediate-Preference* (*n-by-i-p*) and *Navigation-by-Cumulative-Preference* (*n-by-c-p*). Both make use of the user's long-term preferences (given by L); they differ in how they handle short-term preferences. In *n-by-i-p*, only feedback from the most recent cycle affects the next cycle; in *n-by-c-p*, we use item weights to allow feedback from earlier cycles to also affect the next cycle. We now present each in detail.

7.2.1 Neighbour-based *n-by-i-p*

Neighbour-based *n-by-i-p* is shown as Algorithm 4. It initializes the selection-consistent candidates, S, to candidate items that are neighbours of the user-provided seed, N_s. It repeatedly makes a set of n recommendations, R, drawn from S. In each cycle, the user makes a selection, $s \in R$. She also chooses an action, a. In the case that $a = STOP$, the dialog is over; the user has chosen to consume s; in the case that $a = CONTINUE$, s is not ideal but it is the member

Algorithm 4 Neighbour-based *n-by-i-p*

```
Input: $s$: seed item, chosen by the user
       $L$: candidate items that are neighbours of items in $P$
       $\pi$: update policy
       $n$: number of recommendations per cycle
Output: $i \in I$, a candidate item to consume
1: $S \leftarrow N_s$
2: $Tabu \leftarrow \varnothing$
3: while $|S| > n$ do
4:     $R \leftarrow \textsc{Recommend}(S, L, n, Tabu)$
5:     $s, a \leftarrow$ user chooses $s \in R$ and $a \in \{STOP, CONTINUE\}$
6:     if $a = STOP$ then
7:         return $s$
8:     $S \leftarrow \textsc{Update}(s, R \setminus \{s\}, \pi)$
9:     $Tabu \leftarrow Tabu \cup R$
```

of R that comes closest to satisfying the user, and the dialog continues. In the latter case, S is updated based on the item that the user selected (s), the ones she did not select ($R \setminus s$) and an update policy (π), yet to be explained. But it excludes from S members of R, since we choose not to recommend an item more than once in a given dialog.

We will look at recommendation and update in more detail.

7.2.1.1 Recommending

Recommendation in *n-by-i-p* (Algorithm 5) greedily selects the n members of S that have highest score. The score for an item i, score(i, S, L, R), depends on the selection-consistent candidates S (to capture short-term preferences), the candidates that are neighbours of items in the user profile L (to capture long-term preferences), and the incrementally-constructed set of recommendations R (so that the next item to be added to R can be different from the ones that have already been added, thus ensuring a level of diversity to the final set of recommendations).

More formally, the score for inserting a candidate i into a (partial) recommendation list R given S and L is a linear combination of short- and long-term scores:

$$\text{score}(i, S, L, R) = (1-\eta) \cdot \text{ovrlp}(i, S, R) + \eta \cdot \text{ovrlp}(i, L \setminus S, R) \tag{7.1}$$

η in $[0, 1]$ controls the balance between the short- and long-term scores. In the

Algorithm 5 Neighbour-based *n-by-i-p*'s Greedy Recommender

```
Input: $S$: selection-consistent candidates
       $L$: candidate items that are neighbours of items in $P$
       $n$: number of recommendations per cycle
Output: $R$, a list of $n$ recommendations
1: function Recommend($S, L, n, Tabu$)
2:     $Candidates \leftarrow S \setminus Tabu$
3:     $R \leftarrow [\,]$
4:     while $|R| < n$ and $|Candidates| > 0$ do
5:         $i^* \leftarrow \operatorname*{arg\,max}_{i \in Candidates} \operatorname{score}(i, S, L, R)$
6:         append $i^*$ to $R$
7:         $Candidates \leftarrow Candidates \setminus \{i^*\}$
8:     return $R$
```

second term, we pass in $L \setminus S$ instead of L, to ensure that members of S do not get double-counted in the scoring.

ovrlp(i, X, R) simply measures the overlap between i's neighbours (excluding any that are already covered by R) and a set of items X (where X is either S or $L \setminus S$; see Eq. 7.1):

$$\operatorname{ovrlp}(i, X, R) = \frac{2 \cdot |(N_i \setminus \operatorname{cov}(X, R)) \cap X|}{|N_i \setminus \operatorname{cov}(X, R)| + |X \setminus \operatorname{cov}(X, R)|} \tag{7.2}$$

In essence, the numerator is the size of the intersection of the candidate items that are neighbours of i (N_i) and the set X, $N_i \cap X$. However, we do not want to reward i with a high score if it is similar to items that we have already decided to recommend, R. We define cov(X, R) to be the items in X that are already covered by neighbours of items in the partial recommendation list R, i.e. $\operatorname{cov}(X, R) = \bigcup_{j \in R} N_j \cap X$.

The denominator in Eq. 7.2 is the sum of the sizes of both N_i and X (excluding cov(X, R)). If we divide only by the size of X, we would not penalize items that have high overlap simply by virtue of having more neighbours (large N_i). Since we divide by both, we also double the numerator in compensation, resulting in a Harmonic mean.

Notice how the definition of ovrlp makes no explicit reference to features. We are reasoning about items through their neighbours (N_i) and considering how the set of neighbours covers the set of items X. Features are being used, but only implicitly: the set N_i contains candidate item that have high feature similarity

Table 7.1: Update policies: $\textsc{Update}(s, R \setminus \{s\}, \pi)$
Here, s is the selected item; for brevity we write R' for the set of rejected items, $R' = R \setminus \{s\}$; and we write $\mathrm{Sims}(j)$ for the set $\{\mathrm{sim}(j, j') : j' \in R'\}$.

$\pi = \textit{Strict}$: $S \leftarrow N_s \setminus \bigcup_{j \in R'} N_j$
Discard a member of N_s if it is a neighbour of any member of R'.

$\pi = \textit{Relaxed}$: $S \leftarrow N_s \setminus \bigcap_{j \in R'} N_j$
Discard a member of N_s if it is a neighbour of every member of R'.

$\pi = \textit{Open}$: $S \leftarrow N_s$
Do not discard any members of N_s (i.e. ignore R').

$\pi = \textit{Mean}$: $S \leftarrow N_s \setminus \{j \in N_s : \mathrm{sim}(j, s) < \mathrm{mean}(\mathrm{Sims}(j))\}$
Discard a member of N_s if its similarity to s is less than the mean of its similarities to the members of R'.

$\pi = \textit{Max}$: $S \leftarrow N_s \setminus \{j \in N_s : \mathrm{sim}(j, s) < \max(\mathrm{Sims}(j))\}$
Discard a member of N_s if its similarity to s is less than the greatest of its similarities to the members of R'.

with i.

7.2.1.2 Updating

Suppose a user selects an item $s \in R$ and chooses action *CONTINUE*. Then we must update the selection-consistent candidates S. Remember that, in *n-by-i-p*, previous rounds of feedback are forgotten. In essence, S becomes N_s, candidate items that are neighbours of the most recently selected item, s. But, we might also take into account the negative feedback: the rejected items $R \setminus \{s\}$. We have defined five different update policies π, which differ in how they make use of the members of $R \setminus \{s\}$. One policy (*Open*) ignores the rejected items entirely; another (*Strict*) ensures that no rejected item will be recommended in the next cycle; and three policies (*Relaxed*, *Mean* and *Max*) lie somewhere between these two extremes. The details are given in Table 7.1.

7.2.2 Neighbour-based *n-by-c-p*

Navigation-by-Immediate-Preference ignores feedback that the user gives in all but the most recent cycle of the dialog. This means that the current set of recommended items may contain items that are not related to ones that the user se-

Algorithm 6 Neighbour-based *n-by-c-p*

```
Input: $s$: seed item, chosen by the user
       $L$: candidate items that are neighbours of items in $P$
       $\rho$: re-weighting policy
       $n$: number of recommendations per cycle
Output: $i \in I$, a candidate item to consume
 1: $S \leftarrow N_s$
 2: $Tabu \leftarrow \varnothing$
 3: Reweight$(s, \varnothing, \rho)$
 4: while $|S| > n$ do
 5:     $R \leftarrow$ Recommend$(S, L, n, Tabu)$
 6:     $s, a \leftarrow$ user chooses $s \in R$ and $a \in \{STOP, CONTINUE\}$
 7:     if $a = STOP$ then
 8:         return $s$
 9:     $S \leftarrow$ Update$(s, R \setminus \{s\}, \pi = Open)$
10:     Reweight$(s, R \setminus \{s\}, \rho)$
11:     $Tabu \leftarrow Tabu \cup R$
```

lected earlier, or items that are related to ones that the user rejected earlier. This may confuse the user or prolong the dialog. To better utilize user feedback, we formalize *Navigation-by-Cumulative-Preference* (*n-by-c-p*): each candidate item $i \in I$ has a weight w_i; weights are initially zero; but items are re-weighted based on the user's feedback; and, when scoring items for recommendation, overlap is weight-sensitive.

n-by-c-p (Algorithm 6) is very similar to *n-by-i-p* (Algorithm 4). There are two main differences. The first is that, when it calls Update, it always uses the *Open* update policy. This means that the selection-consistent candidates for the next cycle are all candidates that are neighbours of the most recently selected item: no item is discarded. The second difference is that the algorithm calls Reweight. It calls it at the start, so that item weights reflect the user's choice of seed; and it calls it after the user has given feedback, so that weights reflect the user's most recent selection. We will explain recommendation and re-weighting in more detail.

7.2.2.1 Recommending

Recommendation in *n-by-c-p* is almost identical to recommendation in *n-by-i-p* (shown earlier as Algorithm 5). The only difference is that in line 5, *n-by-c-p* selects the item using a different scoring function. Line 5 becomes $i^* \leftarrow$

Algorithm 7 Neighbour-based *n-by-c-p*'s Greedy Recommender

```
Input: $S$: selection-consistent candidates
       $L$: candidate items that are neighbours of items in $P$
       $n$: number of recommendations per cycle
Output: $R$, a list of $n$ recommendations
1: function RECOMMEND($S, L, n, Tabu$)
2:     $Candidates \leftarrow S \setminus Tabu$
3:     $R \leftarrow [\,]$
4:     while $|R| < n$ and $|Candidates| > 0$ do
5:         $i^* \leftarrow \arg\max_{i \in Candidates} \text{wscore}(i, S, L, R)$
6:         append $i^*$ to $R$
7:         $Candidates \leftarrow Candidates \setminus \{i^*\}$
8:     return $R$
```

$\arg\max_{i \in Candidates} \text{wscore}(i, S, L, R)$. Complete pseudocode is presented in Algorithm 7. The weighted score, wscore(i, S, L, R), is given by:

$$\text{wscore}(i, S, L, R) = (1 - \eta) \cdot \text{wovrlp}(i, S, R) + \eta \cdot \text{wovrlp}(i, L \setminus S, R) \tag{7.3}$$

We define wovrlp(i, X, R) as follows:

$$\text{wovrlp}(i, X, R) = \frac{2 \cdot \sum_{j \in (N_i \setminus \text{cov}(X,R)) \cap X} w_j}{|N_i \setminus \text{cov}(X, R)| + |X \setminus \text{cov}(X, R)|} \tag{7.4}$$

This is very similar to Eq. 7.2 except that overlap between an item j in $N_i \setminus \text{cov}(X, R)$ and X now counts for w_j, whereas in Eq. 7.2 it is as if $w_j = 1$ for all j. The weights will give prominence to items that are more important, based on user feedback during the dialog. So we turn now to how the weights are modified.

7.2.2.2 Re-weighting

In each cycle, *n-by-c-p* updates the weight w_i of each candidate item i to incorporate the most recent feedback:

$$w_i \leftarrow w_i + \Delta w_i \qquad \forall i \in I \tag{7.5}$$

We have seven different policies ρ for computing Δw_i, and they are given in Table 7.2. In the policies in Table 7.2, we use a binary indicator C_{ij}, whose value indicates whether items i and j are related. Specifically, they are related if i is

Table 7.2: Re-weighting policies: $\textsc{Reweight}(s, R \setminus \{s\}, \rho)$
Here, s is the selected item; for brevity we write R' for the set of rejected items, $R' = R \setminus \{s\}$; and d is the depth of the tree.

$\rho =$ *Directional (Direc)*:

$$\Delta w_i = C_{is} - \textstyle\sum_{j \in R'} C_{ij}$$

Only considers whether i is a neighbour of s or members of R'.

$\rho =$ *Similarity (Sim)*:

$$\Delta w_i = C_{is} \cdot \mathrm{sim}(i,s) - \textstyle\sum_{j \in R'} C_{ij} \cdot \mathrm{sim}(i,j)$$

Considers similarities when i is a neighbour of s or members of R'.

$\rho =$ *Similarity-Mean (Smean)*:

$$\Delta w_i = C_{is} \cdot \mathrm{sim}(i,s) - \mathrm{mean}(\{C_{ij} \cdot \mathrm{sim}(i,j) : j \in R'\})$$

Considers similarity when i is a neighbour of s, and the mean similarity when i is a neighbour of members of R'.

$\rho =$ *Similarity-Max (Smax)*:

$$\Delta w_i = C_{is} \cdot \mathrm{sim}(i,s) - \max(\{C_{ij} \cdot \mathrm{sim}(i,j) : j \in R'\})$$

Considers similarity when i is a neighbour of s, and the maximum similarity when i is a neighbour of members of R'.

$\rho =$ *Recency (Rcy)*:

$$\Delta w_i = C_{is} \cdot \mathrm{sim}(i,s)^{1/d} - \textstyle\sum_{j \in R'} C_{ij} \cdot \mathrm{sim}(i,j)^{1/d}$$

As per *Sim* above, but with updates counting more for later recommendations.

$\rho =$ *Recency-Mean (Rmean)*:

$$\Delta w_i = C_{is} \cdot \mathrm{sim}(i,s)^{1/d} - \mathrm{mean}(\{C_{ij} \cdot \mathrm{sim}(i,j)^{1/d} : j \in R'\})$$

As per *Smean* above, but with updates counting more for later recommendations.

$\rho =$ *Recency-Max (Rmax)*:

$$\Delta w_i = C_{is} \cdot \mathrm{sim}(i,s)^{1/d} - \max(\{C_{ij} \cdot \mathrm{sim}(i,j)^{1/d} : j \in R'\})$$

As per *Smax* above, but with updates counting more for later recommendations.

one of the candidate items that are neighbours of j: $C_{ij} = 1$ if $i \in N_j$ and 0 otherwise.

The policies differ in the ways they increase Δw_i when i is related to the item that the user has just selected (given by C_{is}) and decrease Δw_i when i is related to items that the user has just rejected (given by C_{ij} for $j \in R \setminus \{s\}$). In all policies except *Direc*, the amounts added or subtracted are based on the similarities of i to s and to the members of $R \setminus \{s\}$. In three of the policies (*Rcy*, *Rmean* and

Rmax), updates that come later in the dialog count for more.

7.2.2.3 Restoring

We have implied that, in every cycle, the user must select an item s from the current set of recommendations R. In fact, in our implementation, we display all the previous recommendations on the screen also (see Figure 7.1). This affords an option that we have not explained so far. We allow a user to 'jump' back to a previous recommendation. In other words, she can decide that no member of R suits her but that some item that was recommended earlier is more suitable. She can select that earlier item, either to consume ($a = STOP$) or as the basis for a new round of recommendations ($a = CONTINUE$). We have excluded this from the pseudocode shown in this chapter (Algorithms 4 and 6) in order to keep the pseudocode simple and intelligible. In *n-by-i-p*, 'jumps' are straightforward because updates are based only on the most recent selection. In *n-by-c-p*, 'jumps' are more complicated because the weights must be restored to previous values. This can be achieved either by storing the weights for all items on every cycle or, as in our implementation, through a form of backtracking that reverses changes in weights by multiplying them by -1.

7.3 Feature-Based *n-by-p*

In this section, we will describe versions of *n-by-p* that use the item features more directly.

Each item i has a set of features (e.g. keywords or tags), denoted F_i. When reasoning about an item i, this version of *n-by-p* uses i's features, F_i, rather than its neighbouring items, N_i. Now, symbol S denotes the set of *selection-consistent features* in place of *selection-consistent candidates*. The set of selection-consistent features is a subset of F_s, i.e. $S \subseteq F_s$. Similarly, the user's long-term preferences L are now represented by the features of the items in the user's profile: $L = \cup_{i \in P} F_i$.

Feature-based *n-by-p* works on item features but still recommends items. Therefore, we need a function that will map features to items. For this, we define a function τ which takes a set of features, say F, as an input and returns a set of candidate items that have at least one feature in common with F: $\tau(F) = \{i \in I \mid F_i \cap F \neq \phi\}$.

Algorithm 8 Feature-based *n-by-i-p*

```
Input: $s$: seed item, chosen by the user
       $L$: set of features of items in $P$
       $\pi$: update policy
       $n$: number of recommendations per cycle
Output: $i \in I$, a candidate item to consume
1: $S \leftarrow F_s$
2: $Tabu \leftarrow \varnothing$
3: while $|S| > n$ do
4:     $R \leftarrow$ Recommend$(S, L, n, Tabu)$
5:     $s, a \leftarrow$ user chooses $s \in R$ and $a \in \{STOP, CONTINUE\}$
6:     if $a = STOP$ then
7:         return $s$
8:     $S \leftarrow$ Update$(s, R \setminus \{s\}, \pi)$
9:     $Tabu \leftarrow Tabu \cup R$
```

Algorithm 9 Feature-based *n-by-i-p*'s Greedy Recommender

```
Input: $S$: selection-consistent features
       $L$: set of features of items in $P$
       $Tabu$: set of already recommended items
       $n$: number of recommendations per cycle
Output: $R$, a list of $n$ recommendations
1: function Recommend$(S, L, n, Tabu)$
2:     $Candidates \leftarrow \tau(S) \setminus Tabu$
3:     $R \leftarrow [\,]$
4:     while $|R| < n$ and $|Candidates| > 0$ do
5:         $i^* \leftarrow \underset{i \in Candidates}{\arg\max}$ score$(i, S, L, R)$
6:         append $i^*$ to $R$
7:         $Candidates \leftarrow Candidates \setminus \{i^*\}$
8:     return $R$
```

7.3.1 Feature-based *n-by-i-p*

Feature-based *n-by-i-p* is shown as Algorithm 8. It is very similar to neighbour-based *n-by-i-p*. The only difference is that it initializes the selection-consistent features, S, to the features of the user-provided seed, F_s, in place of its neighbours, N_s.

We will look at recommendation and update in more detail.

7.3.1.1 Recommending

Recommendation in feature-based *n-by-i-p* (Algorithm 9) greedily selects the n members of $\tau(S)$ that have highest score. In scoring an item i, $\text{score}(i, S, L, R)$ looks the same as in Eq. 7.1 except that, in feature-based *n-by-i-p*, S denotes the selection-consistent features (to capture short-term preferences), L is the set of features of the items in the user profile (to capture long-term preferences), and R is the incrementally-constructed set of recommendations.

$\text{ovrlp}(i, X, R)$ measures the overlap between i's features (excluding any that are already covered by R) and a set of features X (where X is either S or $L \setminus S$; see Eq. 7.1):

$$\text{ovrlp}(i, X, R) = \frac{2 \cdot |(F_i \setminus \text{cov}(X, R)) \cap X|}{|F_i| + |X|} \tag{7.6}$$

In essence, the numerator is the size of the intersection of the features of i (F_i) and the set X, $F_i \cap X$.

Notice that the denominator in feature-based overlap (above) is slightly different from the denominator in neighbour-based overlap (Eq. 7.2). It does not remove already covered features from the size of the sets F_i and X. This is because we found in experiments that covering an item's features results in less diverse recommendations than covering its set of neighbours; removing already-covered features from the size of the respective sets causes even more similar recommendations and so we chose not to do this here.

7.3.1.2 Updating

When a user selects an item $s \in R$ and chooses action *CONTINUE*, the set of selection-consistent features S is updated.

Feature-based update policies differ from neighbour-based ones (see Table 7.1) in two ways: i) these policies are defined on features (e.g. F_s); and ii) they operate on item-specific feature weights (e.g. w_{fs}), as defined in Eq. 5.2.

For instance, for feature-based *n-by-i-p*, the *Max* update policy becomes:

$\pi = \textit{Max}$: $S \leftarrow F_s \setminus \{f \in F_s : w_{fs} < \max(w_{fR'})\}$.

In this way, F_s replaces N_s and w_{fs} replaces $\text{sim}(j, s)$ in all neighbour-based *n-by-i-p* update policies to become feature-based *n-by-i-p* ones.

Algorithm 10 Feature-based *n-by-c-p*

```
Input: s: seed item, chosen by the user
       L: set of features of items in P
       ρ: re-weighting policy
       n: number of recommendations per cycle
Output: i ∈ I, a candidate item to consume
 1: $S \leftarrow F_s$
 2: $Tabu \leftarrow \varnothing$
 3: Reweight($s, \varnothing, \rho$)
 4: while $|S| > n$ do
 5:     $R \leftarrow$ Recommend($S, L, n, Tabu$)
 6:     $s, a \leftarrow$ user chooses $s \in R$ and $a \in \{STOP, CONTINUE\}$
 7:     if $a = STOP$ then
 8:         return $s$
 9:     $S \leftarrow$ Update($s, R \setminus \{s\}, \pi = Open$)
10:     Reweight($s, R \setminus \{s\}, \rho$)
11:     $Tabu \leftarrow Tabu \cup R$
```

7.3.2 Feature-based *n-by-c-p*

Feature-based *n-by-c-p* (Algorithm 10) is very similar to neighbour-based *n-by-c-p* (shown earlier as Algorithm 6). The main difference is that feature-based *n-by-c-p* assigns item specific weights to the features simply by calculating their *TF-IDF* values as mentioned in Eq. 5.2.

7.3.2.1 Recommending

Recommendation in feature-based *n-by-c-p* is almost identical to recommendation in feature-based *n-by-i-p* (shown earlier as Algorithm 9). The only difference is that in line 5, *n-by-c-p* selects the item using a different scoring function. Line 5 becomes $i^* \leftarrow \underset{i \in Candidates}{\arg\max}\ \text{wscore}(i, S, L, R)$. The weighted score, wscore(i, S, L, R), is given by Eq 7.1.

For feature-based *n-by-c-p*, we define wovrlp(i, X, R) as follows:

$$\text{wovrlp}(i, X, R) = \frac{2 \cdot \sum_{f \in (F_i \setminus \text{cov}(X,R)) \cap X} w_f}{|F_i| + |X|} \tag{7.7}$$

This is very similar to Eq. 7.6 except that overlap between a feature f in $F_i \setminus \text{cov}(X, R)$ and X now counts for w_f, whereas in Eq. 7.6 it is as if $w_f = 1$ for all f. The weights will give prominence to features that are more important, based on user feedback during the dialog. So we turn now to how the weights are modified.

7.3.2.2 Re-weighting

In each cycle, feature-based *n-by-c-p* updates the weight w_f of each feature f to incorporate the most recent feedback:

$$w_f \leftarrow w_f + \Delta w_f \qquad \forall f \in F \tag{7.8}$$

As with neighbour-based *n-by-c-p*, we define seven different policies ρ for computing Δw_f in feature-based *n-by-c-p*. In their formulation, they are the same as the policies defined in Table 7.1 except w_{fj} replaces $\text{sim}(i, j)$.

7.3.2.3 Restoring

Feature-based *restoring* is identical to neighbour-based *restoring* except the fact that here weights are assigned to item features instead of item neighbours.

7.4 Offline Experiments

We designed an offline experiment, with simulated users, to evaluate the different approaches to *n-by-p*. We wanted the experiment to reveal the effect of the differences between the following:

- *neighbour-based* versus *feature-based*: The former represents an item as a set of its neighbours (similar items) in which the features are used only indirectly, while the latter represents an item as a set of its features (e.g. keywords or tags) which makes use of the item features more directly.
- *immediate* versus *cumulative*: The former takes into account only the most recent user feedback and the latter takes into account the feedback across all cycles of the dialog so far.
- *immediate*'s five update policies (Table 7.1) and *cumulative*'s seven re-weighting policies (Table 7.2). The update / re-weighting policies represent different ways of taking negative feedback into account. In all cases, once the user selects item $s \in R$, the next set of recommendations will be drawn from the selection consistent candidates (N_s) or features (or F_s). But the

policies afford different ways of handling the rejected items $R \setminus \{s\}$ or their features.

- The influence of η: η controls the balance between short-term and long-term preferences (Eqs. 7.1 and 7.3). We vary η from 0 (short-term preferences only) to 1 (long-term preferences only) in steps of 0.25. When $\eta = 0$, overlap with selection-consistent neighbours (or features) S contributes to the score but overlap with profile neighbours (or features) L does not, hence only short-term preferences are taken into account. Increasing η (up to 1) shows the effect of taking long-term preferences into account and, indeed, when $\eta = 1$, short-term preferences are ignored.

We also considered different values (0.03, 0.06, 0.09) for the threshold θ in the definition of N_i. However, we found that values for $\theta \in \{0.06, 0.09\}$ result in recommendations with very low hit-rates. Therefore, we only show results for $\theta = 0.03$.

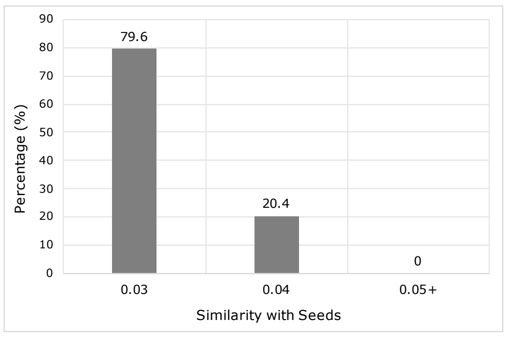
Twelve variations of *n-by-p* with five values of η gives 60 configurations for each of the two representations. This justifies the use of an offline experiment: we could not recruit enough participants to compare so many configurations in a user trial. Instead, we use the offline experiment to help us decide which configurations to use in a user trial.

7.4.1 Experiment settings

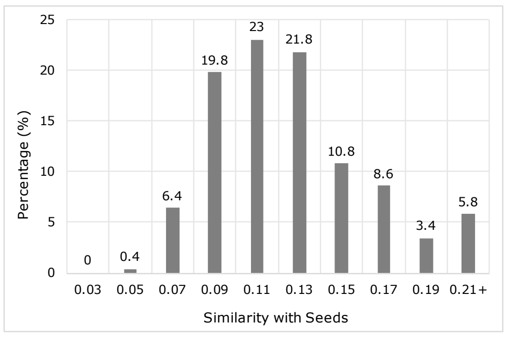
We used the hetrec2011-movielens-2k dataset that was described in Section 2.4.

We randomly selected 500 users from the dataset to use in the experiments. In *n-by-p*, user profiles simply contain items that the user likes (Section 7.2). We treated ratings in the dataset of 4 and 5 as 'likes', so active user u's profile P is given by $\{i : r_{u,i} \geq 4\}$. Ratings are otherwise not used in our experiments.

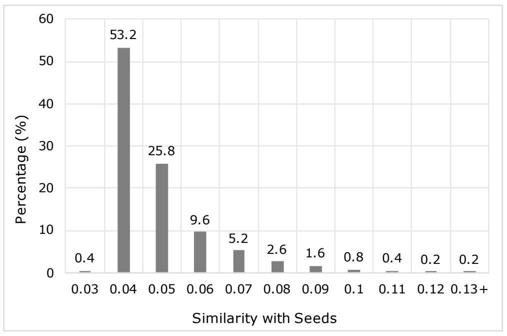
We want to simulate dialogs between each of these users and each of the different configurations of *n-by-p*. The initial seed is chosen at random from the user's profile. But there comes a problem in modeling the simulated user's preferences. Her long-term preferences are obvious: they are given by her profile P. But how do we simulate her short-term preferences? Given a set of $n = 3$ recommendations in each cycle, how do we simulate her preference for one of these over the others? We cannot have her choose the $s \in R$ randomly: that is not the same as exhibiting a short-term preference. Neither can we have her choose the one that is most



(a) *least similar*



(b) *most similar*



(c) *random*

Figure 7.2: Distribution of *targets* for different strategies by their similarity with *seeds*.

similar to the items in her profile P because this would make her short-term preferences the same as her long-term preferences. We follow, e.g., [MGS02a]: in advance, we choose at random a *target item* t from N_s. In each cycle, from the current set of $n = 3$ recommendations, the user will select the one that is most similar to the target: $\arg\max_{i \in R} \text{sim}(i, t)$. The simulated dialog stops when the target is one of the recommendations, $t \in R$, or after 15 cycles in the case where t itself does not get recommended.

We take three approaches to selecting the target item, in order to simulate dialogs with different 'difficulty' levels. In one set of dialogs, the target item is the candidate item that is *least similar* to the initial seed and thus represents the most difficult case. In another set of dialogs, the target item is the candidate that is *most similar* to the seed, and thus represents the easiest case. Finally, we have a set of dialogs where the target item is chosen at *random* from the candidates with uniform probabilities. Figure 7.2 contains histograms that show how similar the targets are to the initial seeds under these three different 'difficulty' levels. In principle, random targets are intermediate in difficulty, although in practice random targets are not very similar to the initial seeds and so the distributions in Figures 7.2a and 7.2c have similar skew.

In each cycle, we measure hit-rate up to that cycle (i.e. the proportion of users who have been recommended their target item) and the Jaccard similarity between the item that the simulated user selects in that cycle and her target ($\text{sim}(s, t)$), which we average over all users. We have also measured the diversity of the $n = 3$ recommendations in that cycle, again averaged over all users; and the average surprise of the recommendations in that cycle, averaged over all users. For diversity and surprise, we used definitions given in Section 7 of [KB16].

Table 7.3: Offline experiment for neighbour-based *n-by-p*. The table shows the hit-rate, i.e. the proportion of the 500 simulated users who find their target item within 15 cycles. All systems use $\theta = 0.03$. All differences except those shown in italics are statistically significant with respect to the corresponding version for $\eta = 0.0$ (two-sample Z-test, with $p < 0.05$).

Target		*Neighbour-based n-by-i-p*					*Neighbour-based n-by-c-p*						
type	η	*Strict*	*Relaxed*	*Open*	*Mean*	*Max*	*Direc*	*Sim*	*Smean*	*Smax*	*Rcy*	*Rmean*	*Rmax*
	0.00	0.044	0.076	0.152	0.118	0.080	0.096	0.046	0.242	0.240	0.014	0.156	0.146
	0.25	*0.024*	0.028	0.102	0.054	0.042	*0.076*	*0.032*	*0.282*	*0.244*	*0.014*	*0.140*	*0.140*
least similar	0.50	*0.030*	0.014	0.030	0.020	0.032	0.060	*0.026*	**0.320**	*0.256*	*0.006*	*0.118*	*0.142*
	0.75	*0.026*	0.012	0.004	0.012	0.030	0.048	*0.032*	0.300	*0.232*	*0.006*	*0.114*	*0.136*
	1.00	*0.032*	0.004	0.006	0.014	0.022	0.062	*0.038*	*0.196*	*0.202*	*0.010*	0.094	*0.130*
	0.00	0.342	0.274	0.504	0.426	0.358	0.438	0.312	0.576	0.514	0.262	0.448	0.422
	0.25	*0.332*	*0.270*	*0.466*	*0.382*	*0.348*	*0.402*	*0.300*	0.686	*0.522*	*0.262*	0.520	*0.402*
most similar	0.50	0.122	0.086	0.142	0.176	0.152	*0.384*	*0.342*	**0.746**	*0.500*	*0.304*	0.552	*0.430*
	0.75	0.052	0.074	0.058	0.068	0.106	0.338	0.146	0.720	0.414	0.122	*0.402*	0.308
	1.00	0.032	0.074	0.024	0.058	0.066	0.192	0.084	0.546	0.384	0.046	0.298	0.246
	0.00	0.102	0.042	0.204	0.146	0.098	0.112	0.070	0.308	0.282	0.028	0.204	0.174
	0.25	0.058	*0.034*	0.152	0.072	0.044	*0.112*	0.040	*0.338*	*0.270*	*0.016*	*0.196*	*0.158*
random	0.50	0.032	*0.046*	0.036	0.038	0.050	*0.084*	0.040	**0.390**	*0.250*	*0.018*	*0.222*	*0.154*
	0.75	0.016	*0.038*	0.014	0.014	0.042	0.072	*0.042*	0.388	*0.266*	*0.018*	*0.162*	*0.134*
	1.00	0.012	*0.030*	0.010	0.024	0.042	0.070	0.040	0.230	0.220	*0.020*	0.112	0.122

Table 7.4: Offline experiment for feature-based *n-by-p*. The table shows the hit-rate, i.e. the proportion of the 500 simulated users who find their target item within 15 cycles. All systems use $\theta = 0.03$. All differences except those shown in italics are statistically significant with respect to the corresponding version for $\eta = 0.0$ (two-sample Z-test, with $p < 0.05$).

Target type	η	*Feature-based n-by-i-p*					*Feature-based n-by-c-p*						
		Strict	*Relaxed*	*Open*	*Mean*	*Max*	*Direc*	*Sim*	*Smean*	*Smax*	*Rcy*	*Rmean*	*Rmax*
least similar	0.00	0.144	0.162	0.154	0.154	0.148	0.166	0.132	0.228	0.160	0.168	0.218	0.206
	0.25	*0.160*	*0.162*	*0.164*	*0.172*	0.190	*0.166*	*0.150*	*0.250*	*0.192*	*0.150*	*0.226*	*0.210*
	0.50	*0.144*	*0.162*	*0.146*	*0.174*	*0.142*	*0.178*	*0.136*	*0.252*	*0.178*	*0.138*	*0.238*	*0.200*
	0.75	0.072	0.096	0.084	0.090	0.074	*0.166*	*0.154*	***0.272***	*0.160*	*0.148*	*0.260*	*0.170*
	1.00	0.024	0.026	0.026	0.020	0.028	0.216	*0.136*	*0.258*	*0.138*	0.090	*0.242*	0.102
most similar	0.00	0.999	0.999	0.999	0.999	0.998	0.998	0.824	0.912	0.846	0.818	0.896	0.854
	0.25	0.952	0.952	0.950	0.954	0.950	*0.998*	*0.828*	*0.912*	*0.840*	*0.810*	*0.900*	*0.856*
	0.50	0.814	0.816	0.798	0.812	0.812	*0.998*	*0.830*	*0.910*	*0.846*	*0.826*	*0.900*	*0.856*
	0.75	0.386	0.392	0.402	0.400	0.386	***1.000***	*0.822*	*0.942*	*0.846*	*0.822*	*0.902*	*0.846*
	1.00	0.056	0.056	0.056	0.056	0.054	0.746	0.616	0.790	0.612	0.520	0.728	0.542
random	0.00	0.134	0.210	0.204	0.210	0.186	0.208	0.200	0.312	0.230	0.186	0.266	0.250
	0.25	*0.152*	*0.216*	*0.192*	*0.188*	0.198	*0.234*	*0.186*	*0.302*	*0.210*	*0.170*	*0.286*	*0.258*
	0.50	*0.126*	0.154	0.134	0.148	0.152	*0.246*	*0.188*	*0.334*	*0.218*	*0.184*	*0.304*	*0.236*
	0.75	0.068	0.066	0.066	0.058	0.058	*0.232*	*0.154*	***0.352***	*0.190*	*0.152*	*0.300*	*0.220*
	1.00	0.016	0.012	0.018	0.016	0.014	*0.258*	0.020	0.256	*0.206*	0.096	*0.260*	0.128

7.4.2 Experiment results

Table 7.3 and Table 7.4 show the results for hit-rate. The columns of the table are the different versions of *n-by-p*. The rows are divided into blocks, one block per strategy for choosing the target item. Rows within blocks are for different values of η.

Neighbour-based *n-by-p*. For *n-by-i-p*, the highest hit-rate for each target type is obtained by using the *Open* update policy and with $\eta = 0$. The *Open* policy is the one that does not take the negative feedback into account; and using $\eta = 0$ means that long-term preferences are ignored. We see that, for *n-by-i-p*, increasing the value of η nearly always reduces hit-rates. But there are exceptions where values of η other than 0 give better hit-rates. We also see that policies, such as *Relaxed*, that make most use of the negative feedback, have among the lowest hit-rates.

n-by-c-p for the most part has higher hit-rates than *n-by-i-p*, which means that taking previous feedback into account is advantageous. For several of the *n-by-c-p* re-weighting policies, $\eta = 0$ again gives the best results, with hit-rates decreasing as η is increased, but again with exceptions. Of the seven different re-weighting policies, *Smean* is clearly the best. *Smean* with $\eta = 0.5$ attains the highest hit-rates among all twelve approaches for all three target types (values in bold in the Table 7.3). Since it is a clear winner, we plot further results for this approach only.

Feature-based *n-by-p*. Different update policies attain the highest hit-rate for different target types. In *n-by-i-p*, for 'least similar' targets, the *Max* update policy attains the highest hit-rate at $\eta = 0.25$; for 'random' targets, the *Relaxed* update policy attains the highest hit-rate at $\eta = 0.25$. However, for 'most similar' targets, the highest hit-rate is obtained by nearly all the five update policies with $\eta = 0$. It indicates that in the 'most similar' case, targets are easily reached and indeed they can be reached without taking the user's long-term preferences into account.

We also see that *n-by-i-p* nearly always attains higher hit-rates for low values of η (i.e. $\eta \in \{0, 0.25\}$). We also see that policies such as *Strict*, that make most use of the negative feedback, have among the lowest hit-rates. But there are exceptions where the target type is 'most similar' and $\eta \in \{0, 0.25\}$.

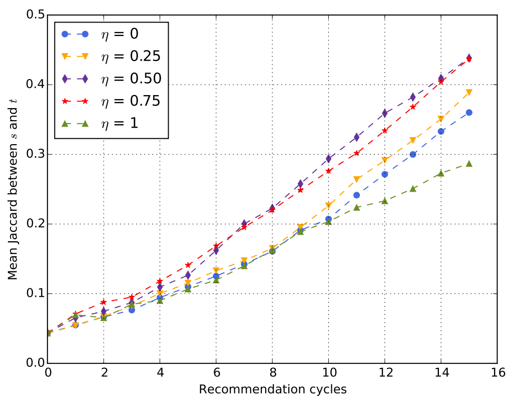
Here again, *n-by-c-p* for the most part has higher hit-rates than *n-by-i-p*, which means that taking previous feedback into account is advantageous. All seven of the different re-weighting policies tend to attain higher hit-rates for $\eta \in \{0.25, 0.5, 0.75\}$, but their differences with their corresponding $\eta = 0$ versions are not statistically significant. Of the seven re-weighting policies, *Smean* with $\eta = 0.75$ is clearly the best for 'least similar' and 'random' targets, while *Direc* with $\eta = 0.75$ attains 100% hit-rate for 'most similar' target types. In the case of feature-based *n-by-p*, the results show that 'most similar' targets are easily reached. Hence, we will focus on 'least similar' and 'random' target types.

Among all twelve approaches, we find that *Smean* with $\eta = 0.75$ attains the highest hit-rate for 'least similar' and 'random' target types. (However, this is not statistically significant with respect to *Smean* at $\eta = 0$.) Since *Smean* with $\eta = 0.75$ is a clear winner, we plot further results for this approach only.

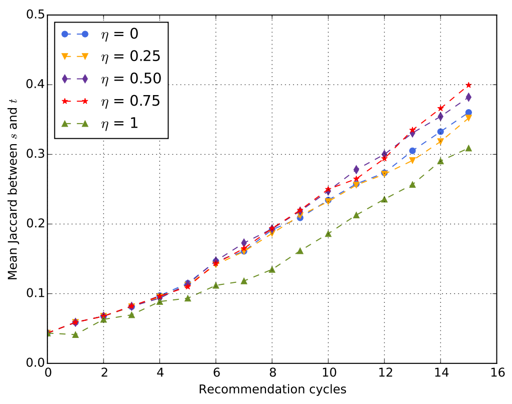
Neighbour-based vs. feature-based *n-by-p*. For reasons given earlier, we now focus on the *Smean* re-weighting strategy in *n-by-c-p*. For random targets, Figure 7.3 shows how neighbour-based and feature-based *n-by-c-p* perform over 15 cycles for different values of η. The different graphs show the different metrics that we presented in Section 7.4.1. As explained, a dialog stops when the target item is one of the recommendations, therefore not all dialogs run for the full 15 cycles. In Figure 7.3, if a dialog stops before the 15th cycle, we *forward fill* the value of the metric to subsequent cycles. For example if a dialog stops at cycle 8 then, when computing the mean similarity, etc. in cycles 9 to 15, we include that dialog's values from cycle 8. This ensures that each value that we plot is an average over 500 users. If we did not do this then, in later cycles, we would be plotting an average for a smaller number of users than in the earlier cycles. Plotting over a smaller number of users makes it harder to see trends: differences arise simply by the extra variation that comes from averaging over fewer users.

In Figures 7.3(a) and (b), we see that similarity with the target increases near linearly: as the interaction proceeds, the system leads the user ever closer to her target. The slope of *neighbour-based n-by-c-p* is steeper than its *feature-based* equivalent: it gets closer to the target more quickly.

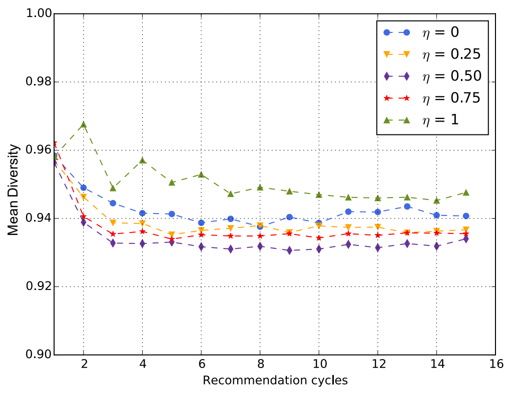
Figures 7.3(c) and (d) show recommendation diversity. For all configurations, it mostly decreases up to 4 cycles (but by only a small amount) and then remains almost the same. Decreasing diversity implies convergence on the item of interest.



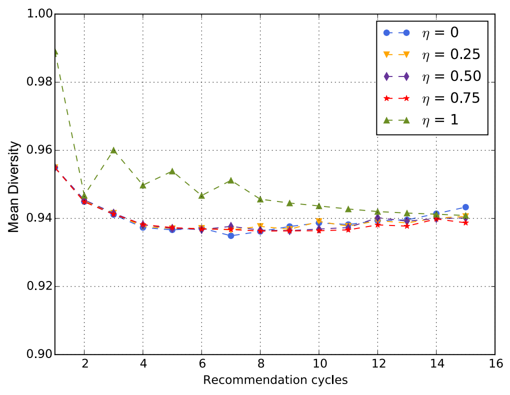
(a) Mean Jaccard similarity in each cycle between the user's selected item and their target item for neighbour-based *n-by-p*



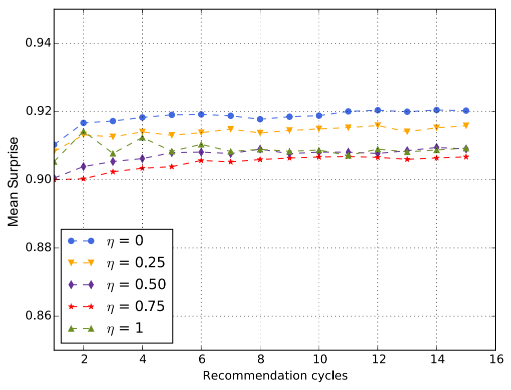
(b) Mean Jaccard similarity in each cycle between the user's selected item and their target item for feature-based *n-by-p*



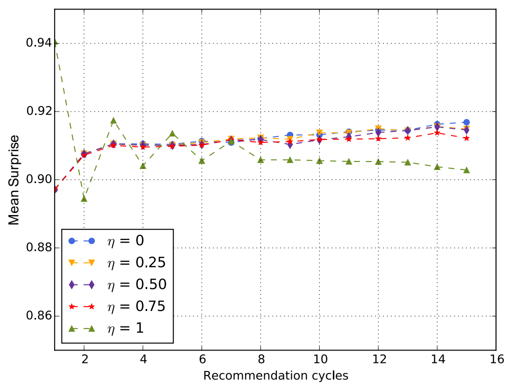
(c) Mean *Diversity* of the recommendation sets in each cycle for neighbour-based *n-by-p*



(d) Mean *Diversity* of the recommendation sets in each cycle for feature-based *n-by-p*



(e) Mean *Surprise* of the recommendations in each cycle for neighbour-based *n-by-p*



(f) Mean *Surprise* of the recommendations in each cycle for feature-based *n-by-p*

Figure 7.3: Results per cycle for *neighbour-based* and *feature-based* *n-by-c-p* with $\rho = Smean$ and η ranges in $[0.0 - 1.0]$ for *random* targets.

Figures 7.3(e) and (f) show values for surprise. For all configurations, surprise increases very slightly, which indicates that the process takes the user away from her profile.

7.5 User Trial

RQ-1: Does *neighbour-based n-by-p* generate more diverse, serendipitous, and relevant recommendations than *feature-based n-by-p*?

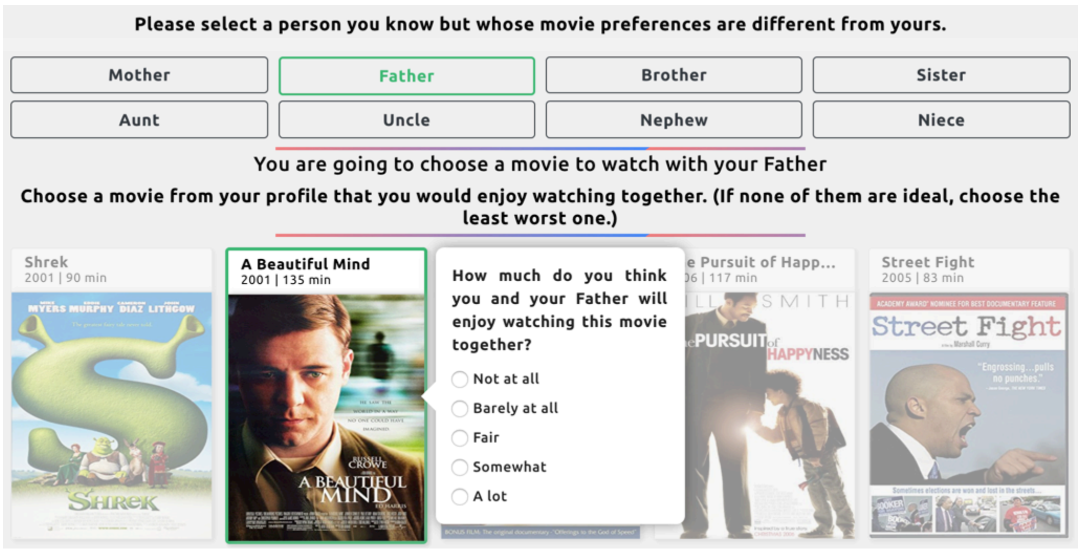
RQ-2: How beneficial is it to exploit a user's long-term preferences along with her short-term preferences?

We built a web-based system in order to conduct a user trial. In this trial, we wanted to reveal the effect of using: i) *neighbour-based* and *feature-based* item representations; and ii) long-term preferences along with short-term preferences. Hence, for both the representations, we chose to use a system with η not equal to 0.0. The obvious choice was the best-performing configuration from our offline experiment, namely *n-by-c-p* with ρ = *Smean* as its re-weighting policy and $\eta = 0.5$ for *neighbour-based* and $\eta = 0.75$ for *feature-based*. We compare it with a baseline system that is as similar as possible but which does not take long-term preferences into account, namely *n-by-c-p* with ρ = *Smean* but with $\eta = 0.0$. In this section, we will designate these systems by *nb-smean@0.5* and *nb-smean@0.0* for *neighbour-based*, and *fb-smean@0.75* and *fb-smean@0.0* for *feature-based* representations, respectively.

We recruited participants through personal email lists and Twitter. In total, 278 people attempted the trial, of whom 204 completed it and have their results reported here. Participants were fully anonymized and we collected no demographic data. However, our method of recruiting participants leads us to speculate that our participants are predominantly young, educated males. They were not rewarded for participation in any way.

We use the movie dataset that we used for the offline experiments. However, to increase the chances of user familiarity with the movies, we use only movies released between the years 2000 and 2011 inclusive: 1851 ($\approx 30\%$) of the 5992 movies in the whole dataset.

The user trial is a between-subject trial: participants are assigned at random to interact with one of the four recommenders. Of those who completed the trial,



Please select a person you know but whose movie preferences are different from yours.

Mother | Father | Brother | Sister
Aunt | Uncle | Nephew | Niece

You are going to choose a movie to watch with your Father

Choose a movie from your profile that you would enjoy watching together. (If none of them are ideal, choose the least worst one.)

How much do you think you and your Father will enjoy watching this movie together?

- Not at all
- Barely at all
- Fair
- Somewhat
- A lot

Figure 7.4: Parts of screenshots showing selection of a companion and a seed movie.

exactly 51 interacted with each of them.

7.5.1 User trial protocol

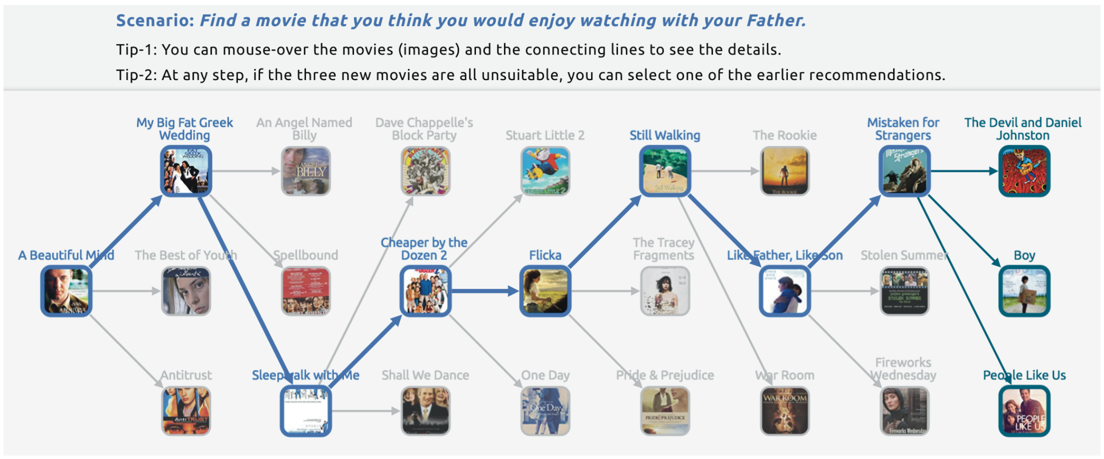
Each participant began by creating a user profile containing 10 movies. The instructions were that the movies should be ones that the user likes. The user interface offers both a scrollable grid of movies and a search box to enable the user to find these movies.

The user profile captures a user's long-term preferences. The challenge in designing an experiment of this kind is to create the conditions under which a user also has ephemeral (short-term) preferences that she wants to satisfy [PK04]. Most likely, because of these ephemeral preferences, the user should be dissatisfied to some extent with recommendations based purely on her profile, because these will satisfy only her long-term preferences. We rejected the idea of picking a target item and showing it to the user. We felt that this would lead to an approximation to the offline experiments that we have already described, where in every cycle the (simulated) user always selects the recommended item that is most similar to the target. What we wanted was a scenario in which a user would have an ephemeral goal, but where she would not know exactly what movie she wanted to watch, and yet where she would be able to make reasonably consistent judgments about a set of recommendations on the basis of that ephemeral goal.

The strategies for doing this in [PK04] rely on having structured item descriptions (e.g. sets of attribute-value pairs), which we do not have. In the end, we designed a novel protocol, which we believe is one of the contributions of our work. The scenario is that the user is trying to find a movie to watch with another person, hence she has to find one that she thinks both she and her putative companion will enjoy watching *together*. From a list of eight people (Mother, Father, Brother, Sister, Aunt, Uncle, Nephew, Niece), we ask her to select a person she knows but whose movie preferences differ from her own (see the top third of Figure 7.4). We did not include options such as Partner or Friend to make it more likely that her putative companion's preferences would differ from her own. We tell her that she must find a movie to watch with this person and we ask her to choose from her profile the movie that she thinks is the least worst movie to watch together with this person (see the middle third of Figure 7.4). We ask her how much she thinks they will enjoy watching the movie together (*Not at all*, *Barely at all*, *Fair*, *Somewhat*, *A lot*) (see the lower third of Figure 7.4). If she thinks they will enjoy watching the movie *Somewhat* or *A lot*, we ask her to repeat the whole process (selecting a different person) in the hope of finding a scenario where short-term preferences will differ from long-term preferences. At most, a user goes through this process a total of three times. The movie that she has selected from her profile at this point becomes the initial seed in the dialog.

We believe this scenario satisfies the criteria above: the user has an ephemeral goal, does not know exactly what movie she wants, but can make judgments when faced with descriptions of movies that we recommend. We emphasize that this protocol is simply a way of creating a scenario in which a user has an ephemeral goal. We are not building a group recommender system. In most work on group recommender systems, the recommender has representations of the tastes of each member of the group (e.g. a user profile for each group member) and it must reconcile these possibly conflicting tastes, which it does using one or more of a variety aggregation methods [JS07]. That is not the situation here: in our user trial, the recommender does not have any explicit representation of the other person's tastes.

Now that the scenario has been established and the seed has been chosen, a dialog of eight cycles begins. In each cycle, the system displays the next $n = 3$ recommendations, building a tree from left to right on the screen (Figure 7.5). The user can mouse-over the nodes and edges to find out movie details and keywords that connect movies, respectively. She must choose the recommendation that she thinks she and her putative companion will most enjoy watching together. If none of the three recommendations seem right, the user can choose a movie



Scenario: ***Find a movie that you think you would enjoy watching with your Father.***

Tip-1: You can mouse-over the movies (images) and the connecting lines to see the details.

Tip-2: At any step, if the three new movies are all unsuitable, you can select one of the earlier recommendations.

Figure 7.5: A screenshot showing a completed dialog with the system.

from earlier in the tree, in which case the system reverts to an earlier state (see Section 7.2.2.3). We require every user to run the system for a full eight cycles, so that the tree has a depth of eight, even if she sees a movie earlier that she thinks is ideal, so that every participant's responses are based on the same number of movies on the screen.

At the end of the dialog, the screen will be displaying a tree, rooted by the seed and containing 24 recommended movies (see Figure 7.5). We ask the user to select one of the 24 movies, the one that she thinks she and her putative companion will most enjoy watching together. Then we ask her five questions:

- *Familiarity*: Have you actually seen the movie <*selected movie*> before?
- *Relevance (Rel.)*: How much do you think you and your <*selected person*> will enjoy watching <*selected movie*> together?
- *Serendipity (Srdp.)*: Is <*selected movie*> a pleasantly surprising recommendation?
- *Effectiveness (Effc.)*: Did you find the recommendations helpful?
- *Satisfaction (Sats.)*: Did you enjoy using the system?

The user chooses between Yes and No in answer to the question about *Familiarity*. For the other questions, she chooses from a 5-point scale: *Not at all*; *Barely at all*; *Fair*; *Somewhat*; and *A lot*.

7.5.2 User trial results

204 participants completed the trial, 51 per system. Table 7.5 and Table 7.6 summarize their responses for the two representations. We describe them below separately.

Neighbour-based *n-by-p*

- *Familiarity*: 62.8% of users of *nb-smean@0.5* have actually seen their selected movie compared with 54.9% of users of *nb-smean@0.0*.
- *Relevance (Rel.)*: 76.5% of the users of *nb-smean@0.5* judged their selected movie to be one that they and their putative companion would enjoy *Somewhat* or *A lot*; in the case of *nb-smean@0.0*, this was just 49.0% of the users.
- *Serendipity (Srdp.)*: 64.7% of users of *nb-smean@0.5* thought their selected movie was pleasantly surprising (*Somewhat* or *A lot*); for *nb-smean@0.0*, this was just 41.2% of the users.
- *Effectiveness (Effc.)*: 62.7% of the users of *nb-smean@0.5* found the recommendations to be helpful (*Somewhat* or *A lot*); in the case of *nb-smean@0.0*, this was just 49.0% of the users.
- *Satisfaction (Sats.)*: 64.7% of the users of *nb-smean@0.5* enjoyed using the system (*Somewhat* or *A lot*); in the case of *nb-smean@0.0*, this was just 58.9% of the users.

On all criteria, *nb-smean@0.5* produced better recommendations. However, the difference was statistically significant only for the *Relevance* and *Serendipity* questions. (We used a one-sided Z-test for proportions, with significance level $p < 0.05$. The null hypothesis was that those preferring *nb-smean@0.0* are greater than or equal to those preferring *nb-smean@0.5*, ignoring those who were neutral i.e. who answered *Fair.*)

Feature-based *n-by-p*

- *Familiarity*: 52.9% of users of *fb-smean@0.75* have actually seen their selected movie; the percentage was the same for the users of *fb-smean@0.0*.

Table 7.5: Users' responses for *neighbour-based* systems to survey questions.

User's	*nb-smean@0.5*				*nb-smean@0.0*			
Response	*Rel.*	*Srdp.*	*Effc.*	*Sats.*	*Rel.*	*Srdp.*	*Effc.*	*Sats.*
Not at all	0	2	0	1	3	5	5	2
Barely at all	1	4	5	5	7	4	4	5
Fair	11	12	14	12	16	21	17	14
Somewhat	18	19	21	20	12	12	16	18
A lot	21	14	11	13	13	9	9	12

Table 7.6: Users' responses for *feature-based* systems to survey questions.

User's	*fb-smean@0.75*				*fb-smean@0.0*			
Response	*Rel.*	*Srdp.*	*Effc.*	*Sats.*	*Rel.*	*Srdp.*	*Effc.*	*Sats.*
Not at all	2	7	4	3	1	6	4	1
Barely at all	1	9	7	5	3	8	9	6
Fair	13	13	14	11	17	18	14	14
Somewhat	16	14	16	14	18	14	15	18
A lot	19	8	10	18	12	5	9	12

- *Relevance (Rel.)*: 68.6% of the users of *fb-smean@0.75* judged their selected movie to be one that they and their putative companion would enjoy *Somewhat* or *A lot*; in the case of *fb-smean@0.0*, this was 58.8% of the users.
- *Serendipity (Srdp.)*: 43.1% of users of *fb-smean@0.75* thought their selected movie was pleasantly surprising (*Somewhat* or *A lot*); for *fb-smean@0.0*, this was just 37.3% of the users.
- *Effectiveness (Effc.)*: 51.0% of the users of *fb-smean@0.75* found the recommendations to be helpful (*Somewhat* or *A lot*); in the case of *fb-smean@0.0*, this was just 47.0% of the users.
- *Satisfaction (Sats.)*: 62.7% of the users of *fb-smean@0.75* enjoyed using the system (*Somewhat* or *A lot*); in the case of *fb-smean@0.0*, this was just 58.8% of the users.

On all criteria except familiarity, *fb-smean@0.75* produced better recommendations. However, the differences were not statistically significant for any of the five questions. (We used a one-sided Z-test for proportions, with significance level $p < 0.05$. The null hypothesis was that those preferring *fb-smean@0.0* are greater than or equal to those preferring *fb-smean@0.75*, ignoring those who were neutral i.e. who answered *Fair*.)

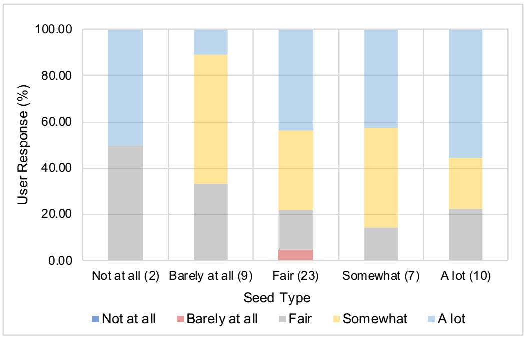
Neighbour-based vs. feature-based. We have found that *nb-smean@0.5* and *fb-smean@0.75* performed better with respect to their corresponding baselines. It confirms that user's long-term preferences should be combined with her short-term preferences. Now, we compare the winning neighbour-based and feature-based systems.

- *Familiarity*: 62.8% of users of *nb-smean@0.5* have actually seen their selected movie compared with 52.9% of users of *fb-smean@0.75.*
- *Relevance (Rel.)*: 76.5% of the users of *nb-smean@0.5* judged their selected movie to be relevant; in the case of *fb-smean@0.75*, this was just 68.6% of the users.
- *Serendipity (Srdp.)*: 64.7% of users of *nb-smean@0.5* thought their selected movie was pleasantly surprising; for *fb-smean@0.75*, this was just 43.1% of users.
- *Effectiveness (Effc.)*: 62.7% of the users of *nb-smean@0.5* found the recommendations to be helpful; in the case of *fb-smean@0.75*, this was just 51.0% of the users.
- *Satisfaction (Sats.)*: 64.7% of the users of *nb-smean@0.5* enjoyed using the system; in the case of *fb-smean@0.75*, this was just 62.7% of the users.

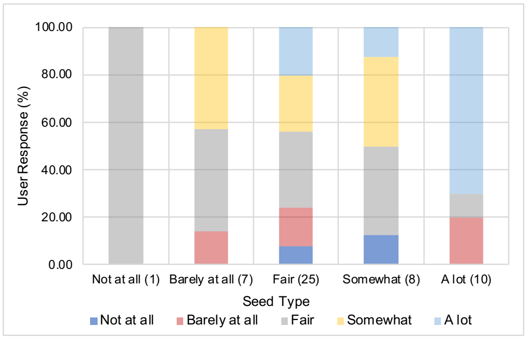
Thus, it can be seen that *nb-smean@0.5* produced better recommendations. However, the difference was statistically significant only for the *Serendipity* and *Effectiveness* questions. (We used a one-sided Z-test for proportions, with significance level $p < 0.05$. The null hypothesis was that those preferring *fb-smean@0.75* are greater than or equal to those preferring *nb-smean@0.5*, ignoring those who were neutral i.e. who answered *Fair*).

7.5.2.1 Change in relevance

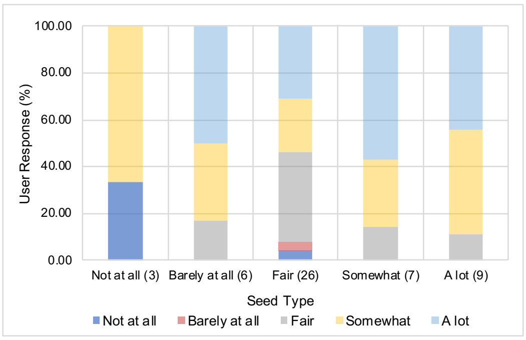
As we have reported, participants in the user trial who used *nb-smean@0.5* judged their selected movie to be more relevant than did users of the other three recommenders. However, this result ignores the user's opinion of the initial seed: do the systems *improve* upon the initial seed? Figure 7.6 shows the data. For example, in the top-left diagram (*nb-smean@0.5*), the first bar shows that there were 2 people who judged the initial seed to be *Not at all* relevant; and, of these two people, one judged their final selection to be *Fair* and the other judged their final selection to be one that they would enjoy *A lot.* Similarly, the second bar shows



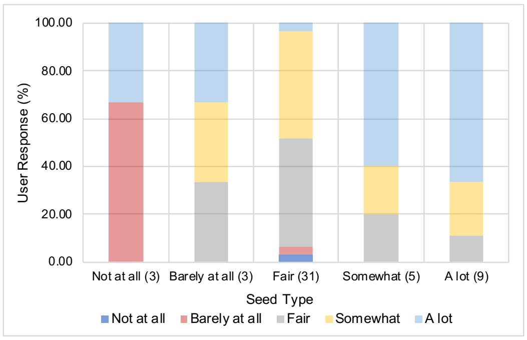
(a) *nb-smean@0.5*



(b) *nb-smean@0.0*



(c) *fb-smean@0.75*



(d) *fb-smean@0.0*

Figure 7.6: Distribution of users' opinions of their final selection with respect to their opinions of the initial seed.

that 9 people chose *Barely at all* when asked about the seed; and, of these, when asked about their final selected movie, 3 answered *Fair*, 4 answered *Somewhat* and one answered *A lot*, and so on.

We have designed a statistic that allows us to summarize all this data. We assign integers in $[1, 5]$ to the responses, $1 =$ *Not at all*, $2 =$ *Barely at all*, etc. We let α_v be the number of participants who assigned a value of v to the initial seed, i.e. α_1 is the number of people who judged the seed to be 1 ($=$ *Not at all*) suitable, α_2 is the number who judged the seed to be 2 ($=$ *Barely at all*) suitable. Similarly, let ω_v be the number of participants who assigned a value of v to the final selected movie. Then, we can compute the improvement that the system makes by taking the difference in the responses divided by the maximum improvement that could be made:

$$\text{improvement} = \frac{\sum_{v=1}^{5} v \cdot \omega_v - \sum_{v=1}^{5} v \cdot \alpha_v}{\sum_{v=1}^{5} 5 \cdot \omega_v - \sum_{v=1}^{5} v \cdot \alpha_v} \tag{7.9}$$

For *nb-smean@0.5*, improvement $= 0.5114$, whereas for *nb-smean@0.0*, improvement $= 0.0723$. Similarly, for *fb-smean@0.75* improvement $= 0.4045$, whereas for *fb-smean@0.0*, improvement $= 0.2614$. It is clear that in terms of expected movie enjoyment, *nb-smean@0.5* does a better job of taking users from their initial seeds to a final movie selection.

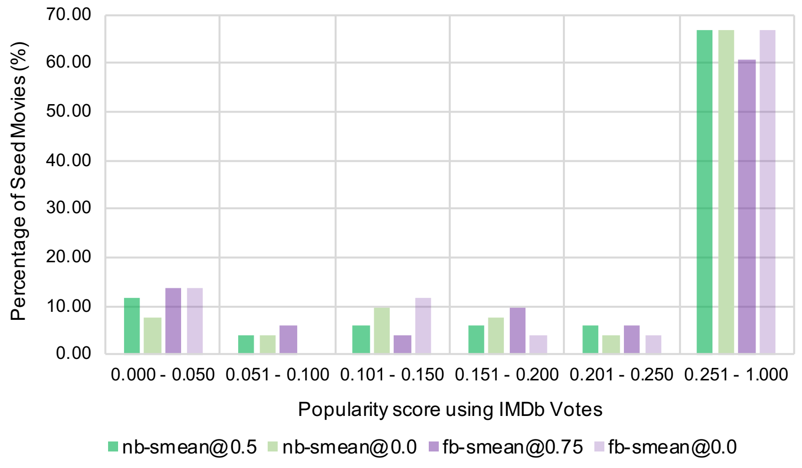
We further analyze this data by recomputing the improvement, this time excluding those cases where there is little or no scope for improvement. For example, in those cases where the user thinks that she and her putative companion would enjoy watching the seed movie *Somewhat* or *A lot*, then there is not much the recommender can do to improve on this. Another way of thinking about this is that this time we are considering only the first three bars for each of the four recommenders in Figure 7.6. Now, there are only 34 users of interest for *nb-smean@0.5* and 33 for *nb-smean@0.0*. For these users only, we obtain improvement = 0.6049 for *nb-smean@0.5* and improvement = 0.2533 for *nb-smean@0.0*. Again, there are 35 users of interest for *fb-smean@0.75* and 37 for *fb-smean@0.0* which results in improvement = 0.4756 for *fb-smean@0.75* and improvement = 0.3012 for *fb-smean@0.0*. This gives a fairer picture of the baselines (*nb-smean@0.0* and *fb-smean@0.0*), but it also shows that *nb-smean@0.5* performs even better in these more difficult cases.

7.5.2.2 Popularity bias

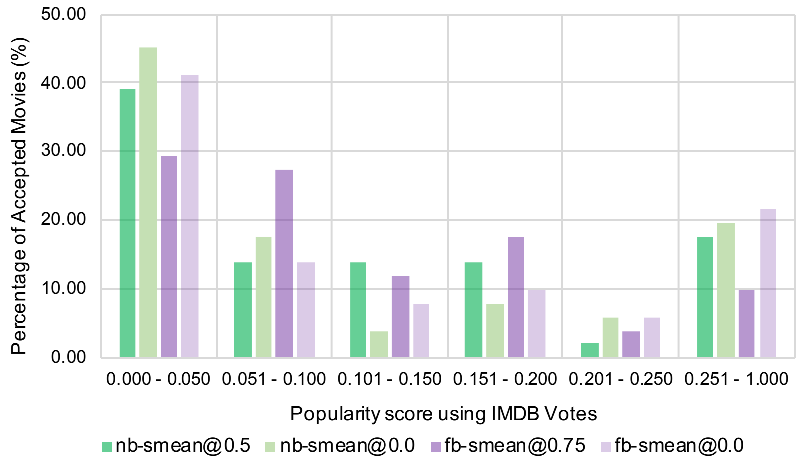
We also measure the *popularity* of the accepted movies in order to know whether their selection was biased towards more popular movies. To answer this question, we measured popularity of both the *seed* and the *accepted* movies. We define the *popularity* of a movie *i* to be the ratio of IMDB votes given to the movie *i* and the maximum IMDB votes given to a movie among those used for the user-trial.

$$Popularity(i) = \frac{IMDBVotes(i)}{IMDBVotes_{max}} \tag{7.10}$$

Figure 7.7(a) and 7.7(b) show the distribution of the popularity of the users' selected *seed* and their final *accepted* movies respectively. Over 60% of the *seed* movies are popular (i.e. *popularity* > 0.25). Nearly 20% of *accepted* movies are popular. Most of the *accepted* movies have low popularity (i.e. $0 \leq popularity \leq 0.05$). This suggests that *n-by-p* systems are not especially susceptible to popularity bias.



(a) Popularity distribution for *seed* movies



(b) Popularity distribution for *accepted* movies

Figure 7.7: Popularity distribution for all four recommenders for user selected *seed* and *accepted* movies in the trial.

7.5.2.3 User effort

Finally, we consider how much effort users expended. Table 7.7 summarizes the effort for users whose final selected movie was one they thought that they and their putative companions would like *Somewhat* or *A lot*. (We excluded other users because, in some sense, their dialog is incomplete since they have not found a satisfactory movie. Since this gives only 25 users for *nb-smean@0.0*, we used a one-sided t-test, with $p < 0.05$, with null hypothesis that *nb-smean@0.0* needs less or equal effort.)

Neighbour-based *n-by-p*

- *Nodes displayed*: As described before, we required users to explore for eight cycles. In these eight cycles, every user is shown 25 nodes (1 seed and 24 recommended movies). But a user can jump (reverting to an earlier set of recommendations), which leads to more recommendations being made. In both systems, the average is a little above 25, which shows that there was some jumping. But the average was higher for *nb-smean@0.0*, which implies a greater need for jumping ($p = 0.022$, which is statistically significant).
- *Node mouse-overs*: This refers to the average number of movies whose descriptions were viewed by mousing-over the node. More movies were examined by users of *nb-smean@0.0* ($p = 0.016$, which is statistically significant).
- *Edge mouse-overs*: Mousing over an edge reveals keywords that the movies at each end have in common. On average, more of this information was viewed by users of *nb-smean@0.0* ($p = 0.077$, which is not statistically significant).
- *Cycles*: This refers to the average number of cycles needed in order for the final selected movie to be shown. Outside of a user trial, this is the point at which the user should, in principle, stop the dialog, having found a satisfactory movie. It was slightly higher for users of *nb-smean@0.5* ($p = 0.021$, which is statistically significant). But it must be remembered that users of *nb-smean@0.5* find movies that they regard as better final choices.
- *Time taken*: This is the average task completion time in seconds. It was higher for users of *nb-smean@0.0* ($p = 0.211$, which is not statistically significant).

It can be seen that both systems require quite similar effort from users. There seems to be a little more effort in the case of *nb-smean@0.0* (more jumps and more time spent making sense of the recommendations by mousing-over their details). On the other hand, the final movie is found around the 5th or 6th cycle on average for users of *nb-smean@0.5* and around the 4th cycle for users of *nb-smean@0.0*, but it is a less satisfactory movie in the latter case.

Feature-based *n-by-p*

- *Nodes displayed*: In both systems, the average is a little above 25, which

Table 7.7: Comparison of decision effort for neighbour-based *n-by-p*. All values are averaged over participants who liked their final selected movie *Somewhat* or *A lot*.

Measure of effort	*nb-smean@0.5*	*nb-smean@0.0*
Nodes displayed	26.85	29.56
Node mouse-overs	19.28	22.84
Edge mouse-overs	10.03	11.92
Cycles	5.36	4.12
Time taken (secs.)	251.93	300.58

Table 7.8: Comparison of decision effort for feature-based *n-by-p*. All values are averaged over participants who liked their final selected movie *Somewhat* or *A lot*.

Measure of effort	*fb-smean@0.75*	*fb-smean@0.0*
Nodes displayed	29.46	27.00
Node mouse-overs	22.40	18.70
Edge mouse-overs	11.91	9.97
Cycles	4.60	5.00
Time taken (secs.)	271.67	183.07

shows that there was some jumping. But the average was higher for *fb-smean@0.75*, which implies a greater need for jumping ($p = 0.089$, which is not statistically significant).

- *Node mouse-overs*: More movies were examined by users of *fb-smean@0.75* ($p = 0.008$, which is statistically significant).
- *Edge mouse-overs*: On average, more of this information was viewed by users of *fb-smean@0.75* ($p = 0.062$, which is not statistically significant).
- *Cycles*: This was slightly higher for users of *fb-smean@0.0* ($p = 0.267$, which is not statistically significant). It must also be remembered that users of *fb-smean@0.75* find movies that they regard as better final choices.
- *Time taken*: This was higher for users of *fb-smean@0.75* ($p = 0.029$, which is statistically significant).

It can be seen that both systems require quite similar effort from users. There seems to be a little more effort in the case of *fb-smean@0.75* (more jumps and more time spent making sense of the recommendations by mousing-over their details). On the other hand, the final movie is found around the 4th or 5th cycle on average for users of *fb-smean@0.75* and around the 5th cycle for users of *fb-smean@0.0*, but it is a less satisfactory movie in the latter case.

Neighbour-based vs. feature-based *n-by-p*. Here, we are comparing *nb-smean@0.5* with *fb-smean@0.75*. We found that *fb-smean@0.75* seemed to require a little more effort (more jumps and more time spent making sense of the recommendations by mousing-over their details), but this is not statistically significant. On the other hand, the final movie is found around the 5th or 6th cycle on average for users of *nb-smean@0.5* and around the 4th or 5th cycle for users of *fb-smean@0.75*, but it is a less satisfactory movie in the latter case.

7.6 Conclusion

Navigation-by-Preference (*n-by-p*) is a conversational recommender system that works on unstructured item descriptions to help a user construct and articulate her short-term preferences, while aiming to minimize the effort of reaching an item of interest. We believe that *n-by-p* has the following characteristics:

- *Preference-based feedback*: Preference-based feedback (where the user simply selects one of the current recommendations) is the simplest form of feedback. It does not require the user to articulate which features of the item she likes or dislikes or how she wants to change them. This simplicity for the user means ambiguity for the system: there is no explicit feedback about the features [MS06].
- *Configurability*: *n-by-p* is highly configurable. We have described two variants, the first with five update policies; the second with seven re-weighting policies. These allow us to choose how to combine the user's preferences in ways that are best suited to the domain of application.
- *Interpretability*: *n-by-p* is a content-based approach based on keywords or tags that items have in common, which makes it easy to understand the relationship between pairs of consecutive items in a preference chain.

We presented an offline experiment, with simulated users, that selected the best of 60 different configurations of *n-by-p* for each of the two representations. Then we used a web-based system to conduct a user trial with a novel protocol. For neighbour-based *n-by-p*, the trial showed with statistical significance that the *nb-smean@0.5* configuration, which combines short-term preferences with long-term preferences, produces more accurate and serendipitous recommendations without greater effort from its users than its baseline *nb-smean@0.0* configuration. Similarly, for feature-based *n-by-p*, the trial confirmed that the *fb-smean@0.75* configuration, which also combines short-term preferences with long-term preferences, produces more accurate and serendipitous recommendations (but only for difficult cases) without greater effort from its users than its baseline *fb-smean@0.0* configuration. However, the difference was not statistically significant.

We also compared the winning configurations of the two representations. We found with statistical significance that *nb-smean@0.5* produces more serendipitous and effective recommendations than *fb-smean@0.75*, while both achieve almost similar accuracy with roughly equal efforts from their users. On the whole, *nb-smean@0.5* provides much greater improvement over the initial seed.

Part IV

Concluding Remarks

Chapter 8

Conclusions & Future Work

In this dissertation, we introduced a new recommendation framework, chain-based recommendations. This framework models the recommendation problem as a path search problem in the item-item similarity graph using coverage-based heuristics. We instantiated the chain-based framework in two recommendation engines under the two different themes with different motivations: i) the role of explanation in the recommendation process; and ii) the role of conversational recommendation to help the user reveal her ephemeral needs.

We evaluated both the systems using offline experiments and user trials. We note in particular that user responses in the trials were mostly positive: the proposed systems helped them understand the complex item-space, make better decisions, and find an item of interest in an efficient and informed manner.

In this chapter, we present the conclusions derived from the main contributions of this dissertation and we give some ideas for future work.

8.1 Summary of Contributions

In this section, we summarize the main contribution of this dissertation.

8.1.1 Recommendation-by-Explanation

In Chapter 4, we have proposed Recommendation-by-Explanation (*r-by-e*), a novel approach that unifies recommendation and explanations and finds relevant recommendations with explanations that have a high degree of fidelity. We presented a basic form of *r-by-e*, where items are represented as a set of their features (e.g. keywords). The empirical comparison of 25 configurations on a movie recommendation dataset showed that *r-by-e* attains higher precision than a customized version of classic content-based system, while remaining competitive on measures of diversity and surprise. Our user trial also confirmed that *r-by-e*'s recommendations are relevant, more diverse and surprising; and that their explanations are more helpful in making accurate judgements about recommendation quality.

In Chapter 5, we further extended *r-by-e* by carrying out experiments using feature weighting as well as with a more indirect item representation (i.e. *neighbour-based*), where an item is represented by a set of its neighbours (i.e. similar items) in place of its features. We defined a normalized overlap function for obtaining these extended representations. We also generalized *r-by-e*'s *chain selection* by redefining its chain scoring function.

Experimental results on a movie recommendation dataset showed that, from all four of our approaches, the *weighted feature-based* approach provides more relevant sets of recommendations, while remaining competitive on measures of diversity and serendipity. We also found that overall *neighbour-based* approaches provide more serendipitous recommendations than *feature-based* ones.

We then extended *r-by-e* so that it could apply to datasets where item features are concepts extracted from user-generated reviews and assigned sentiment score as weights to generate sentiment-aware explanation chains. We conducted another user trial in which we found that the *weighted feature-based* approach provides more relevant recommendations that are also diverse and serendipitous. The explanations that it provides are also more effective than the baseline.

8.1.2 Navigation-by-Preference

In Chapter 7, we proposed Navigation-by-Preference (*n-by-p*), a new preference-based conversational recommender that works on unstructured item representations to help a user construct and articulate her short-term preferences, while aiming to minimize the effort of reaching an item of interest. We presented *n-by-p* for both neighbour-based and feature-based representations. For each representation, we described two variants: *immediate* and *cumulative*, based on how to combine the user's long- and short-term preferences. We defined twelve techniques, five for the former and seven for the latter.

We reported results of offline experiments with simulated users that found that, out of 120 configurations (60 configurations for each approach), a configuration that includes long-term preferences, that uses both positive and negative feedback, and that uses previous rounds of feedback is the one with highest hit-rate. It also obtains the best survey responses and lowest measures of effort in a trial with real users that we conducted with a web-based system. It is noteworthy that the user trial has a novel protocol for experimenting with short-term preferences.

8.2 Future Work

Our contributions in this dissertation have introduced recommendation tools to offer a better user experience without compromising the quality of the recommendations. Mainly, we have focused on the issues of explanations and recommendation feedback. We believe that we have covered many angles on the main research topic. But our findings open lines of future work that we briefly describe in this section.

8.2.1 Explanation chain length vs. surprise

In our offline experiments for *r-by-e*, we have found a correlation between surprise and chain length for a top-n recommendation. For higher values of α, *r-by-e* generates longer chains, which, on the whole, results in more surprising recommendations.

It would be valuable to conduct user trials to measure how users perceive the surprise of the recommendations generated, for example, on different values of α.

8.2.2 Actionable explanation chains

If a system incorrectly infers preferences, then it may make poor recommendations. In most recommender systems, there is a little a user can do in this case. In principle, the user could remove items from her profile or modify their ratings in the hope of getting better recommendations. But it is not easy to know which items to remove or which ratings to modify. It is especially cumbersome to exclude an entire set of items (such as those that have specific genre in movie recommendations) when, for example, a user's interests shift.

An interesting avenue for future work would be to allow a user to interact with explanation chains to tell the system where it is going wrong. For example, removing an inappropriate chain member may result in instant revision of recommendations. Such actionable explanation chains may make the system more correctable.

8.2.3 Collaborative *r-by-e*

Another interesting research direction would be to extend the explanation chains from content-based setting to collaborative settings. In principle, chains can still be constructed using coverage heuristics but now coverage would be of ratings rather than of features. However, explaining item-item relationships among chain members will become more challenging especially if we do not want to compromise the fidelity of the recommender.

8.2.4 Parameter free *n-by-p*

Our conversational recommender *n-by-p* uses a hyperparameter to balance short- and long- term preferences of user while measuring overlap between the seed and the candidates. There is a possibility of learning this parameter using her feedback within the current session. One way of doing this is to think about how to combine work on *n-by-p* with work in recommender systems on reinforcement learning and multi-armed bandits.

8.2.5 Sequence recommendation using *n-by-p*

Short-term preferences can also be inferred from browsing behaviour, as well as the kind of feedback that we have explored in our work. Recommender systems that infer preferences and make recommendations from temporally ordered interaction events are referred to as sequence-aware recommender systems [QCJ18]. We could try situate *n-by-p* also within this body of work. As a special case, we also could explore domains in which the recommendations themselves are sequences of items (e.g. music playlists, tours of art galleries). Implicit feedback on early items in the sequence might trigger updates to the remainder of the sequence.

8.2.6 Applications to other domains

Chain-based recommendation relies on content-based principles. It can be easily extended to domains other than movie recommendations where there unstructured item descriptions are available such as news, blogs, and research papers. More challenging, and in need of further research, is the question of how easily chain-based recommendation can be extended to domains with structured item descriptions, such as e-commerce. But particularly interesting are domains where users already interact with sequences of items. In fact, e-commerce is one such example, since users tend to browse through sequences of items before making purchase decision. In entertainment and perhaps education, it is common to recommend whole sequences, such as a music playlist or a sequence of lesson. Sequence recommendation is common too in the culture and tourism domains; for example, a tour of an art gallery, museum or historic city.

Taken all together, we believe that we have motivated a new way of thinking about content-based systems that addresses several challenges (e.g. over-specialization, concentration bias) and improves the overall user experience. We have shown how chains of items, where each item reinforces its successor, can lead to more diverse and serendipitous recommendations. We hope that this work will stimulate more research into recommender systems that whose explanations achieve a higher degree of fidelity while giving more control to the user without requiring much more effort from her.

References

[Agg16] Charu C Aggarwal. An introduction to recommender systems. In *Recommender Systems*, pages 1–28. Springer, 2016.

[AGH99] JE Allen, Curry I Guinn, and E Horvtz. Mixed-initiative interaction. *IEEE Intelligent Systems and their Applications*, 14(5):14–23, 1999.

[AMRT11] Gediminas Adomavicius, Bamshad Mobasher, Francesco Ricci, and Alexander Tuzhilin. Context-aware recommender systems. *AI Magazine*, 32(3):67–80, 2011.

[AN16] Behnoush Abdollahi and Olfa Nasraoui. Explainable matrix factorization for collaborative filtering. In *Proceedings of the 25th International Conference Companion on World Wide Web*, pages 5–6. International World Wide Web Conferences Steering Committee, 2016.

[AN18] Behnoush Abdollahi and Olfa Nasraoui. Transparency in fair machine learning: the case of explainable recommender systems. In *Human and Machine Learning*, pages 21–35. Springer, 2018.

[APTO09] Xavier Amatriain, Josep M. Pujol, Nava Tintarev, and Nuria Oliver. Rate it again: Increasing recommendation accuracy by user re-rating. In *Proceedings of the 3rd ACM Conference on Recommender SSystems*, pages 173–180, 2009.

[AT05] Gediminas Adomavicius and Alexander Tuzhilin. 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):734–749, 2005.

[AT14] Panagiotis Adamopoulos and Alexander Tuzhilin. On over-specialization and concentration bias of recommendations: Probabilistic neighborhood selection in collaborative filtering systems. In *Proceedings of the 8th ACM Conference on Recommender systems*, pages 153–160. ACM, 2014.

[BCL$^+$12] Roi Blanco, Diego Ceccarelli, Claudio Lucchese, Raffaele Perego, and Fabrizio Silvestri. You should read this! let me explain you why: explaining news recommendations to users. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, pages 1995–1999. ACM, 2012.

[BD14] Derek Bridge and Kevin Dunleavy. 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*, page 22, 2014.

[BGMS05] Derek Bridge, Mehmet H. Göker, Lorraine McGinty, and Barry Smyth. Case-based recommender systems. *Knowledge Engineering Review*, 20(3):315–320, 2005.

[BHY97] Robin D. Burke, Kristian J. Hammond, and Benjamin C. Young. The FindMe approach to assisted browsing. *IEEE Expert: Intelligent Systems and Their Applications*, 12(4):32–40, 1997.

[BM05] Mustafa Bilgic and Raymond J Mooney. Explaining recommendations: Satisfaction vs. promotion. In *Beyond Personalization Workshop, IUI*, volume 5, page 153, 2005.

[BRA19] Krisztian Balog, Filip Radlinski, and Shushan Arakelyan. Transparent, scrutable and explainable user models for personalized recommendation. page 265–274, 2019.

[Bri02] Derek G Bridge. Towards conversational recommender systems: A dialogue grammar approach. In *ECCBR Workshops*, pages 9–22, 2002.

[Bur02] Robin Burke. Hybrid recommender systems: Survey and experiments. *User modeling and user-adapted interaction*, 12(4):331–370, 2002.

[BZIL18] Ramesh Baral, XiaoLong Zhu, SS Iyengar, and Tao Li. Reel: R eview aware explanation of location recommendation. In *Proceedings of the 26th Conference on User Modeling, Adaptation and Personalization*, pages 23–32. ACM, 2018.

[CBL$^+$18] Konstantina Christakopoulou, Alex Beutel, Rui Li, Sagar Jain, and Ed H Chi. Q&r: A two-stage approach toward interactive recommendation. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 139–148. ACM, 2018.

[CER$^+$08] Henriette Cramer, Vanessa Evers, Satyan Ramlal, Maarten Van Someren, Lloyd Rutledge, Natalia Stash, Lora Aroyo, and Bob Wielinga. The effects of transparency on trust in and acceptance of a content-based art recommender. *User Modeling and User-Adapted Interaction*, 18(5):455, 2008.

[CES$^+$08] Henriette Cramer, V Evers, MV Someren, S Ramlal, Lloyd Rutledge, Natalia Stash, L Aroyo, and B Wielinga. The effects of transparency on perceived and actual competence of a content-based recommender. In *Semantic Web User Interaction workshop at CHI, Florence, Italy*, 2008.

[CFLH14] Sergio Cleger, Juan M Fernández-Luna, and Juan F Huete. Learning from explanations in recommender systems. *Information Sciences*, 287:90–108, 2014.

[CHL13] Wei Chen, Wynne Hsu, and Mong Li Lee. Tagcloud-based explanation with feedback for recommender systems. In *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*, pages 945–948. ACM, 2013.

[CHT16] Shuo Chang, F Maxwell Harper, and Loren Gilbert Terveen. Crowd-based personalized natural language explanations for recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems*, pages 175–182. ACM, 2016.

[CK02] Marek Czarkowski and Judy Kay. A scrutable adaptive hypertext. In *International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems*, pages 384–387. Springer, 2002.

[CLA$^+$03] Dan Cosley, Shyong K Lam, Istvan Albert, Joseph A Konstan, and John Riedl. Is seeing believing?: how recommender system interfaces affect users' opinions. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 585–592. ACM, 2003.

[CODL18] Felipe Costa, Sixun Ouyang, Peter Dolog, and Aonghus Lawlor. Automatic generation of natural language explanations. In *Proceedings of the 23rd International Conference on Intelligent User Interfaces Companion*, page 57. ACM, 2018.

[CP05] Li Chen and Pearl Pu. 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*, pages 135–145. Citeseer, 2005.

[CP06] Li Chen and Pearl Pu. Evaluating critiquing-based recommender agents. In *AAAI*, pages 157–162, 2006.

[CP09] Li Chen and Pearl Pu. Interaction design guidelines on critiquing-based recommender systems. *User Modeling and User-Adapted Interaction*, 19(3):167, 2009.

[CP10] Li Chen and Pearl Pu. Experiments on the preference-based organization interface in recommender systems. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 17(1):5, 2010.

[CRH16] Konstantina Christakopoulou, Filip Radlinski, and Katja Hofmann. Towards conversational recommender systems. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 815–824, 2016.

[CS05] Maurice Coyle and Barry Smyth. Explaining search results. In *Proceedings of the International Joint Conference on Artificial Intelligence*, volume 19, page 1553. Lawrence Erlbaum Associates Ltd, 2005.

[CTFLH12] Sergio Cleger-Tamayo, Juan M Fernandez-Luna, and Juan F Huete. Explaining neighborhood-based recommendations. In *Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval*, pages 1063–1064. ACM, 2012.

[CW17] Li Chen and Feng Wang. Explaining recommendations based on feature sentiments in product reviews. In *Proceedings of the 22nd International Conference on Intelligent User Interfaces*, pages 17–28, 2017.

[DC00] Michelle Doyle and Pádraig Cunningham. A dynamic approach to reducing dialog in on-line decision guides. In *Proceedings of the 5th European Workshop on Case-Based Reasoning*, pages 49–60, 2000.

[DDGR07] Abhinandan S Das, Mayur Datar, Ashutosh Garg, and Shyam Rajaram. Google news personalization: scalable online collaborative filtering. In *Proceedings of the 16th international conference on World Wide Web*, pages 271–280. ACM, 2007.

[DFDCM18] Rafael M D'Addio, Eduardo P Fressato, Arthur F Da Costa, and Marcelo G Manzato. Incorporating semantic item representations to soften the cold start problem. In *Proceedings of the 24th Brazilian Symposium on Multimedia and the Web*, pages 157–164. ACM, 2018.

[DLQW18] Yingpeng Du, Hongzhi Liu, Yuanhang Qu, and Zhonghai Wu. Online personalized next-item recommendation via long short term preference learning. In *Procs. of the Pacific Rim International Conference on Artificial Intelligence*, pages 915–927, 2018.

[DMM19] Rafael M D'Addio, Ronnie S Marinho, and Marcelo G Manzato. Combining different metadata views for better recommendation accuracy. *Information Systems*, 83:1–12, 2019.

[dSBV18] Cecilia di Sciascio, Peter Brusilovsky, and Eduardo Veas. A study on user-controllable social exploratory search. In *Proceedings of the 23rd International Conference on Intelligent User Interfaces*, pages 353–364, 2018.

[DSOS13] Ruihai Dong, Markus Schaal, Michael P O'Mahony, and Barry Smyth. Topic extraction from online reviews for classification and recommendation. In *IJCAI*, volume 13, pages 1310–1316, 2013.

[dSSV16] Cecilia di Sciascio, Vedran Sabol, and Eduardo E. Veas. Rank as you go: User-driven exploration of search results. In *Proceedings of the 21st International Conference on Intelligent User Interfaces*, pages 118–129, 2016.

[EAVSG09] Khalid El-Arini, Gaurav Veda, Dafna Shahaf, and Carlos Guestrin. Turning down the noise in the blogosphere. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 289–298. ACM, 2009.

[EHWK14] Michael D Ekstrand, F Maxwell Harper, Martijn C Willemsen, and Joseph A Konstan. User perception of differences in recommender algorithms. In *Proceedings of the 8th ACM Conference on Recommender systems*, pages 161–168. ACM, 2014.

[FPTV04] Boi Faltings, Pearl Pu, Marc Torrens, and Paolo Viappiani. Designing example-critiquing interaction. In *Proceedings of the 9th international conference on Intelligent user interfaces*, pages 22–29. ACM, 2004.

[Fri04] Gerhard Friedrich. Elimination of spurious explanations. In *ECAI*, volume 16, page 813, 2004.

[FTBE$^+$16] Ignacio Fernández-Tobías, Matthias Braunhofer, Mehdi Elahi, Francesco Ricci, and Iván Cantador. Alleviating the new user problem in collaborative filtering by exploiting personality information. *User Modeling and User-Adapted Interaction*, 26(2-3):221–255, 2016.

[FZ11] Gerhard Friedrich and Markus Zanker. A taxonomy for generating explanations in recommender systems. *AI Magazine*, 32(3):90–98, 2011.

[GC13] Saurabh Gupta and Sutanu Chakraborti. Utilsim: Iteratively helping users discover their preferences. In *International Conference on Electronic Commerce and Web Technologies*, pages 113–124. Springer, 2013.

[GCN$^+$18] Yangyang Guo, Zhiyong Cheng, Liqiang Nie, Yinglong Wang, Jun Ma, and Mohan S. Kankanhalli. Attentive long short-term preference modeling for personalized product search. *CoRR*, abs/1811.10155, 2018.

[GF17] Bryce Goodman and Seth Flaxman. European union regulations on algorithmic decision-making and a “right to explanation”. *AI Magazine*, 38(3):50–57, 2017.

[GGJ11] Fatih Gedikli, Mouzhi Ge, and Dietmar Jannach. Understanding recommendations by reading the clouds. In *International Conference on Electronic Commerce and Web Technologies*, pages 196–208. Springer, 2011.

[GGL18] Jianfeng Gao, Michel Galley, and Lihong Li. Neural approaches to conversational AI. In *Proceedings of the 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pages 1371–1374, 2018.

[GH06] Scott A Golder and Bernardo A Huberman. Usage patterns of collaborative tagging systems. *Journal of information science*, 32(2):198–208, 2006.

[GJG14] Fatih Gedikli, Dietmar Jannach, and Mouzhi Ge. How should i explain? a comparison of different explanation types for recommender systems. *International Journal of Human-Computer Studies*, 72(4):367–382, 2014.

[GKBP14] Ihsan Gunes, Cihan Kaleli, Alper Bilge, and Huseyin Polat. Shilling attacks against recommender systems: a comprehensive survey. *Artificial Intelligence Review*, 42(4):767–799, 2014.

[GLK$^+$09] Fan Guo, Chao Liu, Anitha Kannan, Tom Minka, Michael Taylor, Yi-Min Wang, and Christos Faloutsos. Click chain model in web search. In *Proceedings of the 18th international conference on World wide web*, pages 11–20. ACM, 2009.

[GUH16] Carlos A Gomez-Uribe and Neil Hunt. The netflix recommender system: Algorithms, business value, and innovation. *ACM Transactions on Management Information Systems (TMIS)*, 6(4):13, 2016.

[GW15] Mark P. Graus and Martijn C. Willemsen. Improving the user experience during cold start through choice-based preference elicitation. In *Proceedings of the 9th ACM Conference on Recommender Systems*, pages 273–276, 2015.

[HEZ$^+$12] Jeff Huang, Oren Etzioni, Luke Zettlemoyer, Kevin Clark, and Christian Lee. Revminer: An extractive interface for navigating reviews on a smartphone. In *Proceedings of the 25th annual ACM symposium on User interface software and technology*, pages 3–12. ACM, 2012.

[HGE$^+$12] M Shahriar Hossain, Joseph Gresock, Yvette Edmonds, Richard Helm, Malcolm Potts, and Naren Ramakrishnan. Connecting the dots between pubmed abstracts. *PloS one*, 7(1):e29509, 2012.

[HKR00] Jonathan L Herlocker, Joseph A Konstan, and John Riedl. Explaining collaborative filtering recommendations. In *Proceedings of the 2000 ACM conference on Computer supported cooperative work*, pages 241–250. ACM, 2000.

[HL04] Minqing Hu and Bing Liu. Mining and summarizing customer reviews. In *Proceedings of the 10th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 168–177. ACM, 2004.

[HLZ08] Nan Hu, Ling Liu, and Jie Jennifer Zhang. Do online reviews affect product sales? the role of reviewer characteristics and temporal effects. *Information Technology and management*, 9(3):201–214, 2008.

[HMB14] Negar Hariri, Bamshad Mobasher, and Robin Burke. Context adaptation in interactive recommender systems. In *Proceedings of the 8th ACM Conference on Recommender Systems*, pages 41–48, 2014.

[HPV16] Chen He, Denis Parra, and Katrien Verbert. Interactive recommender systems: A survey of the state of the art and future research challenges and opportunities. *Expert Systems with Applications*, 56:9–27, 2016.

[IL00] Sheena S Iyengar and Mark R Lepper. When choice is demotivating: Can one desire too much of a good thing? *Journal of personality and social psychology*, 79(6):995, 2000.

[JBB11] Nicolas Jones, Armelle Brun, and Anne Boyer. Comparisons instead of ratings: Towards more stable preferences. In *Proceedings of the IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology*, volume 1, pages 451–456, 2011.

[JJ17] Michael Jugovac and Dietmar Jannach. Interacting with recommenders—overview and research directions. *ACM Transactions on Interactive Intelligent Systems (TiiS)*, 7(3):10, 2017.

[JLJ15] Dietmar Jannach, Lukas Lerche, and Michael Jugovac. Adaptation and evaluation of recommendations for short-term shopping goals. In *Procs. of the Ninth ACM Conference on Recommender Systems*, pages 211–218, 2015.

[JM] Daniel Jurafsky and James H. Martin. Speech and language processing. Available at `https://web.stanford.edu/~jurafsky/slp3/ed3book.pdf` (2019/08/28).

[JS07] Anthony Jameson and Barry Smyth. Recommendation to groups. In Peter Brusilovsky, Alfred Kobsa, and Wolfgang Nejdl, editors, *The Adaptive Web*, pages 596–627. Springer-Verlag, 2007.

[KB16] Marius Kaminskas and Derek Bridge. 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):2:1–2:42, December 2016.

[KBOK12] Bart P Knijnenburg, Svetlin Bostandjiev, John O'Donovan, and Alfred Kobsa. Inspectability and control in social recommenders. In *Proceedings of the 6th ACM conference on Recommender systems*, pages 43–50. ACM, 2012.

[Kor08] Yehuda Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 426–434. ACM, 2008.

[KR12] Joseph A Konstan and John Riedl. Recommender systems: from algorithms to user experience. *User modeling and user-adapted interaction*, 22(1-2):101–123, 2012.

[KSB$^+$13] Todd Kulesza, Simone Stumpf, Margaret Burnett, Sherry Yang, Irwin Kwan, and Weng-Keen Wong. 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*, pages 3–10. IEEE, 2013.

[LAW14] Béatrice Lamche, Ugur Adıgüzel, and Wolfgang Wörndl. Interactive explanations in mobile shopping recommender systems. In *Joint Workshop on Interfaces and Human Decision Making in Recommender Systems*, page 14, 2014.

[LC13] Xin Li and Hsinchun Chen. Recommendation as link prediction in bipartite graphs: A graph kernel-based machine learning approach. *Decision Support Systems*, 54(2):880–890, 2013.

[LDGS11] Pasquale Lops, Marco De Gemmis, and Giovanni Semeraro. Content-based recommender systems: State of the art and trends. In *Recommender systems handbook*, pages 73–105. Springer, 2011.

[LHZ14] Benedikt Loepp, Tim Hussein, and Jüergen Ziegler. Choice-based preference elicitation for collaborative filtering recommender systems. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 3085–3094, 2014.

[LKH14] Blerina Lika, Kostas Kolomvatsos, and Stathes Hadjiefthymiades. Facing the cold start problem in recommender systems. *Expert Systems with Applications*, 41(4):2065–2073, 2014.

[LOL$^+$18] Bruno Lepri, Nuria Oliver, Emmanuel Letouzé, Alex Pentland, and Patrick Vinck. Fair, transparent, and accountable algorithmic decision-making processes. *Philosophy & Technology*, 31(4):611–627, 2018.

[LSY03] Greg Linden, Brent Smith, and Jeremy York. Amazon. com recommendations: Item-to-item collaborative filtering. *IEEE Internet computing*, 7(1):76–80, 2003.

[LZ12] Bing Liu and Lei Zhang. A survey of opinion mining and sentiment analysis. In Charu C. Aggarwal and ChengXiang Zhai, editors, *Mining Text Data*, pages 415–463. Springer US, 2012.

[McS05] David McSherry. Explanation in recommender systems. *Artificial Intelligence Review*, 24(2):179–197, 2005.

[MGS02a] Lorraine Mc Ginty and Barry Smyth. Comparison-based recommendation. In *Proceedings of the 6th European Conference on Case-Based Reasoning*, pages 575–589. Springer, 2002.

[MGS02b] Lorraine Mc Ginty and Barry Smyth. Evaluating preference-based feedback in recommender systems. In *Irish Conference on Artificial Intelligence and Cognitive Science*, pages 209–214. Springer, 2002.

[MGS03] Lorraine Mc Ginty and Barry Smyth. On the role of diversity in conversational recommender systems. In *Proceedings of the 5th International Conference on Case-Based Reasoning*, pages 276–290. Springer, 2003.

[MLKR03] Sean M McNee, Shyong K Lam, Joseph A Konstan, and John Riedl. Interfaces for eliciting new user preferences in recommender systems. In *International Conference on User Modeling*, pages 178–187. Springer, 2003.

[MLRS15] Khalil Muhammad, Aonghus Lawlor, Rachael Rafter, and Barry Smyth. Great explanations: Opinionated explanations for recommendations. In *International Conference on Case-Based Reasoning*, pages 244–258. Springer, 2015.

[MLS16] Khalil Muhammad, Aonghus Lawlor, and Barry Smyth. On the use of opinionated explanations to rank and justify recommendations. In *Proceedings of the 28th FLAIRS Conference*, pages 554–559, 2016.

[MNL$^+$16] Cataldo Musto, Fedelucio Narducci, Pasquale Lops, Marco De Gemmis, and Giovanni Semeraro. ExpLOD: A Framework for Explaining Recommendations Based on the Linked Open Data Cloud. In *Proceedings of the 10th ACM Conference on Recommender Systems*, pages 151–154, 2016.

[MNL$^+$19] Cataldo Musto, Fedelucio Narducci, Pasquale Lops, Marco de Gemmis, and Giovanni Semeraro. Linked open data-based explanations for transparent recommender systems. *International Journal of Human-Computer Studies*, 121:93–107, 2019.

[MRMS04] Kevin McCarthy, James Reilly, Lorraine McGinty, and Barry Smyth. Thinking positively-explanatory feedback for conversational recommender systems. In *Proceedings of the European Conference on Case-Based Reasoning (ECCBR-04) Explanation Workshop*, pages 115–124, 2004.

[MRMS05] Kevin McCarthy, James Reilly, Lorraine McGinty, and Barry Smyth. Experiments in dynamic critiquing. In *Proceedings of the 10th international conference on Intelligent user interfaces*, pages 175–182. ACM, 2005.

[MRN14] Andrea Moro, Alessandro Raganato, and Roberto Navigli. Entity linking meets word sense disambiguation: a unified approach. *Transactions of the Association for Computational Linguistics*, 2:231–244, 2014.

[MS06] Lorraine Mcginty and Barry Smyth. Adaptive selection: An analysis of critiquing and preference-based feedback in conversational recommender systems. *International Journal of Electronic Commerce*, 11(2):35–57, 2006.

[MSB+14] Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. The stanford corenlp natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60, 2014.

[MSS10] Kevin McCarthy, Yasser Salem, and Barry Smyth. Experience-based critiquing: Reusing critiquing experiences to improve conversational recommendation. In *International Conference on Case-Based Reasoning*, pages 480–494. Springer, 2010.

[MZR18] Sina Mohseni, Niloofar Zarei, and Eric D Ragan. A survey of evaluation methods and measures for interpretable machine learning. *arXiv preprint arXiv:1811.11839*, 2018.

[NDZ18] Sidra Naveed, Tim Donkers, and Jürgen Ziegler. Argumentation-based explanations in recommender systems: Conceptual framework and empirical results. In *UMAP 2018 - Adjunct Publication of the 26th Conference on User Modeling, Adaptation and Personalization*, page 293–298, New York, NY, USA, 2018. ACM.

[NJ17] Ingrid Nunes and Dietmar Jannach. A systematic review and taxonomy of explanations in decision support and recommender systems. *User Modeling and User-Adapted Interaction*, 27(3-5):393–444, 2017.

[NK11] Xia Ning and George Karypis. Slim: Sparse linear methods for top-n recommender systems. In *2011 IEEE 11th International Conference on Data Mining*, pages 497–506. IEEE, 2011.

[NMLDL12] Ingrid Nunes, Simon Miles, Michael Luck, and Carlos JP De Lucena. Investigating explanations to justify choice. In *International Conference on User Modeling, Adaptation, and Personalization*, pages 212–224. Springer, 2012.

[NP12] Roberto Navigli and Simone Paolo Ponzetto. Babelnet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network. *Artificial Intelligence*, 193:217–250, 2012.

[NR04] Quang Nhat Nguyen and Francesco Ricci. User preferences initialization and integration in critique-based mobile recommender systems. In *Proceedings of Workshop on Artificial Intelligence in Mobile Systems*, 2004.

[NR08] Quang Nhat Nguyen and Francesco Ricci. Long-term and session-specific user preferences in a mobile recommender system. In *Proceedings of the 13th international conference on Intelligent user interfaces*, pages 381–384. ACM, 2008.

[NR17] Thuy Ngoc Nguyen and Francesco Ricci. Combining long-term and discussion-generated preferences in group recommendations. In *Proceedings of the 25th Conference on User Modeling, Adaptation and Personalization*, pages 377–378. ACM, 2017.

[NR18] Thuy Ngoc Nguyen and Francesco Ricci. Situation-dependent combination of long-term and session-based preferences in group recommendations: an experimental analysis. In *Proceedings of the 33rd Annual ACM Symposium on Applied Computing*, pages 1366–1373. ACM, 2018.

[OHS09] Antti Oulasvirta, Janne P Hukkinen, and Barry Schwartz. When more is less: the paradox of choice in search engine use. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pages 516–523. ACM, 2009.

[PB07] Michael J Pazzani and Daniel Billsus. Content-based recommendation systems. In *The adaptive web*, pages 325–341. Springer, 2007.

[PC07] Pearl Pu and Li Chen. Trust-inspiring explanation interfaces for recommender systems. *Knowledge-Based Systems*, 20(6):542–556, 2007.

[PC08] Pearl Pu and Li Chen. User-involved preference elicitation for product search and recommender systems. *AI magazine*, 29(4):93, 2008.

[PCH12] Pearl Pu, Li Chen, and Rong Hu. Evaluating recommender systems from the user's perspective: survey of the state of the art. *User Modeling and User-Adapted Interaction*, 22(4-5):317–355, 2012.

[PCK08] Pearl Pu, Li Chen, and Pratyush Kumar. Evaluating product search and recommender systems for e-commerce environments. *Electronic Commerce Research*, 8(1-2):1–27, 2008.

[PCL$^+$16] Roberto Pagano, Paolo Cremonesi, Martha Larson, Balázs Hidasi, Domonkos Tikk, Alexandros Karatzoglou, and Massimo Quadrana. The contextual turn: From context-aware to context-driven recommender systems. In *Proceedings of the 10th ACM Conference on Recommender Systems*, pages 249–252, 2016.

[PCM18] Weike Pan, Li Chen, and Zhong Ming. Personalized recommendation with implicit feedback via learning pairwise preferences over item-sets. *Knowledge and Information Systems*, pages 1–24, 2018.

[PK04] Pearl Huan Z Pu and Pratyush Kumar. Evaluating example-based search tools. In *Proceedings of the 5th ACM Conference on Electronic Commerce*, pages 208–217, 2004.

[PLH07] Do-Hyung Park, Jumin Lee, and Ingoo Han. The effect of on-line consumer reviews on consumer purchasing intention: The moderating role of involvement. *International journal of electronic commerce*, 11(4):125–148, 2007.

[PSM12] Alexis Papadimitriou, Panagiotis Symeonidis, and Yannis Manolopoulos. A generalized taxonomy of explanations styles for traditional and social recommender systems. *Data Mining and Knowledge Discovery*, 24(3):555–583, 2012.

[QCJ18] Massimo Quadrana, Paolo Cremonesi, and Dietmar Jannach. Sequence-aware recommender systems. *ACM Comput. Surv.*, 51(4):66:1–66:36, 2018.

[RB18] Arpit Rana and Derek Bridge. Explanations that are intrinsic to recommendations. In *Proceedings of the 26th Conference on User Modeling, Adaptation and Personalization*, pages 187–195. ACM, 2018.

[RFGST09] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. In *Proceedings of the 25th conference on uncertainty in artificial intelligence*, pages 452–461. AUAI Press, 2009.

[RL03] Ian Ruthven and Mounia Lalmas. A survey on the use of relevance feedback for information access systems. *Knowl. Eng. Rev.*, 18(2):95–145, 2003.

[RMMS04] James Reilly, Kevin McCarthy, Lorraine McGinty, and Barry Smyth. Dynamic critiquing. In *European Conference on Case-Based Reasoning*, pages 763–777. Springer, 2004.

[RMMS05] James Reilly, Kevin McCarthy, Lorraine McGinty, and Barry Smyth. Incremental critiquing. *Knowledge-Based Systems*, 18(4-5):143–151, 2005.

[RMTM10] Haggai Roitman, Yossi Messika, Yevgenia Tsimerman, and Yonatan Maman. Increasing patient safety using explanation-driven personalized content recommendation. In *Proceedings of the 1st ACM International Health Informatics Symposium*, pages 430–434. ACM, 2010.

[RN07] Francesco Ricci and Quang Nhat Nguyen. Acquiring and revising preferences in a critique-based mobile recommender system. *IEEE Intelligent systems*, 22(3):22–29, 2007.

[RRS11] Francesco Ricci, Lior Rokach, and Bracha Shapira. Introduction to recommender systems handbook. In *Recommender systems handbook*, pages 1–35. Springer, 2011.

[RSG16] Marco Túlio Ribeiro, Sameer Singh, and Carlos Guestrin. "why should I trust you?": Explaining the predictions of any classifier. *CoRR*, abs/1602.04938, 2016.

[RSZ13] Marco Rossetti, Fabio Stella, and Markus Zanker. Towards explaining latent factors with topic models in collaborative recommender systems. In *Database and Expert Systems Applications (DEXA), 2013 24th International Workshop on*, pages 162–167. IEEE, 2013.

[SAN$^+$18] Masahiro Sato, Budrul Ahsan, Koki Nagatani, Takashi Sonoda, Qian Zhang, and Tomoko Ohkuma. Explaining recommendations using contexts. In *23rd International Conference on Intelligent User Interfaces*, pages 659–664. ACM, 2018.

[SB88] Gerard Salton and Christopher Buckley. Term-weighting approaches in automatic text retrieval. *Information processing & management*, 24(5):513–523, 1988.

[SC00] Barry Smyth and Paul Cotter. A personalised tv listings service for the digital tv age. *Knowledge-Based Systems*, 13(2-3):53–59, 2000.

[Sch02] Sascha Schmitt. *simVar*: A similarity-influences question selection criterion or e-sales dialogs. *Artificial Intelligence Review*, 18(3–4):195–221, 2002.

[SCLDL12] Christian Scheel, Angel Castellanos, Thebin Lee, and Ernesto William De Luca. The reason why: A survey of explanations for recommender systems. In *International Workshop on Adaptive Multimedia Retrieval*, pages 67–84. Springer, 2012.

[SH13] Yasser Salem and Jun Hong. History-aware critiquing-based conversational recommendation. In *Proceedings of the 22nd International Conference on World Wide Web*, pages 63–64. ACM, 2013.

[Shi02] Hideo Shimazu. ExpertClerk: A conversational case-cased reasoning tool for developing salesclerk agents in e-commerce webshops. *Artificial Intelligence Review*, 18(3-4):223–244, 2002.

[Shi07] Clay Shirky. Ontology is overrated–categories, links, and tags. 2007.

[SHKL14] Christian Sandvig, Kevin Hamilton, Karrie Karahalios, and Cedric Langbort. Auditing algorithms: Research methods for detecting discrimination on internet platforms. *Data and discrimination: converting critical concerns into productive inquiry*, 22, 2014.

[SHY04] Kazunari Sugiyama, Kenji Hatano, and Masatoshi Yoshikawa. Adaptive web search based on user profile constructed without any effort from users. In *Proceedings of the 13th International Conference on the World Wide Web*, pages 675–684, 2004.

[SM03a] Barry Smyth and Lorraine McGinty. An analysis of feedback strategies in conversational recommenders. In *Proceedings of the 14th Irish Artificial Intelligence and Cognitive Science Conference*, 2003.

[SM03b] Barry Smyth and Lorraine McGinty. The power of suggestion. In *Proceedings of the International Joint Conference on Artificial Intelligence*, volume 3, pages 127–132, 2003.

[Smy07] Barry Smyth. Case-based recommendation. In Peter Brusilovsky, Alfred Kobsa, and Wolfgang Nejdl, editors, *The Adaptive Web*, pages 342–376. Springer-Verlag, 2007.

[SNM08] Panagiotis Symeonidis, Alexandros Nanopoulos, and Yannis Manolopoulos. Providing justifications in recommender systems. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, 38(6):1262–1272, 2008.

[SNM09] Panagiotis Symeonidis, Alexandros Nanopoulos, and Yannis Manolopoulos. Moviexplain: a recommender system with explanations. In *Proceedings of the 3rd ACM conference on Recommender systems*, pages 317–320. ACM, 2009.

[SRS$^+$13] Guy Shani, Lior Rokach, Bracha Shapira, Sarit Hadash, and Moran Tangi. Investigating confidence displays for top-n recommendations. *Journal of the American Society for Information Science and Technology*, 64(12):2548–2563, 2013.

[SS02] Rashmi Sinha and Kirsten Swearingen. The role of transparency in recommender systems. In *CHI'02 extended abstracts on Human factors in computing systems*, pages 830–831. ACM, 2002.

[SZ18] Yueming Sun and Yi Zhang. Conversational recommender system. In *Proceedings of the 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pages 235–244, 2018.

[TGL04] Cynthia A Thompson, Mehmet H Goker, and Pat Langley. A personalized system for conversational recommendations. *Journal of Artificial Intelligence Research*, 21:393–428, 2004.

[Tin07] Nava Tintarev. Explanations of recommendations. In *Proceedings of the 1st ACM conference on Recommender systems*, pages 203–206. ACM, 2007.

[TM07a] Nava Tintarev and Judith Masthoff. Effective explanations of recommendations: user-centered design. In *Proceedings of the 1st ACM conference on Recommender systems*, pages 153–156. ACM, 2007.

[TM07b] Nava Tintarev and Judith Masthoff. A survey of explanations in recommender systems. In *Data Engineering Workshop, 2007 IEEE 23rd International Conference on*, pages 801–810. IEEE, 2007.

[TM08a] Nava Tintarev and Judith Masthoff. The effectiveness of personalized movie explanations: An experiment using commercial metadata. In *International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems*, pages 204–213. Springer, 2008.

[TM08b] Nava Tintarev and Judith Masthoff. Over-and underestimation in different product domains. In *Workshop on Recommender Systems associated with ECAI*, pages 14–19, 2008.

[TM15] Nava Tintarev and Judith Masthoff. Explaining recommendations: Design and evaluation. In *Recommender systems handbook*, pages 353–382. Springer, 2015.

[VSR09] Jesse Vig, Shilad Sen, and John Riedl. Tagsplanations: explaining recommendations using tags. In *Proceedings of the 14th international conference on Intelligent user interfaces*, pages 47–56. ACM, 2009.

[VSR11] Jesse Vig, Shilad Sen, and John Riedl. Navigating the tag genome. In *Proceedings of the 16th International Conference on Intelligent User Interfaces*, pages 93–102, 2011.

[WASB16] Chao-Yuan Wu, Christopher V. Alvino, Alexander J. Smola, and Justin Basilico. Using navigation to improve recommendations in real-time. In *Proceedings of the 10th ACM Conference on Recommender Systems*, pages 341–348, 2016.

[WCY$^+$18] Xiting Wang, Yiru Chen, Jie Yang, Le Wu, Zhengtao Wu, and Xing Xie. A reinforcement learning framework for explainable recommendation. In *2018 IEEE International Conference on Data Mining (ICDM)*, pages 587–596. IEEE, 2018.

[WHL11] Chen Wei, Wynne Hsu, and Mong Li Lee. A unified framework for recommendations based on quaternary semantic analysis. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, pages 1023–1032. ACM, 2011.

[WMF17] Sandra Wachter, Brent Mittelstadt, and Luciano Floridi. Why a right to explanation of automated decision-making does not exist in the general data protection regulation. *International Data Privacy Law*, 7(2):76–99, 2017.

[WWP$^+$13] Rainer Wasinger, James Wallbank, Luiz Pizzato, Judy Kay, Bob Kummerfeld, Matthias Böhmer, and Antonio Krüger. Scrutable user models and personalised item recommendation in mobile lifestyle applications. In *International Conference on User Modeling, Adaptation, and Personalization*, pages 77–88. Springer, 2013.

[YLAY09] Cong Yu, Laks VS Lakshmanan, and Sihem Amer-Yahia. Recommendation diversification using explanations. In *2009 IEEE 25th International Conference on Data Engineering*, pages 1299–1302. IEEE, 2009.

[ZC18] Yongfeng Zhang and Xu Chen. Explainable recommendation: A survey and new perspectives. *arXiv preprint arXiv:1804.11192*, 2018.

[ZCA+18] Yongfeng Zhang, Xu Chen, Qingyao Ai, Liu Yang, and W Bruce Croft. Towards conversational search and recommendation: System ask, user respond. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pages 177–186. ACM, 2018.

[ZLZ+14] Yongfeng Zhang, Guokun Lai, Min Zhang, Yi Zhang, Yiqun Liu, and Shaoping Ma. Explicit factor models for explainable recommendation based on phrase-level sentiment analysis. In *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval*, pages 83–92. ACM, 2014.

[ZN10] Markus Zanker and Daniel Ninaus. Knowledgeable explanations for recommender systems. In *Web Intelligence and Intelligent Agent Technology (WI-IAT), 2010 IEEE/WIC/ACM International Conference on*, volume 1, pages 657–660. IEEE, 2010.

[ZP06] Jiyong Zhang and Pearl Pu. A comparative study of compound critique generation in conversational recommender systems. In *International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems*, pages 234–243. Springer, 2006.