CORA Cork Open Research Archive
Cartlann Taighde Oscailte Chorcaí

Title	Extended recommendation-by-explanation
Authors	Rana, Arpit;D'Addio, Rafael M.;Manzato, Marcelo G.;Bridge, Derek G.
Publication date	2022-03-07
Original Citation	Rana, A., D'Addio, R. M., Manzato, M. G. and Bridge, D. (2022) 'Extended recommendation-by-explanation', User Modeling and User-Adapted Interaction. doi: 10.1007/s11257-021-09317-4
Type of publication	Article (peer-reviewed)
Link to publisher's version	10.1007/s11257-021-09317-4
Rights	© 2022, the Authors, under exclusive licence to Springer Nature B.V. This is a post-peer-review, pre-copyedit version of an article published in User Modeling and User-Adapted Interaction. The final authenticated version is available online at: https://doi.org/10.1007/s11257-021-09317-4
Download date	2026-11-04 21:18:31
Item downloaded from	https://hdl.handle.net/10468/13055

UCC
University College Cork, Ireland
Coláiste na hOllscoile Corcaigh

User Modeling and User Adapted Interaction manuscript No.
(will be inserted by the editor)

Extended Recommendation-by-Explanation

Arpit Rana · Rafael M. D'Addio · Marcelo G. Manzato · Derek Bridge

Received: date / Accepted: date

Abstract Studies have shown that there is an intimate connection between the process of computing recommendations and the process of generating corresponding explanations, and that this close relationship may lead to better recommendations for the user. However, to date, most recommendation explanations are post-hoc rationalizations; in other words, computing recommendations and generating corresponding explanations are two separate and sequential processes. There is, however, recent work *unifies* recommendation and explanation, using an approach that is called Recommendation-by-Explanation (*r-by-e*). In *r-by-e*, 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. However, the way it constructs and se-

Arpit Rana (✉)
Dhirubhai Ambani Institute of Information and Communication Technology, Gandhinagar, India
E-mail: arpit_rana@daiict.ac.in
This research work was conducted when the author was at University College Cork, Ireland.

Rafael M. D'Addio
Dynaccurate SARL, Luxembourg
E-mail: rafael.daddio@dynaccurate.com
This research work was conducted when the author was at Institute of Mathematics and Computer Science, University of São Paulo, Brazil.

Marcelo G. Manzato
Institute of Mathematics and Computer Science, University of São Paulo, Brazil
E-mail: mmanzato@icmc.usp.br

Derek Bridge
Insight Centre for Data Analytics, University College Cork, Ireland
E-mail: derek.bridge@insight-centre.org

lects chains is relatively simple, and it considers only one way of representing item's elements — in terms of their features.

In this article, we extend *r-by-e*. We present a number of different ways of generating chains from a user's profile. These methods mainly differ in their item representations (i.e. whether using item elements as features or neighbors) and in the weighting schemes that they use to generate the chains. We also explore *r-by-e*'s approach to chain selection, allowing the system to choose whether to cover more aspects of the candidate item or the user profile. We compare the extended versions with corresponding classic content-based methods on two datasets that mainly differ on their item feature sets. We find that the versions of *r-by-e* that make explicit use of item features have several advantages over the ones that use neighbors, and the empirical comparison shows that one of these versions —the one that assigns weights to the item features based on their importance to that item— 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. It also obtains the best survey responses for its recommendations and corresponding explanations in a trial with real users.

Keywords Explanation · Recommendation · Sentiments · User trials

1 Introduction

Explanations can serve a multiplicity of aims: they give credibility to recommendations (Sinha and Swearingen, 2002; Pu and Chen, 2007), help users make better choices (Bilgic and Mooney, 2005), positively contribute to a better user experience (Konstan and Riedl, 2012), and so on. Due to the wide adoption of recommender systems in many aspects of our lives, explaining recommendations has attracted considerable attention (Zhang and Chen, 2018; Tintarev and Masthoff, 2015).

However, 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 information in the explanation that is different from the information that it used to compute the recommendation (Abdollahi and Nasraoui, 2016). For example, in (Rossetti et al., 2013), a recommendation generated by factorization of a ratings matrix is explained using topic models mined from textual data associated with items — data that was not used when when building the recommendation model. Such differences are one cause of low fidelity (Kulesza et al., 2013). Fidelity (also called objective transparency (Gedikli et al., 2014)) is 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 higher the fidelity between the explanations and the underlying recommender, the greater the users' trust in the recommender and the better their understanding (Kulesza et al., 2013). This finding indicates that there are

reasons to see a more intimate connection between the process of computing recommendations and generating corresponding explanations; and that a closer relationship may even ultimately lead to better recommendations.

To investigate the role and importance of explanations in the process of computing recommendations, we have previously introduced a novel approach, *Recommendation-by-Explanation* (*r-by-e*) (Rana and Bridge, 2017, 2018). *r-by-e unifies* recommendation and explanation to a greater degree than has been achieved hitherto. The main idea in *r-by-e* is to recommend the items that have the best explanations. This turns recommendation and explanation 'on their heads'. First, for each candidate item, we compute reasons to recommend the candidate item –these are explanations. Next, we recommend a top-n set of recommendations: those items with the strongest explanations. More specifically, our current version of *r-by-e* implement the two steps of this process in the following way: *chain generation* and *chain selection*. In the former, for each candidate item, the system constructs and scores an explanation, a chain of items from the user's profile, referred to as an *explanation chain*, based on *overlap* between the representations of items in the chain; then, in the second step, it recommends those candidate items that have the best explanations based on scores that we compute for the chains. Examples of chains can be seen in Section 6.3.2. In *r-by-e*, an *explanation chain* 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}. *r-by-e* describes each item using a set of elements. In our original work, elements are item features, but in one of our extensions (section 4.2), an item's elements are other items — those that are its neighbours.*r-by-e* constructs an explanation chain by iteratively adding items from the user profile in an effort to cover potentially different elements of the candidate item. Hence, in the chain, the item closest to the candidate shares more of its elements with the candidate and the item farthest from the candidate shares the least with the candidate. Consecutive items in the chain must also share elements. Thus, the explanation chain enables a user to understand the mutual relationships between adjacent items as well as relationships between items from her profile and the candidate item in an incremental manner. In some sense, the chain 'leads' the user through ever more relevant items from her profile towards the candidate. This, we believe, has the potential to explain recommendations in an effective manner that is sensitive to user understanding.

This paper, drawn from the first author's PhD research (Rana, 2020), significantly extends *r-by-e* in three ways (see Figure 1). First, as stated above, we previously described each item using a set of elements (and, previously, these elements were item features). This treated each element in a description as having equal importance. In our extensions, we use a scheme for assigning weights to the elements in each item description, based on their informativeness. We define *weighted overlap* to take advantage of these weights. Second, as already mentioned, 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

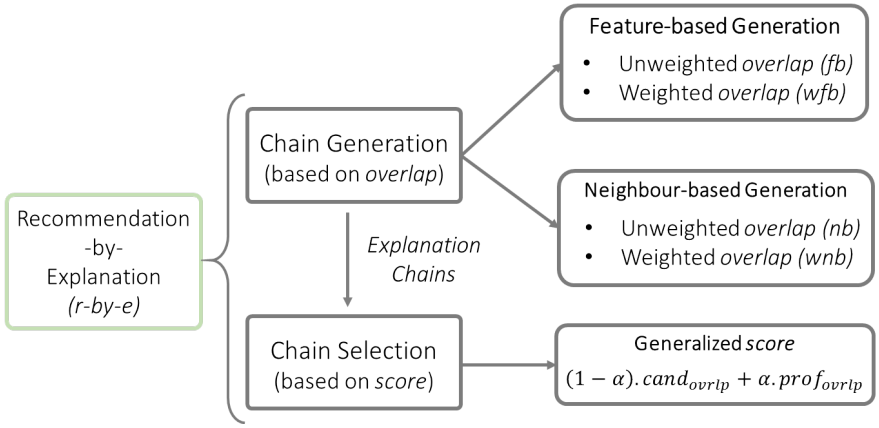


Fig. 1: Extended Recommendation-by-Explanation

and weighted overlap. Finally, we also generalize *r-by-e*'s *chain selection*. In place of simply adding two types of overlap (as in (Rana and Bridge, 2018), summarized in Section 3.2), we define the score to be a linear combination of the two but controlled by a parameter α that allows us to vary the importance of the two components of the definition.

We explore these variants using two versions of a movie recommendation dataset. The first version describes each item (movie) as a set of its *keywords*. In the second version of the dataset, we use sentiment data extracted from user reviews and describe each item as a set of its concepts associated with sentiment values. The complete details of these datasets are covered in Section 6.1. In a set of offline experiments on the first version of the dataset, we compare all four versions of *chain generation*: i) unweighted feature-based (*fb*), ii) weighted feature-based (*wfb*), iii) unweighted neighbour-based (*nb*), and iv) weighted neighbour-based (*wnb*). Notice that unweighted feature-based is what we covered already in our previous work (Rana and Bridge, 2018). We include it here again for better understanding the difference among all four approaches.[1] In these experiments, we found that the approaches that explicitly use item features perform better than their neighbour-based counterparts, and weight-based approaches are able to produce more accurate recommendations while being competitive in terms of diversity and surprise. We also show in the results that by varying the balancing parameter (α), we can configure *r-by-e* so that it 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

[1] In fact, the results we present here are not identical to the ones in (Rana and Bridge, 2018) because in this work we have included some extra normalization of the scores that this version uses in order to give a fair comparison with the three other variants.

unweighted forms of *chain generation* do not apply to the sentiment version of the dataset.) In this experiment, we find again that the feature-based and neighbour-based variants produce sets of recommendations that are comparable in their diversity, but the feature-based approach attains higher levels of precision at the cost of somewhat lower results for surprise and novelty.

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

The paper is organized as follows. After a review of related work (Section 2), Section 3 briefly summarises the notion of *Recommendation-by-Explanation* and the explanation chains, while Section 4 describes the extensions and refinements over the original *r-by-e* formulation. In Section 5, we evaluate those extensions in offline experiments with keywords as features, while in Section 6 we detail our Sentiment-Aware Explanation Chains and present both online and offline experiments to evaluate them. In Section 7, we discuss results the of our experiments; and, finally, in Section 8, we offer some concluding remarks.

2 Related Work

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

Explanations are especially important in high-risk domains where the cost of making a wrong decision is higher (e.g. buying a laptop, planning a holiday, etc.) than in low-risk domains (e.g. selecting a song to play) (Herlocker et al., 2000). The level of detail in an explanation may also vary with the level of risk associated with the decision-making process (Chen and Pu, 2005). 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 (Tintarev and Masthoff, 2008; Kouki et al., 2019).

Explanations of recommendations vary in many ways. They may vary in their goals: they may be intended to help the user make a better decision (effectiveness), change the user's behaviour (persuasion), make a system more correctable (scrutability), and so on (Tintarev and Masthoff, 2007b). In our work, we are interested in *effectiveness*, which is why one of our user trials is a *re-ranking* task.

Explanations often relate the recommended item to the user through *intermediary entities* (Vig et al., 2009). These intermediary entities may be other users, other items, item elements, or context. Based on these intermediary entities, explanations can be described as either *user-based*, *item-based*, *element-based*, *context-based* or, in the case of combinations, *hybrid* (Bilgic and Mooney, 2005; Papadimitriou et al., 2012; Kouki et al., 2017).

User-based explanations say that an item is recommended because users who are similar to the active user liked it. For example, social networks such as Facebook[2] often use user-based explanations when recommending a person to add as a friend or to follow. However, these methods do not scale up to user-based collaborative filtering systems, where: the number of neighbours is usually larger; most, if not all, of the neighbours are not known to the active user; and the number of co-rated items between the active user and any neighbour can be too large to be readily comprehended (Bridge and Dunleavy, 2014).

Item-based explanations say that the item is being recommended because the user liked similar items. Famously, Amazon[3] uses item-based explanations for its recommendations (Linden et al., 2003). 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 (Bilgic and Mooney, 2005). Accordingly, in (Bridge and Dunleavy, 2014), 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 (Tintarev and Masthoff, 2007a).

Element-based explanations say that the recommended item has elements that the user likes. For instance, Pandora uses altogether 450 musical attributes for representing each music track[4] and provides explanations such as: *Based on what you have told us so far, we are playing this track because it features a leisurely tempo, a sparse piano solo, a lazy swing groove, major tonality and many other similarities identified* (Tintarev and Masthoff, 2015). In the literature, elements take numerous different forms, e.g. attribute-value pairs (Tintarev, 2007; Scheel et al., 2012), item content (Bilgic and Mooney, 2005), user-generated tags (Vig et al., 2009; Gedikli et al., 2011, 2014), opinions mined from user reviews (McAuley and Leskovec, 2013; Muhammad et al., 2015; Chang et al., 2016; Dong and Smyth, 2017; Costa et al., 2018; Ni and McAuley, 2018; Lu et al., 2018; Ni et al., 2019), and linked data (Musto et al., 2016, 2019). Most recently, contextual information has been used in explanations too. In (Sato et al., 2018), Sato et al. proposed explanations that include contexts suitable for consuming the recommended item.

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

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

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

Hybrid explanations are also possible. For example, in the case of item-based explanations, we often want to show why the items in the explanation are similar to the recommended item, and this is typically done by showing the elements that they have in common. Since these item-based explanations combine items and elements, they are hybrid explanations (Papadimitriou et al., 2012; Kouki et al., 2017). Explanation Chains are of this kind: they are item-based but they expose item relationships through elements.

In Artificial Intelligence in general, explanations are sometimes categorized as white-box (also sometimes called model-based) or black-box (sometimes called model-agnostic) (Herlocker et al., 2000; Friedrich and Zanker, 2011). The distinction typically reflects on the fidelity of the explanations to the underlying reasoning done by the AI system.

White-box explanations are built from traces of the system's reasoning. These explanations disclose something of the underlying model in order to reveal 'how' the system has reached its conclusions. For example, if we have a user-based nearest-neighbours recommender that makes recommendations by finding items liked by the active user's nearest neighbours, then a histogram of the neighbours' ratings (Herlocker et al., 2000) is a white-box explanation.

Black-box explanations, by contrast, make no use of knowledge of how the system produced its decision. Black-box explanations are post-hoc rationalizations. For example, the LIME system explains classification decisions by interrogating the classifier to obtain a dataset from which LIME builds a distinct explanation model (Ribeiro et al., 2016). Since they make no use of traces of the system's reasoning, black-box explanations may make use of other sources of information that were not used in the decision-making. In (Rossetti et al., 2013), 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 raise the issue of fidelity (Kulesza et al., 2013) (also called objective transparency (Gedikli et al., 2014)): 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 element 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 (Kulesza et al., 2013).

Recommendation-by-Explanation seeks to achieve quite high fidelity since, in *r-by-e*, explanation is intrinsic to recommendation. Indeed, *r-by-e*'s expla-

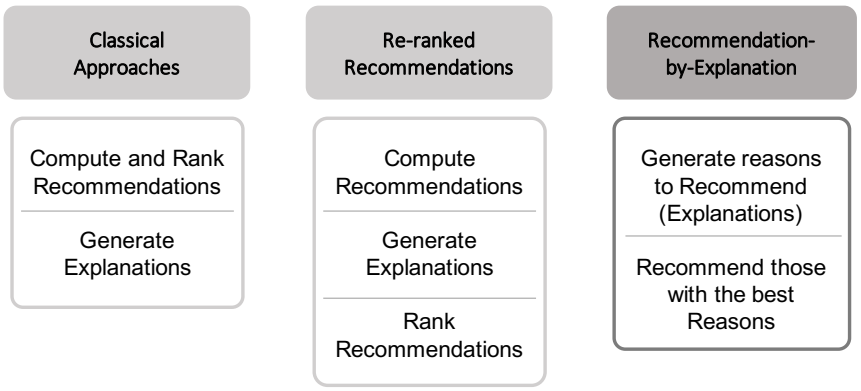


Fig. 2: Role of explanations in producing recommendations.

nations are not black-box: the recommendation are exclusively guided by explanation quality, and the recommender's model is driven by the explanations themselves. This leads to sound and complete explanations, i.e., they do not oversimplify the model nor expose only a limited part of its reasoning.

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

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

There have, however, been a few efforts that modify the classical approach a little. These are shown in the middle of Figure 2. In Re-ranked Recommendations, for example, the system finds some recommendations; it generates explanations for the recommendations; it scores the explanations; and it re-ranks the recommendations based on their explanation scores before showing them to the user (Muhammad et al., 2015, 2016). In (Yu et al., 2009), Yu et al., for example, use this strategy to increase the recommendation diversity.

Another strategy that falls within the Re-ranked Recommendations category is to use explanations to help the users to analyse he recommendation quality and, if appropriate, to allow the user to propose a change to the recommender's predictions (Cleger-Tamayo et al., 2012). For example, Cleger et al. (Cleger et al., 2014) use explanations to learn a regression model that can predict the error and thus 'correct' the predicted rating for a target item.

Our new approach, Recommendation-by-Explanation, is shown rightmost in Figure 2. Uniquely, as far as we are aware, it reverses the process. First, it finds explanations for all the candidate items. Then, it recommends the candidates that have the best explanations. Hence, Recommendation-by-Explanation is an approach that unifies the two processes: computing recommendations

and generating corresponding explanations. This, we believe, gives it high fidelity.

3 *Recommendation-by-Explanation*

Recommendation-by-Explanation is a novel approach for recommendation and explanation: the system constructs an explanation (a chain of items from the user's profile) for each candidate item (we call this step *chain generation*); then it recommends those candidate items that have the best explanations (which we call *chain selection*). By unifying recommendation and explanation, *r-by-e* finds relevant recommendations with explanations that have a high degree of fidelity. In this section, we briefly describe the formulations of basic *r-by-e*. For more details on this, please refer to Rana and Bridge (2018).

3.1 Chain generation

Given a candidate item, *r-by-e* works backwards to construct a chain: starting with the candidate item, it finds predecessors from the user's profile (i.e. items that the user likes), greedily selects one, finds its predecessors, selects one; and so on. At each step, the predecessor that gets selected is the one that most increases the degree to which the chain 'covers' the elements of the candidate item. This is measured by the notion of *overlap*. The overlap $\text{ovrlp}(p, i, C)$ of adding predecessor p to partial chain C that explains candidate item i is given by:

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

Here, F_i and F_p denote the elements of items i and p. $\text{covered}(i, C)$ is the set of elements of candidate i that are already covered by members of the chain C, i.e. $\text{covered}(i, C) = \bigcup_{j \in C} F_j \cap F_i$. The above formula measures the overlap with respect to both $|F_i|$ and $|F_p|$ (the two denominators). Readers who are interested in a detailed rationale behind this and other formulas in this section, are referred to Rana and Bridge (2018).

3.2 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 greedily in such a way as to ensure a degree of diversity between the chains that are associated with the items in the top-n. 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:

$$\text{score}(\langle C, i \rangle, L^*) = \mathit{cand}_{\text{ovrlp}}(\langle C, i \rangle, L^*) + \mathit{prof}_{\text{ovrlp}}(\langle C, i \rangle, L^*) \quad (2)$$

where

$$cand_{\text{ovrlp}}(\langle C, i\rangle, L^*) = \frac{sum_ovrlps}{|C|+1} \tag{3}$$

$$prof_{\text{ovrlp}}(\langle C, i\rangle, L^*) = \frac{\left|C \setminus \bigcup_{j \in L^*} j\right|}{|C|+1} \tag{4}$$

Eq. 3 defines $cand_{\text{ovrlp}}$ as the average of the overlaps of the candidate elements in the chain: the higher is the term, the greater is the coverage of the candidate's elements by the other chain members. Here, *sum_ovrlps* is the sum of the overlaps (Eq. 1) of the items in chain C.

Eq. 4 defines, $prof_{\text{ovrlp}}$ as the coverage of items in the user profile with respect to the length of the chain: the higher is the term, the more items in the user's profile and, hence, a greater variety of the user's tastes, are covered by the chain.

In future, whenever we need to refer informally to these two parts of Eq. 2, we will refer to $cand_{\text{ovrlp}}$ as the 'overlap term' and $prof_{\text{ovrlp}}$ as the 'profile term'.

4 Extended *Recommendation-by-Explanation*

As we mentioned earlier, *r-by-e* employs two main steps: *chain generation* and *chain selection*. We extend *r-by-e* by redefining the formulations used in these steps.

4.1 Feature-based generation

In *feature-based* settings, an item is described by 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.

4.1.1 Unweighted overlap

In our new versions of *r-by-e*, the unweighted overlap $\text{ovrlp}(p, i, C)$ of adding predecessor p to partial chain C that explains candidate item i is given by:

$$\text{ovrlp}(p, i, C) = \frac{2 \cdot |(F_p \setminus \text{covered}(i, C)) \cap F_i|}{|F_i| + |F_p|} \tag{5}$$

using the same definitions of F_i, F_p and $\text{covered}(iC)$ as before. This is different from Eq. 1 that we defined in Section 3.1. We now use the *harmonic mean* of the two terms of Eq. 1 and return 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.

4.1.2 Weighted overlap

Features associated with an item can be assigned weights based on how representative or informative they are to that item. In the information retrieval domain, for example, there are many ways to weight the terms of a corpus of documents (Manning et al., 2008).

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

$$\text{wovrlp}(p, i, C) = \frac{2 \cdot \left(\sum_{f \in (F_p \setminus \text{covered}(i,C)) \cap F_i)} w_{max} - |w_{fp} - w_{fi}| \right)}{|F_i| + |F_p|} \tag{6}$$

Here, w_{fi} denotes the weight of feature f in item i, and similarly for w_{fp}. The weights of the features are defined depending on the approach being used. In this paper, we evaluate with a weighted keyword approach (detailed in Section 5.1.1) and our sentiment-based weighted approach (detailed in Section 6.1.3).

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

4.2 Neighbour-based generation

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

4.2.1 Unweighted overlap

We will denote the unweighted overlap of adding predecessor p to partial chain C that explains candidate item i in the neighbour-based setting by $\text{ovrlp}(p, i, C)$, 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:

$$\text{ovrlp}(p, i, C) = \frac{2 \cdot |(N_p \setminus \text{covered}(i, C)) \cap N_i|}{|N_i \setminus \text{covered}(i, C)| + |N_p \setminus \text{covered}(i, C)|} \tag{7}$$

Here N_i and N_p are the neighbours of items i and p. $\text{covered}(i, C)$ 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.

4.2.2 Weighted overlap

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

The weighted neighbour-based overlap $\text{wovrlp}(p, i, C)$ of adding predecessor p to partial chain C that explains candidate item i is given by:

$$\text{wovrlp}(p, i, C) = \frac{2 \cdot \left(\sum_{j \in ((N_p \setminus \text{covered}(i,C)) \cap N_i)} 1 - |w_{jp} - w_{ji}| \right)}{|N_i \setminus \text{covered}(i, C)| + |N_p \setminus \text{covered}(i, C)|} \tag{8}$$

This is analogous to Eq. 6. But notice that in the numerator, we replace w_{max} by 1, which we take to be the maximum value of the similarity between a neighbour and a candidate item.

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

4.3 Generalized chain selection

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

$$\text{score}(\langle C, i \rangle, L^*) = (1 - \alpha) \cdot \mathit{cand}_{\text{ovrlp}}(\langle C, i \rangle, L^*) + \alpha \cdot \mathit{prof}_{\text{ovrlp}}(\langle C, i \rangle, L^*) \tag{9}$$

This is just a more configurable version of Eq. 2, allowing us, through parameter α, to alter the relative contributions of the two terms in the combination.

5 Extended Chains on Keywords

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

5.1 Dataset

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

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

5.1.1 Keyword extraction and weighting

For the feature-weighted approaches, we use the well-known term frequency-inverse document frequency (TF-IDF) weighting scheme (Salton and McGill, 1986). In effect, we treat an item as a document and its features as terms.

As shown in Eq. 10, the weight of a feature f of an item i with respect to the set of all items $\mathbb{I}$ is proportional to the frequency of occurrence of f in i (denoted as o_{fi}), but inversely proportional to the frequency of occurrence of f in $\mathbb{I}$ overall, thus giving preference to the features that help to discriminate each item $i \in \mathbb{I}$ from the other items in the collection. The set of items consisting of the feature f is denoted as $\mathbb{I}_f$.

$$w_{fi} = \frac{(1 + \log(o_{fi})) \cdot \left(\log \frac{|\mathbb{I}|}{|\mathbb{I}_f|}\right)}{\sqrt{\sum_{f' \in i} \left((1 + \log(o_{f'i})) \cdot \left(\log \frac{|\mathbb{I}|}{|\mathbb{I}_{f'}|}\right)\right)^2}} \tag{10}$$

The equation above is a variant of TF-IDF modeling with cosine normalization in feature–item space. Intuitively, it measures the informativeness of a feature f for an item i with respect to the informativeness of all other features in the item (Hossain et al., 2012).

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

5.2 Offline evaluation

We ran an offline experiment to evaluate different versions of *r-by-e* on this dataset. 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. We compare these versions of *r-by-e* with two versions of a content-based recommender, which works as follows.

Given a candidate item i, the first content-based recommender, designated *CB*, finds the items in the user profile 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. Finally, it recommends the n candidates with highest scores (Lops et al., 2011). In our experiments, we tried $k = \{5, 10, 15, 20, 25\}$ and used the best performing value ($k = 15$).

In the second content-based recommender, we 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, $k = |C|$. However, we have more than one version of *r-by-e* which can furnish the values for k. Hence, we obtain more than one version of *CB*-$|C|$. Specifically, we run experiments using *fb-CB*-$|C|$, where k comes from *fb*, and *wfb-CB*-$|C|$, where k comes from *wfb*. Similarly, we create *CB*-$|C|$ versions for *nb* and *wnb*.

We chose these two baselines because one of the main objectives of our experiments is to reveal the usefulness of chains over sets which requires our baselines be as similar as possible to *r-by-e* in all other aspects —to minimise confounders. As we mentioned earlier, 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. But they both use the same item features (keywords), and they both use the same similarity measure (Jaccard). However, there remains a confounder: these two systems may have different number of items in their explanations. By additionally including *CB*-$|C|$, we tried to ensure that we had a baseline that was even more comparable to *r-by-e*: *CB*-$|C|$ and *r-by-e* both have equal number of items in their corresponding explanations. Overall, we will see that, although *r-by-e* relies on content-based principles as the baselines, it has the potential to outperform the classic content-based models irrespective of the number of items that they use to generate their explanations.

In this experiment, we attempt to reveal the effect of the differences 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 a set of its neighbours (similar items) in which the keywords are used only indirectly.

- *unweighted* versus *weighted*: The former computes overlap by counting the number of elements (either features or neighbours) that are shared with the candidate item, while the latter assigns weights to the elements based on their informativeness and uses these when computing overlap.
- *CB* and *CB-|C|* versus versions of *r-by-e*: 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. We 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).
- The influence of α: When selecting the top-n chains, α balances the overlap of candidate elements and the overlap of items in the user profile (see Eq. 9). 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.

5.2.1 Evaluation Measures

In the offline experiments, for each user u, we generate a list of top-n (= 10) 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 we also evaluate using a number of 'beyond-accuracy' measures. 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 Kaminskas and Bridge (2016). We briefly describe these metrics as follows.

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

$$\frac{1}{|\mathbb{U}_T|} \sum_{u \in \mathbb{U}_T} \frac{1}{|R_u|} |R_u \cap T_u| \tag{11}$$

*Mean Average Precision (MAP).*This is an average of the estimated area of the Precision-Recall curve for each user. This metric is rank-aware, since it rewards early positive occurrences in the top-n rankings. Here, m_u is the number of relevant items for u in T_u, k indexes the positions in the ranking and $rel(k)$ is a binary function that indicates whether the item in that position is relevant or not.

$$\frac{1}{|\mathbb{U}_T|} \sum_{u \in \mathbb{U}_T} \frac{1}{m_u} \sum_{k=1}^{N} Precision@k \times rel(k) \tag{12}$$

*Diversity.*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 elements $sim(F_i, F_j)$.

$$\frac{1}{|\mathbb{U}_T|} \sum_{u \in \mathbb{U}_T} \frac{1}{|R_u|(|R_u|-1)} \sum_{i \in R_u} \sum_{j \in R_u \setminus i} 1 - sim(F_i, F_j) \tag{13}$$

*Surprise.*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 $i \in R_u$.

$$\frac{1}{|\mathbb{U}_T|} \sum_{u \in \mathbb{U}_T} \frac{1}{|R_u|} \sum_{i \in R_u} \min_{j \in P_u} 1 - sim(F_i, F_j) \tag{14}$$

*Novelty.*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.

$$\frac{1}{|\mathbb{U}_T|} \sum_{u \in \mathbb{U}_T} \frac{1}{novelty_{max} \cdot |R_u|} \sum_{i \in R_u} -\log_2 \frac{|u \in \mathbb{U},\, r(u,i) \neq 0|}{|\mathbb{U}|} \tag{15}$$

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 catalog.

$$\frac{|\cup_{u \in \mathbb{U}_T} R_u|}{|\mathbb{I}|} \tag{16}$$

5.2.2 Experiment settings

In *r-by-e*, user profiles simply contain items the user likes. We treated ratings of 4 and 5 as 'likes', so user u's profile is given by $\{i \mid r_{u,i} \geq 4\}$. We split each user's ratings into training, validation and test sets in the ratio $60 : 20 : 20$, repeated five times.

We vary the α parameter from Eq. 9 in a $[0, 1]$ interval in steps of 0.1. We consider the values (0.03, 0.06, 0.09) for the similarity threshold (θ) in the definition of N_i and different sets of values for the marginal gain threshold (ϵ): for the feature-based representation, we experimented with (0.03, 0.06, 0.09); and for neighbour-based representation, we experimented with (0.05, 0.10, 0.15).

The reason behind using different values for the marginal gain threshold ϵ for the two representations is the difference in the size of an item's set of keywords (for the feature-based representation) and the size of its set of neighbours (for the neighbour-based representation). A typical item has on average only 107 keywords while it may have over 4600 neighbours (the average out-degree

of each node in the item-item similarity graph that we mentioned earlier). This is the average number of neighbours for a similarity threshold θ of zero. As we increase θ, the average number of neighbours may decrease; however, it still remains higher than the average number of keywords for an item. 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*. When choosing the best configuration, there is an issue about what to optimize. It makes sense, for example, to choose the configuration that 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. 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 also report results where we choose configurations which optimize this on the validation set.

We use this offline experiment to decide which version performs the best and how it works on different values of α.

5.2.3 Experiment results

Table 1 and Table 2 summarize the results of the feature-based and neighbour-based approaches for top-n (= 10) recommendations. 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 MAP results, they are statistically significant. They are also 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. When comparing MAP, the difference is even greater. Similarly, the *CB-|C|* recommenders have higher diversity, surprise, and novelty but lower precision, MAP and coverage. There are no results for the *CB* recommender when optimizing for % of explanations of size 2–4 as this criteria is not applicable to *CB*.

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

Table 1: Results of the offline experiment that uses *feature-based* representations, where features are keywords. Highlighted values indicate the best results of a metric for each of the optimization criteria.

All of the *fb* and *wfb* results are statistically significant with respect to *fb-CB-|C|* and *wfb-CB-|C|* respectively and also to *CB* (t-test with $p < 0.05$) except the one shown in italics.

Recommender	θ, ϵ & α optimized for	Precision	MAP	Diversity	Surprise	Novelty	Coverage	% of explanations of size 2–4
CB		0.0209	0.0011	**0.9803**	**0.9444**	**0.5791**	**0.7606**	NA
fb		0.1076	0.0201	*0.9274*	0.7682	0.3715	0.2026	0.2899
fb-CB-|C|	Precision	0.0124	0.0009	0.9251	0.8894	0.4391	0.0585	0.2085
wfb		**0.1093**	**0.0205**	*0.9256*	0.7687	0.3640	0.1990	**0.3115**
wfb-CB-|C|		0.0124	0.0004	0.9251	0.8893	0.4391	0.0585	0.2048
CB		0.0203	0.0009	**0.9825**	**0.9498**	**0.6062**	**0.6727**	NA
fb		0.0694	0.0114	0.9498	0.7932	0.4429	0.2556	0.4598
fb-CB-|C|	Diversity	0.0063	0.0004	0.9613	0.9255	0.5198	0.0391	**0.5726**
wfb		**0.0730**	**0.0119**	0.9489	0.7915	0.4312	0.2521	0.4851
wfb-CB-|C|		0.0063	0.0002	0.9613	0.9255	0.5201	0.0391	0.5724
fb		0.0146	**0.0014**	0.9307	0.8906	0.4390	**0.2697**	**0.9882**
fb-CB-|C|	% of explanations	0.0050	0.0004	**0.9747**	**0.9346**	**0.4930**	0.0288	0.0074
wfb	of size 2–4	**0.0152**	**0.0014**	0.9302	0.8901	0.4381	0.2660	0.9878
wfb-CB-|C|		0.0053	0.0002	0.9742	0.9338	0.4915	0.0296	0.0057

Table 2: Results of the offline experiment that uses *neighbour-based* representations, where features are keywords. Highlighted values indicate the best results of a metric for each of the optimization criteria.

All of the *nb* and *wnb* results are statistically significant with respect to *nb-CB-|C|* and *wnb-CB-|C|* respectively and also to *CB* (t-test with $p < 0.05$).

Recommender	θ, ϵ & α optimized for	Precision	MAP	Diversity	Surprise	Novelty	Coverage	% of explanations of size 2–4
CB		0.0209	0.0011	**0.9803**	**0.9444**	**0.5791**	**0.7606**	NA
nb		**0.0361**	**0.0061**	0.9129	0.8410	0.4130	0.1771	0.6518
nb-CB-|C|	Precision	0.0125	0.0009	0.9240	0.8914	0.4467	0.0641	0.4266
wnb		0.0357	0.006	0.9128	0.8410	0.4104	0.1772	**0.7409**
wnb-CB-|C|		0.0124	0.0009	0.9240	0.8913	0.4464	0.0642	0.4245
CB		**0.0203**	0.0009	0.9825	0.9498	**0.6062**	**0.6727**	NA
nb		0.0138	0.0019	0.9456	0.8896	0.4896	0.2904	**0.8174**
nb-CB-|C|	Diversity	0.0041	0.0003	**0.9885**	**0.9524**	0.5596	0.0264	0.4102
wnb		0.0177	**0.002**	0.9463	0.8895	0.4664	0.2907	0.8131
wnb-CB-|C|		0.0040	0.0003	**0.9885**	**0.9524**	0.5598	0.0265	0.4078
nb		0.0157	**0.0015**	0.9121	0.8775	0.4101	**0.1602**	**0.9756**
nb-CB-|C|	% of explanations	0.0041	0.0003	**0.9837**	**0.9469**	0.5483	0.0215	0.0870
wnb	of size 2–4	**0.0159**	**0.0015**	0.9124	0.8774	0.4087	0.1599	0.9750
wnb-CB-|C|		0.0041	0.0003	**0.9837**	0.9468	**0.5490**	0.0627	0.0868

very low and in no case are they statistically significant. Overall, the *CB-|C|* recommenders attain higher values of diversity, surprise, and novelty, but have lower precision, MAP and coverage. Moreover, one can see that both *nb* and *wnb* have produced more explanations (in their case, chains) of size 2–4 size than the *CB-|C|* systems in all of the optimization criteria.

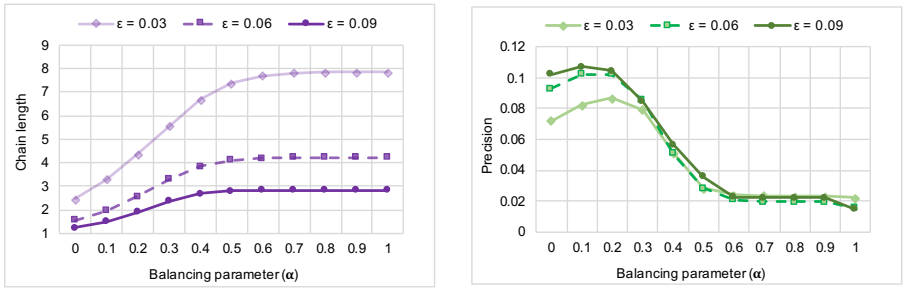
Feature-based vs. neighbour-based chain generation. We will now compare the two types of item representation by looking at results from both Table 1 and Table 2 together. When optimizing hyperparameters for precision, feature-based approaches attain three times higher precision and MAP, with similar levels of diversity and % of explanations of size 2–4. The neighbour-based approaches result in more surprising and novel recommendations. In the case of optimizing for diversity, feature-based approaches give five times more relevant recommendations with a nearly equal level of diversity. Again, neighbour-based approaches attain higher levels of surprise and novelty with over 81% of chains of size 2–4. Finally, when hyperparameters are optimized for % of explanations of size 2–4, the two types of approaches recommend items with almost equal precision and MAP, with *nb* and *wnb* producing greater, but not statistically significant, values. However, feature-based approaches produce recommendations with greater variety, higher levels of surprise and novelty and with a somewhat greater percentage of chains of size 2–4.

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).

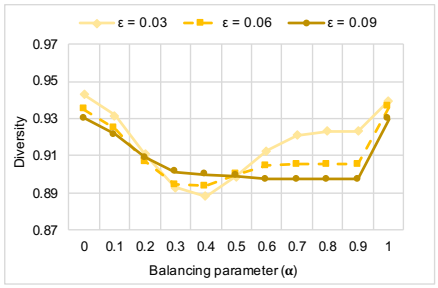
Figure 3 has six sub-figures — one for each evaluation measure. Each line that we plot in a sub-figure is for one of the three different values of ϵ. In most cases, increasing ϵ does not change the trend of the measure; it only 'shifts' the values of the measure because higher values of ϵ impose a stricter 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. 9: recall that, for conciseness, we refer to $cand_{\text{ovrlp}}$, which indicates the average amount of overlap of candidate elements, as the *overlap term*; and we refer to $prof_{\text{ovrlp}}$, 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 3(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, which gives more weight to the profile term, enables the system to select longer chains. It is also noteworthy that increasing ϵ imposes a stricter constraint on chains such that their average length reduces substantially.

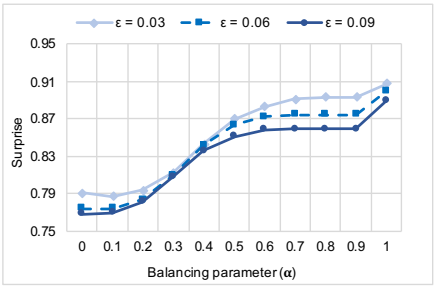


(a) Chain length

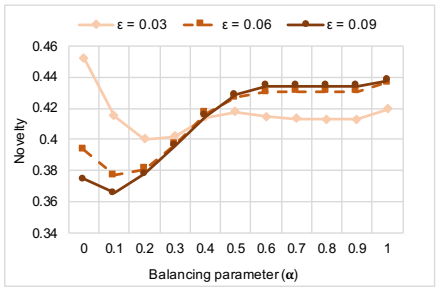
(b) Precision



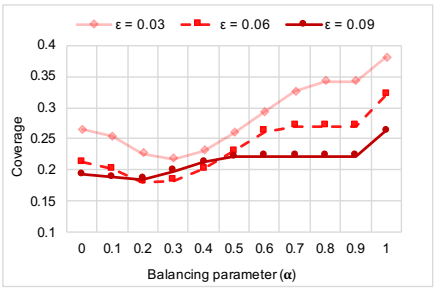
(c) Diversity



(d) Surprise



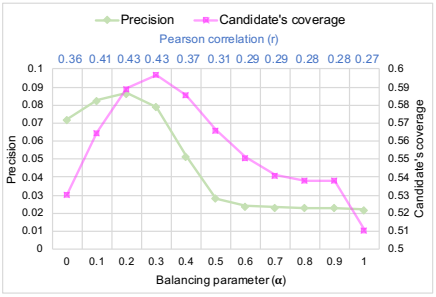
(e) Novelty



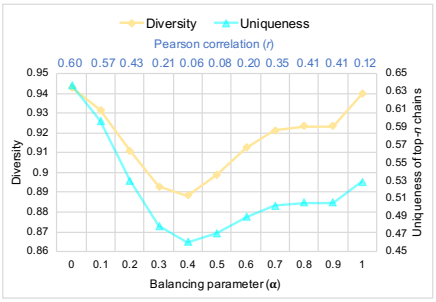
(f) Coverage

Fig. 3: 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.

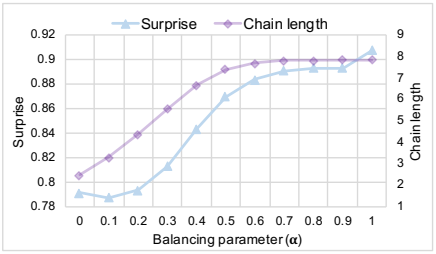
Precision: In Figure 3(b), we see that precision varies in four different ways as we increase the value of α: i) up to 0.2, it increases; ii) from 0.2 to 0.7, it decreases; iii) from 0.7 to 0.9, it remains almost constant; and iv) at 1.0, it slightly decreases. We find that, in explanation chains, precision is proportional to the candidate's coverage: the greater the candidate's coverage, the higher is the precision. We will define the candidate's coverage as the ratio of the number of candidate's elements covered by the chain members to the size of the candidate's element set. The variation in precision with respect



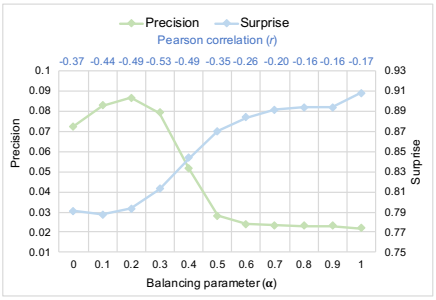
(a) Precision vs. Candidate coverage



(b) Diversity vs. Uniqueness



(c) Surprise vs. Chain length



(d) Precision vs. Surprise

Fig. 4: 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. $p < 0.05$).

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 element set that cannot be covered easily, which reduces the coverage and so also reduces the precision; for (iii), the system selects almost similar chains; and for (iv), the system totally ignores the overlap term and so the chains it selects do not try to cover the candidate, they only try to cover the profile, hence precision decreases. We plot the relationship between the precision and the candidate's coverage in Figure 4(a).

Diversity: In Figure 3(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 4(b), we show that both diversity and uniqueness follow the same trend for increasing α.

Surprise: Figure 3(d) shows that up to $\alpha = 0.6$, surprise increases; then it remains almost unchanged (up to $\alpha = 0.9$); and finally, at $\alpha = 1.0$, it increases again. Increasing α gives more weight to the profile term. In effect, the system selects longer chains that increases their surprise. The intuition behind this relationship is that chains are shorter when they easily cover the candidate's elements, while they are longer when the candidate's elements are not easily covered. We find a correlation between the surprise and the chain length that we show in Figure 4(c). We also see in Figure 4(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 3(e) shows that novelty decreases up to $\alpha = 0.2$; then, increases up to $\alpha = 0.6$; up to $\alpha = 0.9$, it remains almost unchanged; and finally, at $\alpha = 1.0$, it slightly increases. It can be seen that novelty varies in a way that is similar to surprise on increasing α. This indicates that for lower values of α, the system selects those chains that have high coverage of candidates: intuitively, popular items can be easily covered. As α increases, the system suggests novel items which cannot be covered easily and need more items from the user profile to support them.

Coverage: In Figure 3(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 catalog, while longer chains with high uniqueness cover a larger part of it.

This experiment on keyword-based representations clearly presents some interesting relationships. First and foremost, *wfb* attains the best results of all the approaches. By analysing its results more thoroughly, we find that surprise and chain length are directly proportional while precision and surprise are inversely proportional. We now would like to confirm these findings on another dataset, which is presented in the next section.

6 Extended Chains on Sentiments

Thus far, we have described some extensions to *r-by-e* and evaluated them in a keyword-based scenario. But those extensions, especially the weighted approaches, enabled us to develop *Sentiment-Aware Explanation Chains*.

Using keywords as item features presents some issues, mainly: i) where we use weights, they only convey frequency and rarity information; ii) they do not necessarily convey the elements of the items that interest the users; iii) they may be ambiguous if taken out of context. Taking these issues into

consideration, we turn to user-provided texts, i.e. user reviews, in order to further aggregate semantics to our explanation chains.

Using these texts, allied with state-of-the-art natural language processing tools, we are able to produce semantically richer item features which can both help produce better explanation chains and help users understand their recommendations. These features are 'concepts', which convey an idea, and hence they do not suffer from the same issues of synonymy and polysemy that keywords suffer from. For instance, before we could have a keyword *bank*, which could mean a land mass or a financial institution; now we would have a concept for each of those meanings. Moreover, we can extract *sentiments* towards those features, since user reviews are opinionated texts.

In this section we evaluate *Sentiment-aware Explanation Chains*, which are extended chains that use sentiment-aware concepts instead of keywords. These chains guarantee that items are connected only if they share features with close polarity (sentiment) scores.

6.1 Dataset

In this study, to increase the chances of user familiarity with the movies, we only use hetrec2011-movielens-2k movies that were released between the years 2000 and 2011 inclusive. This results in trials that use 1851 ($\approx 30\%$) of the 5992 movies in the dataset. In the offline experiments and user trial that we report in the remainder of this paper, we use sentiment data extracted from user reviews for each of these 1851 movies.

6.1.1 User reviews to concepts

A concept, in the approach given in (D'Addio et al., 2018, 2019), describes an idea or a notion. Concepts can be seen as *synsets*, i.e. sets of word synonyms which define an idea. As stated before, using concepts instead of words reduces the problems of synonymy and polysemy.

In order to extract concepts and sentiments from user reviews, two different natural language processing resources were used: Stanford CoreNLP[7] (Manning et al., 2014) for sentence splitting, parsing and sentence-level sentiment analysis; and BabelFy[8] (Moro et al., 2014) for word sense disambiguation and entity linking.

First, the items' reviews were processed with Stanford CoreNLP using the following pipeline: tokenization, part-of-speech (POS) tagging, parsing, sentence splitting and sentiment analysis.

Next, BabelFy processes the texts, returning disambiguated concepts in the form of BabelNet synsets (Navigli and Ponzetto, 2012). BabelNet[9] is a knowledge base that links several linguistic resources, such as Wikipedia, Wikidata,

[7] https://stanfordnlp.github.io/CoreNLP/
[8] http://babelfy.org/
[9] https://babelnet.org/

and WordNet, among others. Its unified ontology organizes all these resources into BabelNet synsets, which define both concepts (such as "romance", "action movie") and named entities (such as "Steven Spielberg" or "Willem DaFoe"), and provides links between them.

The synsets that are selected to compose our vocabulary (i.e., our features) are only those that come from common and proper nouns, and noun phrases. Our vocabulary is built only with these parts-of-speech because, in sentiment analysis, features most commonly come from nouns and noun phrases (such as 'action', 'plot' and 'special effects' in a movie review). Adjectives and adverbs are opinion words, i.e. words that indicate sentiment towards features, and thus are used in calculating the sentiment of a feature, sentence or document (Liu and Zhang, 2012). This vocabulary, in its current state, is very large and contains many noisy and useless features. Before assigning sentiments to them, first we need to filter out some of the concepts, reducing the vocabulary size. In the following, we employ TF-IDF weights to aid in that filtering. After that, we assign sentiments to the remaining concepts, which will be used as weights in the generation of explanation chains.

6.1.2 Filtering concepts

In this experiment, explanation chains are built over concepts. In order to improve their quality and informativeness, concepts were filtered using the following two steps:

- Concepts that appear in only one item were removed from the vocabulary. Since our chains are constructed from links between features of the target item and items present in the user profile, it is natural that features which appear in a single item are removed because they will not influence *chain generation*. Similarly, concepts that were present in all the items in the dataset were also removed since they are too general.
- Concepts obtained from the previous step were assigned weights using cosine normalized TF-IDF scores as in Eq. 10. Concepts whose average weight across the reviews in which they appeared were less than 0.01 were removed. The remaining concepts constitute the vocabulary which was used to produce item representations. Notice that here we use TF-IDF scores only to filter out the concepts and not to assign weights to them. We explain the weights separately below.

With that filtering done, the vocabulary contains 34,088 concepts. On average, a typical movie has 324 concepts ranging from 104 to 646, which shows a very high variance in the number of concepts.

6.1.3 Representing items and producing explanation chains

Finally, items are represented as a set of concepts. Each item-concept pair may have a sentiment, which gives the average quality of that concept for that movie. Sentiments are measured as scores in the range of $[1, 5]$, which can be

classified as positive (>= 4), negative (< 3), or neutral (= 3). The overall sentiment score for a feature of an item is the average of the sentiment scores related to its appearances in the reviews of the corresponding item. In order to calculate this, we use the Stanford CoreNLP sentiment analysis tool, which assigns polarities to each sentence of a document. These polarities are converted into the $[1, 5]$ scale, with 1 symbolizing a very negative sentiment and 5 a very positive sentiment. For each concept of each item, we take the average sentiment across the sentences in which it is mentioned. As an example, the concept "photography" may be mentioned in reviews related to the movie *Life of Pi*. If three out of four sentences in which "photography" was mentioned were 'very positive' (= 5) and the remaining sentence classified it as 'negative' (= 2), then the final sentiment score would be the average value, 4.25. 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*. From this, we can also define the *neighbour-based representation*, where each item is a set of its neighbours in the graph. On this dataset, we use the cosine similarity measure (Manning et al., 2008) to build the item-item similarity graph. In this graph, 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 graph which described items using keywords and defined similarity as Jaccard similarity, 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.

The concepts' sentiments 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*).

6.2 Offline evaluation

We ran an offline experiment to evaluate the performance of sentiment-aware *r-by-e*. We compare i) *wfb* and *wnb* with each other; ii) both *wfb* and *wnb* with a classic content-based recommender (*CB*); and iii) both *wfb* and *wnb* with their corresponding dynamic content-based recommenders: *wfb-CB-*$|C|$ and *wnb-CB-*$|C|$.

6.2.1 Experiment settings

We use the same experiment settings as we described previously. The only difference is in the values of the similarity threshold (θ) and the marginal gain threshold (ϵ). We experimented with θ from (0.06, 0.09, 0.12) for both *wfb* and *wnb*; and we use ϵ from (0.03, 0.06, 0.09) for *wfb*, and from (0.10, 0.20, 0.30) for *wnb*. Greater average similarity among items than before (i.e.

Table 3: Results of the offline experiment that uses *feature-based* representations on concepts. Highlighted values indicate the best results of a metric for each of the optimization criteria.

All of the *wfb* results are statistically significant with respect to *wfb-CB-|C|* *and CB* (t-test with $p < 0.05$).

Recommender	θ, ϵ & α optimized for	Precision	MAP	Diversity	Surprise	Novelty	Coverage	% of explanations of size 2–4
CB	Precision	0.0145	0.0044	**0.9383**	**0.9126**	0.5048	0.3068	NA
wfb		**0.1053**	**0.0502**	0.9130	0.8697	0.3468	**0.3271**	**0.4040**
wfb-CB-|C|		0.0036	0.0014	0.9300	0.9094	**0.5597**	0.0282	0.0000
CB	Diversity	0.0145	0.0044	**0.9383**	**0.9126**	0.5048	0.3068	NA
wfb		**0.0223**	**0.0064**	0.9225	0.9006	0.4056	**0.5185**	**0.9025**
wfb-CB-|C|		0.0033	0.0012	0.9308	0.9102	**0.5619**	0.0245	0.0000
wfb	% of explanations of size 2–4	**0.0838**	**0.0432**	0.9104	0.8778	0.3752	**0.4930**	**0.9731**
wfb-CB-|C|		0.0014	0.0011	**0.9483**	**0.9234**	**0.6102**	0.0361	0.2054

in the keyword dataset) causes us to use higher similarity thresholds. Greater similarity between chain members means they are less likely to cover different elements of the candidate item and therefore the marginal gain threshold (ϵ) for *wfb* has to be lower. In contrast, higher similarity values result in greater overlap among the neighbours of chain members and the candidate item which causes us to use higher ϵ for *wnb*.

As before, three values of each of θ and ϵ with eleven values of α gives 99 configurations for each of the two versions of *r-by-e* used here, optimized on validation data for three criteria: precision, diversity, and the percentage of explanations of size 2–4. We use this offline experiment to decide which version performs the best and how it works on different values of α.

6.2.2 Experiment results

Table 3 and Table 4 summarize the results of the weighted feature-based and weighted neighbour-based approaches.

Feature-based chain generation. We looked at the differences in the results between: i) *CB* and *wfb*; and ii) *wfb* and *wfb-CB-|C|*. The *wfb* results for both the comparisons are statistically significant. In comparison to the *wfb* recommender, the *CB* and *wfb-CB-|C|* recommenders attain higher levels of diversity, surprise, and novelty but lower values of precision, MAP and coverage. In particular, the *wfb-CB-|C|* recommender covers only around 3% of the catalog with almost irrelevant recommendations. For the same reason as before, there are no results for the *CB* recommender when optimizing for % of explanations of size 2–4. One thing worth noticing is that *wfb-CB-|C|* is capable of producing a very small number of explanations of size 2–4; in fact, while optimizing hyperparameters for precision and diversity, it selected no explanations of that size. *wfb*, on the other hand, was able to attain over 90% of explanations of that size for two out of three of the optimization criteria.

Table 4: Results of the offline experiment that uses *neighbour-based* representations on concepts. Highlighted values indicate the best results of a metric for each of the optimization criteria.

All of the *wnb* results are statistically significant with respect to *wnb-CB-|C| and CB* (t-test with $p < 0.05$).

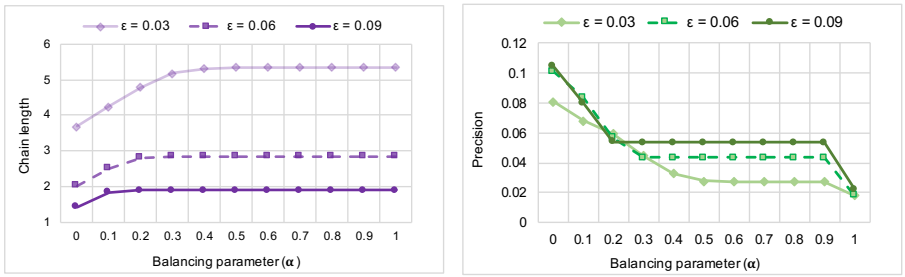
Recommender	θ, ϵ & α optimized for	Precision	MAP	Diversity	Surprise	Novelty	Coverage	% of explanations of size 2–4
CB	Precision	0.0145	0.0044	**0.9383**	**0.9126**	0.5048	**0.3068**	NA
wnb		**0.0342**	**0.0101**	0.9003	0.8884	0.4072	0.1305	**0.8448**
wnb-CB-|C|		0.0024	0.0015	0.9327	0.9108	**0.5926**	0.0348	0.0000
CB	Diversity	**0.0145**	**0.0044**	**0.9383**	0.9126	0.5048	0.3068	NA
wnb		0.0088	0.0024	0.9276	0.9053	0.4695	**0.7462**	**0.9857**
wnb-CB-|C|		0.0020	0.0015	0.9448	**0.9211**	**0.6241**	0.0739	0.7061
wnb	% of explanations of size 2–4	**0.0117**	**0.0027**	0.9186	0.9005	0.4250	**0.2700**	**0.9997**
wnb-CB-|C|		0.0018	0.0014	**0.9513**	**0.9247**	**0.5989**	0.0976	0.1804

Neighbour-based chain generation. Now, we see the differences in the results between: i) *CB* and *wnb*; and ii) *wnb* and *wnb-CB-|C|*. They are statistically significant in all cases. Changing the item representation does not apply to *CB*, so its results are the same as before. The *CB* and *wnb-CB-|C|* recommenders attain higher values of diversity, surprise, and novelty than *wnb*, while having lower precision, MAP and % of explanations of size 2–4, in two out of three configurations – while optimizing for Diversity, *wnb* was not able to have higher precision and MAP than *CB*. In particular, the *wnb-CB-|C|* recommender has lower catalog coverage with explanations satisfying the size constraint than *wnb*.

Feature-based vs. neighbour-based chain generation. We will now compare the two types of item representation by looking at results from both Table 1 and Table 2 together. One can see that *wfb* attained, in all configurations, a much higher level of precision and MAP. When optimized for % of explanations with size 2–4, it achieved around eight times more relevant suggestions than *wnb*. Furthermore, a MAP score 160 times higher suggests that relevant items were ranked much higher in *wfb* than *wnb*. When optimized for precision and % of explanations of size 2–4, *wfb* also presented greater coverage. On the other hand, *wnb* presented greater levels of novelty and % of explanations with size 2–4 in every configuration, with surprise being greater in two out of three configurations. As for diversity, they both present similar results in every configuration.

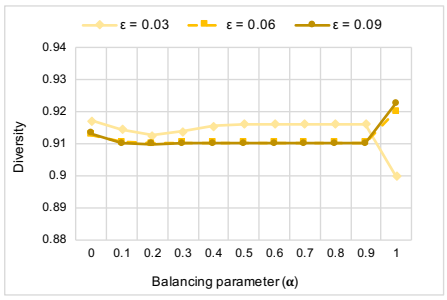
We now select one of the two representations for further study. As before, to pick one, we assume that optimizing for the percentage of explanations of size 2–4 is best, since it generally gives explanations that are not so long as to be uninterpretable. In this setting, *wfb* performs better than *wnb*, and so this is the version that we will explore further.

We will study the effect of hyperparameters ϵ and α on the performance of *wfb*. Figure 5 shows six sub-figures — one for each evaluation measure. It

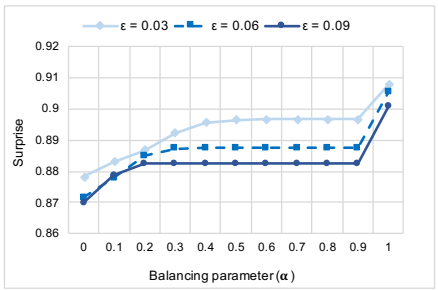


(a) Chain length

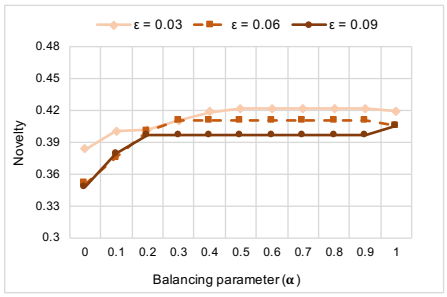
(b) Precision



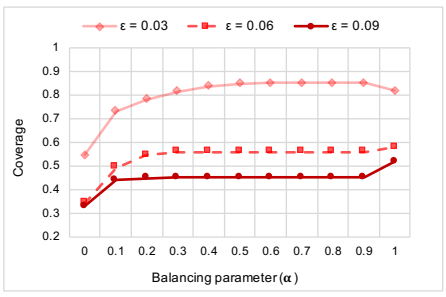
(c) Diversity



(d) Surprise



(e) Novelty

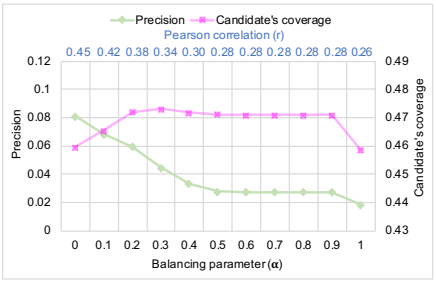


(f) Coverage

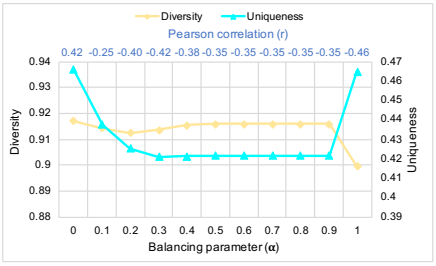
Fig. 5: Results for *wfb* with $\theta = 0.06$, $\epsilon \in \{0.03, 0.06, 0.09\}$, and α ranges in $[0.0 - 1.0]$ for extended chains on sentiments.

can be seen that in, all the plots, for $\epsilon \in \{0.06, 0.09\}$, values of the evaluation measures remain almost constant for values of α in the range of $[0.02 - 0.09]$. Only for $\epsilon = 0.03$ is there some variation in the evaluation measures but this variation occurs only for the initial and last values of α; the measures remain largely constant in between. This shows the effect of high similarity among items in this experiment.

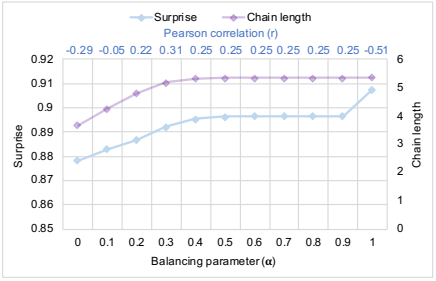
We look in detail at the results for each evaluation measure individually. Again, we refer to Eq. 9 and, as before, for conciseness, we refer to the two



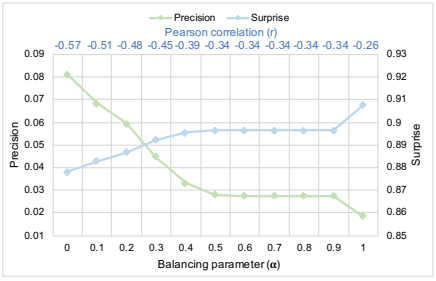
(a) Precision vs. Candidate's coverage



(b) Diversity vs. Uniqueness



(c) Surprise vs. Chain length



(d) Precision vs. Surprise

Fig. 6: 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. $p < 0.05$).

terms that make up its definition as the *overlap term* and the *profile term* respectively.

Chain length: In Figure 5(a), we see that the length of top-n chains increases up to $\alpha = 0.4$; then, increasing α does not show a noticeable effect on the length. This indicates that for lower α, the overlap term dominates, and the system selects shorter chains. For higher values of ϵ, the system imposes a stricter constraint on chains so their average length reduces substantially.

Precision: In Figure 5(b), for $\epsilon = 0.03$, we see that precision varies in three different ways when increasing the value of α: i) up to 0.5, it decreases; ii) from 0.5 to 0.9, it remains unchanged; and iii) at 1.0, it decreases again. As we mentioned before, in explanation chains, precision is proportional to the candidate's coverage. In particular, for this experiment, in the case of (i), items are very similar to each other and adding more members to the chain (i.e. increasing chain length) does not necessarily increase the candidate's coverage; for (ii), the system selects almost similar chains; and for (iii), the system totally ignores the overlap term, selects chains based only on the profile term and these chains do not try to cover the candidate, only the profile; therefore, precision decreases. For $\epsilon \in \{0.06, 0.09\}$, where there is a stricter constraint, precision

becomes constant even earlier. We show the relationship between the precision and the candidate's coverage for different values of α in Figure 6(a).

Diversity: In Figure 5(c), we see diversity remains almost unchanged up to $\alpha = 0.9$ and decreases at $\alpha = 1.0$ for $\epsilon = 0.03$ (and increases for $\epsilon \in \{0.06, 0.09\}$). In our system, as we have shown in the previous experiment, the diversity of the top-n recommendations depends upon their uniqueness: the lower the overlap among the members of top-n chains, the higher is the diversity. However, in this experiment, items are quite similar to each other so varying α does not affect the uniqueness of the top-n chains except at $\alpha = 0.0$ when the profile term is totally ignored. We show in Figure 6(b) that, in all but one case, uniqueness of chain members is negatively correlated with diversity.

Surprise: Figure 5(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 elements 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 6(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 6(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(e) shows that novelty increases up to $\alpha = 0.4$, then remains almost at the same level. It shows that on lower values of α, the overlap term dominates, enabling the system to 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 may slightly increase with the increase in chain length.

Coverage: Figure 5(f) shows that coverage increases up to $\alpha = 0.4$, then remains almost unchanged. In this experiment, this indicates that up to $\alpha = 0.4$, coverage increases with the increase in the 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.

The experimental results on sentiment-based representations confirm that *wfb* performs the best out of all *r-by-e* approaches. Again, we find that surprise and chain length are directly proportional while precision and surprise are inversely proportional. However, because of greater similarities among items, the balancing parameter α does not have as much impact as it had when we used keyword-based representations.

6.3 User trials

In (Rana and Bridge, 2018), we reported the results of a user trial for the basic version of *r-by-e* (as described in Section 3) on the keyword dataset that we summarised in Section 5.1. We updated our web-based system (Rana and Bridge, 2018) in order to conduct a new user trial on the sentiment-aware version of the dataset. From our offline experiments, we considered configurations of *r-by-e* that optimized the percentage of explanations of size 2–4. With this criterion, *wfb* performs better than other versions of *r-by-e*, and so this is the version that we picked for the new user trial. We compare *wfb* with *wfb-CB-*$|C|$ with both using sentiment-aware concepts as features.

r-by-e is, above all, a recommender and so we designed one trial to measure recommendation quality and a second trial to measure explanation quality. We recruited participants through personal email lists and Twitter. In total, 144 people attempted the trials. The majority of them were undergraduate and postgraduate students from universities in Ireland, Brazil, and India. Participants were fully anonymized and we collected no demographic data. 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, 55 for the recommendation trial and 45 for the explanation trial. To familiarize our participants with the interface functionalities, we provided instructions in our invitation email. Also, on-screen instructions were given at every stage of the trial.

6.3.1 Recommendation trial

In this trial, we investigate whether *wfb* generates more diverse, serendipitous and relevant recommendations than *wfb-CB-*$|C|$ or not.

Experiment settings. The recommendation trial is defined as follows. We asked users to evaluate two different lists of recommendations produced by the two recommenders we were comparing. These lists of recommendations have length 5 and are sorted in decreasing order of recommender scores.

Before displaying the recommendations, we ensured that the two lists contained different movies. Each movie that was common to both lists was removed and the next best recommendations from the top-10 were added to the end of the lists. If it was not possible to create two different lists of length 5 from the top-10 recommendations, the user's responses to the survey were discarded. We did this to avoid skewing responses about the diversity of recommendations: shorter lists are less likely to be diverse. In our experiments, there were only two users whose responses were discarded for this reason.

For half the users, the list on the left ('List A') came from *r-by-e* and the list on the right ('List B') from *CB-*$|C|$; for the other half of the users, List A was from *CB-*$|C|$ and List B from *r-by-e*. Users were not aware of which list belonged to which recommender.

Table 5: Results of the Recommendation Trial for Extended *r-by-e* on Sentiments.

User's opinion	Diversity	Serendipity	Satisfaction
Much more *r-by-e*	11	8	20
More *r-by-e*	17	16	14
About the same	9	12	5
More *CB-*$\|C\|$	11	12	7
Much more *CB-*$\|C\|$	7	7	9

Participants were required to answer three questions on Diversity, Serendipity and Satisfaction.

- Diversity: Which list has a greater variety of movies?
- Serendipity: Which list has more pleasantly surprising recommendations?
- Satisfaction: Which list has more recommendations that you would be likely to try?

Their answers were on a 5-point: Much more List A than List B; More List A than List B; About the Same; More List B than List A; and Much more List B than List A.

Experiment results. Fifty-five participants completed this trial. Table 5 summarizes their responses.

- *Diversity question:* 50.9% of participants found *r-by-e* recommendations to be much more diverse or more diverse than *CB-*$|C|$ recommendations, 16.4% found the recommendation lists to be equally diverse, leaving 32.7% finding *CB-*$|C|$ to be much more or more diverse.
- *Serendipity question:* 43.7% of participants found *r-by-e* recommendations to be much more or more pleasantly surprising, 21.8% found the recommendation lists to be equally surprising, leaving 34.5% finding *CB-*$|C|$ to be much more or more surprising.
- *Satisfaction question:* 61.8% of participants found *r-by-e* recommendations to be ones they would be much more or more likely to try, 9.1% found the recommendations to be equally worthy of trying, leaving 29.1% finding *CB-*$|C|$ to be much more or more worth trying.

On all criteria *r-by-e* produced the better recommendation lists. However, only in the case of the satisfaction question was this statistically significant. (We used two-tailed proportion tests with significance level $p_0 = 0.05$. The null hypothesis was that those preferring *r-by-e* was equal to those preferring *CB-*$|C|$, i.e. ignoring those who thought the two lists were about the same.) This is also in line with the results of the offline experiments where the weighted feature-based approach attains better precision than the other approaches, while remaining competitive on measures of diversity and serendipity. Since

precision is directly related to user satisfaction, i.e. the former directly evaluates the recommender's ability to generate relevant suggestions, *r-by-e* is able to provide in both offline and online settings more relevant (and thus, more satisfactory) suggestions than its baseline competitor.

6.3.2 Explanation trial

In this trial, we investigate whether *wfb* generates more effective explanations than *wfb-CB-*$|C|$ or not.

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 (Bilgic and Mooney, 2005; Gedikli et al., 2014). The users are initially asked to rate a recommendation in the case where they are given only the explanation and not the identity of the movie. This is called the *explanation-rating*. The users are asked later to re-rate the recommended item in the case where they are given information about the item, including its identity. This is called the *actual-rating*. An effective explanation is one where the explanation-rating is close to the actual-rating. Effective explanations will be ones for which (a) μ_d (the mean difference between explanation-ratings and corresponding actual-ratings) is close to zero; (b) σ_d (their standard deviation) is small; and (c) r (their Pearson correlation) is highest.

Experiment settings. Explanation Chains were displayed in the fashion shown in Figures 7 and 8: arrows connect a movie to its successor in the chain. *CB-*$|C|$'s explanations (sets of neighbours, rather than chains), on the other hand, were displayed in the fashion shown in Figure 9: arrows connect each movie to the recommended movie. Note how the movie identity is redacted.

In both cases, users can mouse over parts of the explanation, which causes the system to display features that movies have in common. A maximum of three features are displayed in any box. The challenge in selecting the features is to rank them. We cannot use features' sentiment scores: positive sentiment features will always have higher scores than the other two sentiment types and so this ranking approach will most likely return positive sentiment features. Therefore, we use the following steps: i) we split the features into three lists, one for each type of sentiment; ii) we calculate the proportion of the full feature list that is accounted for by each of the three lists; iii) we rank each list using features' TF-IDF scores —specifically, in order to make sure that the feature is important for both the movies we sum up both the feature's TF-IDF scores for both movies and rank the list based on these; iv) finally, we select features from the three ranked lists in the order of their proportion (i.e. the highest proportion sentiment type features will be added first) such that the top three features maintain nearly the same proportion of sentiment types as they have in the original list. Each feature is also associated with an emoji indicating the sentiment of the feature. We used different colors for different sentiments:

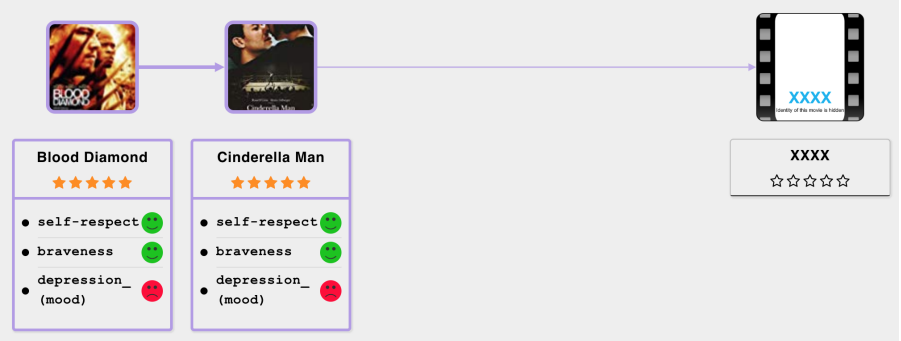


Fig. 7: A screenshot of an explanation chain. The user has moused over the arrow that connects the first two movies, which causes the system to bring up boxes of sentiments that these two movies have in common.

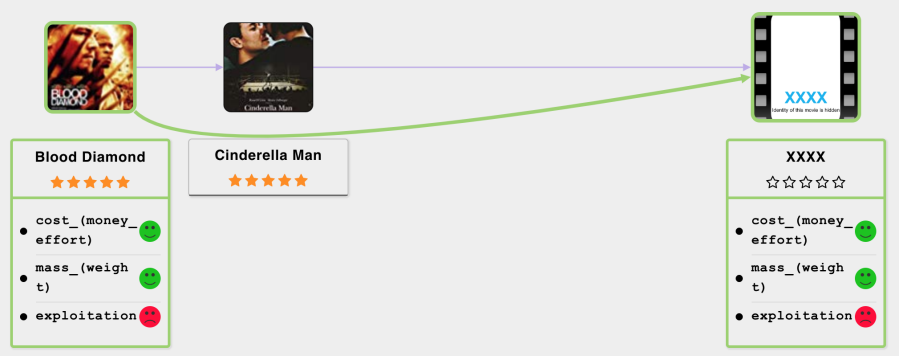


Fig. 8: A screenshot of an explanation chain. The user has moused over the icon for the first movie, which causes the system to display an arrow between that movie and the recommended movie and to bring up boxes of sentiments that these two movies have in common.

positive (green smiley face), negative (red frowny face), and neutral (yellow neutral face).

We asked the user to supply an explanation-rating (1-5 stars): how much they thought they might like the movie based only on the explanation. We do it for both *r-by-e* and *CB-|C|* explanations. After the users have given these $2n$ ratings, the system then shows them in a random order each of the n recommended movies again. This time, the identity of the movie is not redacted but no explanation is shown. Instead, we show genre, plot synopsis, main cast members, directors, writers, duration, and release date. Again we ask them for ratings to indicate how much they think they will like the movies. Note that, although users have rated the same movie three times, nothing in the on-screen instructions makes this apparent.

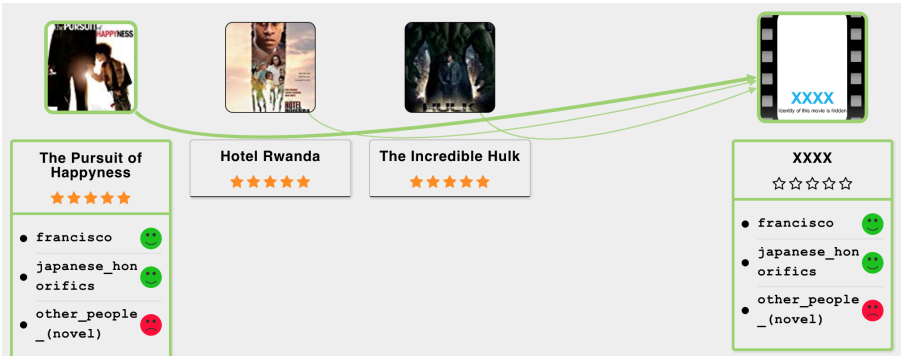


Fig. 9: A screenshot of a *CB-|C|* explanation. The user has moused over the icon for the first movie, which causes the system to increase the width of the arrow between that movie and the recommended movie and to bring up boxes of sentiments that these two movies have in common.

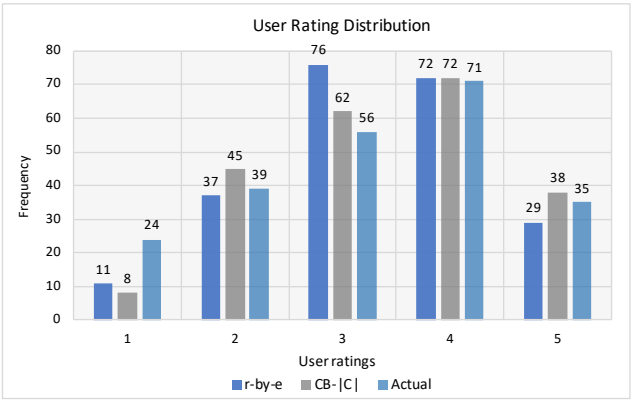


Fig. 10: Ratings from the Explanation Trial for Extended chains on Sentiments.

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 10 shows the distribution of the users' ratings; Table 6 gives summary statistics.

We can see that users mostly think they will like the movies that the system recommends, both when they see explanations only and when they see movie identities. For the differences between explanation-ratings and actual-ratings, Figure 11 shows the distribution of values and Table 7 gives summary statistics.

The mean difference between *r-by-e* ratings and actual ratings is 0.0756; for *CB-|C|*, it is 0.1467. Hence, both kinds of explanations cause users to

Table 6: Mean (μ), standard deviation (σ) and Pearson correlation (r) of ratings from the Explanation Trial for Extended *r-by-e* on Sentiments.

Rating type	μ	σ	r
Actual	3.2400	1.2193	–
r-by-e	3.3156	1.0492	0.5338
CB-|C|	3.3867	1.0925	0.1613

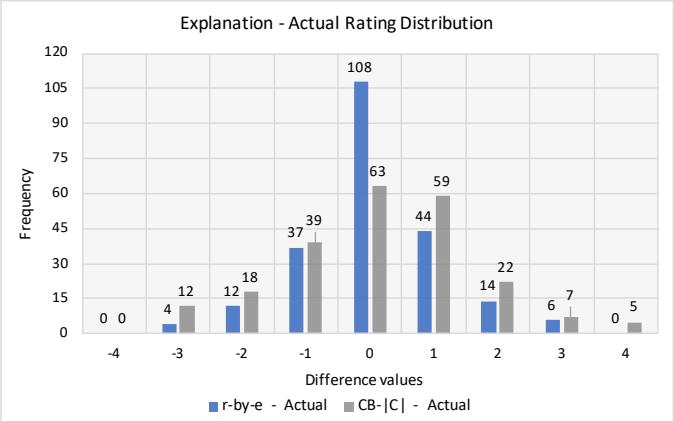


Fig. 11: Differences in ratings from the Explanation Trial for Extended *r-by-e* on Sentiments.

Table 7: Differences in ratings from the Explanation Trial for Extended *r-by-e* on Sentiments.

Explanation type	μ_d	σ_d	95% Conf. Int.
r-by-e	0.0756	1.1054	(-0.0688, 0.2199)
CB-|C|	0.1467	1.5002	(-0.0494, 0.3427)

overestimate their actual-ratings. Using a two-tailed paired t-test ($p_0 = 0.05$), we observed that in this study, i) the difference between *r-by-e*-ratings and actual-ratings are not statistically different; ii) the differences between *CB-|C|*ratings and actual-ratings are also not statistically significant; and iii) *r-by-e*-ratings and *CB-|C|*-ratings are not statistically different. In terms of μ_d and σ_d, then, neither kind of explanation is better than the other. But there is still the question of correlation with the actual-ratings.

Table 6 shows r, the Pearson correlation between explanation-ratings and actual-ratings. We see that *r-by-e*-ratings are better correlated with actual-ratings. We calculated the probability of getting this correlation due to chance to be 0 in both cases. This is evidence that the relationships between items de-

picted in *r-by-e*'s explanation chains are capable of providing a more thorough explanation to users, thus helping them make accurate and informed decisions towards items.

7 Discussion

We have presented a comprehensive empirical comparison of all four versions of *r-by-e* with their corresponding customized classic content-based methods on two variants of a movie recommendation dataset. We saw that feature-based approaches performed the best in our experiments. On both the datasets, feature-based approaches attained greater recommendation accuracy and, in most cases, remain competitive on beyond-accuracy measures. Indeed, content-based methods generate more diverse and serendipitous recommendations, but they were less accurate. Such recommendations are concentrated on a small fraction of the catalog that results in lower catalog coverage than the versions of *r-by-e*. In general, a random recommender, for example, may generate diverse set of recommendations, but they are less likely to be relevant to the user. Alternatively, a popularity-based recommender may, in many cases, achieve high accuracy but such recommendations may be less surprising (Adomavicius and Kwon, 2008). Content-based approaches only consider item-item similarities and are unable to provide a balance between recommendation accuracy, diversity and surprise. As stated before, feature-based *r-by-e* attempts to cover the features of the candidate item as well as the items in the user profile. This enables *r-by-e* to generate more relevant recommendations while remaining competitive in its diversity and serendipity.

Further, the results for neighbour-based *r-by-e* indicate that a comparison between these approaches and the content-based baselines follow roughly the same pattern of comparison as with the feature-based approaches. This highlights *r-by-e*'s capability of understanding user's preferences better than the content-based approaches even when the item representations make no explicit use of item features.

On comparing feature-based approaches to neighbour-based ones, we find that the feature-based *r-by-e* performs better. Neighbour-based approaches try to cover neighbours of the candidate items instead of their features. Intuitively, this may lead to higher levels of diversity and surprise than the feature-based approaches. Our offline results confirm that the neighbour-based recommendations are more diverse and surprising but there is a trade-off with their relevance. However, they still perform better than the content-based baselines.

We also observed that the use of sentiment-based features in the construction of the chains adds another layer of information: users are able to perceive not only the features shared by the movies they like, as well as the reasoning behind the chain, but also whether those features are perceived as positive or negative elements of the films. All of this semantic information, coupled with the chain itself, provides sufficient support to users to make appropriate decisions towards recommended movies.

Finally, even though the user trial described in this article cannot be directly comparable to the one presented previously in (Rana and Bridge, 2018), if we look at the results of both Explanation Trials, we can see that the sentiment-based representation, on *average*, produces more closely related explanations and actual ratings, with a greater correlation among them. This is an indication that sentiment-based representations do help *r-by-e* construct more effective explanations than keyword-based representations; however, more experiments are required to support this claim.

8 Conclusion

In this paper, we considered various extensions to *r-by-e*. We presented two item representations: i) feature-based; and ii) neighbour-based. The former describes an item by a set of its features, and the latter makes no explicit reference to features but describes an item as a set of its neighbours instead. For each of these representations, we also explored weighting schemes to assign weights to the features (or neighbours) 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.

We evaluated these extensions to *r-by-e* on two different datasets: first, where items are described by keywords, and second, where items are described by features weighted by sentiment scores. Our offline experiments show that *weighted feature-based (wfb)* version gives more relevant recommendations than all other versions of *r-by-e* and the baselines while being competitive on measures of diversity and serendipity.

We also conducted user trials with *sentiment-aware explanation chains* to evaluate the quality of recommendations and the effectiveness of the corresponding explanations. We found that *r-by-e* produces recommendations that are more diverse and serendipitous than those of a baseline (although this is not statistically significant) and with statistically significantly higher levels of user satisfaction. User responses also confirmed that the sentiment-aware explanation chains allow users to make more accurate judgements about the quality of the recommended items than do the baseline's explanations.

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.

Another interesting research direction would be to extend the explanation chains from a 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 items' elements. 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.

Acknowledgements We thank Arpit Jain for help with web design. This paper emanates from research supported by a grant from Science Foundation Ireland (SFI) under Grant Number SFI/12/RC/2289-P2 which is co-funded under the European Regional Development Fund.

References

Abdollahi B, Nasraoui O (2016) Explainable matrix factorization for collaborative filtering. In: Proceedings of the 25th International Conference Companion on World Wide Web, International World Wide Web Conferences Steering Committee, pp 5–6

Adomavicius G, Kwon Y (2008) Overcoming accuracy-diversity tradeoff in recommender systems: A variance-based approach. In: Proceedings of WITS, Citeseer, vol 8

Bilgic M, Mooney RJ (2005) Explaining recommendations: Satisfaction vs. promotion. In: Beyond Personalization Workshop, IUI, vol 5, p 153

Bridge D, Dunleavy K (2014) If you liked herlocker et al.'s explanations paper, then you might like this paper too. In: Joint Workshop on Interfaces and Human Decision Making in Recommender Systems, p 22

Chang S, Harper FM, Terveen LG (2016) Crowd-based personalized natural language explanations for recommendations. In: Proceedings of the 10th ACM Conference on Recommender Systems, ACM, pp 175–182

Chen L, Pu P (2005) Trust building in recommender agents. In: Proceedings of the Workshop on Web Personalization, Recommender Systems and Intelligent User Interfaces at the 2nd International Conference on E-Business and Telecommunication Networks, Citeseer, pp 135–145

Cleger S, Fernández-Luna JM, Huete JF (2014) Learning from explanations in recommender systems. Information Sciences 287:90–108

Cleger-Tamayo S, Fernandez-Luna JM, Huete JF (2012) Explaining neighborhood-based recommendations. In: Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval, ACM, pp 1063–1064

Costa F, Ouyang S, Dolog P, Lawlor A (2018) Automatic generation of natural language explanations. In: Proceedings of the 23rd International Conference on Intelligent User Interfaces Companion, ACM, p 57

D'Addio RM, Fressato EP, Da Costa AF, Manzato MG (2018) Incorporating semantic item representations to soften the cold start problem. In: Proceedings of the 24th Brazilian Symposium on Multimedia and the Web, ACM, pp 157–164

D'Addio RM, Marinho RS, Manzato MG (2019) Combining different metadata views for better recommendation accuracy. Information Systems 83:1–12

Dong R, Smyth B (2017) User-based opinion-based recommendation. In: Proceedings 26th IJCAI, Melbourne, Australia, pp 4821–4825

Friedrich G, Zanker M (2011) A taxonomy for generating explanations in recommender systems. AI Magazine 32(3):90–98

Gedikli F, Ge M, Jannach D (2011) Understanding recommendations by reading the clouds. In: International Conference on Electronic Commerce and Web Technologies, Springer, pp 196–208

Gedikli F, Jannach D, Ge M (2014) How should i explain? a comparison of different explanation types for recommender systems. International Journal of Human-Computer Studies 72(4):367–382

Herlocker JL, Konstan JA, Riedl J (2000) Explaining collaborative filtering recommendations. In: Proceedings of the 2000 ACM conference on Computer supported cooperative work, ACM, pp 241–250

Hossain MS, Gresock J, Edmonds Y, Helm R, Potts M, Ramakrishnan N (2012) Connecting the dots between pubmed abstracts. PloS one 7(1):e29509

Kaminskas M, Bridge D (2016) Diversity, serendipity, novelty, and coverage: A survey and empirical analysis of beyond-accuracy objectives in recommender systems. ACM Transactions on Interactive Intelligent Systems 7(1):2:1–2:42

Konstan JA, Riedl J (2012) Recommender systems: From algorithms to user experience. User modeling and user-adapted interaction 22(1-2):101–123

Kouki P, Schaffer J, Pujara J, O'Donovan J, Getoor L (2017) User preferences for hybrid explanations. In: Proceedings of the Eleventh ACM Conference on Recommender Systems, ACM, p 84–88

Kouki P, Schaffer J, Pujara J, O'Donovan J, Getoor L (2019) Personalized explanations for hybrid recommender systems. In: Proceedings of the 24th International Conference on Intelligent User Interfaces, ACM, p 379–390

Kulesza T, Stumpf S, Burnett M, Yang S, Kwan I, Wong WK (2013) Too much, too little, or just right? Ways explanations impact end users' mental models. In: Visual Languages and Human-Centric Computing (VL/HCC), 2013 IEEE Symposium on, IEEE, pp 3–10

Linden G, Smith B, York J (2003) Amazon. com recommendations: Item-to-item collaborative filtering. IEEE Internet computing 7(1):76–80

Liu B, Zhang L (2012) A survey of opinion mining and sentiment analysis. In: Aggarwal CC, Zhai C (eds) Mining Text Data, Springer US, pp 415–463

Lops P, De Gemmis M, Semeraro G (2011) Content-based recommender systems: State of the art and trends. In: Recommender systems handbook, Springer, pp 73–105

Lu Y, Dong R, Smyth B (2018) Coevolutionary recommendation model: Mutual learning between ratings and reviews. In: Proceedings of the 2018 World Wide Web Conference, pp 773–782

Manning CD, Raghavan P, Schütze H (2008) Introduction to Information Retrieval. Cambridge University Press, USA

Manning CD, Surdeanu M, Bauer J, Finkel J, Bethard SJ, McClosky D (2014) The stanford corenlp natural language processing toolkit. In: Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations, pp 55–60

McAuley J, Leskovec J (2013) Hidden factors and hidden topics: understanding rating dimensions with review text. In: Proceedings of the 7th ACM Conference on Recommender Systems, ACM, p 165–172

Moro A, Raganato A, Navigli R (2014) Entity linking meets word sense disambiguation: a unified approach. Transactions of the Association for Computational Linguistics 2:231–244

Muhammad K, Lawlor A, Rafter R, Smyth B (2015) Great explanations: Opinionated explanations for recommendations. In: International Conference on Case-Based Reasoning, Springer, pp 244–258

Muhammad K, Lawlor A, Smyth B (2016) On the use of opinionated explanations to rank and justify recommendations. In: Proceedings of the 28th FLAIRS Conference, pp 554–559

Musto C, Narducci F, Lops P, De Gemmis M, Semeraro G (2016) ExpLOD: A Framework for Explaining Recommendations Based on the Linked Open Data Cloud. In: Proceedings of the 10th ACM Conference on Recommender Systems, pp 151–154

Musto C, Narducci F, Lops P, de Gemmis M, Semeraro G (2019) Linked open data-based explanations for transparent recommender systems. International Journal of Human-Computer Studies 121:93–107

Navigli R, Ponzetto SP (2012) Babelnet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network. Artificial Intelligence 193:217–250

Ni J, McAuley J (2018) Personalized review generation by expanding phrases and attending on aspect-aware representations. In: Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), ACM, pp 706–711

Ni J, Li J, McAuley J (2019) Justifying recommendations using distantly-labeled reviews and fine-grained aspects. In: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), Association for Computational Linguistics, pp 188–197

Papadimitriou A, Symeonidis P, Manolopoulos Y (2012) A generalized taxonomy of explanations styles for traditional and social recommender systems. Data Mining and Knowledge Discovery 24(3):555–583

Pu P, Chen L (2007) Trust-inspiring explanation interfaces for recommender systems. Knowledge-Based Systems 20(6):542–556

Rana A (2020) Chain-based recommendations. PhD thesis, Insight Centre for Data Analytics, School of Computer Science & Information Technology, University College Cork, Ireland

Rana A, Bridge D (2017) Explanation chains: Recommendations by explanation. In: Tikk D, Pu P (eds) Proceedings of the Poster Track of the 11th ACM Conference on Recommender Systems, CEUR Workshop Proceedings, vol-1905

Rana A, Bridge D (2018) Explanations that are intrinsic to recommendations. In: Proceedings of the 26th Conference on User Modeling, Adaptation and Personalization, ACM, pp 187–195

Ribeiro MT, Singh S, Guestrin C (2016) "Why should I trust you?": Explaining the predictions of any classifier. CoRR abs/1602.04938

Rossetti M, Stella F, Zanker M (2013) Towards explaining latent factors with topic models in collaborative recommender systems. In: Database and Expert Systems Applications (DEXA), 2013 24th International Workshop on, IEEE, pp 162–167

Salton G, McGill M (1986) Introduction to modern information retrieval

Sato M, Ahsan B, Nagatani K, Sonoda T, Zhang Q, Ohkuma T (2018) Explaining recommendations using contexts. In: 23rd International Conference on Intelligent User Interfaces, ACM, pp 659–664

Scheel C, Castellanos A, Lee T, De Luca EW (2012) The reason why: A survey of explanations for recommender systems. In: International Workshop on Adaptive Multimedia Retrieval, Springer, pp 67–84

Sinha R, Swearingen K (2002) The role of transparency in recommender systems. In: CHI'02 extended abstracts on Human factors in computing systems, ACM, pp 830–831

Tintarev N (2007) Explanations of recommendations. In: Proceedings of the 1st ACM conference on Recommender systems, ACM, pp 203–206

Tintarev N, Masthoff J (2007a) Effective explanations of recommendations: user-centered design. In: Proceedings of the 1st ACM conference on Recommender systems, ACM, pp 153–156

Tintarev N, Masthoff J (2007b) A survey of explanations in recommender systems. In: Data Engineering Workshop, 2007 IEEE 23rd International Conference on, IEEE, pp 801–810

Tintarev N, Masthoff J (2008) The effectiveness of personalized movie explanations: An experiment using commercial meta-data. In: International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems, Springer, pp 204–213

Tintarev N, Masthoff J (2015) Explaining recommendations: Design and evaluation. In: Recommender systems handbook, Springer, pp 353–382

Vig J, Sen S, Riedl J (2009) Tagsplanations: explaining recommendations using tags. In: Proceedings of the 14th international conference on Intelligent user interfaces, ACM, pp 47–56

Yu C, Lakshmanan LV, Amer-Yahia S (2009) Recommendation diversification using explanations. In: 2009 IEEE 25th International Conference on Data Engineering, IEEE, pp 1299–1302

Zhang Y, Chen X (2018) Explainable recommendation: A survey and new perspectives. arXiv preprint arXiv:180411192

Arpit Rana is an Assistant Professor at Dhirubhai Ambani Institute of Information and Communication Technology (DA-IICT), Gandhinagar, India. He did his Ph.D. from University College Cork, Ireland in 2020. Prior to joining DA-IICT, he worked as a Postdoctoral Fellow at the Department of Industrial Engineering, University of Toronto, Canada. His research interests are recommender systems, human-computer interaction, and machine learning.

Rafael M. D'Addio currently works as Lead Data Scientist on Dynaccurate SARL. He received his M.Sc. and Ph.D. on Computer Science from the University of São Paulo, Brazil, in 2015 and 2020, respectively. His fields of interest are recommender systems, machine learning, natural language processing and linked data.

Marcelo G. Manzato is Associate Professor at the Mathematics and Computer Science Institute of the University of São Paulo, SP, Brazil. He received his BSc in Computer Science from the State University of Londrina, Brazil, in 2000, and his M.Sc. and Ph.D. in Computer Science from the University of São Paulo, Brazil, in 2006 and 2011, respectively. His research interests are recommender systems, machine learning and human-computer interaction.

Derek Bridge is a senior lecturer in the School of Computer Science and Information Technology at University College Cork, Ireland. He is also a Principal Investigator in Insight, the Science Foundation Ireland Centre for Data Analytics. His research interests lie within Artificial Intelligence, especially recommender systems and case-based reasoning.