ClusterExplorer: Enable User Control over Related Recommendations via Collaborative Filtering and Clustering

Denis Kotkov
kotkov.denis.ig@gmail.com
Åbo Akademi University
Finland, Turku

Qian Zhao
qzhao101@bloomberg.net
Bloomberg L.P.
United States, New York

Kati Launis
kati.launis@helsinki.fi
University of Helsinki
Finland, Helsinki

Mats Neovius
mats.neovius@abo.fi
Åbo Akademi University
Finland, Turku

ABSTRACT

Related item recommendations have a long history in recommender systems, but they tend to be a static list of similar items with respect to a target item of interest without any support of user control. In this paper, we propose ClusterExplorer, a novel approach for enabling user control over related recommendations. The approach allows users to explore the latent space of user-item interactions through controlling related recommendations. We evaluated ClusterExplorer in the book domain with 42 participants recruited in a public library and found that our approach has higher user satisfaction of browsing items and is more helpful in finding interesting items compared to traditional related item recommendations.

CCS CONCEPTS

• **Information systems → Recommender systems**; *Users and interactive retrieval.*

KEYWORDS

recommender systems, user control, related item recommendations, information exploration tool, interactive recommendation, user interfaces, conversational recommender systems, critiquing recommender systems

ACM Reference Format:
Denis Kotkov, Qian Zhao, Kati Launis, and Mats Neovius. 2020. ClusterExplorer: Enable User Control over Related Recommendations via Collaborative Filtering and Clustering. In *Fourteenth ACM Conference on Recommender Systems (RecSys '20), September 22–26, 2020, Virtual Event, Brazil.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3383313.3412221

1 INTRODUCTION

Related item recommendations are a list of items similar to the target item typically shown to the user on the item page. These recommendations have a long history in recommender systems and have been adopted by many popular e-commerce websites, such as Amazon ("Customers who viewed this item also viewed"), LinkedIn ("Similar Jobs") and Quora ("Related Questions"). Related recommendations are the main source of recommendations for many users [12].

Traditionally, related recommendations are static without any support of user control, while it has been shown that users prefer to have control over recommendations [5]. One of the ways to provide users with more control is through critiquing [3]. Critiquing recommender systems use item attributes to provide control [11]. For example, in a recommender system that suggests laptops, the user can specify if they want to see cheaper models or laptops with bigger screens. However, these systems require rich metadata regarding each item, for example, laptop prices, screen sizes and other characteristics, which is labor-intensive. One way of mitigating this problem is to use keywords. For example, MovieTuner utilizes keyword submissions, text based reviews and ratings users assigned to movies to automatically generate critiques in the movie domain [11].

Another way to provide users with control over recommendations is to let them navigate in the latent space retrieved from the matrix factorization (MF) algorithm [6]. For example, the MovieExplorer system allows users to explore a latent space retrieved from MF by examples in the movie domain [10]. In this system, users iteratively specify movies that they find interesting. In each iteration, the system navigates the user in the latent space based on user feedback by showing a set of movies similar to those picked in the previous iterations. Another example is the choice-based system [8], which asks the user to iteratively indicate their preference for items from the extremes of each latent factor, positions them in the latent space based on the received feedback and recommends items closest to the user position.

In this paper, we design ClusterExplorer[1], a novel approach to related recommendations, which allows users to use critiques for navigation in the latent space based on user-item interactions. Since latent factors are not interpretable [8], to generate critiques, we apply MF to user-item interactions, retrieve latent space and cluster items in this space. Each cluster corresponds to a critique and the similarity between the item representation and the cluster centroid corresponds to the intensity, with which the critique applies to the item. If the metadata is missing, the clusters are described by an

[1]The current version of ClusterExplorer is available at `recommendabook.me/clusterexplorer`

expert, which is not as labor-intensive as annotating each item. If the metadata is poor, for example, tags only (our case), cluster descriptions can also be generated automatically. In this paper, we implement both versions of our approach: automatically generated descriptions and descriptions edited by an expert. We implement an application in the book domain and evaluate it in a within subject experiment with 42 participants. Our research questions are as follows: (RQ1) Do users understand how their actions affect recommended items in ClusterExplorer? (RQ2) Does ClusterExplorer improve user experience of related item recommendations?

Similar to MovieTuner in prior work [11], our approach allows users to indicate critiques that they want to see with different intensity in the list of related recommendations. In MovieTuner, these critiques are based on item tags enriched with item metadata and user-item interactions, while in ClusterExplorer, critiques are based on clusters retrieved from latent space generated based on only user-item interactions. Our approach is also similar to MovieExplorer [10] and the choice-based system [8], as it allows users to navigate in the latent space, but it is different from these systems in the navigation process. ClusterExplorer allows users to control related recommendations with critiques.

We found that most users understand how their actions affect recommendations in ClusterExplorer. We also found that our approach has higher user satisfaction of browsing books and is more helpful in finding interesting books to read compared to traditional related recommendations.

The contributions of this paper are as follows: (1) we present a novel approach to related recommendations and a novel user interface, (2) we evaluate our approach with real users in terms of different metrics, such as utility, serendipity and user satisfaction, and (3) we evaluate two versions of our approach and demonstrate that it improves traditional related item recommendations.

2 APPLICATION DESIGN

The application of our approach is implemented in the book domain, where the user chooses a book and the application recommends books related to the chosen one. The user can make this list also related to a particular critique, which is called *topic* in the user interface. The process is described in Figure 1.

Let us assume that the dataset contains 10 books represented as points in a two-dimensional space. Books form three clusters: drama, fantasy and sci-fi (Figure 1(b)). Each cluster corresponds to a topic in the user interface (on the left in Figure 1(a)), which is described with keywords. First, the user chooses any book they think about at the moment to start exploration. In our example, it is Harry Potter (Figure 1(a)). The application displays a number of books (on the right) and topics (on the left) related to Harry Potter. By choosing the book, the user locates their exploration point to the position of Harry Potter in the two-dimensional space (Figure 1(b)). If the user decides to continue exploring, they can either (1) choose a different book by typing its title to the search field or by selecting a book from the list of relevant books ("Show related"), or (2) adjust the list of related books with topics ("More like this"). In our example, the user decides to add more drama by hitting the "More like this" button. Their exploration point then moves closer to the drama cluster (Figure 1(d)), which updates their nearest books (Figure 1(c)).

3 DATASET

Our dataset was provided to us by the Vantaa Ciry Library. The original dataset contained 1,550,695 loans performed between January 8, 2016 and December 10, 2017 by 57,118 users to 790,653 items, where an item is a product offered by the library, such as a book, CD or DVD. To preprocess the dataset, we first removed the items that were not books. The remaining data consisted of 546,378 books, 54,917 users and 1,195,165 loans. Second, we removed duplicates and books that were loaned less than 20 times. The resulting dataset consisted of 12,885 books, 47,087 users and 796,529 loans.

4 GENERATING LATENT SPACE

We model books as points in the latent space and cluster them. ClusterExplorer shows n-nearest neighbors of the user exploration point. The exploration point moves closer to a particular cluster centroid, as the user hits "More like this" in the user interface (Figure 1).

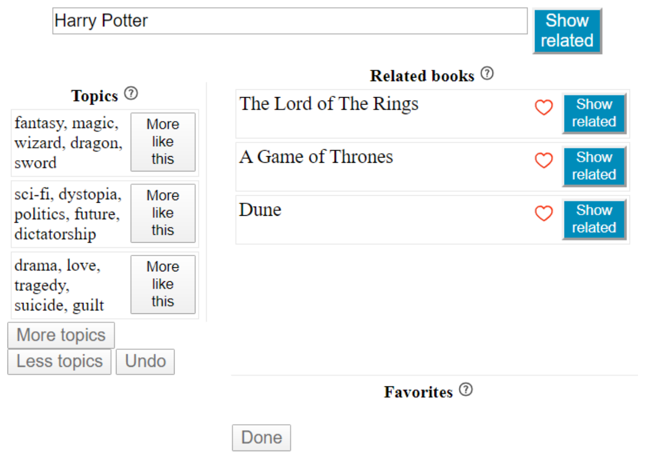
To generate latent factors, we use matrix factorization (MF), the common algorithm used in recommender systems [6]. MF decomposes user item matrix into two lower dimensionality matrices, where one matrix represents users and another – items. For our experiments, we only consider the item matrix, which corresponds to the latent space. We pick the following MF implementation: [6], as it is designed for implicit data and commonly used in the industry [1].

To make sure, that generated latent factors are meaningful: (1) we split the dataset into test and training datasets. The test dataset contains 30% last user loans, while the training dataset contains the rest of the loans, (2) we perform 3-fold cross validation based on the training dataset to tune MF (find the most efficient hyper parameters), (3) we train MF on the training dataset and compare its performance with that of baseline algorithms on the test dataset, (4) we train MF on the whole dataset and retrieve the latent factors (item matrix) for clustering.

To tune MF, we used hill climbing (HC) [9], which is a local search algorithm. It starts with an arbitrary solution of the problem, incrementally changes the solution, compares the results and either accepts the new solution and starts the next iteration or keeps looking for a better solution. Based on our tuning phase, HC detected the following most efficient parameters for MF: number of latent factors: 650, regularization term: 6, number of iterations: 3 (with the increase of iteration number, the performance drops).

To assess the quality of latent factors generated by MF, we compared its performance with that of the following baseline algorithms: (a) Random – orders books randomly, (b) POP – orders books according to the number of loans these books received in the dataset and IBCF – item-based collaborative filtering [2].

To measure performance of the algorithms, we picked normalized discounted cumulative gain (NDCG) [7], as this measure is commonly used for ranking systems and it takes into account the order of recommended items. The values of NDCG@5 were as follows: Random: 0.0007, POP: 0.0168, IBCF: 0.0692 and MF: 0.0938. For lists of other lengths the trend was the same. The low values of



Harry Potter | Show related

Topics ⓘ

fantasy, magic, wizard, dragon, sword	More like this
sci-fi, dystopia, politics, future, dictatorship	More like this
drama, love, tragedy, suicide, guilt	More like this

More topics
Less topics | Undo

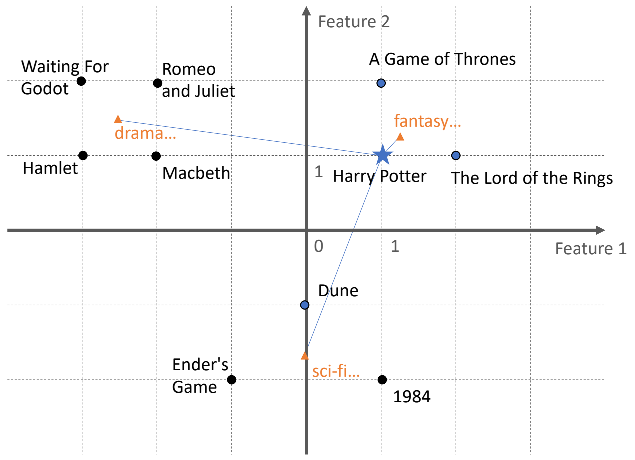
Related books ⓘ

The Lord of The Rings	♡	Show related
A Game of Thrones	♡	Show related
Dune	♡	Show related

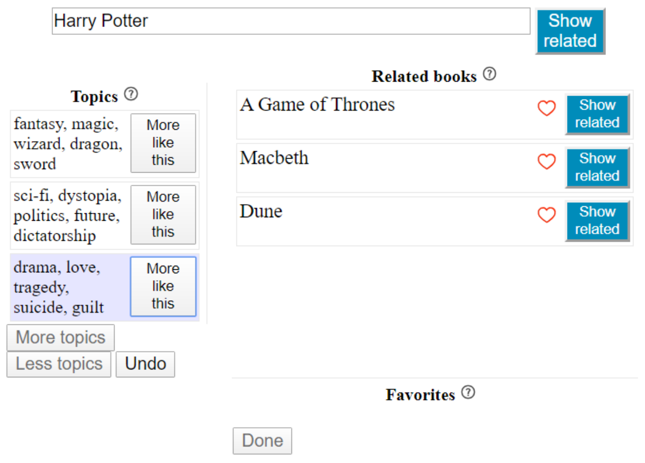
Favorites ⓘ

Done

(a) The user selects Harry Potter



(b) The exploration point is located at the Harry Potter book



Harry Potter | Show related

Topics ⓘ

fantasy, magic, wizard, dragon, sword	More like this
sci-fi, dystopia, politics, future, dictatorship	More like this
drama, love, tragedy, suicide, guilt	More like this

More topics
Less topics | Undo

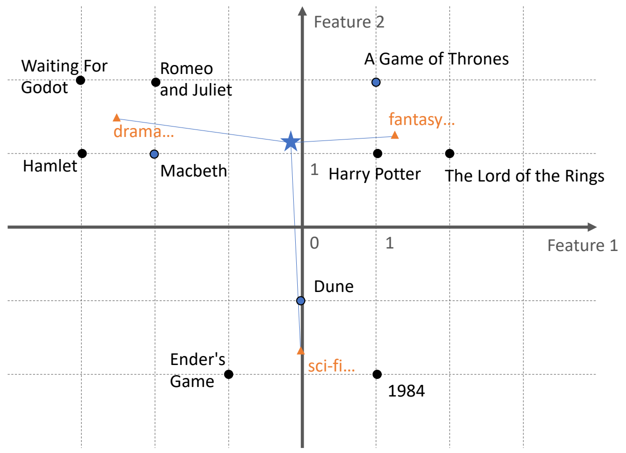
Related books ⓘ

A Game of Thrones	♡	Show related
Macbeth	♡	Show related
Dune	♡	Show related

Favorites ⓘ

Done

(c) The user hits "More like this" next to "drama, love, tragedy, suicide, guilt"



(d) The exploration point moves closer to the drama cluster

Figure 1: Application design

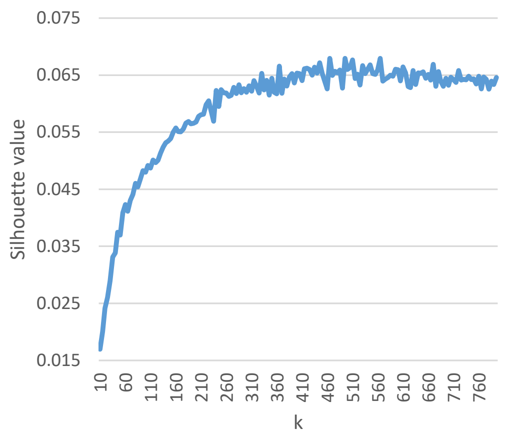
NDCG for all the algorithms are related to the low density of the dataset (0.13%) and the evaluation protocol (the algorithms ranked books from the whole catalog). Based on the obtained results, MF outperformed other algorithms.

5 CLUSTERING

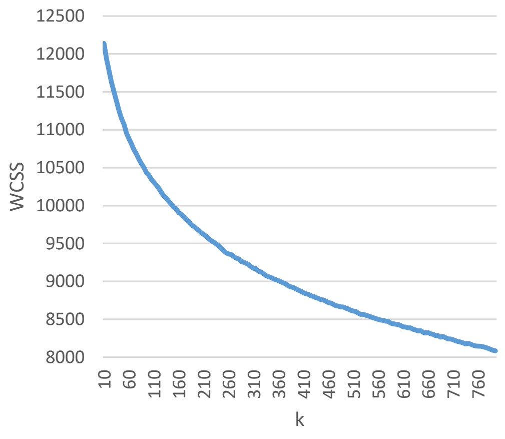
The goal of clustering in our case was not to detect all the books belonging to a particular topic, but to detect points (centroids) surrounded by books of the same topic, as we then use these points to navigate in the latent space. To cluster books, we used k-means, which requires the number of clusters (k) to be set. We gradually increased k and measured WCSS (within-cluster sums of squares) and Silhouette index (Figure 2). Based on the results, we set the number of clusters to 400. Although each cluster had different books, some clusters had very similar themes. For example, two clusters that include different books about Donald Duck.

To remove similar clusters, we clustered centroids of the 400 obtained clusters with DBSCAN and extracted keywords of 5 books nearest to each centroid. For centroids that fall into the same cluster, we calculated Jaccard similarity in term of keywords and removed centroids that were very similar to each other in terms of this similarity measure. The similarity threshold has been set manually. This resulted in 378 clusters for the automatically generated application. To generate description for each cluster, we needed to select the most representative keywords of a cluster centroid, while taking into account popularity of each keyword. We therefore selected 5 books nearest to the cluster centroid, extracted keywords of each of these books, merged them into a single document and ordered them based on their tf-idf scores. Top 5 keywords of each cluster described the corresponding critique in the user interface. We picked only 5 most representative keywords and books, as they fit well in the user interface (Figure 1) and represent the topic of books that surround a cluster centroid.

For the expert edited application, we asked a library expert to remove clusters, where the 5 books nearest to the cluster centroid were a mix of different themes, for example travel guides mixed



(a) Silhouette value



(b) WCSS (within-cluster sums of squares)

Figure 2: Cluster metrics

with children's literature. We also asked the expert to remove similar clusters that have the same theme and edit the top 5 keywords of each cluster. This resulted in 320 clusters. Overall, the two applications had an overlap of 308 clusters with 2.7 common keywords per cluster on average.

6 USER EXPERIMENT

In our user experiment, we evaluated the following three applications:

- Expert edited (EE) application. The interface of the application is depicted in Figure 1. The 320 selected clusters correspond to critiques (topics in the user interface), while the top 5 keywords correspond to descriptions of these critiques. The user can see the 5 books nearest to the cluster centroid by hovering over the crtique. The user can also see the book keywords by hovering over a book. The critiques and keywords for this application are generated by the algorithm and then edited by the library expert.
- Automatically generated (AG) application. The application has the same user interface as EE, but different input data. Critiques (based on 378 clusters) and keywords of this application are generated by the algorithm, but not edited by the library expert.
- Baseline (B) application. The interface of this application is missing critiques (on the left in Figure 1) (traditional related item recommendations). The book vectors are the same as in EE and AG.

In the related books section of the application (Figure 1), each application displays 10 books most similar to the exploration point, where the similarity is based on cosine similarity between the book vector and the exploration point. We picked cosine similarity, as it was the most descriptive measure in our pilot experiment. The exploration point does not rotate, but moves in Euclidean space for convenience.

As the target audience for our approach are Finnish library users, we recruited participants in the main hall of the Turku City Library. For participation in the experiment, we gave each participant a voucher, which allowed them to get a cup of coffee and a bun in the library restaurant. After a participant has agreed to take part in our experiment, we provided them with a laptop to go through the experiment script.

Figure 3 demonstrates the flow of the experiment. The welcome page (1) explained what the experiment is about, who is conducting that, for what reason and what data is being collected. The welcome survey inquired basic information, such as how often the user reads books, age and gender. A tutorial for each application (3, 6, 9) taught users to use the application by guiding them through a few simple tasks, such as find a certain book and adjust related books with critiques. The tutorial has been included in the experiment to mitigate the learning effect of using different applications. An application (4, 7, 10) was presented along with the task to complete, which had the following description: "Your task is to find books that are interesting to you and that you might want to read later. Add these books to your list of favourite books by hitting [button icon]. When you are done, hit Done". The survey regarding the user experience of using the application included statements listed in figure 4. The final page (12) indicated that the experiment is over. Applications were shown to users in a counter-balanced order. The script and application interface have been translated into Finnish.

Overall, we surveyed 42 participants (7 participants per experimental condition) with the following gender distribution: 18 males, 21 females and 3 others, and the following age distribution: 18-24: 22 participants, 25-34: 17 participants and 35-44: 3 participants. Prior to the experiment, we also asked participants to indicate how often they read books, look for them online and whether they are looking for a book to read at the moment. The majority indicated that they read books more often than once a month (30 participants), look for books online at least once in half a year (28) and are looking for a book to read at the moment (29).

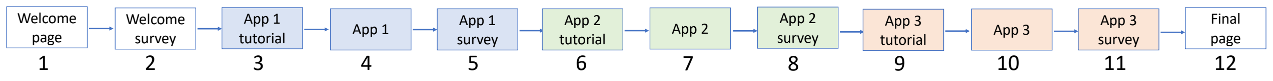


Figure 3: Flow of the experiment (applications were shown in a counter-balanced order.)

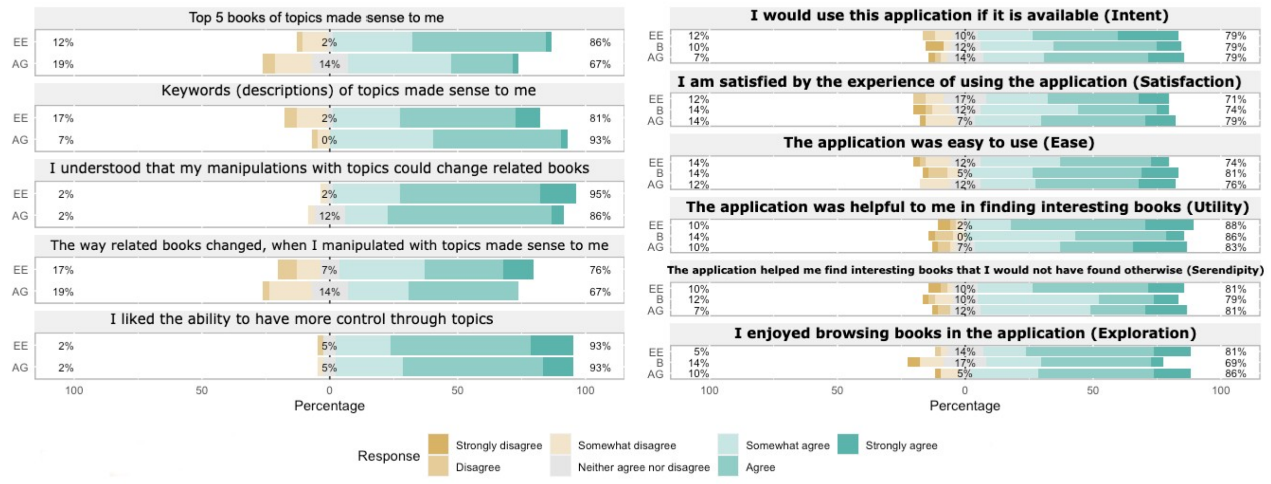


Figure 4: Ratings that users gave to statements after using an application. Statements that apply to each of the three applications are highlighted with bold

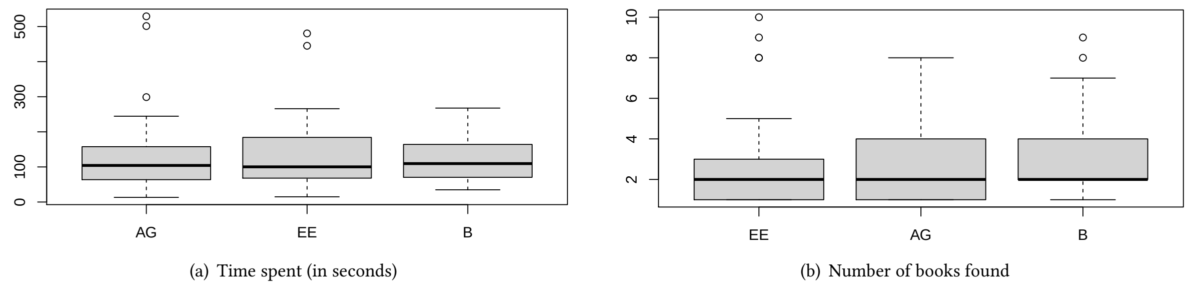


Figure 5: Behavioral activities

7 RESULTS

Survey data summarized in Figure 4 suggest that the majority of participants understand how their actions affect recommended books (RQ1): the number of participants who agree that they understand that critiques affect related books is 36 for AG, and 40 for EE, while for the statement that users also understand how their actions affect related books, these numbers are 28 and 32, respectively. The vast majority of users also indicated that they liked the ability to have more control over recommendations, the top 5 books were representative and keywords described critiques well.

Based on the survey data (table 1), ClusterExplorer makes related recommendations more helpful at finding interesting books (Utility) and more enjoyable at browsing books (Exploration) than traditional related recommendations (RQ2). We ran statistical tests only for survey data for statements related to each of the three applications to control for false discovery. In terms of behavioral activities, the difference is very small (Figure 5). For each application, medians for the number of books users found are the same (EE: 2; AG: 2; B: 2) (Figure 5(b)) and for the time users spent looking for books are very similar (EE: 100; AG: 104; B: 109) (Figure 5(a)).

Table 1: Fixed effects (coefficients) of cumulative link mixed-effect regression models. Each cell corresponds to a coefficient of an ordinal regression model with a single independent variable run on a dataset consisting of two applications. The first application is assigned the value of 1, while the other – 0. Dependent variables are ratings that users gave to statements. Significance codes: '' 0.01 '*' 0.05**

Applications	Ease	Utility	Serendipity	Exploration	Intent	Satisfaction
EE vs. B	-0.54	**1.44****	0.75	**0.91***	0.59	0.51
AG vs. B	0.01	0.47	0.38	**1***	0.39	0.82

Based on survey and action data, the two versions of our approach do not differ significantly, which users also mentioned in comments. However, collected data indicate that most users provided positive feedback on each application.

8 CONCLUSION AND DISCUSSION

In this paper, we designed and implemented a novel approach to related item recommendations, ClusterExplorer, which is based on latent factors retrieved from MF in the book domain. We evaluated two versions of this approach: based on automatically generated and expert edited topics. We also compared our approach to the traditional related item recommendations. Based on the obtained results, most users understand how their actions of navigating in the latent space affect related book recommendations (RQ1). We also found that ClusterExplorer improves utility in terms of finding interesting books to read and user satisfaction of browsing books (RQ2). The difference between automatically generated and expert edited versions of ClusterExplorer is not significant based on collected data. Generally, both ClusterExplorer and traditional recommendations received positive feedback from users.

Users provided similar feedback on each of the applications in terms of the survey data and their activities. One of the reasons for that might be that our approach was affected by the low density (0.13%) of the dataset. For example, the density of 1M MovieLens dataset is 4% [4]. We believe that further experimentation with denser datasets, user interfaces, latent spaces and similarity measures will reveal more insights on our approach. However, the trend seems to be in favor of ClusterExplorer, as the statistically significant results indicate its advantages over the traditional related recommendations and most users indicated that they enjoyed having more control over related item recommendations.

ACKNOWLEDGMENTS

The research performed in the LibDat project was supported by the Academy of Finland, grant #309495.

REFERENCES

[1] [n.d.]. Machine learning @ Spotify - Madison Big Data Meetup. https://www.slideshare.net/AndySloane/machine-learning-spotify-madison-big-data-meetup.

[2] Michael D. Ekstrand, John T. Riedl, and Joseph A. Konstan. 2011. Collaborative Filtering Recommender Systems. *Found. Trends Hum.-Comput. Interact.* 4, 2 (Feb. 2011), 81–173. https://doi.org/10.1561/1100000009

[3] Alexander Felfernig, Gerhard Friedrich, Dietmar Jannach, and Markus Zanker. 2015. Constraint-based recommender systems. In *Recommender systems handbook*. Springer, 161–190.

[4] F. Maxwell Harper and Joseph A. Konstan. 2015. The MovieLens Datasets: History and Context. *ACM Transactions on Interactive Intelligent Systems* 5, 4 (2015), 19:1–19:19.

[5] F Maxwell Harper, Funing Xu, Harmanpreet Kaur, Kyle Condiff, Shuo Chang, and Loren Terveen. 2015. Putting users in control of their recommendations. In *Proceedings of the 9th ACM Conference on Recommender Systems*. ACM, 3–10.

[6] Yifan Hu, Yehuda Koren, and Chris Volinsky. 2008. Collaborative Filtering for Implicit Feedback Datasets.. In *ICDM*, Vol. 8. Citeseer, 263–272.

[7] Kalervo Järvelin and Jaana Kekäläinen. 2002. Cumulated Gain-based Evaluation of IR Techniques. *ACM Trans. Inf. Syst.* 20, 4 (Oct. 2002), 422–446. https://doi.org/10.1145/582415.582418

[8] Benedikt Loepp, Tim Hussein, and Jüergen Ziegler. 2014. Choice-based preference elicitation for collaborative filtering recommender systems. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 3085–3094.

[9] Steven S Skiena. 1998. *The algorithm design manual: Text*. Vol. 1. Springer Science & Business Media.

[10] Taavi T Taijala, Martijn C Willemsen, and Joseph A Konstan. 2018. Movieexplorer: building an interactive exploration tool from ratings and latent taste spaces. In *Proceedings of the 33rd Annual ACM Symposium on Applied Computing*. ACM, 1383–1392.

[11] Jesse Vig, Shilad Sen, and John Riedl. 2012. The tag genome: Encoding community knowledge to support novel interaction. *ACM Transactions on Interactive Intelligent Systems (TiiS)* 2, 3 (2012), 13.

[12] Yuan Yao and F Maxwell Harper. 2018. Judging similarity: a user-centric study of related item recommendations. In *Proceedings of the 12th ACM Conference on Recommender Systems*. ACM, 288–296.