RecSys'15 Joint Workshop on Interfaces and Human Decision Making for Recommender Systems (IntRS'15)

Proceedings of the

Joint Workshop on Interfaces and Human Decision Making for Recommender Systems

September 19, 2015

In conjunction with the
9th ACM Conference on Recommender Systems
Vienna, Austria

Edited by

John O'Donovan, Alexander Felfernig, Nava Tintarev,
Peter Brusilovsky, Giovanni Semeraro, Pasquale Lops

Preface

As an interactive intelligent system, recommender systems are developed to give recommendations that match users' preferences. Since the emergence of recommender systems, a large majority of research focuses on objective accuracy criteria and less attention has been paid to how users interact with the system and the efficacy of interface designs from users' perspectives. The field has reached a point where it is ready to look beyond algorithms, into users' interactions, decision making processes, and overall experience. This workshop will focus on the aspect of integrating different theories of human decision making into the construction of recommender systems. It will focus particularly on the impact of interfaces on decision support and overall satisfaction, and on ways to compare and evaluate novel techniques and applications in this area.

The aim of the workshop is to bring together researchers and practitioners around the topics of designing and evaluating novel intelligent interfaces for recommender systems in order to: (1) share research and techniques, including new design technologies and evaluation methodologies (2) identify next key challenges in the area, and (3) identify emerging topics.

This workshop aims at establishing an interdisciplinary community with a focus on the interface design issues for recommender systems and promoting the collaboration opportunities between researchers and practitioners.

The workshop consists of a mix of ten presentations of papers in which results of ongoing research as reported in these proceedings are presented and one invited talk by Anthony Jameson presenting "Recommender Systems Seen Through the Lens of Choice Architecture". The workshop is closed by a final discussion session.

John O'Donovan, Alexander Felfernig, Nava Tintarev, Peter Brusilovsky, Giovanni Semeraro and Pasquale Lops

August 2015

Organizing Committee

Workshop Co-Chairs

Program Committee

Table of Contents

Recommender Systems Seen Through the Lens of Choice Architecture

Anthony Jameson
German Research Center for Artificial Intelligence (DFKI)
jameson@dfki.de

Abstract

"How do people make choices?" "How can we help them make better choices?" It's helpful to have compact, coherent answers to these questions if we want to build recommender systems that support choice processes. This talk begins with a brief summary of the ASPECT and ARCADE models (introduced in "Choice Architecture for Human-Computer Interaction"), which answer these questions. It then uses this framework to shed new light on a sample of subtle questions such as: "How can explanations of recommendations help people make better choices?" and "How can recommender systems help people choose via trial and error?" The talk is a concrete and selective presentation of key ideas from the chapter "Human Decision Making and Recommender Systems" in the second edition of the "Recommender Systems Handbook".

Parsimonious and Adaptive Contextual Information Acquisition in Recommender Systems

Matthias Braunhofer
Faculty of Computer Science
Free University of Bozen-Bolzano
Bozen-Bolzano, Italy
mbraunhofer@unibz.it

Ignacio Fernández-Tobías
Escuela Politécnica Superior
Universidad Autónoma de Madrid
Madrid, Spain
ignacio.fernandezt@uam.es

Francesco Ricci
Faculty of Computer Science
Free University of Bozen-Bolzano
Bozen-Bolzano, Italy
fricci@unibz.it

ABSTRACT

Context-Aware Recommender System (CARS) models are trained on datasets of context-dependent user preferences (ratings and context information). Since the number of context-dependent preferences increases exponentially with the number of contextual factors, and certain contextual information is still hard to acquire automatically (e.g., the user's mood or for whom the user is buying the searched item) it is fundamental to identify and acquire those factors that truly influence the user preferences and the ratings. In particular, this ensures that (i) the user effort in specifying contextual information is kept to a minimum, and (ii) the system's performance is not negatively impacted by irrelevant contextual information. In this paper, we propose a novel method which, unlike existing ones, directly estimates the impact of context on rating predictions and adaptively identifies the contextual factors that are deemed to be useful to be elicited from the users. Our experimental evaluation shows that it compares favourably to various state-of-the-art context selection methods.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*information filtering*

Keywords

Context-Aware Recommender Systems; Contextual Information; Feature Selection

1. INTRODUCTION

Context-Aware Recommender Systems (CARSs) generate more relevant recommendations than traditional Recommender Systems (RSs) by adapting them to the specific contextual situation of the user (e.g., time, weather, location) [1]. The development of an effective CARS faces many challenges [2]. First, it is necessary to identify the contextual factors that could potentially influence individual's preferences (ratings) and the decision-making process, and hence are worth to be collected, either automatically (e.g., the time, or the location), or by querying the user. The second challenge is to develop a predictive model that is capable of predicting the users' ratings for items under various contextual situations. Finally, the design of a proper human-computer interaction layer on top of the predictive model is the third and last but not least challenge for building a CARS.

In this paper we are focusing on the first challenge. In this respect, previous approaches have mainly applied feature selection techniques to identify which contextual factors should be used in the rating prediction phase. The downside of this approach is that it may force users to add to ratings contextual information that later on, when the prediction model is built, may be found not to be useful for improving the system performance. Because of that, here we propose a new method for identifying which contextual factors should be acquired from the user upon rating an item, so that the user will not enter the value of many contextual factors (parsimonious), and the accuracy of the subsequent recommendations is improved the most.

As a concrete motivation, consider the places of interest (POIs) CARS that is illustrated in Figure 1 and Figure 2 [5]. That system is called STS (South Tyrol Suggests) and it uses 14 contextual factors (e.g., weather, mood, distance, time available). Users may specify any of them when entering a rating for a POI (and also when the user requests context-aware recommendations). These are, however, not all equally important for different user-item pairs, in the sense that they contribute differently to the improvement of the system's rating prediction and recommendation accuracy. In fact, we must avoid any possible waste of time and effort of the user while entering this information and also keep away from the potential degradation of the system performance that could be caused by the usage of irrelevant information. For example, the user's mood may be extremely important to predict the ratings only of certain users, and weather may be an essential factor for one class of items, while negligible for others.

Unlike current state-of-the-art strategies, which measure the relevance of contextual factors on a global basis, our strategy dynamically and adaptively selects the contextual factors to be elicited from the user when she enters a rat-

.

ing for an item. This is achieved by using the CARS rating prediction model itself, and asking the user to specify, when she is rating an item, those contextual factors that if considered in the model would produce a rating prediction for that item that is most different from the prediction computed by a context-free model. We consider this as a heuristics: if this contextual information has an impact on rating prediction it should be acquired and used in the model.

Several CARS algorithms can be used to implement the above mentioned solution; here we employ a new variant of Context-Aware Matrix Factorization (CAMF) [3] that leverages latent correlations and patterns between users, items as well as contextual conditions, thus making it well-suited for selective context acquisition, but also for prediction and recommendation as well. We have compared our proposed method with several state-of-the-art context selection strategies in an offline experiment on two contextually-tagged rating datasets. The results show that the proposed parsimonious and personalized acquisition of relevant contextual factors is efficient and effective, and allows to elicit ratings augmented with contextual factor values that best improve the recommendation performance in terms of accuracy, precision and recall.

We note that parsimoniously acquiring from the user relevant contextual information can be considered as an Active Learning problem [8]. But, while in previous work [6, 4] we focused on the active identification of the *items* to present to the user to rate, in this article we focus on the subsequent decision of identifying which *contextual factors* the user should enter, i.e., under which conditions the user experienced the item.

The rest of the paper is structured as follows. In Section 2, we review the related work. Section 3 introduces our main application scenario. Section 4 presents in detail the proposed context acquisition method. Then, we describe the experimental evaluation in Section 5, and detail the obtained results in Section 6. Finally, conclusions are drawn and future work directions are described in Section 7.

2. RELATED WORK

Finding the most relevant features for building a prediction model has been extensively studied in machine learning. Feature selection is aimed at improving the performance of learning algorithms and gaining insight into the unknown generative process of the data [9]. There are three main approaches to feature selection: wrappers, filters and embedded methods. While wrapper methods optimize the selection within the prediction model, filter methods employ statistical characteristics of the training data to select features independently of any prediction model, and thus are substantially faster to compute. Popular examples of filter methods used in machine learning include mutual information, t statistic in Student test, χ^2 test for independence, F statistic in ANOVA and minimum Redundancy Maximum Relevance (mRMR) [14], which uses the mutual information of a feature and a class as well as the mutual information of features to infer features' relevance and redundancy, respectively. Differently from the two previous methods, embedded methods use internal parameters of some prediction model to perform feature selection (e.g., the weight vector in support vector machines).

Focussing now on CARSs, previous research has explored methods: a) for identifying a priori the factors that should be considered by the system, or b) for selecting, a posteriori, after the ratings and context data was acquired, those factors that are most influential for computing rating predictions. The first task was tackled by exploiting domain knowledge of the RS's designer or market expert [2], whereas the second one was addressed by using feature selection algorithms.

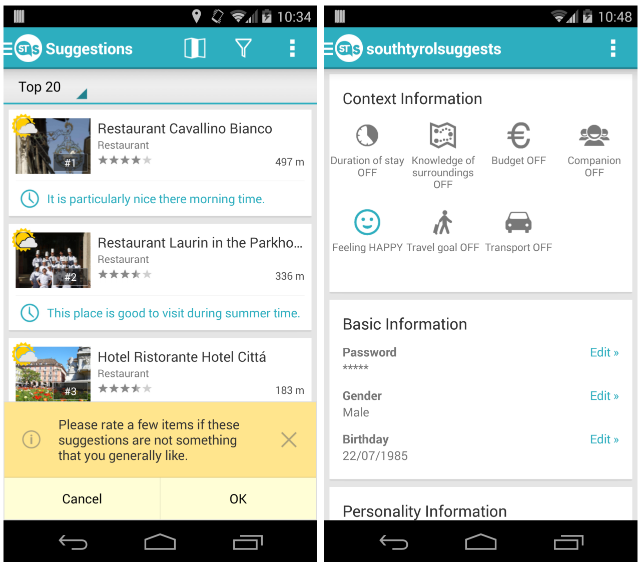
In order to tackle the second task, Odić et al. [13] provide several statistical measures for relevant-context detection (i.e., unalikeability, entropy, variance, χ^2 test and Freeman-Halton test), and show that there exists a significant difference in the prediction of ratings when using relevant and irrelevant context. Another example can be found in [16], where a Las Vegas Filter (LVF) algorithm [12] is employed: it repeatedly generates random subsets of contextual factors, evaluates them based on an inconsistency criterion and finally returns the subset with the best evaluation measure. Finally, Zheng et al. [17] presented a set of approaches based on multi-label classification for the task of recommending the most suitable contexts in which a user should consume a specific item.

Rather than post filtering (after the rating data was acquired) the contextual factors in the rating prediction phase, we are interested in detecting which contextual factors should be acquired upfront from the user in the first place. Hence, when a specific user rates a particular item, our goal is to parsimoniously request and possibly elicit only the contextual factors that improve the most the system performance. These factors can differ for each user-item pair. Moreover, instead of relying on statistical measures, which has been the major trend so far, our work uses a CARS rating prediction model itself to estimate the usefulness of contextual factors. Our approach is similar to some Active Learning [8] solutions of the cold-start problem that also use the rating prediction model to identify which items are better to propose to the users to rate. An example of such an Active Learning method can be found in [10]; it asks users to rate the items whose ratings, if known, contribute most to reduce the system prediction error on a set of held-out test ratings. Another similar approach is the influence-based method presented in [15], which selects those items whose ratings are estimated to have the highest influence on the rating predictions of other items.

3. APPLICATION SCENARIO

Our application scenario is a mobile CARS called STS (South Tyrol Suggests) [5] that is available on Google Play Store and recommends POIs to visit in the South Tyrol region of Italy. STS can generate POI recommendations (Figure 1, left) adapted to the user's and items' current contextual situation by exploiting 14 contextual factors whose conditions (values) are partially acquired automatically by the system (e.g., weather at the POI, season, daytime) and partially entered manually by the user through an appropriate screen (e.g., user's budget, companion, feeling), as shown in Figure 2 (right). More information about the used contextual factors and their possible values, which are called contextual conditions, can be obtained from Table 1. The user's preference model is learned using a set of in-context ratings that the system actively collects from the users and that describe the users' evaluations for the POIs together with the contextual situations in which the users visited the POIs (see Figure 2). However, in our application scenario, given the relatively large number of contextual factors we faced the problem of choosing the contextual factors to ask to the end user upon rating a POI. This is an important and practical problem: asking the value of all the contextual factors is not effective, as it would take too much time and effort for the user to specify them. Moreover, asking the wrong subset of contextual factors may result in the degradation of the prediction model performance and in poor recommendations.

In order to cope with this problem we propose here a novel method that is able to dynamically and adaptively identify the most important contextual factors to be elicited from a specific user upon rating a particular POI. This method serves the purpose of minimizing the amount of information that the users have to input manually, while at the same time allowing the system to still obtain all the relevant information needed to maintain a high level of rating prediction performance. Referring to Figure 2, by means of our proposed method, we can identify for instance the three most relevant contextual factors for "Restaurant Pizzeria Amadè" and then present the user with three screens that step-by-step elicit the contextual conditions for these factors. Otherwise, the user would be required to go through 14 screens, one for each available contextual factor.



Restaurant Cavallino Bianco
Restaurant
#1
497 m
It is particularly nice there morning time.

Restaurant Laurin in the Parkho...
Restaurant
#2
336 m
This place is good to visit during summer time.

Hotel Ristorante Hotel Cittá
Restaurant
#3
183 m

Please rate a few items if these suggestions are not something that you generally like.

Cancel OK

Context Information

Duration of stay OFF
Knowledge of surroundings OFF
Budget OFF
Companion OFF
Feeling HAPPY
Travel goal OFF
Transport OFF

Basic Information

Password
***** Edit »

Gender
Male Edit »

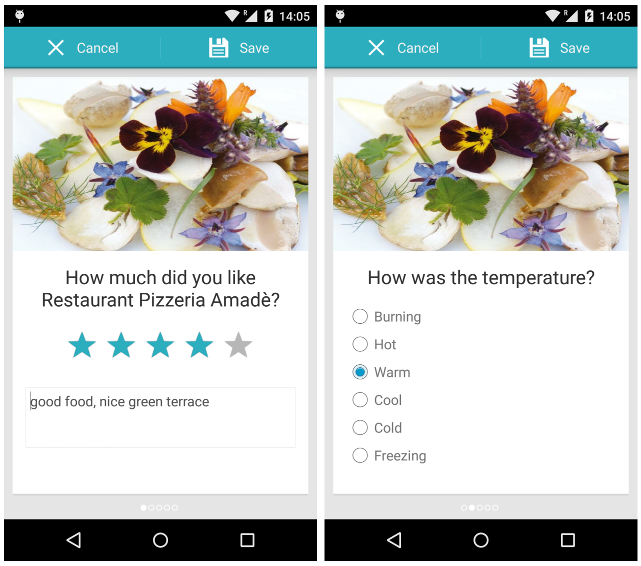
Birthday
22/07/1985 Edit »

Personality Information

Figure 1: Context-aware suggestions in STS

4. SELECTIVE CONTEXT ACQUISITION

Before presenting the proposed selective context acquisition method, we introduce the CARS predictive model that we have adopted in this study. It is a new variant of the context-aware predictive model CAMF [3] that treats contextual conditions similarly to either item or user attributes and uses a distinct latent factor vector corresponding to each user- and item-associated attribute. More specifically, a contextual condition is treated as a user attribute if it corresponds to a dynamic characteristic of a user, e.g., the mood, budget or companion of the user, whereas it is considered as an item attribute if it describes a dynamic characteristic of the item, e.g., the weather and temperature at the POI. The model is scalable and flexible, and is able to capture latent correlations and patterns between a potentially wide range of knowledge sources (e.g., users, items, contextual conditions, demographics, item categories), making it ideal to derive the usefulness of contextual factors in rating prediction. Given a user u with user attributes $A(u)$, an item i with item attributes $A(i)$ and a contextual situation consisting of the conjunction of individual contextual conditions $c_1, ..., c_k$ that can be decomposed into the subset of user-related contextual conditions $C(u)$ and the subset of item-related contextual conditions $C(i)$, it predicts a rating using the following rule:



How much did you like Restaurant Pizzeria Amadè?

good food, nice green terrace

How was the temperature?

Burning
Hot
[✓ Warm]
Cool
Cold
Freezing

Figure 2: Rating interface of STS

Table 1: Contextual factors used in STS

Contextual factors	Associated contextual conditions
Weather	Clear sky, sunny, cloudy, rainy, thunderstorm, snowing
Season	Spring, summer, autumn, winter
Budget	Budget traveler, high spender, none of them
Daytime	Morning, noon, afternoon, evening, night
Companion	Alone, with friends/colleagues, with family, with girlfriend/boyfriend, with children
Feeling	Happy, sad, excited, bored, relaxed, tired, hungry, in love, loved, free
Weekday	Working day, weekend
Travel goal	Visiting friends, business, religion, health care, social event, education, scenic/landscape, hedonistic/fun, activity/sport
Transport	No transportation means, a bicycle, a motorcycle, a car, public transport
Knowledge of the travel area	New to area, returning visitor, citizen of the area
Crowdedness	Crowded, some people, almost empty
Duration of stay	Some hours, one day, more than one day
Temperature	Burning, hot, warm, cool, cold, freezing
Distance	Far away (over 3 km), nearby (within 3 km)

$$\hat{r}_{uic_1,\ldots,c_k} = (q_i + \sum_{a \in A(i) \cup C(i)} x_a)^\top (p_u + \sum_{b \in A(u) \cup C(u)} y_b) + \bar{r}_i + b_u \quad (1)$$

where q_i is the latent factor vector associated to item i, p_u is the latent factor vector associated to user u, x_a is the latent factor vector associated to an attribute of the item i, that may either describe a conventional attribute (e.g., genre, item category) or a contextual attribute (e.g., weather, temperature), y_b is the latent factor vector associated to an (contextual or not) attribute of the user u. Finally, $\bar{r}_i$ is the average rating for item i, and b_u is the bias associated to the user u, which indicates the observed deviation of user u's ratings from the global average.

CARSs can generate recommendations only after having gathered ratings from the users that are augmented with information about the contextual conditions (values of the contextual factors) observed at the time the item was experienced and rated. It is, however, not always easy to identify which contextual information should be requested and acquired from the users upon rating an item, given the numerous conditions that might or might not be relevant to predict new ratings (in various contextual situations). This is where parsimonious and adaptive context acquisition comes in. Parsimonious and adaptive context acquisition aims at predicting, for a given user-item pair, the most useful contextual factors, i.e., those that when elicited together with the rating from the user improve more the quality of future recommendations, both for that user and for other users of the system.

As we mentioned in the related work section, there exist many algorithms that even though principally designed for context / feature selection (i.e., selection of the most useful contextual factors / features to be used for prediction) can be used also for the purpose of parsimonious context acquisition (i.e., selection of the contextual factors to be elicited from the user upon rating an item). In this paper, we propose a new strategy, which we call *Largest Deviation*. Differently from several state-of-the-art context / feature selection strategies, it personalizes the selection of the contextual factors to ask to the user when rating an item by computing a personalized relevance score for a contextual factor C_j and user-item pair (u, i). To achieve this, for each user u and item i pair (whose rating is acquired) we first measure the "impact" of each contextual condition $c_j \in C_j$, denoted as $\hat{w}_{uic_j}$, by calculating the absolute deviation between the rating prediction when the condition holds (i.e., $\hat{r}_{uic_j}$) and the predicted context-free rating (i.e., $\hat{r}_{ui}$):

$$\hat{w}_{uic_j} = f_{c_j} |\hat{r}_{uic_j} - \hat{r}_{ui}|, \quad (2)$$

where f_{c_j} denotes the normalized frequency of the contextual condition c_j, and is calculated as the fraction of ratings in the entire dataset that are tagged with contextual condition c_j (i.e., $\frac{|R_{c_j}|}{|R|}$). The normalized frequency adjusts the raw absolute deviation by taking into account that the contextual conditions with largest frequency are more reliable. For example, suppose that you want to estimate the impact of *Sunny* weather on the user-item pair (*Alice*, *Skiing*). Let us assume that the rating prediction for *Alice* of *Skiing* is 5 under *Sunny* weather (i.e., $\hat{r}_{Alice\,Skiing\,Sunny} = 5$), and that the corresponding context-free rating prediction is 3.5 (i.e., $\hat{r}_{Alice\,Skiing} = 3.5$). Furthermore, assume that 20% of the ratings in the rating dataset are tagged with *Sunny* weather. Then, the impact of *Sunny* weather on the user-item pair (*Alice*, *Skiing*), i.e., $\hat{w}_{Alice\,Skiing\,Sunny}$, is 0.3 $(0.2 \cdot |5 - 3.5|)$.

Finally, these individual scores for the contextual conditions are aggregated into a single relevance score for the contextual factor C_j by simply computing the arithmetic mean of the scores of the various conditions/values for that contextual factor. We conjectured that the contextual factors with largest estimated deviation are more useful to optimize the system performance. Note that this is quite similar to the influence-based Active Learning strategy proposed in [15], which estimates the influence of an item's rating on the rating predictions of other items, and selects the items with the largest influence for rating acquisition.

5. EXPERIMENTAL EVALUATION

5.1 Datasets

In order to evaluate the proposed selective context acquisition method, we have considered two contextually-tagged rating datasets with different characteristics. Table 2 provides some descriptive statistics of both datasets.

- The *CoMoDa* movie-rating dataset was collected by Odić et al. [13]. It consists of ratings acquired in contextual situations that are described by the conjunction of multiple conditions coming from 12 different factors, for instance, time, daytype, season and mood. In addition to the ratings data, this dataset also includes well-defined user attributes (i.e., age, gender, city, country) and movie attributes (i.e., director, country, language, year, budget, genres, actors).
- The *TripAdvisor* dataset is a dataset that we crawled from the TripAdvisor[1] website, which is one of the largest travel sites in the world. It contains ratings for POIs in the South Tyrol region of Italy that are tagged with contextual situations described by the conjunction of contextual conditions coming from three contextual factors, namely, type (e.g., couple, family or business trip), month (e.g., January, February) and year (e.g., 2015, 2014) of the trip. Additionally, also the TripAdvisor dataset has well-defined user (e.g., user location, member type) and POI attributes (e.g., item type, amenities, item locality).

We note that other rating datasets, which are commonly used in CARS research, are not suitable for our analysis since they contain ratings augmented only with the knowledge of a *subset* of all the contextual factors. For instance, in STS, the POIs RS that we mentioned in Section 3, when a user rates a POI she commonly specifies only the value of two or three of the fourteen contextual factors that the system manages (see Table 1). The lack of knowledge of all the contextual factors for each rating is a problem in our case, because, as we will describe in Section 5.2, we wanted to simulate a rating acquisition process where, for a given item, the system requests the user to rate it and to enter the values of the contextual factors identified by the proposed method. Therefore, every contextual factor must be available in the dataset in order to be acquired during the simulated interactions.

[1]http://www.tripadvisor.com/

Table 2: Datasets' characteristics

Dataset	CoMoDa	TripAdvisor
Domain	Movies	POIs
Rating scale	1-5	1-5
Ratings	2,098	4,147
Users	112	3,916
Items	1,189	569
Contextual factors	12	3
Contextual conditions	49	31
User attributes	4	2
Item features	7	12

5.2 Evaluation Procedure

In the evaluation we have simulated system/user interactions where the users rate items specifying only the values of contextual factors (contextual conditions) that have been identified by a context selection strategy. To achieve this, we adapted a procedure which was employed to evaluate Active Learning strategies for RSs [7]. This procedure first randomly partitions all the available ratings into three subsets in the ratio 25:50:25%, respectively: (i) *training set* that contains the ratings that are used to train the context acquisition strategies; (ii) *candidate set* containing the ratings that can be potentially transferred into the training set with the contextual conditions matched by the context acquisition strategies; and finally (iii) *testing set* which contains the part of the ratings that is withheld from the system in order to calculate various performance metrics, i.e., user-averaged MAE (U-MAE), Precision@10 and Recall@10. Then, for each user-item pair (u, i) in the candidate set, the N most relevant contextual factors according to a context usefulness strategy are computed, with N (in different experiments) varying from 1 to the total number of contextual factors in the rating dataset, and the corresponding rating r_{uic} in the candidate set is transferred to the training set as $r_{uic'}$ with $c' \subseteq c$ containing the associated contextual conditions for these contextual factors. For instance, if the top two contextual factors for the user-item pair $(Alice, Skiing)$ are $Season$ and $Weather$, and $Alice$'s rating is $r_{Alice\, Skiing\, Winter, Sunny, Warm, Morning} = 5$, then $r_{Alice\, Skiing\, Winter, Sunny} = 5$ is added to the training set. Since in the considered rating datasets all the contextual factors were specified for each rating, we could always acquire the contextual conditions for the top contextual factors. Finally, the evaluation metrics were measured on the testing set, after training the rating prediction model on the new extended training set.

The above process was repeated 20 times with different random seeds and the results were averaged over the splits to yield more robust estimates (i.e., repeated random subsampling validation [11]).

5.3 Baseline Methods for Evaluation

We have compared the performance of our proposed *Largest Deviation* method with the following three state-of-the-art context / feature selection strategies, in addition to *Random* which we used as a baseline (see Table 3 for a summary of all the tested methods):

- *Mutual Information*: the usage of mutual information for context selection was proposed in [2]. Given a user-item pair (u, i), it computes the relevance score for contextual factor C_j as the normalized mutual information between the ratings for items belonging to i's category and C_j; the higher the mutual information, the better the contextual factor can explain the user ratings for items of a particular category. We note that this strategy depends on the item category but is not personalized, i.e., the same contextual factors are requested to any user upon rating an item belonging to a particular category.
- *Freeman-Halton Test*: proposed as context selection strategy in [13], it calculates the relevance of a contextual factor C_j using the Freeman-Halton test. The Freeman-Halton test is the Fisher's exact test extended to contingency tables larger than 2×2, which is a common alternative to the χ^2 test in case the Cochran's rule about small expected frequencies is not satisfied. The null hypothesis of the test is that the contextual factor C_j and the ratings are independent. If the null hypothesis can be rejected, one can conclude that the contextual factor C_j and the ratings are dependent and thus that the contextual factor C_j is relevant. This test is performed on the full dataset and therefore the selected factors do not depend on the user or the item to be rated.
- *Minimum Redundancy Maximum Relevance (mRMR)*: mRMR [14] is a widely used feature selection algorithm, which, to the best of our knowledge, has not yet been used for the purpose of context selection. It ranks each contextual factor C_j according to its relevance to the rating variable and redundancy to other contextual factors, where both relevance and redundancy are measured based on mutual information. Analogous to the Freeman-Halton test, it is calculated on the full dataset and the selected factors are used for all user-item rating combinations.
- *Random*: the score for a contextual factor C_j is simply a random float in the interval [0, 1). Hence, the top N contextual factors for a user-item pair are simply randomly chosen. This is a baseline strategy used for comparison.

Table 3: Overview of tested strategies for selective context acquisition

Strategy	User Personalization	Item Dependence
Largest Deviation	✓	✓
Mutual Information	✗	✓
Freeman-Halton Test	✗	✗
mRMR	✗	✗
Random	✗	✗

6. EVALUATION RESULTS

Figure 3 and Figure 4 show the U-MAE, Precision@10 and Recall@10 results of the CARS algorithm obtained by applying the various context acquisition strategies on the CoMoDa and TripAdvisor dataset, respectively. In the figures, the x-axis represents the number of acquired contextual

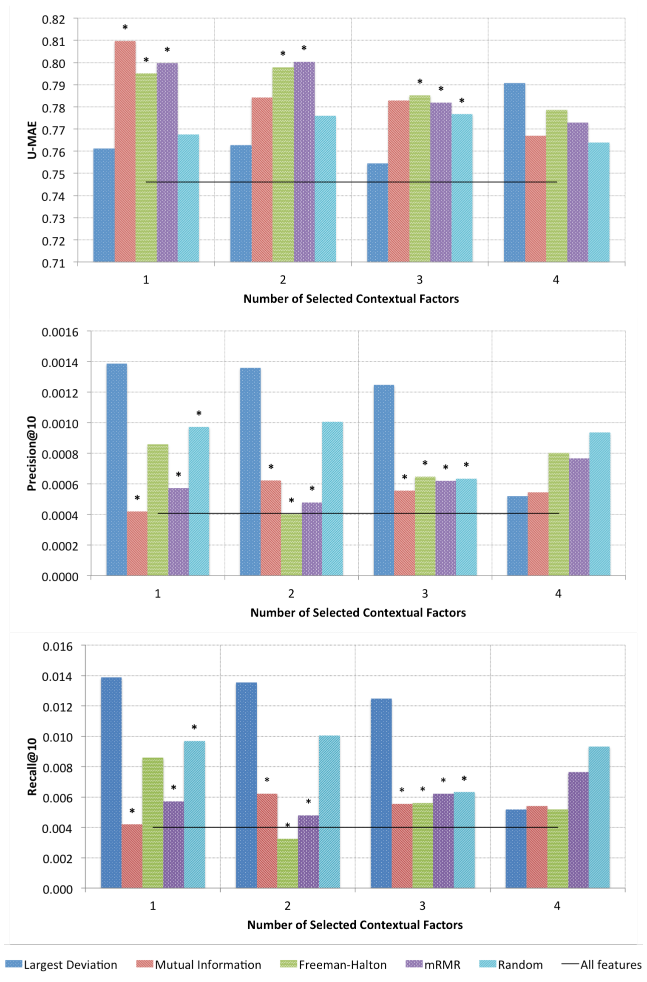


Figure 3: Accuracy, precision and recall results for the CoMoDa dataset

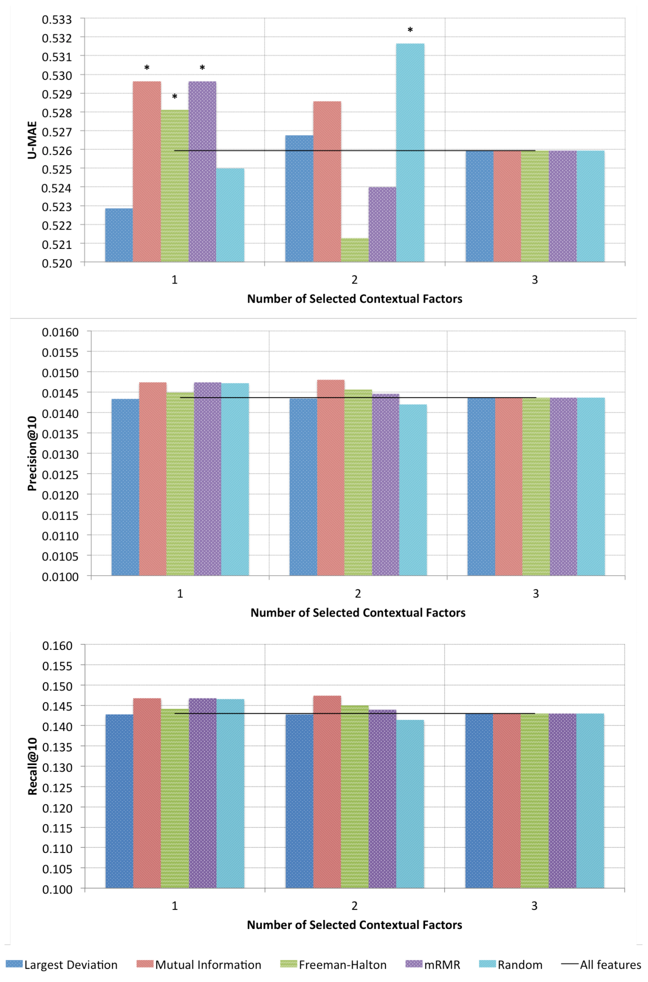


Figure 4: Accuracy, precision and recall results for the TripAdvisor dataset

factors, and statistically significant improvements (paired t-test, $p < 0.05$) of the proposed *Largest Deviation* strategy over the other considered strategies are indicated by asterisks on top of the bars. On the CoMoDa dataset, by using up to three contextual factors, *Largest Deviation* strategy can achieve a significantly better performance in terms of U-MAE, Precision@10 and Recall@10 when compared with the other strategies, i.e., *Mutual Information*, *Freeman-Halton Test* and *mRMR*. With four contextual factors selected, however, there is a notable increase in the U-MAE of *Largest Deviation*, which also causes Precision@10 and Recall@10 to drop. We note that in the graph the number of selected contextual factors goes only up to 4 (out of 12) in order to focus the presentation on the selection of a small subset of factors. In fact, the performance differences between the strategies vanish when more than 4 contextual factors are acquired. We also note that all these 12 contextual factors were supposed to be relevant in the movie recommendation domain [13]. Hence our results clearly indicate that a parsimonious context acquisition strategy is highly beneficial.

Experimental results also indicate that the *Random* strategy has a relatively good performance. Our explanation is that in this strategy, every contextual factor has the same chance of being selected. As a side effect, this allows to better explore the effect of individual contextual conditions on users and/or items. However, the *Random* strategy cannot be practically used since it can often request meaningless contextual factors to the user, e.g., the budget for a POI that can be visited for free. Hence, the random strategy is not directly applicable in a realistic scenario and can only be used in combination with other strategies. This is in line with the findings of Elahi et al. [7], who suggested to consider "partially randomized" strategies that add a small portion of randomly selected items to those identified by another baseline strategy.

Looking at the results for the TripAdvisor dataset, one can note that minor differences (especially in Precision@10 and Recall@10) between the considered context acquisition strategies are present. This is due to the fact that in this dataset in total only three contextual factors are available, thus providing only little potential for parsimonious and adaptive contextual factor selection. Nevertheless, it can be seen that *Largest Deviation* achieves even here a very good accuracy for the tested number of selected contextual factors (1 - 3).

7. CONCLUSIONS AND FUTURE WORK

In this paper, we have proposed a new method for parsimonious context acquisition, i.e., for identifying, for a given user-item pair the contextual factors that when acquired together with the rating from the user let the system to generate better predictions. This is an important and challenging problem for CARSs, since usually many contextual factors (e.g., location, weather, time of day, mood) may be available, but only a small subset may be useful and should be asked to the user to avoid an unnecessary waste of time and effort as well as to avoid any degradation of the recommendation model performance.

We have formulated the experimental hypothesis that the proposed parsimonious and personalized selective context acquisition strategy is able to elicit ratings with contextual information that improve more the recommendation performance in terms of accuracy, precision and recall, and also compares favourably with state-of-the-art (context selection) alternatives. In an offline experiment on two rating datasets we were able to confirm these hypotheses.

Selective context acquisition is still a new and under-researched topic, and there are some research questions that deserve future work. Firstly, what is the effect on system performance of employing an Active Learning method for adaptively selecting both the item to rate and the contextual information to add. In this paper we have addressed only partially the problem, by identifying the contextual factors that should be acquired, when a user is rating an item. Secondly, it is interesting to understand how the proposed selective context acquisition method can be extended to generate requests for contextual data that takes into account the possible correlation between contextual factors. Thirdly, it would be interesting to update the evaluation procedure so that it can be used also on datasets of contextually-tagged ratings for which only a subset of the contextual factors is known; as it occurs in the rating dataset collected by our STS app. Finally, we plan to integrate the developed context acquisition method into our STS app so that we can perform a live user study and assess the impact and the benefit of the proposed dynamic and personalized parsimonious acquisition of contextual factors.

8. REFERENCES

[1] G. Adomavicius, B. Mobasher, F. Ricci, and A. Tuzhilin. Context-aware recommender systems. *AI Magazine*, 32(3):67–80, 2011.

[2] L. Baltrunas, B. Ludwig, S. Peer, and F. Ricci. Context relevance assessment and exploitation in mobile recommender systems. *Personal and Ubiquitous Computing*, 16(5):507–526, 2012.

[3] L. Baltrunas, B. Ludwig, and F. Ricci. Matrix factorization techniques for context aware recommendation. In *Proceedings of the Fifth ACM Conference on Recommender Systems*, pages 301–304. ACM, 2011.

[4] M. Braunhofer, M. Elahi, M. Ge, and F. Ricci. Context dependent preference acquisition with personality-based active learning in mobile recommender systems. In *Learning and Collaboration Technologies. Technology-Rich Environments for Learning and Collaboration*, pages 105–116. Springer, 2014.

[5] M. Braunhofer, M. Elahi, and F. Ricci. Usability assessment of a context-aware and personality-based mobile recommender system. In *E-Commerce and Web Technologies*, pages 77–88. Springer, 2014.

[6] M. Elahi, M. Braunhofer, F. Ricci, and M. Tkalcic. Personality-based active learning for collaborative filtering recommender systems. In *AI* IA 2013: Advances in Artificial Intelligence*, pages 360–371. Springer, 2013.

[7] M. Elahi, F. Ricci, and N. Rubens. Active learning strategies for rating elicitation in collaborative filtering: a system-wide perspective. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 5(1):13, 2013.

[8] M. Elahi, F. Ricci, and N. Rubens. Active learning in collaborative filtering recommender systems. In *E-Commerce and Web Technologies*, pages 113–124. Springer, 2014.

[9] I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3:1157–1182, 2003.

[10] R. Karimi, A. Nanopoulos, and L. Schmidt-Thieme. A supervised active learning framework for recommender systems based on decision trees. *User Modeling and User-Adapted Interaction*, 25(1):39–64, 2015.

[11] R. Kohavi et al. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Ijcai*, volume 14, pages 1137–1145, 1995.

[12] H. Liu, R. Setiono, et al. A probabilistic approach to feature selection-a filter solution. In *ICML*, volume 96, pages 319–327. Citeseer, 1996.

[13] A. Odić, M. Tkalčič, J. F. Tasič, and A. Košir. Predicting and detecting the relevant contextual information in a movie-recommender system. *Interacting with Computers*, 25(1):74–90, 2013.

[14] H. Peng, F. Long, and C. Ding. Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 27(8):1226–1238, 2005.

[15] N. Rubens and M. Sugiyama. Influence-based collaborative active learning. In *Proceedings of the 2007 ACM Conference on Recommender Systems*, pages 145–148. ACM, 2007.

[16] B. Vargas-Govea, G. González-Serna, and R. Ponce-Medellın. Effects of relevant contextual features in the performance of a restaurant recommender system. *ACM RecSys*, 11, 2011.

[17] Y. Zheng, B. Mobasher, and R. Burke. Context recommendation using multi-label classification. In *Proceedings of the 13th IEEE/WIC/ACM International Conference on Web Intelligence*, pages 288–295. IEEE/WIC/ACM, 2014.

Fostering Knowledge Exchange Using Group Recommendations

Alexander Felfernig
Institute for Software Technology
Inffeldgasse 16b
A-8010 Graz, Austria
felfernig@ist.tugraz.at

Martin Stettinger
Institute for Software Technology
Inffeldgasse 16b
A-8010 Graz, Austria
stettinger@ist.tugraz.at

Gerhard Leitner
Institute for Informatics Systems
Universitätsstraße 65-67
A-9020 Klagenfurt, Austria
gerhard.leitner@aau.at

ABSTRACT

The more domain knowledge individual participants of a group decision process share with each other, the higher the probability of high-quality decision outcomes. In this paper we report the results of an initial empirical study conducted on the basis of a group decision support environment. In this study, groups were confronted with recommendations with a varying degree of diversity. The higher the diversity of recommendations provided to groups, the higher was the degree of knowledge exchange.

Keywords

Group Recommenders, Decision Support, Hidden Profiles.

Categories and Subject Descriptors

H.5.3. [**Group and Organization Interfaces**]: Computer-supported cooperative work.

1. INTRODUCTION

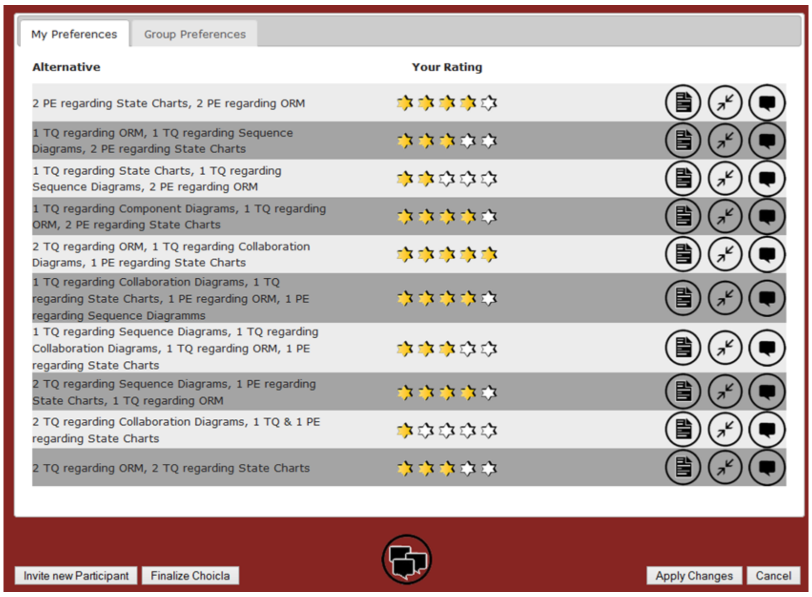
In contrast to single user recommenders [7], group recommenders determine relevant items for whole groups [6, 10]. For example, Jameson [5] introduces a prototype application that supports groups of users in the identification of a holiday destination. Masthoff [9] introduces concepts for sequencing television items for groups of users on the basis of different models from social choice theory (see also [10]). O'Connor et al. [16] introduce a collaborative filtering based movie recommender system that determines recommendations for groups of users. Ninaus et al. [15] show the application of group recommendation technologies in requirements engineering scenarios where stakeholders are in charge of cooperatively developing, evaluating, and prioritizing requirements. Finally, McCarthy et al. [12] introduce a critiquing-based recommender that supports groups of users in a skiing holiday package selection process. CHOICLA[1] is a group decision support environment which includes group recommendation technologies – this system was used as a basis for the user study presented in this paper.

Psychological aspects of group decision making play an increasingly important role in the development of (group) recommendation technologies [8]. Especially decision biases which denote suboptimal shortcuts in decision making can lead to low-quality decision outcomes. Masthoff and Gatt [11] discuss approaches to the prediction of user (group member) satisfaction with recommendations – in this context, conformity and emotional contagion are mentioned as major influence factors. Felfernig et al. [4, 20] analyze the impact of conformity in the context of group decision making and report an increasing diversity of the preferences of group members the later individual preferences are disclosed to the whole group. Chen and Pu [2] show how emotional feedback from group members can be integrated in group (music) recommendation. An outcome of their study is that emotional feedback can enhance mutual awareness of user preferences in the group. For a short overview of decision biases in recommender systems we refer to Felfernig [3, 21].

The frequency of knowledge exchange within a group can have a major impact on the quality of the decision outcome [14]. The more decision-relevant knowledge is exchanged between individual group members, the higher is the probability of discovering the hidden profile which can be characterized as the relevant knowledge to take a good (if optimality criteria exist, also an optimal) decision [22]. A consequence for group decision environments is that decision support has to include mechanisms that *pro-actively encourage knowledge exchange*. One reason for increased knowledge exchange between group members is *group diversity* (in terms of dimensions such as demographic and educational background), i.e., the higher the degree of diversity the higher the probability of higher quality decision outcomes (measured, e.g., in terms of the degree of susceptibility to the framing effect [23]). Schulz-Hardt et al. [17] discuss the role of *dissent* in group decision making: the higher the dissent in initial phases of a group decision process, the higher the probability that the group manages to share the decision-relevant information (discover the hidden profile).

The major focus of our empirical study was to analyze the impact of *recommendation diversity* on the frequency of knowledge exchange between group members. A major reason for increasing the diversity of recommendations is the

[1]www.choicla.com.



Alternative	Your Rating
2 PE regarding State Charts, 2 PE regarding ORM	★★★★☆
1 TQ regarding ORM, 1 TQ regarding Sequence Diagrams, 2 PE regarding State Charts	★★★☆☆
1 TQ regarding State Charts, 1 TQ regarding Sequence Diagrams, 2 PE regarding ORM	★★☆☆☆
1 TQ regarding Component Diagrams, 1 TQ regarding ORM, 2 PE regarding State Charts	★★★★☆
2 TQ regarding ORM, 1 TQ regarding Collaboration Diagrams, 1 PE regarding State Charts	★★★★★
1 TQ regarding Collaboration Diagrams, 1 TQ regarding State Charts, 1 PE regarding ORM, 1 PE regarding Sequence Diagramms	★★★★☆
1 TQ regarding Sequence Diagrams, 1 TQ regarding Collaboration Diagrams, 1 TQ regarding ORM, 1 PE regarding State Charts	★★★☆☆
2 TQ regarding Sequence Diagrams, 1 PE regarding State Charts, 1 TQ regarding ORM	★★★★☆
2 TQ regarding Collaboration Diagrams, 1 TQ & 1 PE regarding State Charts	★☆☆☆☆
2 TQ regarding ORM, 2 TQ regarding State Charts	★★★☆☆

Figure 1: CHOICLA group decision support environment: in the description of the alternatives (alternative exam modes) TQ denotes *theoretical assignment* and PE denotes a *practical assignment*.

fact that otherwise recommendations are too similar to each other and thus provide only a limited coverage of the whole item space [13, 18]. There is always a trade-off between similarity and diversity since too diverse recommendations can lead to situations were relevant items are omitted, i.e., are not recommended although relevant for the user.

In this paper we do not focus on the prediction quality of recommendation algorithms but analyze in which way recommendations can be used to increase knowledge exchange between the members of a group. In the context of group decision making it is often more important to increase the performance of the group rather than predicting decisions that will be taken by the group. In this paper we analyze three different basic group recommendation heuristics (min, avg, and max group distance) with regard to their impact on the communication behavior inside a group. The basis for our analysis is an empirical study that was conducted in a computer science course at our university. The results of our analysis show that recommendation diversity can trigger additional (decision-relevant) communications.

The remainder of this paper is organized as follows. In Section 2 we describe the design of our user study and discuss related results. In Section 3 we discuss open issues for future work. With Section 4 we conclude the paper.

2. USER STUDY

The task of each group (of undergraduate students) in the empirical study (N=256 participants, 12% female, 88% male) was to select their preferred exam mode for their Software Engineering course, for example, *1 theoretical assignment on Object-Relational Mapping (ORM), 1 theoretical assignment on Sequence Diagrams, and two practical assignments on State Charts* (see Figure 1). The participants were informed about the fact that there is no guarantee that the articulated preferences will be taken into account in upcoming exams. Each participant was a member of exactly one group (team) that had to implement a software within the scope of the course. Alternative exam modes (different topics and different shares of theoretical and practical assignments) were modeled in CHOICLA (see Figure 1).

Each group had the task to use the CHOICLA group decision support environment to cooperatively identify a ranking for the different assignment types. Each group member had to define his/her own ranking (see Figure 1) and was not able to see the preferences of the other group members. Participants of the study were encouraged to take a look at the group recommendations (tab *group preferences*) which was done by 91.41% of the participants at least once. Different group decision heuristics were used in our study and each group was assigned to a CHOICLA version that implemented exactly one of these heuristics.[2] Related group recommendations d differ in terms of their *diversity* compared to the individual user ratings (rating scale: 1-5 stars) of an alternative s determined by $eval(u, s)$ (see Formula 1).

$$diversity(d) = \frac{\sum_{u \in Users} |eval(u, s) - d|}{\#Users} \quad (1)$$

The (low diversity) *minimum group distance* heuristic ($GD_{\min}$) returns a rating d that represents the minimum distance to the ratings of group members (see Formula 2).

$$GD_{\min}(s) = \arg \min_{d \in \{1 \ldots 5\}} \left(\sum_{u \in \text{Users}} |\text{eval}(u, s) - d| \right) \quad (2)$$

The (highly diverse) *maximum group distance* heuristic ($GD_{\max}$) returns a rating d that reflects the maximum distance to current ratings of group members (see Formula 3).

$$GD_{\max}(s) = \arg \max_{d \in \{1 \ldots 5\}} \left(\sum_{u \in \text{Users}} |\text{eval}(u, s) - d| \right) \quad (3)$$

Finally, *average group distance* represents a value between maximum and minimum group distance (see Formula 4).

[2]Note that CHOICLA includes a set of group heuristics from social choice theory [10], GD_{max} and GD_{avg} have been included for the purpose of the empirical study.

$$GD_{avg}(s) = \frac{GD_{\min}(s) + GD_{\max}(s)}{2} \quad (4)$$

An overview of the assignment of groups to the different decision heuristics is depicted in Table 1.

heuristic	#groups	#participants
min	17	92
avg	12	69
max	16	95
total	45	256

Table 1: Group distribution in the empirical study.

Hypotheses. The basic assumption of hypothesis H1 is that group decision heuristics with a higher diversity lead to an increased knowledge exchange between group members. The reason for this is that recommendations can act as an anchor [1] and also have the potential to induce the feeling of dissent in the group which needs to be resolved by the group members. An increased amount of knowledge exchange can help to discover the hidden profile of a group decision [14, 22], i.e., the amount of decision-relevant knowledge is increased. Furthermore, we assume that a higher frequency of knowledge exchange is correlated with higher time efforts per group.

Examples of knowledge exchanged within the scope of our empirical study are the following (see Table 2).[3]

Content-related. A student only took a look at exercises related to *Object-Relational Mapping (ORM)* and asks for further information regarding the topic. Another student of the same group points out that there are only a few slides with very simple and understandable rules which are also very useful in industrial contexts.

Preference-related. A student emphasizes that he/she prefers to include appointments on *UML Class Diagrams* compared to appointments related to the *Unified Process.*

Recommendation-related. A student does not like the group recommendation since it does not take into account his/her preferences. Furthermore, he/she articulates an urgent need to further discuss assignment topics that are acceptable for the group as a whole. For recommendation-related comments we also evaluated the valence, i.e., how positive/negative a recommendation was perceived.

The assumption of hypothesis H2 is that a higher degree of knowledge exchange increases the flexibility of group members to change their initial preferences. Due to the fact that more decision-relevant knowledge is exchanged between group members, the amount of global decision-relevant knowledge is increased which improves the individual capabilities of taking into account additional decision alternatives. Increased knowledge exchange between group members helps to overcome a *discussion bias* (group discussions tend to be dominated by information group members already knew before the discussion [19]).

Hypothesis H1 can be confirmed, i.e., the amount of decision relevant knowledge exchanged between group members increases with the diversity degree of the used group recommendation heuristic. The higher the diversity, the higher the number of decision-relevant comments given within the scope of the decision process (see Table 2). Furthermore, also the overall time investments increase with the diversity of the decision heuristic (see Table 3).

[3]The categorization into the types *content-related*, *preference-related*, and *recommendation-related* was performed manually.

heuristic	content	preferences	recommendation
min	22	0	27 (+4.2)
avg	31	26	35 (+0.9)
max	79	91	108 (-4.4)

Table 2: Content-, preference-, and recommendation-related comments (valence [-5 .. +5]).

heuristic	avg. endtime−starttime (h)	avg. efforts (min)
min	71.06 (13.05)	210.71 (20.19)
avg	85.64 (26.58)	234.56 (17.67)
max	101.18 (19.48)	278.46 (16.74)

Table 3: Time (and std.dev.) invested per group for decision task completion (i.e., rating of alternatives).

We can also confirm hypothesis H2. A higher degree of knowledge exchange between group members also provides flexibility regarding the final group decision. Table 4 provides an overview of the degree of opinion adaptation of groups depending on the supported decision heuristics.

heuristic	avg. change of rating
min	0.67
avg	1.32
max	2.46

Table 4: Changes of initial ratings depending on the supported decision heuristic.

Summarizing, the higher the diversity of the used decision heuristic, the higher the frequency of knowledge exchange between group members. Consequently, recommendations in the context of group decision support can also be exploited to adapt a user's group decision behavior which can lead to higher quality decision outcomes. Diverse recommendations can help to detect hidden profiles [17, 19] which represent an amount of global decision-relevant knowledge needed to take good (or even optimal) decisions. Online group decision support environments have to be aware of this fact and should also take into account diversity in group recommendations.

3. FUTURE WORK

Major issues for future work are the following. Our study is limited in the sense of having investigated a set of basic heuristics (diversity measures) (min, avg, and max group distance). In our future research we will investigate further decision heuristics (see, e.g., [10]) with regard to their capability to increase the frequency of knowledge exchange and to increase decision quality. We will also focus on a more fine-grained analysis of potential optimal degrees of diversity that help to maximize knowledge exchange while decreasing the perceived quality of recommendations as little as possible. The average diversity (Formula 1) of recommendations determined by the three different heuristics is depicted in Table 5. We want to emphasize that the satisfaction with group recommendations significantly decreases if the degree of diversity is too high – Table 6 summarizes user feedback regarding the perceived satisfaction with the group recommendations.

heuristic	min	avg	max
diversity	0.84	1.38	2.23

Table 5: Diversity of group recommendations.

heuristic	very satisfied	satisfied	average	unsatisfied	very unsatisfied
min	67	12	9	2	2
avg	17	14	12	14	12
max	2	1	15	25	52

Table 6: Satisfaction with group recommendations.

4. CONCLUSIONS

In this paper we presented the results of an initial empirical study that focused on possibilities of increasing the amount of knowledge exchange in group decision scenarios. In this context, we showed that the diversity of recommendations can have a significant impact on the frequency of knowledge exchange – the higher the diversity of group recommendations, the higher the corresponding number of comments included in the group decision process. The results presented in this paper are a first step towards the application of recommendation technologies to foster knowledge exchange in group decision making.

5. REFERENCES

[1] Adomavicius, G., Bockstedt, J., Curley, S., and Zhang, J. Recommender systems, consumer preferences, and anchoring effects. In *Decisions@RecSys11 Workshop* (Chicago, IL, USA, 2011), 35–42.

[2] Chen, Y., and Pu, P. Cofeel: Emotional social interface in group recommender systems. In *RecSys'12 Workshop on Interfaces for Recommender Systems* (Dublin, Ireland, 2012), 48–55.

[3] Felfernig, A. Biases in decision making. In *International Workshop on Decision Making and Recommender Systems 2014*, vol. 1278 of *CEUR Proceedings* (2014), 32–37.

[4] Felfernig, A., Zehentner, C., Ninaus, G., Grabner, H., Maaleij, W., Pagano, D., Weninger, L., and Reinfrank, F. Group decision support for requirements negotiation. In *Advances in User Modeling*, vol. 7138 of *LNCS*, Springer (2012), 105–116.

[5] Jameson, A. More than the sum of its members: Challenges for group recommender systems. In *Intl. Working Conf. on Adv. Visual Interf.* (2004), 48–54.

[6] Jameson, A., and Smyth, B. Recommendation to Groups. In *The Adaptive Web: Methods and Strategies of Web Personalization*, P. Brusilovsky, A. Kobsa, and W. Neijdl, Eds. Springer, 2007, ch. 20, 596–627.

[7] Jannach, D., Zanker, M., Felfernig, A., and Friedrich, G. *Recommender Systems: An Introduction.* Cambridge University Press, 2010.

[8] Mandl, M., Felfernig, A., Teppan, E., and Schubert, M. Consumer Decision Making in Knowledge-based Recommendation. *Journal of Intelligent Information Systems (JIIS) 37*, 1 (2010), 1–22.

[9] Masthoff, J. Group modeling: Selecting a sequence of television items to suit a group of viewers. *UMUAI 14*, 1 (2004), 37–85.

[10] Masthoff, J. Group Recommender Systems: Combining Individual Models. *Recommender Systems Handbook* (2011), 677–702.

[11] Masthoff, J., and Gatt, A. In Pursuit of Satisfaction and the Prevention of Embarrassment: Affective State in Group Recommender Systems. *User Modeling and User-Adapted Interaction 16*, 3–4 (2006), 281–319.

[12] McCarthy, K., Salamo, M., Coyle, L., McGinty, L., Smyth, B., and Nixon, P. Group Recommender Systems: A Critiquing based Approach. In *IUI'06*, ACM (2006), 267–269.

[13] McGinty, L., and Smyth, B. On the Role of Diversity in Conversational Recommender Systems. In *5th Intl. Conf. on Case-based Reasoning* (2003), 276–290.

[14] Mojzisch, A., and Schulz-Hardt, S. Knowing Other's Preferences Degrades the Quality of Group Decisions. *Jrnl. of Pers. & Social Psy. 98*, 5 (2010), 794–808.

[15] Ninaus, G., Felfernig, A., Stettinger, M., Reiterer, S., Leitner, G., Weninger, L., and Schanil, W. Intelligent techniques for software requirements engineering. In *Europ. Conf. on AI, Prestigious Applications of Intelligent Systems (PAIS)* (2014), 1161–1166.

[16] O'Connor, M., Cosley, D., Konstan, J., and Riedl, J. PolyLens: A Recommender System for Groups of Users. In *Europ. Conf. on Computer-Supported Cooperative Work*, ACM (2001), 199–218.

[17] Schulz-Hardt, S., Brodbeck, F., Mojzisch, A., Kerschreiter, R., and Frey, D. Group Decision Making in Hidden Profile Situations: Dissent as a Facilitator of Decision Quality. *Journal of Personality & Social Psychology 91*, 6 (2006), 1080–1093.

[18] Smyth, B., and McClave, P. Similarity vs. diversity. In *4th Intl. Conf. on Case-based Reasoning (ICCBR'01)*, Springer (London, UK, 2001), 347–361.

[19] Stasser, G., and Titus, W. Pooling of Unshared Information in Group Decision Making: Biased Information Sharing During Discussion. *Jrnl. of Pers. and Social Psy. 48*, 6 (1985), 1467–1478.

[20] Stettinger, M., Felfernig, A., Leitner, G., and Reiterer, S. Counteracting Anchoring Effects in Group Decision Making. In *UMAP 2015*, vol. 9146 of *LNCS* (Dublin, Ireland, 2015), 118–130.

[21] Stettinger, M., Felfernig, A., Leitner, G., Reiterer, S., and Jeran, M. Counteracting Serial Position Effects in the CHOICLA Group Decision Support Environment. In *ACM IUI2015* (Atlanta, GA, USA, 2015), 148–157.

[22] Wittenbaum, G., Hollingshead, A., and Botero, I. From Cooperative to Motivated Information Sharing in Groups: Moving Beyond the Hidden Profile Paradigm. *Comm. Monographs 71*, 3 (2004), 286–310.

[23] Yaniv, I. Group Diversity and Decision Quality: Amplification and Attenuation of the Framing Effect. *International Journal of Forecasting 27* (2011), 41–49.

Explaining contextual recommendations: Interaction design study and prototype implementation

Joanna Misztal
Jagiellonian University
Cracow, Poland
joanna.misztal@uj.edu.pl

Bipin Indurkhya
Jagiellonian University
Cracow, Poland
bipin.indurkhya@uj.edu.pl

ABSTRACT

We describe an architecture for generating context-aware recommendations along with detailed textual explanations to support the user in the decision-making process. CARE (Context-Aware Recommender with Explanation) incorporates a hierarchical structure, in which independent modules embodying different aspects of the context cooperate together to generate recommendations for the user with accompanying rationales. We follow the Interaction Design principles to develop personas, goals and user scenarios, based on which a prototype system is developed. We present here two examples of its performance when processing movie-ratings data set with contextual information. We argue that our architecture is extensible in that more modules can be added as needed, and the approach can be applied to other domains as well.

Keywords

context-aware recommender system, recommendations explanations, interaction design

1. INTRODUCTION

An increasing number of available resources, and easy online access to diverse goods has resulted in data overload, making it difficult for many users to decide what items to select, which often slows down their decision-making process. A growing number of choices is leading to an emerging interest in the development of decision-support systems to help users in finding the most interesting or suitable items for their personal needs. Most of the research in this domain is focused on improving the accuracy and precision of recommendations. However, it is equally important to provide the user with some rationale for why a particular item is being recommended to them. Moreover, in some domains such as legal decision-making or moral and ethical reasoning, the justifications for recommendations are very crucial. Hence, the main focus of our work is to design a system that can explain why the user should select particular items.

As observed in [3], the user's preferences may be influenced by factors as diverse as time of the day, day of the week, the season or the weather at the moment, and so on. In our system, we incorporate different independent modules such that each module implements a particular approach to generating a recommendation based on a single contextual feature. This architecture allows generating a number of diverse recommendations, as each piece of contextual information is analyzed separately and the most approporiate items are recommended by choosing from among the various recommendations generated by different modules.

As the main focus of our research is to improve user's experience and understanding during the interaction with the system, we designed a Context-Aware Recommender with Explanation (CARE) system, following the Interaction Design paradigm [10]. Personas, user goals and scenarios are developed after interviewing potential recommendation-system users and a domain expert, based on which the prototype of the system is designed.

We motivate here our approach in the context of the current state of the art, summarize the interaction design process, and present examples of persona and scenarios. Then we describe the system architecture and present some results generated by our prototype implementation.

2. BACKGROUND

As defined in [20], the main goal of a recommender system is to support the user in a decision-making process by suggesting items that they might find interesting. Since information overload is a growing problem for Web users, because of an exponential increase in the amount of web content that is being generated, development of such tools has become a thriving research area in recent years. Consequently, several techniques have been developed for predicting users' preferences.

2.1 Content-based recommender systems

Content-based recommenders try to find items similar to what the user previously liked [17]. These work by identifying key features of the items highly rated by the user in the past, and by building a user-preference model from those characteristics [5].

A significant problem in many recommendation techniques is the *cold-start* problem, which occurs when a new user or a new item is presented to the system and there is not enough data to perform a reliable prediction - the user has not rated enough products to define his or her preferences, or an item has not been rated by a sufficient number of users.

Content-based recommenders deal well with situations when a new item is added to the system. It also maintains independence among the users as a particular user's ratings are sufficient to perform the recommendation process for that user. For our research, a major advantage of content-based recommenders is their transparency — the features that triggered the recommendation results may be listed along with the output.

However, content-based recommenders suffer from overspecialization: i.e. they recommend items similar to those seen in the past, preventing a serendipity of recommendations. Also, when a new user, who does not have a previous history with the system and so lacks any ratings, enters the system, the *cold-start* problem may occur.

Content-based analysis operates on vectors representing features of each object. Two basic techniques for filtering similar items are similarity calculation (such as cosine similarity) and distance measurement (such as Euclidean distance).

2.2 Collaborative filtering recommender systems

In collaborative filtering (CF), a user's preferences are predicted based on modelling other users behaviors [24, 16]. The basic idea behind this approach is that the rating of a user for a new item should be close to the ratings of users who have similar tastes.

This approach suffers from the data-sparsity and new-item problems. Another disadvantage is that collaborative-filtering recommenders mostly work as black-box systems, therefore they lack transparency and cannot explain why certain items are being recommended. However, this approach is proving to be an effective technique for making recommendations and is widely used in commercial applications. It has an advantage of being able to recommend items with unknown content. CF also supports serendipity in recommendations, for recommended items may differ significantly from the previous ones.

Common approaches to CF for recommendations use neighbourhood-based or model-based methods. Neighbourhood-based methods try to find the most similar items (item-based approaches) or most similar users (user-based approaches) when predicting the rating of an item for a particular user. Basic technique may incorporate correlation measures (such as Pearson's similarity) when comparing the vector of ratings for users or items [14]. Model-based methods work by finding the latent features that characterize the user's ratings, and build a predictive model of their preferences. Such methods may employ Matrix Factorization algorithms, Bayesian models, Support Vector Machines or other such techniques.

2.3 Context-aware recommender systems

As observed in [3], a person's preferences may be influenced by factors as diverse as time of the day, day of the week, the season, the weather at the moment, and so on. Context-aware recommender systems (CARS) try to model user's preferences considering changing contexts that may affect user's moods and tastes [4]. Contextual data may be collected explicitly by asking the user some questions, or implicitly from the environment (information such as time, day of week, season or location). Some information may also be statistically inferred from the other data (such as the companion or mood). In CARS, the input data from the standard recommendation approach in the form $< user, item, rating >$ is extended by an additional parameter of *context*. Some standard approaches to recommendations have been adapted to model the additional dimension of context. In [15], the authors present a *Tensor Factorization* model-based technique using N-dimensional tensor of User-Item-Context instead of the 2D User-Item matrix.

Standard approaches for CARS implementation incorporate contextual pre-filtering (items filtered by context before recommendation), post-filtering (context applied to recommendation results) and contextual modeling (context as a part of ratings prediction). Common approaches based on standard pre-filtering techniques represent item and user-splitting algorithms [28, 6]. In these methods, items (or users) in different contexts are treated as separate objects for the recommendation algorithm. Context-aware systems are known to increase the accuracy of recommendations [4]. However, they face the data sparsity problem, as the number of ratings is restricted to given context. As discussed in [7], the most efficient approach to context-splitting is *single split*, where objects are split considering a single contextual feature.

2.4 Other approaches

Some recommenders are implemented as knowledge-based systems [8], where the recommendation algorithm is performed by a set of constraints representing the knowledge about the domain. Such systems may be applicable to the domains for which historical data is not available, or when the user does not perform the action often enough, so there is little data to make a prediction (e.g. buying a car).

Another approach is to use demographic information about the users to predict their tastes based on their social group. Such recommendations may depend on user's age, gender, nationality, and so on.

2.5 Hybrid solutions

Each of the techniques mentioned above has some advantages as well as some drawbacks, and each may be effective for a certain domain or a certain type of problem [13, 9]. In order to build a more general recommender system, or to improve the quality of recommendations, *hybrid systems* combine diverse recommendation algorithms. There are different ways to combine outputs of various recommendation strategies, which are classified by Burke [9] as follows:

Weighted: the scores from several recommenders are weighted into one result.

Switching: the most appropriate technique is selected depending on the input data.

Mixed: outputs from diverse algorithms are presented simultaneously.

Feature combination: features of different algorithms are combined into a new feature.

Cascade: the recommendation is performed hierarchically and the outputs are refined by the subsequent recommenders.

Feature augmentation: output from one system is the input to the following one.

Meta-level: the model created by one system is used by another.

2.6 Multi-agent approaches

Hybrid recommender systems are often implemented based on diverse multi-agent system (MAS) architectures. Distributed approaches to recommendations have been previously studied in [27], where a collaborative recommendation algorithm is implemented using cloud computing. Sabater, Singh and Vidal [22] proposed a protocol in which a group of selfish agents can decide how to share their recommendations with the others. In [26], the authors introduce *recommender agents* to enable the user's interaction with the system and to combine the outputs of the recommendation algorithms with other techniques such as other users bookmarks and tags.

Another example of MAS architecture that may be used for implementing a recommender system is the *blackboard architecture.* This architecture may be visualized by the metaphor [11] of a group of independent experts with diverse knowledge who are sharing a common workspace (the blackboard). They work on the solution together and each of them adds some contribution to the blackboard, whenever apossible, until the problem is solved.

The blackboard model provides an efficient platform for problems that require many diverse sources of knowledge. It allows a range of different *experts* represented as diverse computational agents, and provides an integration framework for them. It seems a promising platform for recommendation tasks, and has already been incorporated in [12, 21].

3. OVERVIEW OF CARE SYSTEM

3.1 Architecture

CARE (Context-Aware Recommender with Explanation) is built as a hybrid architecture that adapts a *mixed* approach to recommendations, and incorporates some features of the multi-agent blackboard architecture. We implemented each module as an independent subsystem that uses some particular approach to generating a recommendation by incorporating a particular contextual feature. As some of those factors may be non-deterministic, we present the final result as an array of alternate choices and allow the user to choose from the recommendations generated by analyzing diverse contextual factors. Hence the diversity of final recommendation outputs is ensured by presenting the analysis from multiple points of view. This approach also incorporates serendipity, and gives the users a choice of possible actions. The users actions can be noted and used for ordering future recommendations.

Our solution also embodies some aspects of the *feature augmentation* approach — we perform the recommendations hierarchically on different levels of abstraction. We introduce some inter-level recommenders that are responsible for defining the features of items that are most liked by the users in a given context. Their outputs are used to filter the data with identified characteristics, which is then sent as an input to the higher layer of recommendations.

3.2 Evaluation

Most of the solutions for automatic recommendations are focused on development of techniques improving the overall performance and accuracy of ratings prediction. Accordingly, most popular evaluation approaches incorporate precision metrics for the estimations. However, there are other factors that impact the effectiveness of recommendations and influence the user experience.

A major limitation of the existing recommendation systems is overspecialization and a lack of diversity in recommender outputs [29, 19]. As described in [23], during the challenge on Context-Aware Movie Recommendation (CAMRa 2010), competing systems were evaluated according to diverse factors divided into two groups. The first set of criteria consisted of precision metrics while the other set contained the following *Subjective Evaluation* Criteria: Context, Contextualization of recommendations, Extensibility, Serendipity, Creativity, Scalability, Sparsity, Domain dependence, and Adoptability

In our research, we aim to address some of these subjective criteria to improve user experience, and also develop an architecture that is easily adaptable to other domains. We use contextual filtering to model user preferences. Our architecture enables one to implement flexible and generic recommender systems that can easily be extended with new independent modules in a hierarchical structure. Our approach promotes diversity and serendipity among the recommendation outputs as each module processes information from a different point of view.

3.3 Explaining recommendations

We mentioned above that aspects such as user satisfaction play an important role in the evaluation of a recommender system. As noted by [25], a limitation of many recommenders is that they work as black-box systems and do not provide the users with any reasons for providing a particular recommendation. Some of the commercial systems are striving to overcome this limitation by producing a rationale accompanying each recommendation. A number of diverse styles have emerged to provide this rationale [25]: *Case-Based* (*... because you highly rated Item A...*, used in Netflix [2]), *Collaborative* (Customers who bought this Item were also interested in... used by Amazon), *Content-Based* (We are playing this music because it has a slow tempo by Pandora [1]). In [26], the authors use information visualization techniques to improve interaction with their recommender system. Such system transparency not only increases the user satisfaction, but also helps the users in making easier and faster decision in selecting an item.

In CARE, each module is provided with an explanation-generating function, which produces a description of the features that determined the recommendation. The style of this message is dependent on the module's implementation. The final explanation may combine different styles of messages produced independently by separate modules. We incorporate modules that process information on different levels of abstraction, hence the final description contains rationales at multiple granularity levels. Our goal is to generate a rationale explaining to the user why she or he should find certain items interesting (contextual reason, e.g. *because it is rainy* and what features make this particular item a relevant choice (e.g. *because you like rock music when it is rainy*).

4. GOAL-DIRECTED DESIGN

In designing the CARE system, we follow the principles of Interaction Design [10], according to which the design of a system is developed iteratively through continuous interaction with the user. In the subsequent sections, we summarize the conclusions from the interviews with three users and a movie-domain expert.

4.1 Domain expert's opinion

Following the Interaction Design paradigm, we consulted a film analysis academic to find out what factors may influence the popularity of a movie among the users.

The expert noted that the genre is not the only feature that the users consider when deciding which movie to watch. Other factors which may determine their preferences are narrative description, its tempo, atmosphere and tension.

Moreover, the users often want to watch the same kind of movies that they have already watched. Thus, they select well-known names, plots or recognizable brands. Such a *brand* may be defined by the director, movie star or award such as Oscar or some Film Festival.

Considering all these factors, our system design should embody modules that analyze relevant movie features such as genre, cast, director, awards won as well as information about the atmosphere of the movie. The prototype implementation incorporates the genre filtering modules, and we plan to extend it with modules that will analyze other factors as well.

Another major factor that influences a user's choice is the current trend or fashion. There are some *must-see* movies that many users desire to watch. Moreover, some people rely on the public opinion more than on their own impressions. In our design, the public opinion is modelled by a collaborative filtering recommendation algorithm.

4.2 Defining User Goals

We interviewed three potential users to determine their expectations from a recommender system. The volunteers were technical faculty graduates who are familiar with using recommender systems to find items of their interest. Each of them was interviewed separately, in their natural environment. They were asked to describe their experiences in one of the three domains of recommendations: books, movies and music. First, each person was asked to describe his or her general preferences in the given area and if they could identify some factors that may influence it. Then they were asked to describe some particular situations in which they use recommendations, considering the context details. The final question was about the expected recommendation output in these situations. We are planning to extend this research by incorporating interviews with a broader group of users with more diverse backgrounds.

It was observed that every person has some general tastes, but particular preferences change according to the time and the mood. Thus, the system should consider some contextual information in generating the recommendations. However, some of these factors, such as the user's mood, who they are with, and so on, may be difficult to predict. Hence the approach we chose is to give the user a choice of possible actions considering different contextual or affective states so that the user can decide what she or he needs at the moment.

Finally, we found that the users like to know why any particular item is recommended to them. Hence the system should aggregate information from different levels of abstraction, and present it to the user in an intuitively understandable way. We plan to present a rationale for each recommendation as an accompanying text message.

4.3 User scenarios

Persona: Mark

Goals: getting movies recommendations; finding uncommon yet interesting movies when alone; finding lights comedies to watch with his girlfriend

Scenario 1

It is a cold winter Friday and Mark and his girlfriend want to spend the evening with a light movie and a glass of wine. Mark opens CARE and a message pops out:

Hi Mark! Finally, it's the weekend! It's freezing, isn't it? Are you dreaming of little holidays? What about Woody Allen's "Vicky Cristina Barcelona" to warm you up a little? I know you like this director. Or maybe you had a tough week and feel like watching something to cheer you up with a bit of dark humor, like "Grand Budapest Hotel"?

Scenario 2

On Monday, Mark's girlfriend is off for a ladies night with her friends so finally he can choose a movie on his own. CARE greets him:

Hi Mark! Maybe something positive for the new week? How about "Intouchables"? Or maybe you're fed up with the city life in Krakow and want to watch the story of a man in the heart of nature, like "Into the wild"? You like non-fiction movies!

5. SYSTEM ARCHITECTURE OF CARE

General architecture of the CARE system is presented in Figure 1. Modules in our prototype system work on different levels of abstraction.

First group of modules perform *contextual features filtering* based on the input with current context, user information and ratings history with context. The goal of this processing phase is to determine the most relevant item features for a given context. Each component on this level analyzes the information about a single contextual information. To address the problem of sparsity, we also incorporate a module that considers all users' ratings without any contextual filtering. The output of each filter is a list of items characterized by the identified features. This architecture is extensible with different types of filters that analyze other aspects of recommendations (such as demographic data). In this paper we focus only on the contextual information processing.

In the next stage of recommendation process, we incorporate recommender algorithms that select items that should be most liked by the user, considering each of the item groups received from the former stage as a separate recommendation problem. The modules on this stage may represent diverse recommendation techniques and algorithms, however our prototype implementation includes collaborative filtering algorithms. The result of recommendation is the best choice of items for each set of items.

Each component generates a short description of its results and the reason of recommendation. The messages are finally composed by the *explanation templates* module and presented to the user along with the recommended items.

The hierarchical structure of the system and the inter-level filtering of item features enables a more thorough explanation of the process in generated outputs. Hence the output does not only provide the user with the information *Item A was recommended because it is summer,* but also emphasizes the feature that was crucial for this choice (*Item A was recommended because you seem to like this type of items during summer.*).

The system is being developed using Django, a Web framework for Python. The system interface will be provided as a web application, however at present the system output is

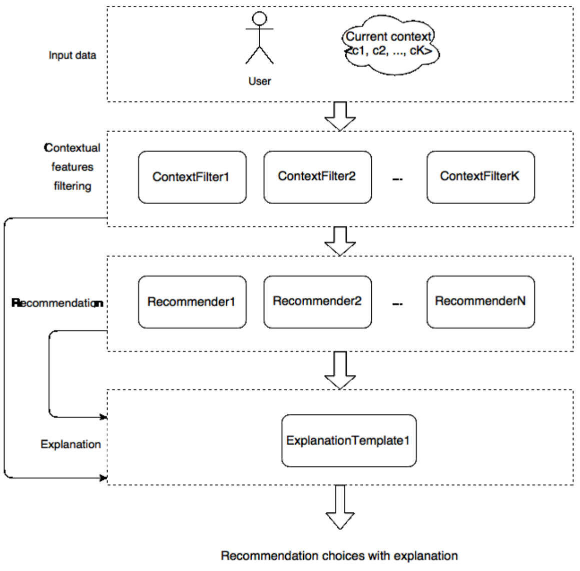


Figure 1: General architecture of the CARE system.

in plain textual form as presented in the results section.

6. EXPERIMENT

We present phases of the recommendation algorithm along with an illustrative example of recommendations for user John.

6.1 Testing data

CARE system architecture is applicable to many recommendation domains where the context may influence user preferences. Possible domains of application include books, music or movies as well as restaurants recommendations as user's choice may depend on aspects such as changing weather or time. Here we present the results from testing our prototype on the LDOS-CoMoDa dataset [18] that contains movie ratings along with contextual information, and user and item characteristics. Contextual data contain information about the season, type of day (weekend, working day or holiday), time of day (morning, afternoon, evening), companion, and so on.

The dataset also contains information about the *mood* of the movie and it could be interesting to consider this data as well. However, the dataset only provides the *dominant* and *end* emotional values, without any information about the user's mood *before* watching the movie. Since we treat contextual factors as facts known at the moment of recommendation (as the initial data), we cannot make a recommendation considering user's mood after or during watching the movie. In future work, we plan to add a feature to query user's mood at the moment of recommendation, and consider this information as a contextual parameter.

Table 2 contains a part of John's ratings for the analyzed example along with contextual information.

6.2 Recommendation process

We present below a brief description of the different stages of our algorithm along with illustrative examples. The main steps of algorithm are listed below and described in the subsequent sections:

1. Initialization with contextual input data.
2. Contextual type-splitting - identifying significant contextual factors and relevant data types.
3. Collaborative filtering items recommendation for each of the types defined in 2.
4. Explanation generation for each of the recommended types and a corresponding contextual factor.

6.2.1 *Input data*

Input for the recommendation is the user data and the information about the context in which the recommendation takes place. This data may contain information explicitly provided by the user (such as whether they are alone or with a companion) or implicit information extracted automatically from the date and location data (such as day of week, time of the day, weather, and so on).

Example:

Context:
Season: Summer
Day type: working day
Time: evening
Weather: sunny
Companion: alone

6.2.2 *Contextual type-splitting*

For the context-aware recommendation process, we introduced *contextual type-splitting* algorithm which is an adaptation of the standard *contextual item-splitting* approach. We incorporated an additional abstraction level and treated the item features as the recommendation objectives. Table 2 illustrates the difference between approaches.

- Contextual item-splitting

  In the first phase of the basic algorithm, each item is associated with the most relevant contextual feature that diversifies its ratings. Then the ratings for this item are split according to this division. The mean rating values for each item are compared for the situation where particular circumstance occurs and otherwise.

  For example, we could compare ratings for a particular movie that were given during the weekend with ratings from all other days. If they are significantly different, we can infer that the user preferences for this item are influenced by the day of the week.

- Contextual type-splitting

  In our approach, we perform analogical computations, but instead of comparing the contexts for each movie, we analyze each context for a group of movies with a common feature (such as a genre). As a result we can, for example, find out that the user prefers to watch horror movies during the weekends than on other days. This step may be generalized to recommend items grouped by other features, such as the director, country of origin, etc.

movie id	genre	rating	daytype	time	weather	season	companion
23	action movie	1	working day	evening	sunny	autumn	alone
160	action movie	2	working day	evening	sunny	autumn	alone
2755	action movie	4	working day	evening	rainy	winter	alone
3898	action movie	5	weekend	night	cloudy	spring	with family
3942	action movie	3	working day	evening	sunny	spring	alone
4020	action movie	5	weekend	night	rainy	summer	alone
3992	action movie	5	working day	afternoon	sunny	summer	with family
3962	animated movie	4	weekend	evening	rainy	spring	alone
65	comedy	3	working day	evening	sunny	autumn	alone
101	comedy	2	working day	afternoon	cloudy	autumn	alone
4025	comedy	5	working day	evening	sunny	summer	alone
4054	comedy	5	holiday	night	cloudy	summer	alone
30	crime movie	4	weekend	evening	rainy	autumn	alone
227	crime movie	2	working day	night	cloudy	autumn	alone
3715	crime movie	1	working day	evening	cloudy	winter	alone
248	crime movie	5	weekend	night	sunny	winter	with family
149	drama	5	weekend	evening	sunny	autumn	alone
61	drama	2	weekend	afternoon	sunny	autumn	alone
239	drama	3	holiday	night	rainy	autumn	in public
242	drama	4	holiday	evening	cloudy	autumn	alone

Table 1: User's ratings with contextual information.

Exemplary ratings with a context				
User	Item	Item type	Rating	Context
U1	I1	T1	1	C1
U1	I3	T1	3	C1
U1	I2	T2	5	C2
U1	I3	T1	5	C2
U1	I2	T2	5	C1

Contextual items splitting		
User	Item	Rating
U1	$I3_{C1}$	3
U1	$I3_{C2}$	5

Contextual types splitting		
User	Item type	Rating
U1	$T1_{C1}$	2
U1	$T1_{C2}$	5

Table 2: Contextual splitting - basic approach and proposed modification.

This approach allows us to give more transparent and intuitive recommendations for the user by presenting the features that lead to the final recommendation. It also addresses a major drawback of the context pre-filtering approaches, namely the data sparsity after applying the filter. As the number of recommendations for a particular type of movies is significantly higher than for each movie separately, we expect the results to be more accurate. Reducing number of comparisons to groups of items also contributes to decreasing complexity of computations and speeds up the recommendation process. In subsequent steps, the recommendations are performed for selected groups only.

We verify the significance of each contextual feature using the two-tailed Student's t-test, assuming the p-value threshold of 10%. Additionally, we consider the significance of only those features where the mean rating is higher when a particular context occurs. The t-test is a basic approach (as presented in [28]), however its use is limited for normally distributed data. In other cases, it may be replaced by other statistical methods such as Wilcoxon signed-rank test.

The features significance testing in a context is performed as follows:

- For each type t_i of items (eg. for each of the genres) we compare the mean rating for items of this type when each of the input contextual circumstances c_i occurs and otherwise (eg. ratings for comedies in summer and other seasons).
  $significance(c_i, t_j) = ttest(Ratings_{t_j|c_i}), Ratings_{t_j|\neg c_i})$
- If the statistical test for both groups of ratings split by the contextual feature c_i indicates a significant difference in mean ratings, and the mean rating for type when c_i occurs ($Ratings_{t_j|c_i}$) is higher then otherwise ($Ratings_{t_j|\neg c_i}$), we conclude that the type t_i is a relevant recommendation in given context c_i.

Example:

Contexts significance testing:
Input context: summer, working day, evening, sunny, alone
Ratings in summer vs other seasons: mean ratings for comedies are significantly higher during summer.
Ratings on working days vs other days: no relation found.
Ratings in the evening vs other time: mean ratings for comedies are significantly higher in the evening.
Ratings on sunny days vs other weather: no relation found.
Ratings for movies watched alone and other companion: no relation found.

6.2.3 Types pre-filtering

After identifying the types of movies that are most relevant for a given context, we filter the set of ratings for each type separately. For example, if we consider a recommendation for Saturday morning and we find out that horror movies are the most preferred genre during the weekend, and comedies get the highest ratings in the mornings, we first consider recommendations for horror movies and then for comedies separately.

We also calculate general recommendations considering the user preferences of all the items, without any pre-filtering. This addresses the sparsity problem for context recommendations and deals with the situation when no relevant contextual information is provided.

6.2.4 Items recommendation

After filtering the items by their types, we perform a standard collaborative-filtering recommendation algorithm to find the most suitable choices considering a particular user's taste. We calculate the similarity between users using Pearson's correlation measure. Then we consider the ratings of the most similar users with the K-Nearest-Neighbours algorithm. In further research we plan to address the problem of scalability by users clustering.

For each of the identified categories we perform a separate recommendation process, hence the final output contains the recommendation results from diverse perspectives. For the current implementation, we incorporated the standard user-based CF algorithm. However the system may be easily extend by other modules performing different recommendation algorithms since the calculations are performed independently.

Example:

Performing user-based collaborative filtering algorithm to find expected highest-rated movies for each category and general user's preferences without any context:
crime story: "The Usual Suspects"
action movie: "Pirates of the Caribbean: At World's End"
all movies: "Le Concert"

6.2.5 Explanations generation

A major goal of our research is to develop a recommendation system that can present the recommendations along with accompanying explanations. Hence, we incorporated modules responsible for generating textual messages to give rationales for recommendations. Each sentence of the accompanying message consists of the following information: item type; recommended item; context.

The message is generated in the form of a textual template. Future improvements of CARE will consider increasing the serendipity aspect of the recommendations by generating more advanced and surprising commentaries for the outputs.

The messages generated by all the modules that analyzed the situation from different perspectives are aggregated in one template and presented to the user as a message.

Example:

Producing a textual explanation containing descriptions from all former steps of algorithm:
Hi John! You might like a comedy "Le Concert" as it is something in your taste. You might like a drama like "Shutter Island" because it is evening. Maybe you feel like watching a comedy like "Intouchables" because it is summer?

7. CONCLUSIONS AND FUTURE WORK

We proposed an architecture to generate context-aware recommendations along with accompanying rationales to help the user choose the most interesting item. In CARE (Context-Aware Recommender with Explanation), the recommendation process is performed hierarchically, and with transparency at each abstraction level so as to produce detailed explanations for the suggested choices. Our approach promotes a diversity of recommendation results since each piece of contextual information is analyzed separately, and the most appropriate items are recommended with a rationale accompanying each suggestion.

Our architecture enables one to implement a flexible and generic recommender systems that can easily be extended with new independent modules in a hierarchical structure. In the current stage of our research, we have tested the performance of the CARE prototype on a movie-ratings dataset. Following the suggestions of a domain expert, we plan to extend the system with modules that incorporate other diverse movie features such as the director, cast, atmosphere and so on. We also plan to follow the Interaction Design principles during the evaluation of our system and will perform user testing with a working system. In future work, we will also address the evaluation of results quality with standard methods such as RMSE or nDCG.

Our approach is applicable to other domains as well. Currently we are working on adapting this architecture for supporting legal decision making, and moral and ethical reasoning.

8. REFERENCES

[1] Pandora. http://www.pandora.com, 2006.

[2] Netflix dataset. http://www.netflixprize.com/, 2009.

[3] G. Adomavicius and A. Tuzhilin. Context-aware recommender systems. In *Proceedings of the 2008 ACM Conference on Recommender Systems*, RecSys '08, pages 335–336, New York, NY, USA, 2008. ACM.

[4] G. Adomavicius and A. Tuzhilin. Context-aware recommender systems. In *Recommender Systems Handbook*, pages 217–253. 2011.

[5] M. Balabanović and Y. Shoham. Fab: Content-based, collaborative recommendation. *Commun. ACM*, 40(3):66–72, Mar. 1997.

[6] L. Baltrunas and F. Ricci. Context-based splitting of item ratings in collaborative filtering. In *Proceedings of the Third ACM Conference on Recommender Systems*, RecSys '09, pages 245–248, New York, NY, USA, 2009. ACM.

[7] L. Baltrunas and F. Ricci. Experimental evaluation of context-dependent collaborative filtering using item splitting. 24(1-2):7–34, 2014.

[8] D. Bridge, M. H. G oker, L. McGinty, and B. Smyth. Case-based recommender systems. *The Knowledge Engineering Review*, 20:315–320, 9 2005.

[9] R. Burke. Hybrid recommender systems: Survey and experiments. *User Modeling and User-Adapted Interaction*, 12(4):331–370, Nov. 2002.

[10] A. Cooper, R. Reimann, and D. Cronin. *About Face: The Essentials of Interaction Design*. John Wiley & Sons, Inc., New York, NY, USA, 2014.

[11] D. D. Corkill. Blackboard systems. *AI Expert*, 6, 1991.

[12] A. H. Dong, D. Shan, Z. Ruan, L. Zhou, and F. Zuo. The design and implementation of an intelligent apparel recommend expert system.

[13] B. S. Francesco Ricci, Lior Rokach. Introduction to recommender systems handbook. In F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, editors, *Recommender Systems Handbook*, pages 1–35. Springer US, 2011.

[14] G. Guo, J. Zhang, and N. Yorke-Smith. A novel bayesian similarity measure for recommender systems. In *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence*, IJCAI '13, pages 2619–2625. AAAI Press, 2013.

[15] A. Karatzoglou, X. Amatriain, L. Baltrunas, and N. Oliver. Multiverse recommendation: n-dimensional tensor factorization for context-aware collaborative filtering. In *Proceedings of the fourth ACM conference on Recommender systems*, pages 79–86. ACM, 2010.

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

[17] P. Lops, M. De Gemmis, and G. Semeraro. Content-based recommender systems: State of the art and trends. In *Recommender systems handbook*, pages 73–105. Springer US, 2011.

[18] A. Odić, M. Tkalčič, J. F. Tasič, and A. Košir. In G. Adomavicius, editor, *Proceedings of the 4th Workshop on Context-Aware Recommender Systems in conjunction with the 6th ACM Conference on Recommender Systems (RecSys 2012)*, volume 889, 2012.

[19] L. Qin and X. Zhu. Promoting diversity in recommendation by entropy regularizer. In *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence*, IJCAI '13, pages 2698–2704. AAAI Press, 2013.

[20] F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, editors. *Recommender Systems Handbook*. Springer, 2011.

[21] A. Ruiz-Iniesta, G. Jiménez-Díaz, M. Gómez-Albarrán, et al. A framework for the rapid prototyping of knowledgebased recommender systems in the learning domain. *Journal of Research and Practice in Information Technology*, 44(2):167, 2012.

[22] J. Sabater, M. Singh, and J. M. Vidal. A Protocol for a Distributed Recommender System, 2005.

[23] A. Said, S. Berkovsky, and E. W. De Luca. Introduction to special section on camra2010: Movie recommendation in context. *ACM Trans. Intell. Syst. Technol.*, 4(1):13:1–13:9, Feb. 2013.

[24] J. Schafer, D. Frankowski, J. Herlocker, and S. Sen. Collaborative filtering recommender systems. In P. Brusilovsky, A. Kobsa, and W. Nejdl, editors, *The Adaptive Web*, volume 4321 of *Lecture Notes in Computer Science*, pages 291–324. Springer Berlin Heidelberg, 2007.

[25] N. Tintarev and J. Masthoff. Designing and evaluating explanations for recommender systems. In F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, editors, *Recommender Systems Handbook*, pages 479–510. Springer US, 2011.

[26] K. Verbert, D. Parra-Santander, P. Brusilovsky, and E. Duval. Visualizing recommendations to support exploration, transparency and controllability. In *Proceedings of the 2013 international conference on Intelligent user interfaces - IUI '13*, page 351. ACM Press, 2013.

[27] Y. ZHANG, H. Liu, and S. Li. A Distributed Collaborative Filtering Recommendation Mechanism for Mobile Commerce Based on Cloud Computing. 2011.

[28] Y. Zheng, B. Mobasher, and R. D. Burke. The role of emotions in context-aware recommendation. In L. Chen, M. de Gemmis, A. Felfernig, P. Lops, F. Ricci, G. Semeraro, and M. C. Willemsen, editors, *Decisions@RecSys*, volume 1050 of *CEUR Workshop Proceedings*, pages 21–28. CEUR-WS.org, 2013.

[29] C.-N. Ziegler, S. M. McNee, J. A. Konstan, and G. Lausen. Improving recommendation lists through topic diversification. In *Proceedings of the 14th International Conference on World Wide Web*, WWW '05, pages 22–32, New York, NY, USA, 2005. ACM.

Inspection Mechanisms for Community-based Content Discovery in Microblogs

Nava Tintarev
University of Aberdeen
Aberdeen, UK
nava.tintarev@gmail.com

Byungkyu Kang
Dept. of Computer Science
University of California
Santa Barbara, USA
bkang@cs.ucsb.edu

Tobias Höllerer
Dept. of Computer Science
University of California
Santa Barbara, USA
holl@cs.ucsb.edu

John O'Donovan
Dept. of Computer Science
University of California
Santa Barbara, USA
jod@cs.ucsb.edu

ABSTRACT
This paper presents a formative evaluation of an interface for inspecting microblog content. This novel interface introduces filters by communities, and network structure, as well as ranking of tweets. It aims to improving content discovery, while maintaining content relevance and sense of user control. Participants in the US and the UK interacted with the interface in semi-structured interviews. In two iterations of the same study (n=4, n=8), we found that the interface gave users a sense of control. Users asked for an active selection of communities, and a more fine-grained functionality for saving individual 'favorite' users. Users also highlighted unanticipated uses of the interface such as iteratively discovering new communities to follow, and organizing events. Informed by these studies, we propose improvements and a mock-up for an interface to be used for future larger scale experiments for exploring microblog content.

Author Keywords
Microblogs, visualization, communities, explanations, interfaces, content discovery

ACM Classification Keywords
H.5.m. Information Interfaces and Presentation (e.g. HCI): Miscellaneous

INTRODUCTION
Filtering of streaming data such as microblog content is inevitable, even if it is done by showing the most recent content as restricted by screen-size. However our live timelines do often get tailored to us, without transparency or a sense of control. Getting the selection of the content right is a delicate matter.

Recommender systems address the challenges of finding 'hidden gems' which are tailored to individuals from a very wide selection. Implemented well, they hold the key to helping users discover items that are *both* unexpected and relevant, while helping catalog holders sell a wider range of items [3].

In trying to help users make such discoveries, recommender systems walk a thin line between a) making unexpected but risky recommendations (increasing the chances of irrelevant recommendations), and on the other hand b) over-tailoring (resulting in unsurprising recommendations). Over-tailoring can also result in filter bubbles [15], whereby users do not get exposed to items outside their existing interests. For current events, such as content in microblogs, personalization algorithms may narrow what we know, and surround us with information that supports what we already believe. This can result in polarization of views, especially as we have a tendency to self-filter [2].

This paper address these issues by supporting controlled filtering of microblog content. It introduces a novel visualization which supports filtering by allowing a user to control: a) which communities influence their feed b) the network structure relating to these communities and c) different ways of ranking tweets. This visualization is evaluated in two iterations of a qualitative study that assesses the value of such controls, as well as the concrete implementation choices applied. We also discuss the ways these filters and controls are perceived by users, and how they envision that they would use them. We conclude with describing our next steps.

BACKGROUND

Inspectability and Control in Recommender Systems
In the domain of recommender systems there is a growing acceptance and interest in user-centered evaluations [12]. For example, [9] argues for a framework that takes a user-centric approach to recommender system evaluation, beyond the scope of recommendation accuracy. Along the same vein, it has also been recognized that many recommender systems function as *black boxes*, providing no transparency into the

working of the recommendation process, nor offering any additional information to accompany the recommendations beyond the recommendations themselves [6].

To address this issue, explanations can be given to improve the transparency and control of recommender systems. Research on textual explanations in recommender systems to date has been evaluated in wide range of domains (varying from movies [18] to financial advice [4]). Increasingly, there has also been a blurring between recommendation and search, making use of information visualization. For example, [19] has looked at how interaction visualization can be used to improve the effectiveness and probability of item selection when users are able to explore and interrelate multiple entities – i.e. items bookmarked by users, recommendations and tags. Similarly [16] found that in addition to receiving transparent and accurate item recommendations, users gained information about their peers, and about the underlying algorithm through interaction with a network visualization.

Inspectability and Control in Microblogs

In order to better deal with the vast amounts of user-generated content in microblogs, a number of recommender systems researchers have studied user experiences through systems that provide transparency of and control over recommendation algorithms. Due to the brevity of microblog messages, many systems provide summary of events or trending topics with detailed explanations [11]. This unique aspect of microblogs makes both inspectability and control of recommender algorithms particularly important, since they help users to more efficiently and effectively deal with fine-grained data. For example, experimental evidence to argue that inspectability and control improve recommendation systems is presented for microblogs in [16], via a commuter traffic analysis experiment, and more generally in [8] using music preference data in their *TasteWeights* system.

Community-based Content Discovery

Serendipity is defined as the act of unexpectedly encountering something fortunate. In the domain of recommender systems, one definition has been the extent to which recommended items are both useful and surprising to a user [7]. This paper investigates how exploration can be supported in a way that improves serendipity.

The intuitions guiding the studies in this paper are based on findings in the area of social recommendations, that is based on people's relationships in online social networks (e.g., [13]) in addition to more classical recommendation algorithms.

The *first intuition* is that weak rather than strong ties are important for content discovery. This intuition is informed by the findings of the cohesive power of weak ties in social networks, and that some information producers are more influential than others in terms of bridging communities and content [5]. Results in the area of social-based explanations also suggest that mentioning which friend(s) influence a recommendation can be beneficial (e.g, [17, 20]). In this case, we support exploring immediate connections or friends, as well as friends-of-friends.

The *second intuition* is that the intersection of groups may be particularly fortuitous for the discovery of new content. This is informed by exploitation of cross-domain model inspiration as a means for serendipitous recommendations, e.g., [1].

VISUALIZATION

In this study, we designed a web-based visualization that allows users to experience the recommender system we propose (see Figure 1). The first two columns represent "groups" (communities) and "people" (users), allow us to filter 'tweets' in the third column by both of these 'facets'. The system supports therefore support a faceted navigation, with the third column representing the resulting information. In addition, the system supports Pivoting (or set-oriented browsing), in that it allows users to navigate the search space by starting from a set of instances (by selecting which groups they would like to follow).

The rational for the visualization follows several intuitions with regards to exploring novel and relevant content in social network, as outlined in the section in related work.

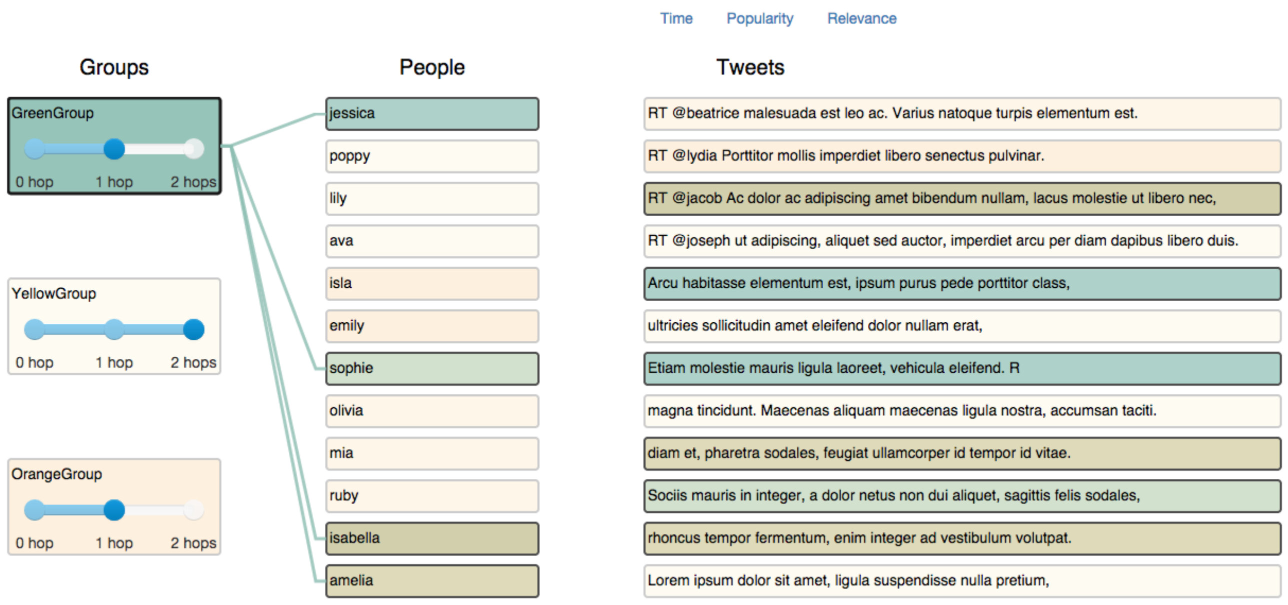
The first is that people can find relevant content in the intersection between multiple communities. In the visualization this is represented by the selection of up to three communities to which a user belongs, and color blending to indicate people and content that represents this type of overlap. Another intuition is that weak ties, or friends of friends, are also good candidates for content discovery. In this visualization they are represented as two hops in a network structure. Consequently we included a slider which included 0-hops (do not consider this community), 1-hop (include people who follow a given community), 2-hops (include people who follow people in a given community).

Finally, the ranking of tweets according to a) relevance to a user compared to b) popularity and c) time is also likely to help users find relevant and unexpected content compared to tweets only ordered by time.

Structure and Interaction

Figure 3 shows a snapshot of the interactive visualization used in the study. Information is presented in three columns. From left to right, these are: group/community, people and tweet columns. Users can interact with entities in any of these three columns to highlight associations to entities in other columns. In the people and tweet columns, entities are clustered and colored based on community associations. In the first column, we visualize a set of communities (also referred to as groups), which by design, may have some membership and content overlapping. Within this column, each entity has a widget to control network distance from that entity. This enables the user to specify how that entity contributes users and content to the other columns. In particular, sliders were used for control in Study 1 and radio buttons in Study 2.

In the second column, a ranked list of users related to each community is visualized. These users serve as sources for information recommened in the third column, but the visualization also supports analysis of the connectivity of these users across communities in addition to the content they distribute.



RT @beatrice malesuada est leo ac. Varius natoque turpis elementum est.

RT @lydia Porttitor mollis imperdiet libero senectus pulvinar.

RT @jacob Ac dolor ac adipiscing amet bibendum nullam, lacus molestie ut libero nec,

RT @joseph ut adipiscing, aliquet sed auctor, imperdiet arcu per diam dapibus libero duis.

Arcu habitasse elementum est, ipsum purus pede porttitor class,

ultricies sollicitudin amet eleifend dolor nullam erat,

Etiam molestie mauris ligula laoreet, vehicula eleifend. R

magna tincidunt. Maecenas aliquam maecenas ligula nostra, accumsan taciti.

diam et, pharetra sodales, feugiat ullamcorper id tempor id vitae.

Sociis mauris in integer, a dolor netus non dui aliquet, sagittis felis sodales,

rhoncus tempor fermentum, enim integer ad vestibulum volutpat.

Lorem ipsum dolor sit amet, ligula suspendisse nulla pretium,

Figure 1. Visualization of the recommendation system used in the study 1.

The third column shows the recommended tweets which are by default filtered and ordered according to recency. A user can change the ranking algorithm for this column to either popularity or relevance.

Color Scheme

Selecting appropriate color scheme is one of the important aspects to consider in user interface design. We examined different sets of colors and carefully selected three major colors that represent each group on the first column. They have been selected among the most popular color palettes on Adobe Color website[1]. These colors are tested under grayscale condition.

Materials

The materials for the experiment were abstracted: people were given random names of both genders, tweets were short lines from a short Latin text ("Lorem Ipsum…"), resulting in a total of 229 tweets. When participants interacted with the system, a random subset of 12 tweets was presented. The top 4 of these tweets included a retweet, to visually increase the similarity with a twitter feed, and was applied consistently across adaptations.

STUDY 1

This section describes a formative study conducted to evaluate the proposed visualization. We used a layered evaluation approach [14], focusing on the decision of an adaptation and how it was applied (in contrast to which data was collected or how it was analyzed). Participants took part in semi-structured interviews, in order to evaluate the user experience (following the guiding scenarios of [10]). More concretely this study aimed to answer the following questions: a) are the three introduced controls (selection of communities, network structure, and ranking of tweets) considered useful for participants? b) is the way they are implemented useful? c) do these controls give users a sense of control? d) do participants use the controls in the way that we envisaged? The version of the system used for this study can be found online[2].

Participants

4 participants were recruited from research staff at computer science department at a UK university. Their ages ranged from 23-51. They all had twitter accounts, but their experience with twitter ranged from inactive to highly experienced (including the use of twitter management and analytics applications). 1 was female, and 3 male. They all had a native or fluent level of English language skills. Participants varied from PhD students, post-doctoral fellows to teaching staff.

One of the participants had done research with visualizations and twitter, the other three had no experience with either. None knew Latin (one had taken Latin course, but professed a very rudimentary level of knowledge).

Procedure

Participants took part in individual semi-structured interviews, following a user test plan[3]. Following the collection of basic demographic data, participants were given a brief introduction to the system. The various interface components were verbally introduced without interacting with the system. Participants were then given several simple tasks such as including people who are connected to other people for a given community, or ranking tweets by relevance (rather than time). Following each interaction participants were asked how the tweets had changed, if new ones had been added, or if tweets had disappeared. The tasks given were:

- Go to the system online. What are your first impressions?

[1] `https://color.adobe.com/explore/most-popular/?time=all`

[2] http://goo.gl/krOvuJ

[3] `https://goo.gl/3KpH9z`

- Select one of three communities that you are a member of and reflect your interests (if user can not think of any tell them to think of conferences that they attend). Have a look at the tweets that are recommended to you.
- Add tweets (1 hop) for a second community of your choice from the above.
- Is there any relevant tweet from this second community you did not see before? Are there any that have disappeared?
- The tweets are currently ranked by time, change this to rank the tweets by popularity.
- Are there now any tweets you did not see before? Are there any that have disappeared?
- Now, change who you get your tweets from to include people who are linked to (2 hops) people that attend your first community. You may want to remove the second community for this too.
- How about now, are there now any tweets you did not see before? Are there any that have disappeared?

Following the interaction with the system, participants took part in an exit interview where they were asked about their perceived control of the system, the usefulness of various functionalities, and how they would use them for exploration. More concretely the questions asked included

- How did it feel? What was your impression? (Positive impressions? Negative impressions?)
- Would you have liked more training on how to interact with the visualization before you got started?
- How helpful did you find the following functionalities (1-7, unhelpful to helpful), and how could they be improved?
  - Tweets organized by community;
  - Changing how the tweets are ordered/ranked
  - Changing who I get tweets from (0,1,2 hops)
  - Being able to interact with the system to specify different preferences
  - The links between different parts of the interface (people, groups, tweets).
- Do you think these functionalities would help you find new and relevant information you would not find otherwise? How would you use them to do this?
- Does the filtering give you a sense of what you might be missing, or does it hide information that you need?
- Did you feel like you had control over which information was presented to you?
- Would you liked to have had any controls that are not present in this interface?

Results

Are the introduced controls (communities, network structure, and ranking of tweets) considered useful by participants?

The scores given to the various controls was generally high (5 or above). There were three exceptions. Participant3 did not find tweet ranking by relevance and popularity useful at all. Participant4 gave low scores to the hop control for network structure, and the links, but this was due to the way they were implemented, and is discussed below.

Is the way they are implemented useful?

All the participants noted that the interface was simple and clean, and had a good first impression. Participant4 noted that it would be well suited for a mobile interface.

- **Hop control** All of the participants found it difficult to understand the control for the network structure. When thinking aloud, several said that pulling the slider further to the right would increase the number of tweets on a certain topic, rather than widen the network (which potentially would dilute the focus of the tweets).
- **Community selection** Participant1 wanted to 'activate' a community by selecting its box. This seems more intuitive than selecting 0 hops for the communities they did not want to follow.
- **People** In addition to filtering on community structure and inclusion, several participants wanted a finer grain control of which users were included in the selection of tweets. Some users wanted to activate users somehow, by either adding them to favorites at the top of the person list, or activating through selection. These participants felt that this should influence the ranking of tweets.
- **Tweets** Participants felt that tweets belonging to the same community should not only have the same color, but be grouped together. Participant3 (experienced twitter user) felt that ranking of tweets by any other measure than recency (time) was not useful.
- **Links** Participant3 found the links and colors between the columns inconsistent. The relationship between the first two columns used links, whereas the relationship between the second two columns used colors.
- **Color-interleaving** Participant1 mistook the color-interleaving to imply significance, as they varied in hue. However, the other participants interpreted this correctly although did ask if the interpretation was correct.

Do these controls give users a sense of control?

All of the participants felt that the interface improved their control over their tweets. They also consistently agreed that they would be missing some content, and that they were not in complete control, but that they were happy with the balance in the trade-off.

However, Participant3 felt that they wanted to be able to scroll through all of their tweets, especially because they did not have the finer grained control of which individuals appeared in their feed.

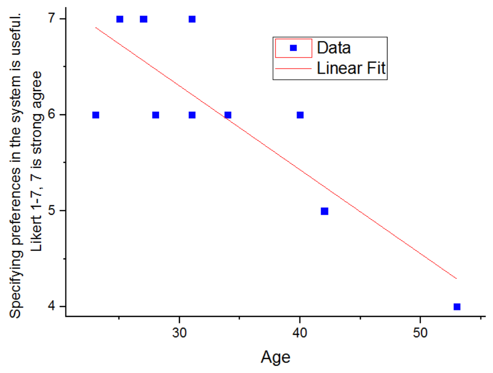


Figure 2. Plot showing correlation between participant age and reported importance of "Being able to interact with the system to specify different preferences".

Do participants use the controls in the way that we envisaged?
All of the participants completed the simple tasks given to them. They all stated that they would find new and relevant content using the interface, although the highly experience twitter user felt they already find novel content using tools such as TweetDeck. When asked how they would you use the functionalities to find new and relevant information, participants suggested two uses we had not initially considered:

Organizing events Participant3 felt that the groups could be defined by other characteristics rather than membership of a community, such as geographic location. This participant suggested that they would use this functionality to identify and coordinate groups of people when organizing events on the topics they were interested in.

Discover new groups Participant2 was confident that they would find new relevant communities when looking at the intersection of existing communities that they follow. This participant listed three music bands that they listen to and would follow on twitter. They would use the system to discover new bands, and would then add them as a new group as a "seed" for further discovery.

Other suggestions
Participants suggested several features they would expect in an interface that was integrated with twitter. For example, they would want to be able to view the profiles (or at least, the first 50 characters) of the people they are receiving tweets from. Others wanted to be able to reply to tweets directly from the feed. Another suggestion was to introduce separate columns for different communities. This may be related to the request by other users to be able to group tweets by community.

STUDY 2
The first study identified several limitations of the system, which were addressed for a second iteration of evaluation. Improvements included: a) using buttons rather than a slider to control the number of hops; b) sorting people by group affinity, e.g. greenGroup people were listed at the top, rather than mixed throughout the list; c) identifying how many people were filtered (i.e. "Showing 12 of 1307"). The improved interface can be seen in Figure 3, with annotations to highlight each improvement. The version of the system used for this study can be found online[4].

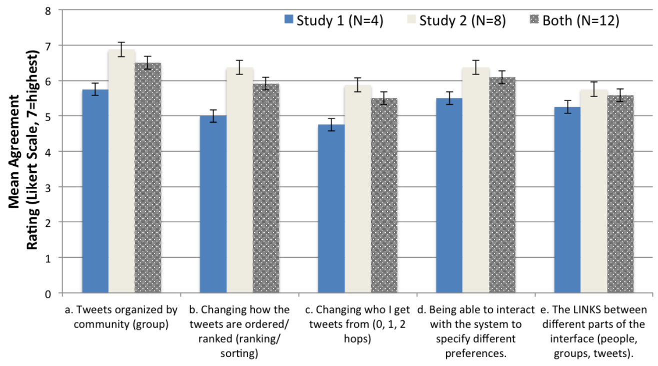


Figure 4. Analysis of subjective results in exit interviews for the two studies. Error bars show standard error.

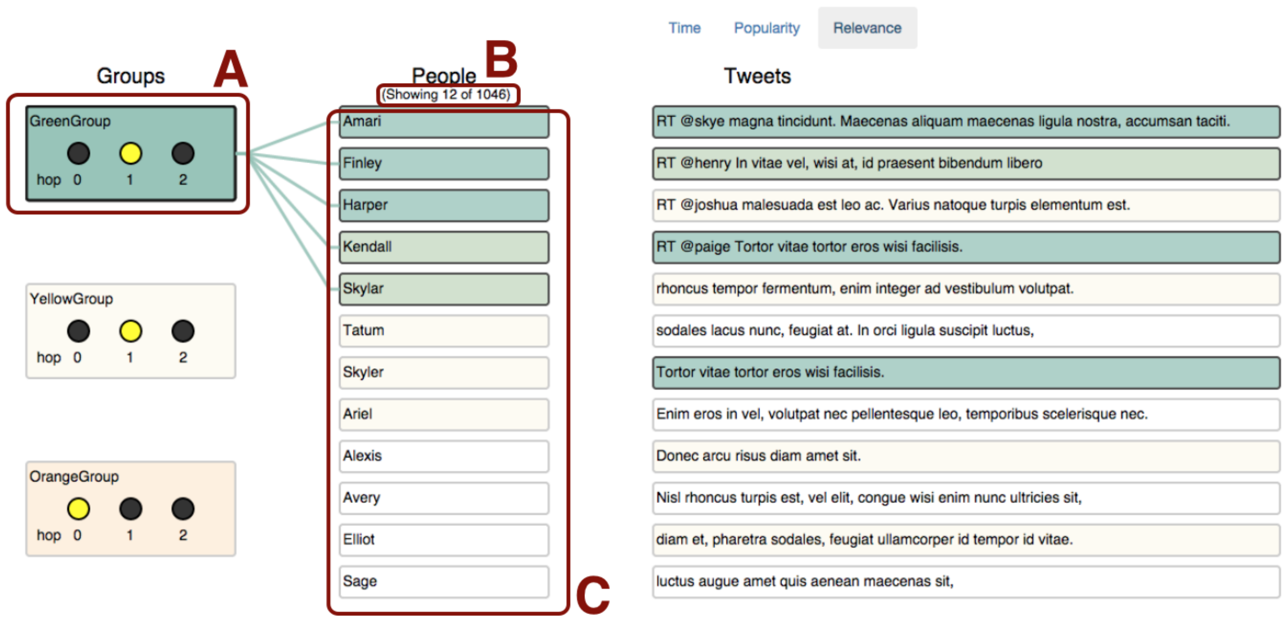
Participants
8 participants were recruited from research staff at computer science department at a US university. Their ages ranged from 20 to 45. 5 participants were female and 3 were male. Participants varied from PhD students, post-doctoral fellows to teaching staff in computer science, engineering, media-arts and physics. They all had a native or fluent level of English language skills. 6 of the participants had Twitter accounts, and one person had done research with Twitter data in the past. 5 had done research with visualization. As with Study1, no participants knew Latin.

Procedure
As in Study1, participants took part in individual semi-structured interviews. Studies were conducted in a computer science lab on campus using two notebook computers. The participant interacted with the UI on one, and the experimenter/interviewer took notes on the other. On average, studies lasted 35 minutes (min 28 minutes, max 43 minutes).

Results
In this section, we revisit questions from Study1 and add additional comments and discussion based on the new participants interacting with the improved UI in Study2. Figure 4 shows a comparison of participants' opinions on the different features of the system between Study1 (N=4) and Study2 (N=8), along with the combined score (N=12). We note that the combined score is based on two slightly different UI designs, and it is only used as a rough estimate of the overall group evaluation.

Are the introduced controls (communities, network structure, and ranking of tweets) considered useful by participants?
The scores shown in Figure 4 range between 5.58 and 6.87 for Study2, shown in the middle column of each group, an average of approximately one point on the 7 point scale. Compared to Study1, the interface modifications appear to have had a positive impact on user experience with the system.

[4]`http://penguinkang.com/intRS/`



RT @skye magna tincidunt. Maecenas aliquam maecenas ligula nostra, accumsan taciti.
RT @henry In vitae vel, wisi at, id praesent bibendum libero
RT @joshua malesuada est leo ac. Varius natoque turpis elementum est.
RT @paige Tortor vitae tortor eros wisi facilisis.
rhoncus tempor fermentum, enim integer ad vestibulum volutpat.
sodales lacus nunc, feugiat at. In orci ligula suscipit luctus,
Tortor vitae tortor eros wisi facilisis.
Enim eros in vel, volutpat nec pellentesque leo, temporibus scelerisque nec.
Donec arcu risus diam amet sit.
Nisl rhoncus turpis est, vel elit, congue wisi enim nunc ultricies sit,
diam et, pharetra sodales, feugiat ullamcorper id tempor id vitae.
luctus augue amet quis aenean maecenas sit,

Figure 3. Improved visualization design used in the main user study. Annotation (A) shows changes to the number-of-hops selection. (B) shows the number of filtered users interactively in the form "m of n", and (C) shows connectivity-based clustering and associated coloring of nodes in the "People" column.

While this is a promising side result, the purpose of the study was to provide a formative evaluation of the interface.

Participants reported the best score for the feature to organize Tweets by community, which is a core contribution of the system. This is encouraging feedback as the authors are designing a larger-scale quantitative evaluation with this as a central feature. The features that elicited the lowest scores were the hop-distance selector and the edge visualizations between the columns.

Participants also reported that they liked the ability to change how Tweets were ordered and ranked through the interface. One participants commented that "I can't do this in Facebook or Twitter – this is great!". Support for expressing real-time preferences through interactive interface components met with strong positive feedback, with all users reporting a sense of increased control over the information feed.

Is the way they are implemented useful?

Similarly to Study1 study, all participants commented that the interface was clean and well organized. One participant complained that it was too complex and could benefit from having less data. 50% of the participants pointed out an issue with the node-coloring in column 2, shown in Figure 3. Note that this figure needs to be viewed in color to see the true effect (see link to system above).

- **Hop control** Some participants did not realize that the 0 position essentially turned the group node off. There were also multiple comments that when hop control was set to 0, showing the nodes opaquely was not a good design choice. One participant explicitly mentioned that it would be better to remove these nodes completely, noting that the visual effect of setting the hop-control to 0 would be much shorter. Unlike Study1, no participants confused the hop slider with a weighting mechanism, and all understood that it sourced users from n-hops farther away in the Twitter network.
- **Community selection** Most participants commented that community selection and analysis was a strong point of the system. Suggested communities included musical artists, pet fan clubs, and conferences or meetings.
- **People** A few participants reported having trouble understanding the coloring and community-based grouping/clustering in this column. All participants understood the data flow correctly by the end of the sessions, but this feature took longer than others for them to master. The main cited reason for this was that the colors – added to distinguish the groups, were too similar, as mentioned above. Two participants mentioned that it would be useful to select or weight people of interest.
- **Tweets** Two participants suggested that a ranking score would be useful to distinguish between tweets in the right column. Participants also requested that when a change is made in the system, the source of that change's effect on the list should be visualized. Our proposed solution to this is shown in Figure 5 as a ranking source indicator for each tweet.
- **Links** Participants were slightly dissatisfied with how links were shown in the system. Three people commented that links should be shown across all columns when a particular group is selected in the left column, or when any other node is selected, to visually communicate the associations of that node. Other participants commented that the on-demand design was a good idea to avoid cluttering the view.
- **Color-interleaving** Half of the users complained that this was too subtle and needed to be made more explicit. This has been addressed through the use of colored icons next

to people to signal group memberships. The color palette has also been changed to make clearer distinction between groups.

Do these features give users a sense of control?

In keeping with Study1, all of the participants felt that the interface improved their control over their tweets. They also consistently agreed that they would be missing some content, and that they were not in complete control, but that they were happy with the balance in the trade-off. Similar to the Study1, two participants suggested use of scrolling or similar mechanism to view filtered-out tweets in case they wanted to.

Do participants use the features in the way that we envisaged?

Generally, participants reported that they would find the system useful for discovering new content and exploring community structure in the domains that they chose (music, conferences, pet fan clubs etc.). In particular, they felt that real-time preference feedback, community selection and algorithm selection (time, relevance or popularity) gave them a good sense of control. Many commented that such features would be useful on everyday social media streams such as in Twitter and Facebook.

Participants suggested similar uses of the controls as in Study1. Many suggested using the system for organizing events and advertising across relevant communities, and for discovering new groups. Echoing the comments of Study1, one participant mentioned that they would like to use the system for exploring a broader network of musical artists. They described selection of three fan club communities as in our experimental setup, but went on to describe iteratively replacing them with new nodes that were discovered on the right column, thereby applying the interface (theoretically) as a network traversal and discovery tool. This is an example of a reported use that was not in our design. Another participant proposed to use the system to analyze which community produced the most popular content on Twitter, by using the popularity ranking algorithm and traversing the edge connections back to the groups.

Other suggestions

Participants suggested a variety of ways to improve the interface. These included addition of multimedia content to the tweet column, and visually distinguishing retweets (compared to original tweets) by color. Participants also suggested creating visually distinct colorings for blended color groups, and displaying links to all group memberships upon clicking a user node (rather than upon hover). Another request was for an indication of how much data has been filtered in all the columns (currently only for the people column). Participants also suggested measuring the usefulness of the system for getting an overview of a new community or topic. Several comments, including from reviewers, focused on the group selection widget. In the current version, a group is activated by clicking on the box that represents the group, then the radio buttons within it are used to control the number of hops that feed to the people column from that group. Other possibilities that are being considered for activation of group nodes are a) a simple check box and b) extending the radio button selection to include an option for 0-hops, thereby disabling the node.

Demographics Analysis

A brief analysis of demographics and responses showed an interesting correlation between participant age and the perceived importance of specifying preferences on-the-fly in the user interface. Figure 2 shows a plot with the Likert-scale responses for the dynamic preferences shown on the Y-axis and participant age shown on the X-axis. The data follows a negative linear trend, with younger participants specifying a higher perceived importance of specifying preferences.

CONCLUSION AND FUTURE WORK

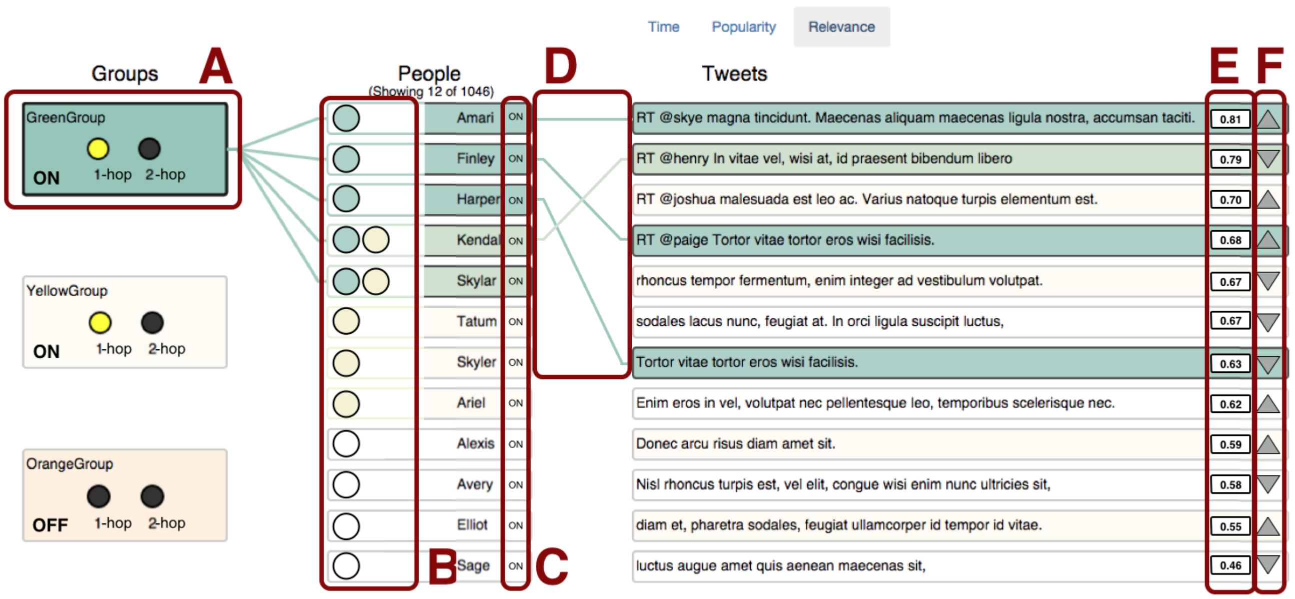
In this paper we evaluated a visualization which allowed users to explore and filter microblog content for communities to which they belong. The ability to organize Tweets by community, the core contribution of the visualization, was rated the most highly. Users also stated that the interface gave them enough control over their content, even if they felt some information would inevitably be hidden – the trade-off was considered acceptable. We also found several unexpected uses of the system. For example two separate participants, in different experimental settings (one in the UK and one in the US) applied the interface (theoretically) as a network traversal and discovery tool for music. Figure 5 introduces an improved mock-up with a number of changes. In addition to these improvements, we are planning larger-scale quantitative evaluations. One of these will explore the use of community-based filters, and the other controls introduced in this paper, on existing twitter feeds.

ACKNOWLEDGMENTS

This research has been carried out within the project Scrutable Autonomous Systems (SAsSY), funded by the UK Engineering and Physical Sciences Research Council, grant ref. EP/J012084/1. This work was also partially supported by the U.S. Army Research Laboratory under Cooperative Agreement No. W911NF-09-2-0053; The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of ARL, NSF, or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

REFERENCES

1. André, P., m.c. schraefel, Teevan, J., and Dumais, S. T. Discovery is never by chance: Designing for (un)serendipity. In *CC* (2009).
2. Bakshy, E., Messing, S., and Adamic, L. A. Exposure to ideologically diverse news and opinion on facebook. *Science 348* (2015), 1130–1132.
3. Castells, P., Hurley, N., and Vargas, S. *Recommender Systems Handbook (second ed)*. (in press), ch. Novelty and Diversity in Recommender Systems.
4. Felfernig, A., Teppan, E., and Gula, B. Knowledge-based recommender technologies for marketing and sales. *Int. J. Patt. Recogn. Artif. Intell. 21* (2007), 333–355.
5. Granovetter, M. S. The Strength of Weak Ties. *The American Journal of Sociology 78*, 6 (1973), 1360–1380.



RT @skye magna tincidunt. Maecenas aliquam maecenas ligula nostra, accumsan taciti. 0.81

RT @henry In vitae vel, wisi at, id praesent bibendum libero 0.79

RT @joshua malesuada est leo ac. Varius natoque turpis elementum est. 0.70

RT @paige Tortor vitae tortor eros wisi facilisis. 0.68

rhoncus tempor fermentum, enim integer ad vestibulum volutpat. 0.67

sodales lacus nunc, feugiat at. In orci ligula suscipit luctus, 0.67

Tortor vitae tortor eros wisi facilisis. 0.63

Enim eros in vel, volutpat nec pellentesque leo, temporibus scelerisque nec. 0.62

Donec arcu risus diam amet sit. 0.59

Nisl rhoncus turpis est, vel elit, congue wisi enim nunc ultricies sit, 0.58

diam et, pharetra sodales, feugiat ullamcorper id tempor id vitae. 0.55

luctus augue amet quis aenean maecenas sit, 0.46

Figure 5. Mock-up of improved UI and interaction design based on study results and analysis: (A) improved representation of the hop-distance controls, (B) iconization to show group memberships, (C) Activation (on/off) control of nodes, (D) visualization of dynamic edges, (E) addition of a ranking score for recommended content, and (F), addition of a provenance arrow to show what the previous interaction did to the ranking of each recommendation.

6. Herlocker, J. L., Konstan, J. A., and Riedl, J. Explaining collaborative filtering recommendations. In *ACM conference on Computer supported cooperative work* (2000), 241–250.

7. Herlocker, J. L., Konstan, J. A., Terveen, L., and Riedl, J. T. Evaluating collaborative filtering recommender systems. *ACM Trans. Inf. Syst. 22*, 1 (2004), 5–53.

8. Knijnenburg, B. P., Bostandjiev, S., O'Donovan, J., and Kobsa, A. Inspectability and control in social recommenders. In *Proceedings of the Sixth ACM Conference on Recommender Systems*, RecSys '12, ACM (New York, NY, USA, 2012), 43–50.

9. Knijnenburg, B. P., Willemsen, M. C., Gantner, Z., Soncu, H., and Newell, C. Explaining the user experience of recommender systems. *User Modeling and User-Adapted Interaction 22*, 4-5 (2012), 441–504.

10. Lam, H., Bertini, E., Isenberg, P., Plaisant, C., and Carpendale, S. Empirical studies in information visualization: Seven scenarios. *IEEE Transactions on Visualization and Computer Graphic 18(9)* (2012), 1520–1536.

11. Marcus, A., Bernstein, M. S., Badar, O., Karger, D. R., Madden, S., and Miller, R. C. Twitinfo: Aggregating and visualizing microblogs for event exploration. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '11, ACM (New York, NY, USA, 2011), 227–236.

12. McNee, S. M., Riedl, J., and Konstan, J. A. Being accurate is not enough: How accuracy metrics have hurt recommender systems. In *Extended Abstracts of the 2006 ACM Conference on Human Factors in Computing Systems (CHI 2006)* (2006).

13. Nagulendra, S., and Vassileva, J. Providing awareness, understanding and control of personalized stream filtering in a p2p social network. In *Conference on Collaboration and Technology (CRIWG)* (2013).

14. Paramythis, A., Weibelzahl, S., and Masthoff, J. Layered evaluation of interactive adaptive systems: Framework and formative methods. *User Modeling and User-Adapted Interaction 20* (2010).

15. Pariser, E. *The filter bubble: What the Internet is hiding from you*. Penguin Books, 2011.

16. Schaffer, J., Giridhar, P., Jones, D., Höllerer, T., Abdelzaher, T., and O'Donovan, J. Getting the message?: A study of explanation interfaces for microblog data analysis. In *Proceedings of the 20th International Conference on Intelligent User Interfaces*, IUI '15, ACM (New York, NY, USA, 2015), 345–356.

17. Sharma, A., and Cosley, D. Do social explanations work? studying and modeling the effects of social explanations in recommender systems. In *World Wide Web (WWW)* (2013).

18. Tintarev, N., and Masthoff, J. Personalizing movie explanations using commercial meta-data. In *Adaptive Hypermedia* (2008).

19. Verbert, K., Parra, D., Brusilovsky, P., and Duval, E. Visualizing recommendations to support exploration, transparency and controllability. In *Proceedings of the 2013 International Conference on Intelligent User Interfaces*, IUI '13, ACM (New York, NY, USA, 2013), 351–362.

20. Wang, B., Ester, M., Bu, J., and Cai, D. Who also likes it? generating the most persuasive social explanations in recommender systems. In *Twenty-Eighth AAAI Conference on Artificial Intelligence* (2014).

uRank: Exploring Document Recommendations through an Interactive User-Driven Approach

Cecilia di Sciascio
Know-Center GmbH
Graz, Austria
cdisciascio@know-center.at

Vedran Sabol
Know-Center GmbH
Graz, Austria
vsabol@know-center.at

Eduardo Veas
Know-Center GmbH
Graz, Austria
eveas@know-center.at

ABSTRACT

Whenever we gather or organize knowledge, the task of searching inevitably takes precedence. As exploration unfolds, it becomes cumbersome to reorganize resources along new interests, as any new search brings new results. Despite huge advances in retrieval and recommender systems from the algorithmic point of view, many real-world interfaces have remained largely unchanged: results appear in an infinite list ordered by relevance with respect to the current query. We introduce *uRank*, a user-driven visual tool for exploration and discovery of textual document recommendations. It includes a view summarizing the content of the recommendation set, combined with interactive methods for understanding, refining and reorganizing documents on-the-fly as information needs evolve. We provide a formal experiment showing that *uRank* users can browse the document collection and efficiently gather items relevant to particular topics of interest with significantly lower cognitive load compared to traditional list-based representations.

General Terms

Theory

Keywords

recommending interface, exploratory search, visual analytics, sensemaking

1. INTRODUCTION

With the advent of electronic archival, seeking for information occupies a large portion of our daily productive time. Thus, the skill to find and organize the right information has become paramount. Exploratory search is part of a discovery process in which the user often becomes familiar with new terminology in order to filter out irrelevant content and spot potentially interesting items. For example, after inspecting a few documents related to robots, sub-topics like human-robot interaction or virtual environments could attract the user's attention. Exploration requires careful inspection of at least a few titles and abstracts, when not full documents, before becoming familiar with the underlying topic. Advanced search engines and recommender systems (RS) have grown as the preferred solution for contextualized search by narrowing down the number of entries that need to be explored at a time.

Traditional information retrieval (IR) systems strongly depend on precise user-generated queries that should be iteratively reformulated in order to express evolving information needs. However, formulating queries has proven to be more complicated for humans than plainly recognizing information visually [6]. Hence, the combination of IR with machine learning and HCI techniques led to a shift towards – mostly Web-based – browsing search strategies that rely on on-the-fly selections, navigation and trial-and-error [15]. As users manipulate data through visual elements, they are able to drill down and find patterns, relations or different levels of detail that would otherwise remain invisible to the bare eye [32]. Moreover, well-designed interactive interfaces can effectively address information overload issues that may arise due to limited attention span and human capacity to absorb information at once.

Sometimes RS can be more limited than IR systems if they do not tackle trust factors that may hinder user engagement in exploration. As Swearingen et al. [27] pointed out in their seminal work, the RS has to persuade the user to try the recommended items. To fulfill such challenge not only the recommendation algorithm has to fetch items effectively, but also the user interfaces must deliver recommendations in a way that they can be compared and explained [22]. The willingness to provide feedback is directly related to the overall perception and satisfaction the user has of the RS [13]. Explanatory interfaces increase confidence in the system (trust) by explaining how the system works (transparency) [28] and allowing users to tell the system when it is wrong (scrutability) [11]. Hence, to warrant increased user involvement the RS has to justify recommendations and let the user customize their generation.

In this work we focus mainly on transparency and controllability aspects and, to some extent, on predictability as well. *uRank* [1] is and interactive user-driven tool that supports exploration of textual document recommendations through:

i) an automatically generated overview of the document collection depicted as augmented keyword tags,

ii) a drag-and-drop-based mechanism for refining search interests, and

iii) a transparent stacked-bar representation to convey document ranking and scores, plus query term contribution. A user study revealed that *uRank* incurs in lower workload compared to a traditional list representation.

[1] `http://eexcessvideos.know-center.tugraz.at/urank-demo.mp4`

2. RELATED WORK

2.1 Search Result Visualization

Modern search interfaces assist user exploration in a variety of ways. For example, query expansion techniques like *Insyder*'s Visual Query [21] address the query formulation problem by leveraging stored related concepts to help the user extend the initial query. Tile-based visualizations like *TileBars* [7] and *HotMap* [9] make an efficient use of space to convey relative frequency of query terms through – gray or color – shaded squares, and in the case of the former, also their distribution within documents and relative document length. This paradigm aims to foster analytical understanding of Boolean-type queries, hence they do not yield any rank or relevance score. All these approaches rely on the user being able to express precise information needs and do not support browsing-based discovery within the already available results.

Faceted search interfaces allow for organizing or filtering items throughout orthogonal categories. Despite being particularly useful for inspecting enriched multimedia catalogs [33, 23], they require metadata categories and hardly support topic-wise exploration.

Rankings conveying document relevance have been discouraged as opaque an under-informative [7]. However, the advantage of ranked lists is that users know where to start their search for potentially relevant documents and that they employ a familiar format of presentation. A study [24] suggests that: *i)* users prefer bars over numbers or the absence of graphical explanations of relevance scores, and *ii)* relevance scores encourage users to explore beyond the first two results. As a tradeoff, lists imply a sequential search through consecutive items and only a small subset is visible at a given time, thus they are mostly apt for sets no larger than a few tens of documents. Focus+Context and Overview+Detail techniques [20, 9] sometimes help overcome this limitation while alternative layouts like *RankSpiral*'s [25] rolled list can scale up to hundreds and maybe thousands of documents. Other approaches such as *WebSearchViz* [16] and *ProjSnippet* [3] propose complementary visualizations to ordered lists, yet unintuitive context switching is a potential problem when analyzing different aspects of the same document.

Although ranked list are not a novelty, our approach attempts to leverage the advantages provided by lists; i.e. user familiarity, and augment them with stacked-bar charts to convey document relevance and query term contribution in a transparent manner. *Insyder*'s bar graph [21] is an example of augmented ranked lists that displays document an keyword relevance relevance with disjoint horizontal bars aligned to separate baselines. Although layered bar dispositions are appropriate for visualizing distribution of values in each category across items, comparison of overall quantities and the contribution of each category to the totals is better supported by stacked-bar configurations [26]. Additionally, we rely on interaction as the key to provide controllability over the ranking criteria and hence support browsing-based exploratory search.

LineUp [4] has proven the simplicity and usefulness of stacked bars to represent multi-attribute rankings. Despite targeting data of different nature – *uRanks*'s domain is rather unstructured with no measurable attributes –, the visual technique itself served as inspiration for our work.

2.2 Recommending Interfaces

In recent years, considerable efforts have been invested into leveraging the power of social RS through visual interfaces [17, 12]. As for textual content, *TalkExplorer* [29] and *SetFusion* [18] are examples of interfaces for exploration of conference talk recommendations. The former is mostly focused in depicting relationships among recommendations, users and tags in a transparent manner, while *SetFusion* emphasizes controllability over a hybrid RS. Rankings are not transparent though, as there is no explanation as to how they were obtained. Kangasraasio et al. [10] highlighted that not only allowing the user to influence the RS is important, but also adding predictability features that produce an effect of causality for user actions.

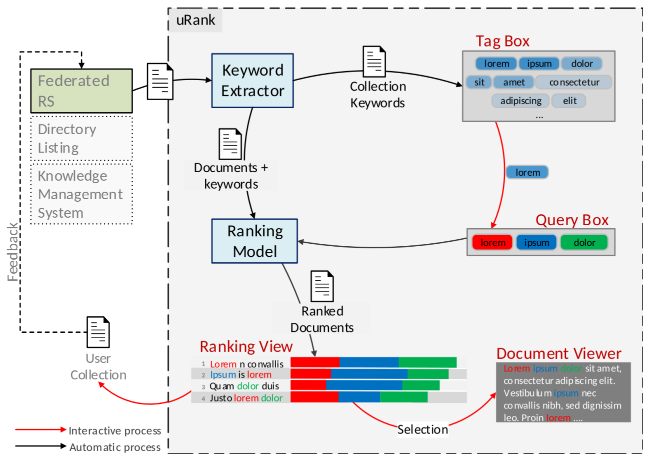


Figure 1: *uRank* visual analytics workflow showing automatic (black arrows) and interactive mechanisms (red arrows)

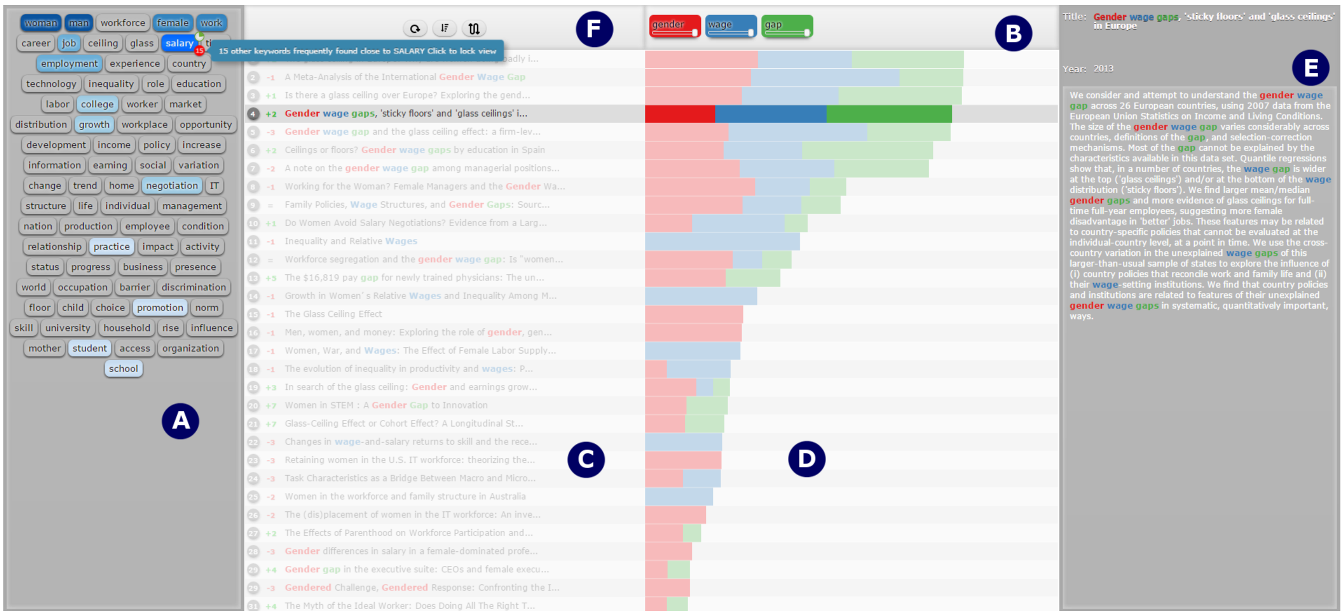
With *uRank* we intend to enhance predictability through document hint previews (section 3.1.1), allow the user to control the ranking by choosing keywords as parameters, and support understanding by means of a transparent graphic representation for scores (section 3.2).

3. URANK VISUAL ANALYTICS

uRank is a visual analytics approach that combines lightweight text analytics and an augmented ranked list to assist in exploratory search of textual recommendations. The Web-based implementation is fed with textual document surrogates by a federated RS (F-RS) connected to several sources. A keyword extraction module analyzes all titles and abstracts and outputs a set of representative terms for the whole collection and for each document. The UI allows users to explore the collection content and refine information needs in terms of topic keywords. As the user selects terms of interest, the ranking is updated, bringing related documents closer to the top and pushing down the less relevant ones. Figure 1 outlines the workflow between automatic and interactive components.

uRank's layout is arranged in a multiview fashion that displays different perspectives of the document recommendations. Following Baldonados's guidelines [30], we decided to limit the number of views to keep display space requirements relatively low. Therefore, instead of multiple overlapping views, we favor a reduced number of perspectives fitting in any laptop or desktop screen. The GUI dynamically scales to the window size, remaining undistorted up to a screen width of approximately 770 px.

The GUI presents the data in juxtaposed views that add to a semantic Overview+Detail scheme [2] with three levels of granularity: *Collection overview.* The *Tag Box* (Figure 2.A) summarizes the entire collection through by representing keywords as augmented tags. *Documents overview.* The *Document List* shows titles augmented with ranking information and the *Ranking View* displays stacked bar charts depicting document relevance scores (Figure 2.C and D, respectively). Together they represent mini-



15 other keywords frequently found close to SALARY Click to lock view

badly i...
2 -1 A Meta-Analysis of the International Gender Wage Gap
3 +1 Is there a glass ceiling over Europe? Exploring the gend...
4 +2 Gender wage gaps, 'sticky floors' and 'glass ceilings' i...
5 -3 Gender wage gap and the glass ceiling effect: a firm-lev...
6 +2 Ceilings or floors? Gender wage gaps by education in Spain
7 -2 A note on the gender wage gap among managerial positions...
8 -1 Working for the Woman? Female Managers and the Gender Wa...
9 = Family Policies, Wage Structures, and Gender Gaps: Sourc...
10 +1 Do Women Avoid Salary Negotiations? Evidence from a Larg...
11 -1 Inequality and Relative Wages
12 = Workforce segregation and the gender wage gap: Is "women...
13 +5 The $16,819 pay gap for newly trained physicians: The un...
14 -1 Growth in Women' s Relative Wages and Inequality Among M...
15 -1 The Glass Ceiling Effect
16 -1 Men, women, and money: Exploring the role of gender, gen...
17 -1 Women, War, and Wages: The Effect of Female Labor Supply...
18 -1 The evolution of inequality in productivity and wages: P...
19 +3 In search of the glass ceiling: Gender and earnings grow...
20 +7 Women in STEM : A Gender Gap to Innovation
21 +7 Glass-Ceiling Effect or Cohort Effect? A Longitudinal St...
22 -3 Changes in wage-and-salary returns to skill and the rece...
23 -3 Retaining women in the U.S. IT workforce: theorizing the...
24 -3 Task Characteristics as a Bridge Between Macro and Micro...
25 -2 Women in the workforce and family structure in Australia
26 -2 The (dis)placement of women in the IT workforce: An inve...
27 +2 The Effects of Parenthood on Workforce Participation and...
28 -3 Gender differences in salary in a female-dominated profe...
29 +4 Gender gap in the executive suite: CEOs and female execu...
29 -3 Gendered Challenge, Gendered Response: Confronting the I...
31 +4 The Myth of the Ideal Worker: Does Doing All The Right T...

Title: Gender wage gaps, 'sticky floors' and 'glass ceilings' in Europe

Year: 2013

We consider and attempt to understand the gender wage gap across 26 European countries, using 2007 data from the European Union Statistics on Income and Living Conditions. The size of the gender wage gap varies considerably across countries, definitions of the gap, and selection-correction mechanisms. Most of the gap cannot be explained by the characteristics available in this data set. Quantile regressions show that, in a number of countries, the wage gap is wider at the top ('glass ceilings') and/or at the bottom of the wage distribution ('sticky floors'). We find larger mean/median gender gaps and more evidence of glass ceilings for full-time full-year employees, suggesting more female disadvantage in 'better' jobs. These features may be related to country-specific policies that cannot be evaluated at the individual-country level, at a point in time. We use the cross-country variation in the unexplained wage gaps of this larger-than-usual sample of states to explore the influence of (i) country policies that reconcile work and family life and (ii) their wage-setting institutions. We find that country policies and institutions are related to features of their unexplained gender wage gaps in systematic, quantitatively important, ways.

Figure 2: *uRank* User Interface displaying a ranking of documents for the keywords "gender", "wage" and "gap". The user has selected the third item in the list. A. The *Tag Box* presents a keyword-based summary, B. the *Query Box* contains the selected keywords that originated the current ranking state, C. the *Document List* presents a list with augmented document titles. D. the *Ranking View* renders stacked bars indicating relevance scores, E. the *Document Viewer* shows the title, year and snippet of the selected document with augmented keywords, and F. the *Ranking Controls* wrap buttons for ranking settings.

mal views of documents where they can be differentiated by title or position in the ranking and compared at a glance basing on the presence of certain keywords of interest. *Document detailed view.* For a document selected in the list, the *Document Viewer* (Figure 2.E) displays the title and snippet with color-augmented keywords.

These views can be modified through interaction with the *Ranking Controls* (Figure 2.F) and the *Query Box* (Figure 2.B). The former provides controls to reset the ranking or switch ranking modes between overall and maximum score. The latter is the container where the user drops keywords tags to trigger changes in the ranking visualization.

3.1 Collection Overview

uRank automatically extracts keywords from the recommended documents with a twofold purpose: i) give an overview of the collection, and ii) provide manipulable elements that serve as input for an on-the-fly ranking mechanism (see section 3.2).

Summarizing the collection in a few representative terms allows the user to scan the recommendations and grasp the general topic at a glance, before even reading any of them. This is particularly important in the context of collections brought by RS, where the user is normally not directly generating the queries that feed the search engine.

3.1.1 Inspecting the Collection

The *Tag Box* provides a summary of the recommended texts as a whole by presenting extracted keywords as tags. Keywords tags are arranged in a bag-of-words fashion, encoding relative frequencies through position and intensity (Figure 2.A). The descending ordering conveys document frequency (DF) while five levels of blue shading help the user identify groups of keywords in the same frequency range. Redundant coding is intentional and aims at maximizing distinctiveness among items in the keyword set [32].

At first glance, the *Tag Box* gives an outline of the covered topic in terms of keywords and their relative frequencies. Nevertheless, a bag-of-words representation per se does not supply further details about how a keyword relates to other keywords or documents. We bridge this gap by augmenting tags with two compact visual hints – visible on mouse over – that reveal additional information: *i)* co-occurence respect to other keywords, and *i)* a preview of the effect of selecting the keyword.

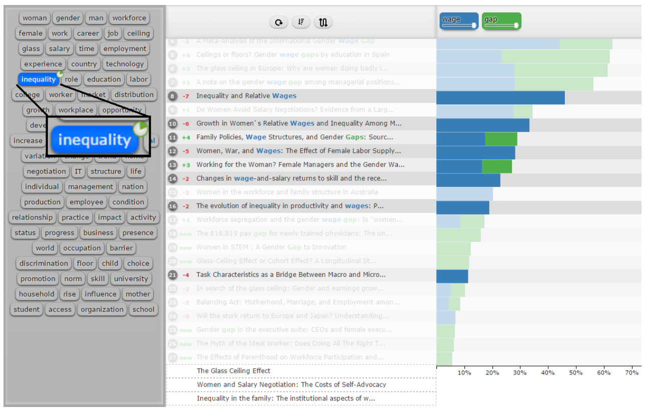
The document hint (Figure 3) consists in a pie chart that conveys the proportion of documents in which the keyword appears. A tooltip indicates the exact quantity and percentage. Upon clicking on the document hint, unrelated documents are dimmed so that documents containing the keyword remain in focus Even unranked documents become discretely visible at the bottom of the *Document List*. This hint provides certain predictability regarding the effect of selecting a keyword, in terms of which ranked items will change their scores and which documents will be added to the ranking.

The co-occurrence hint (Figure 2.A) shows the number of frequently co-occurring keywords in a red circle. Moving the mouse pointer over it brings co-occurring terms to focus by dimming the others in the background. Clicking on the visual hint locks the view so that the user can hover over co-occurring keywords, which shows a tooltip stating the amount of co-occurrences between the hovered and the selected keyword. This hint supports the user in finding possible key phrases and sub-topics within the collection.

3.1.2 Mining a collection of documents

The aforementioned interactive features are supported by a combination of well-known text-mining techniques that extend the recommended documents with document vectors and provide meaningful terms to populate the *Tag Box*.

Document vectors ideally include only content-bearing terms like nouns and frequent adjectives – appearing in at least 50% of the collection –, hence it is not enough to just rely on a list of stop words



4 -2 A Meta-Analysis of the International Gender Wage Gap
5 +6 Ceilings or floors? Gender wage gaps by education in Spain
6 +3 The glass ceiling in Europe: Why are women doing badly i...
7 +1 A note on the gender wage gap among managerial positions...
8 -7 Inequality and Relative Wages
9 +1 Do Women Avoid Salary Negotiations? Evidence from a Larg...
10 -6 Growth in Women' s Relative Wages and Inequality Among M...
11 +4 Family Policies, Wage Structures, and Gender Gaps: Sourc...
12 -5 Women, War, and Wages: The Effect of Female Labor Supply...
13 +3 Working for the Woman? Female Managers and the Gender Wa...
14 -2 Changes in wage-and-salary returns to skill and the rece...
15 -2 Women in the workforce and family structure in Australia
16 -2 The evolution of inequality in productivity and wages: P...
17 +1 Workforce segregation and the gender wage gap: Is "women...
18 new The $16,819 pay gap for newly trained physicians: The un...
19 new Women in STEM : A Gender Gap to Innovation
20 new Glass-Ceiling Effect or Cohort Effect? A Longitudinal St...
21 -4 Task Characteristics as a Bridge Between Macro and Micro...
22 -2 In search of the glass ceiling: Gender and earnings grow...
23 -2 Balancing Act: Motherhood, Marriage, and Employment amon...
24 -5 Will the stork return to Europe and Japan? Understanding...
25 new Gender gap in the executive suite: CEOs and female execu...
26 new The Myth of the Ideal Worker: Does Doing All The Right T...
27 new The Effects of Parenthood on Workforce Participation and...

The Glass Ceiling Effect
Women and Salary Negotiation: The Costs of Self-Advocacy
Inequality in the family: The institutional aspects of w...

Figure 3: Document hints show a preview of documents containing the hovered keyword, even if they are currently unranked

to remove meaningless terms. Firstly, we perform a part-of-speech tagging (POS tagging) [1] step to identify words that meet our criteria, i.e. common and proper nouns and adjectives. Filtering out non-frequent adjectives requires an extra step. Then, plural nouns are singularized, proper nouns are kept capitalized and terms in upper case, e.g. "IT", remain unchanged. We apply the Porter Stemmer method [19] over the resulting terms, in order to increase the probability of matching for similar words, e.g. "robot", "robots" and "robotics" all match the stem "robot". A document vector is thus conformed by stemmed versions of content-bearing terms.

Next, we generate a weighing scheme by computing TF-IDF (term frequency inverse document frequency) for each term in a document vector. The score is a statistical measure of how important the term is to a document in a collection. Therefore, the more frequent a term is in a document and the fewer times it appears in the corpora, the higher its score will be. Documents' metadata are extended with these weighted document vectors.

To fill the *Tag Box* with representative keywords for the collection set, all document keywords are collected in a global keyword set. Global keywords are sorted by document frequency (DF), i.e. the number of documents in which they appear, regardless of the frequency within documents. To avoid overpopulating the *Tag Box*, only terms with DF above certain threshold (by default 5) are taken into account. Note that terms used to label keyword tags are actual words and not plain stems. Scanning a summary of stemmed words would turn unintuitive for users. Thus, we keep a record of all term variations matching each stem, in order to allow for reverse stemming and pick one representative word as follows:
1. if there is only one term for a stem, use it to label the tag,
2. if a stem has two variants, one in lower case and the other in upper case or capitalized, use it in lower case,
3. use a term that ends in 'ion', 'ment', 'ism' or 'ty',
4. use a term matching the stem,
5. use the shortest term.

To feed the document hint (Figure 3), *uRank* attaches a list of bearing documents to each global keyword. For the case of co-occurrence hints (Figure 2.A), keyword co-occurrences with a maximum word distance of 5 and a minimum of 4 repetitions are recorded.

3.2 Ranking Documents On The Fly

In theory, recommendations returned by a RS are already ranked by relevance. However, in practice the lack of control thereof could hinder user engagement if the GUI does not provide enough rationale for the recommendations and features for shaping the recommendation criteria. Hence, one of *uRank*'s major features is the user-driven mechanism for re-organizing documents as information needs evolve, along with its visually transparent logic.

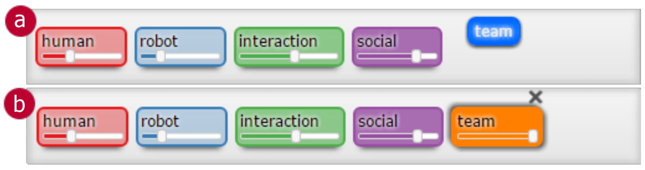


Figure 4: a) Keyword tag before being dropped in Tag Box. b) Keyword tag after dropped: weight slider and delete button added, background color changed according to a categorical color scale. Weight sliders have been tuned.

3.2.1 Ranking Visualization

The ranking-based visualization consists of a list of document titles (Figure 2.C) and stacked bar charts (Figure 2.D) depicting rank and relevance scores for documents and keywords within them. Document titles are initially listed following the order in which they were supplied by the F-RS.

Interactions with the view are the means for users to directly or indirectly manipulate the data [31]. In *uRank*, changes in the ranking visualization originate from keyword tag manipulation inside the *Query Box* (Figure 2.B). As the user manipulates tags, selected keywords are immediately forwarded to the *Ranking Model* as ranking parameters. Selected tags are re-rendered by adding a weight slider, a delete button on the right-upper corner – visible on hover – and a specific background color determined by a qualitative palette (Figure 4). We chose Color Brewer's [5] *9-class Set 1* palette for background color encoding, as it allows the user to clearly distinguish tags from one another. When the user adjusts a weight slider, the intensity of the tag's background color changes accordingly (see Figure 4). We provide three possibilities for keyword tag manipulation:

- **Addition**: keyword tags in the *Tag Box* can be manually unpinned (Figure 4a), dragged with the mouse pointer and dropped into the *Query Box* (Figure 4b).
- **Weight change**: tags in the *Query Box* contain weight sliders that can be tuned to assign a keyword a higher or lower priority in the ranking.
- **Deletion**: tags can be removed from the *Query Box* and returned to their initial position in the *Tag Box* by clicking on the delete button.

As the document ranking is generated, the *Document List* is resorted in descending order by overall score and stacked bars appear in the *Ranking View*, horizontally aligned to each list item. Items with null score are hidden, shrinking the list size to fit only ranked items. The total width of stacked bars indicates the overall score of a document and bar fragments represent the individual contribution of keywords to the overall score. Bar colors match the color encoding for selected keywords in the *Query Box*, enabling the user to make an immediate association between keyword tags and bars. Missing colored bars in a stack denote the absence of certain words in the document surrogate. Additionally, each item in the Document List contains two types of numeric indicators: the first one - in a dark circle - shows the position of a document in the ranking while the adjacent colored number reveals how many positions

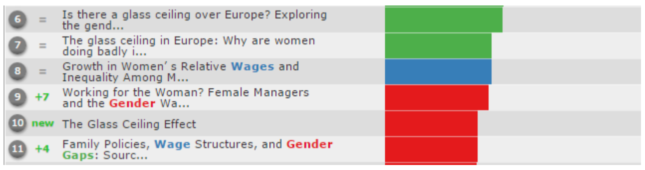

6 = Is there a glass ceiling over Europe? Exploring the gend...
7 = The glass ceiling in Europe: Why are women doing badly i...
8 = Growth in Women' s Relative Wages and Inequality Among M...
9 +7 Working for the Woman? Female Managers and the Gender Wa...
10 new The Glass Ceiling Effect
11 +4 Family Policies, Wage Structures, and Gender Gaps: Sourc...

Figure 5: Ranking visualization in maximum score mode: documents are ranked basing on the keyword with highest score

the document has shifted, encoding upward and downward shifts in green and red, respectively. This graphic representation attempts to help the user concentrate only on useful items and ignore the rest by bringing likely relevant items to the top, pushing less relevant ones to the bottom and hiding those that seem completely irrelevant.

uRank allows for choosing between two ranking modes: overall score (selected by default) and maximum score (Figure 5). In maximum score mode, the Ranking View renders a single color-coded bar per document in order to emphasize its most influential keyword. Finally, resetting the visualization clears the *Query Box* and the *Ranking View*, relocating all selected keywords in the *Tag Box* and restoring the *Document List* to its initial state.

3.2.2 Document Ranking Computation

Quick content exploration in *uRank* depends on its ability to readily re-sort documents according to changing information needs. As the user manipulates keyword tags and builds queries from a subset of the global keyword collection, *uRank* computes documents scores to arrange them accordingly in a document ranking. We assume that some keywords are more important to the topic model than others and allow the user to assign weights to them.

Document scores are relevance measures for documents respect to a query. As titles and snippets are the only content available for retrieved document surrogates, these scores are computed with a term-frequency scheme. Term distribution schemes are rather adequate for long or full texts and are hence out of our scope. Boolean models have the disadvantages that they not only consider every term equally important but also produce absolute values that preclude document ranking.

The *Ranking Model* implements a vector space model to compute document-query similarity using the document vectors previously generated during keyword extraction (section 3.1.2). Nonetheless, a single relevance measure like cosine similarity alone is not enough to convey query-term contribution, given that the best overall matches are not necessarily the ones in which most query terms are found [7, 14]. The contribution that each query term adds to the document score should be clear in the visual representation, in order to give the user a transparent explanation as to why a document ranks in a higher position than another. Therefore, we break down the cosine similarity computation and obtain individual scores for each query term, which are then added up as an overall relevance score.

Given a document collection D and a set of weighted query terms T , such that $\forall t \in T : 0 \leq w_t \leq 1$; the relevance score for term t in document vector $d \in D$ respect to query terms T is calculated as follows:

$$s(t_d) = \frac{tfidf(t_d) \times w_t}{|d| \times \sqrt{|T|}},$$

where $tfidf(t_d)$ is the tf-idf score for term t in document d and $|d|$ is the norm for vector d.

The overall score of a document $S(d)$ is then computed as the sum of each individual term score $s(t_d)$. The collection D is next sorted in descending order by overall score with the quicksort algorithm and ranking positions are assigned in such way that documents with equivalent overall score share the same place.

Alternatively, users can rank documents by maximum score, in which case $S(d) = \max(s(t_d))$.

3.3 Details on Demand

Once the user identifies documents that seem worth further inspecting, the next logical step is to drill down one by one to determine whether the initial assumption holds. The *Document Viewer* (Figure 2.D) gives access to textual content - title and snippet - and available metadata for a particular document. Query terms are highlighted in the text following the same color coding for tags in the *Query Box* and stacked bars in the *Ranking View*. These simple visual cues pop out from their surroundings, enabling the user to preattentively recognize keywords in the text and perceive their general context prior to conscious reading.

3.4 Change-Awareness Cues and Attention Guidance

We favor the use of animation to convey ranking-state transitions rather than abrupt static changes. Animated transitions are inherently intuitive and engaging, giving a perception of causality and intentionality [8]. As the user manipulates a keyword tag in the *Query Box*, *uRank* raises change awareness in the following way:

- Keyword tags are re-styled as explained in section 3.2.1. If the tag is removed from the *Query Box*, animation is used to shift the tag to its original position in the *Tag Box* at a perceivable pace.
- Depending on the type of ranking transition, the *Document List* shows a specific effect:
  - If the ranking is generated for the first time, an accordion-like upward animation shows that its nature has changed from a plain list to a ranked one.
  - If the ranking is updated, list items shift to their new positions at a perceptible pace.
  - If ranking positions remain unchanged, the list stays static as a soft top-down shadow crosses it.
- Green or red shading effects are applied on the left side of list items moving up or down, respectively, disappearing after a few seconds.
- Stacked bars grow from left to right revealing new overall and keyword scores.

The user can closely follow how particular documents shift positions by clicking on the watch - eye-shaped - icon. The item is brought to focus as it is surrounded with a slightly darker shadow and the title is underlined. Also, watched documents remain on top of the z-index during list animations, avoiding being overlaid by other list items.

The same principle of softening changes is applied to re-direct user attention when a document is selected in the *Ranking View*. The selected row is highlighted and the snippet appears in the *Document Viewer* in a fade-in fashion. Animated transitions for ranking-state changes and document selection help the user intuitively switch contexts, either from the *Tag Box* to the *Document List* and *Ranking View*, or from the latter to the Document Viewer. As Baldonado [30] states in the rule of attention management, perceptual techniques lead the users attention to the right view at the right time.

4. EVALUATION

The goal of this study was to find out how people responded when working with our tool. In the current scenario, recommendations were delivered in a sorted list with no relevance information. Since we aim at supporting exploratory search, we hypothesized that participants using *uRank* would be able to gather items faster and with less difficulty, compared to a typical list-based UI.

We were also interested in observing the effect of exposing users to different sizes of recommendation lists. We expected that without this relevance information, a slight growth in the number of displayed items would frustrate the user at the moment of deciding which items should be inspected in detail in the first place. For example, finding the 5 most relevant items in a list of ten appears as an easy task, whereas accomplishing the same task but searching a list of forty or sixty items would be more time consuming and entail a heavier cognitive load.

4.1 Method

We conducted an offline evaluation where participants performed four iterations of the same task with either *uRank* (U) or a baseline list-based UI (L) with usual Web browser tools, e.g. *Control+F* keyword search. Furthermore, we introduced two variations in the number of items to which participants were exposed, namely 30 or 60 items. Therefore, the study was structured in a 2 x 2 repeated measures design with *tool* and *#items* as independent variables, each with 2 levels (*tool* = U/L, *#items* = 30/60).

The general task goal was to "find 5 relevant items" for the given topic and all participants had to perform one task for each combination of the independent variables, i.e. **U-30**, **U-60**, **L-30** and **L-60**.

To counterbalance learning effects, we chose four different topics covering a spectrum of cultural, technical and scientific content: *Women in workforce* (WW), *Robots* (Ro), *Augmented Reality* (AR) and *Circular economy* (CE). Thus, *topic* was treated as a random variable within constraints. We corroborated that participants were not knowledgeable in any of the topics. All variable combinations were randomized and assigned with balanced Latin Square.

Wikipedia provides a well-defined article for each topic mentioned above. We considered them as fictional initial exploration scenarios but participants were not exposed to them. Instead, we simulated a situation in which the user has already received a list of recommendations while exploring certain Wikipedia page. Therefore, we prepared static recommendation lists of 60 and 30 items for each topic and used them as inputs for *uRank* throughout the different participants and tasks. To create each list, portions of texts from the original Wikipedia articles were fed to the F-RS, which preprocessed the text and created queries that were forwarded to a number of content providers. The result was a sorted merged list of items from each provider with no scoring information.

Each task comprised three sub-tasks (Q1, Q2 and Q3) that consisted in finding the 5 most relevant items for a given piece of text. In Q1 and Q2 we targeted a specific search and the supplied text was limited to two or three words. Q3 was designed as a broad-search sub-task where we provided an entire paragraph extracted from the Wikipedia page and the users had to decide themselves which keywords described the topic better. The motivation to ask for the "most relevant" documents was to avoid careless selection.

We recorded completion time for every individual sub-task and for the overall task. To measure workload, we leveraged a 7-likert scale NASA TLX questionnaire covering six workload dimensions.

4.1.1 Participants

Twenty four (24) participants took part in the study (*11* female, *13* male, between 22 and 37 years old). We recruited mainly graduate and post-graduate students from the medical and computer science domains. None of them is majoring in the topic areas selected for the study.

4.1.2 Procedure

A session started with an introductory video explaining the functionality of *uRank*. Each participant got exactly the same instructions. Then came a short training session with a different topic (Renaissance) to let participants familiarize with *uRank* and the baseline the tool. At the beginning of the first task, the system showed a short text describing the topic and the task to be fulfilled. After reading the text, the participant pressed "Start" to redirect the browser to the corresponding UI. At this point, the first sub-task began and the internal timer initiated the count, without disturbing the user. The goal of the task and the reference text were shown in the upper part of the UI. Participants were able to select items by clicking on the star-shaped icon and inspect them later on a drop-down list. In a pilot study, we realized that asking for the "most" relevant items made the experiment overly long, as participants tried to carefully inspect their selections (particularly in the L condition). Then we decided to limit the duration of the three tasks to 3m, 3m and 6m respectively. The time constraint was not a hard deadline. During the study the experimenter reminded the participants when the allotted time was almost over, but did not force them to abandon. The sub-task concluded when the participant clicked on the "Finished" button. The UI alerted participants when attempting to finish without collecting 5 items, but allowed them to continue if desired. The second sub-task started immediately afterward and once the whole task was completed they had to fill the NASA TLX questionnaire. The procedure for the remaining tasks was repeated following the same steps. Finally, participants were asked about comments and preferences.

Table 1: Participants found *uRank* reduces workload in all dimensions

Dimension	$F(1,23)$	p	ε
Mental Demand	19.70	$p < .05$	.10
Physical Demand	14.52	$p < .01$	.07
Temporal Demand	7.72	$p < .05$	.05
Performance	11.80	$p < .01$	.10
Effort	48.60	$p < .001$	.22
Frustration	15.12	$p < .01$	.07
Workload	35.25	$p < .01$	.20

4.2 Results

Workload: A two-way repeated measures ANOVA with *tool* and *#items* as independent variables revealed a significant effect of *tool* on perceived workload F(1,23)=35.254, $p < .01, \varepsilon = .18$. Bonferroni post-hoc tests showed significantly lower workload when using *uRank* ($p < .001$). We also assessed the effect for each workload dimension. Again, ANOVA showed a significant effect of *tool* in all of them, as shown in Table 1. (*#items*) did not have a major effect in any case.

Completion Time: We analyzed the task overall completion time, as well as completion times for each sub-task. A two-way repeated measures ANOVA revealed a significant effect of *tool* on overall completion time F(1,23)=4.94, $p < .05, \varepsilon = .02$. This effect disappeared in a Bonferroni post-hoc comparison. For Q1 and Q2 ANOVA reported no significant effect, but it showed a significant effect of *tool* on completion time for Q3, F(1,23)=6.2,

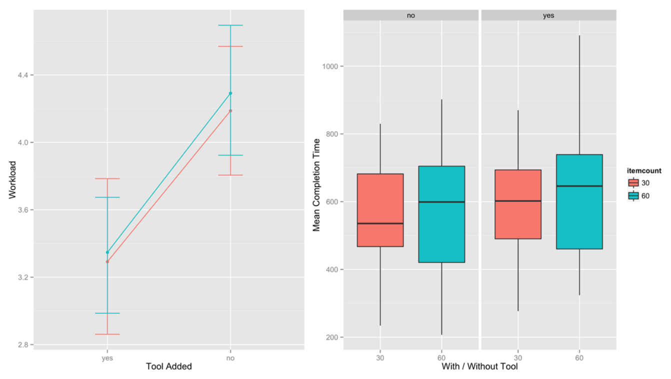


Figure 6: Results. (Left) Workload interaction lines show that *uRank* is significantly less demanding. (Right) Boxplots of time completion for each condition show a regularity towards using all available time.

Table 2: Similarities in collections gathered during evaluation

Sub-task	Comparison	WW	Ro	AR	CE	All topics
Q1	U vs L	.55	.79	.58	.74	.66
	U-30 vs U-60	.71	.83	.94	.67	.79
	L-30 vs L-60	.58	.83	.56	.56	.63
Q2	U vs L	.70	.86	.84	.86	.81
	U-30 vs U-60	.84	.89	.90	.93	.89
	L-30 vs L-60	.82	.74	.81	.87	.81
Q3	U vs L	.75	.72	.75	.63	.72
	U-30 vs U-60	.64	.88	.75	.62	.72
	L-30 vs U-60	.59	.66	.63	.33	.55

$p < .05, \varepsilon = .05$. As a surprise, post-hoc comparison showed that using *uRank* took significantly longer.

Performance: Relevance is a rather subjective measure. Hence, instead of contrasting item selections to some ground truth, we analyzed "consensus" in item selection.

We aggregated the collections gathered under the manipulated conditions and computed cosine similarity across UI (*tool*), data set size (*#items*), topic (WW, Ro, AR, and CE) and sub-task (Q1, Q2 and Q3).

Overall, there was a high similarity between collections produced with *uRank* and those obtained with the list-based UI across all sub-tasks. Choices regarding relevant documents matched three out of four times ($M = .73$, $SD = .1$).

Table 2 shows that collections produced with our tool (U) for the two variations of *#items* (U-30 vs U-60) turned highly similar regardless of topic and sub-task ($M = .8$, $SD = .12$, with a minimum of .62). Comparisons for a typical list-based UI (L) displaying 30 and 60 items (L-30 vs L-60) denote greater diversity ($M = .67$, $SD = .16$, with a minimum of .33) in item selection.

Interestingly, similarity values tend to decrease for broad search task (Q3) ($M = .66$, $SD = .13$) respect to targeted search (Q1 and Q2) ($M = .77$, $SD = .13$).

4.3 Discussion

The study results shed a light on how people interact with a tool like *uRank*. For each hypotheses we contrasted the results with the subjective feedback acquired after evaluation.

Workload: The results support our hypothesis that *uRank* incurs in lower workload during exploratory search, both in specific and broad search tasks. Participants commented feeling alleviated when they could browse the ranking and instantly discard document that did not contain any word of interest. As a remark, the majority claimed that a few tasks were too hard to solve, especially without the *uRank*, because sometimes the terms of interest barely appeared in the titles or were perceived as too ambiguous, e.g. "participation of women in the workforce". Also dealing with technical texts about unfamiliar topics was posed some strain. For example, two participants had to momentarily interrupt exploration to look up a word they did not understand. In spite of that, workload was significantly lower with *uRank* across all dimensions.

Completion Time: We expected people would be faster performing with *uRank* than using a browser-based keyword filter, but completion times were not significantly different. The closing interview revealed that participants who had collected five items before the due time exploited the remainder to refine their selections. In general, participants understood that they were not expected to perform perfectly but to do their best in the given time. However, we noticed that a small group that behaved in the opposite way reported feeling more pressed by time and not satisfied with their performance. The general tendency is reflected in the significant result on temporal demand: participants felt significantly less pressed to finish while performing with *uRank*. The lower subjective time pressure suggests that participants indeed had more time to analyze their choices with *uRank*.

Performance: The results suggest that our tool produces more uniform results as the number of items to which users are exposed grows. Nevertheless, the proportion of matching documents in list-generated collections – two out of three – still conveys a moderate consensus.

The decrease in consensus for broad search task respect to targeted search could be explained by the inherent variability across participants at the moment of chosing the terms of interest for a given text larger than a couple of words.

5. CONCLUSION

We introduced a visual tool for exploration, discovery and analysis of recommendations of textual documents. *uRank* aims to help the user: *i)* quickly overview the most important topics in a collection of documents, *ii)* interact with content to describe a topic in terms of keywords, and *iii)* on-the-fly reorganize the documents along keywords describing a topic.

This paper presented the reasoning line for the visual and interactive design and a comparative user study where we evaluated the experience of collecting relevant items to topics of interest. Participants found it significantly more *relaxing* to work with *uRank*, and most of them wanted to start actively using it in their scientific endeavors (e.g., report or paper writing). Yet, selecting the right keywords to describe a topic is not a trivial task, as it showed on the performance results of the evaluation. We will continue to explore different techniques, e.g. topic modeling, in the near future. As for the GUI, we will work further on solving scaling problems, for example when the amount of tags in the *Tag Box* or the length of the result list becomes unmanageable. Moreover, we will leverage the document selections collected during the evaluation as feedback to improve recommendations, closing the interactive loop with the RS as depicted in Figure 1.

6. REFERENCES

[1] E. Brill. A simple rule-based part of speech tagger. In *Proceedings of the workshop on Speech and Natural Language - HLT '91*, page 112, Morristown, NJ, USA, 1992. Association for Computational Linguistics.

[2] A. Cockburn, A. Karlson, and B. B. Bederson. A review of overview+detail, zooming, and focus+context interfaces. *ACM Computing Surveys*, 41(1):1–31, 2008.

[3] E. Gomez-Nieto, F. San Roman, P. Pagliosa, W. Casaca, E. S. Helou, M. C. F. de Oliveira, and L. G. Nonato. Similarity preserving snippet-based visualization of web search results. *IEEE transactions on visualization and computer graphics*, 20(3):457–70, Mar. 2014.

[4] S. Gratzl, A. Lex, N. Gehlenborg, H. Pfister, and M. Streit. LineUp: visual analysis of multi-attribute rankings. *IEEE transactions on visualization and computer graphics*, 19(12):2277–86, Dec. 2013.

[5] M. Harrower and C. A. Brewer. ColorBrewer.org: An Online Tool for Selecting Colour Schemes for Maps. *The Cartographic Journal*, 40(1):27–37, June 2003.

[6] M. Hearst. User interfaces for search. *Modern Information Retrieval*, 2011.

[7] M. A. Hearst. TileBars: Visualization of Term Distribution Information in Full Text Information Access. In *Proceedings of the SIGCHI conference on Human factors in computing systems - CHI '95*, pages 59–66. ACM Press, 1995.

[8] J. Heer and G. Robertson. Animated transitions in statistical data graphics. *IEEE transactions on visualization and computer graphics*, 13(6):1240–7, 2007.

[9] O. Hoeber and X. D. Yang. The Visual Exploration of Web Search Results Using HotMap. In *Proceedings of the Information Visualization (IV06)*, 2006.

[10] A. Kangasrääsiö, D. Gowacka, and S. Kaski. Improving Controllability and Predictability of Interactive Recommendation Interfaces for Exploratory Search. In *IUI*, pages 247–251, 2015.

[11] J. Kay. Scrutable adaptation: Because we can and must. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 4018 LNCS, pages 11–19, 2006.

[12] B. P. Knijnenburg, S. Bostandjiev, J. O'Donovan, and A. Kobsa. Inspectability and control in social recommenders. *Proceedings of the 6th ACM conference on Recommender systems - RecSys '12*, page 43, 2012.

[13] B. P. Knijnenburg, M. C. Willemsen, Z. Gantner, H. Soncu, and C. Newell. Explaining the user experience of recommender systems. *User Modelling and User-Adapted Interaction*, 22(4-5):441–504, 2012.

[14] C. D. Manning. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[15] G. Marchionini. Exploratory search: from finding to understanding. *Communications of the ACM*, 49(4):41, 2006.

[16] T. N. Nguyen and J. Zhang. A novel visualization model for web search results. *IEEE transactions on visualization and computer graphics*, 12(5):981–8, 2006.

[17] J. O'Donovan, B. Smyth, B. Gretarsson, S. Bostandjiev, and T. Höllerer. PeerChooser: Visual Interactive Recommendation. *Proceeding of the twenty-sixth annual SIGCHI conference on Human factors in computing systems*, pages 1085–1088, 2008.

[18] D. Parra, P. Brusilovsky, and C. Trattner. See what you want to see: Visual User-Driven Approach for Hybrid Recommendation. *Proceedings of the 19th international conference on Intelligent User Interfaces - IUI '14*, pages 235–240, 2014.

[19] M. Porter. An algorithm for suffix stripping. *Program: electronic library and information systems*, 40(3):211–218, 1980.

[20] R. Rao and S. K. Card. The table lens. In *Proceedings of the SIGCHI conference on Human factors in computing systems celebrating interdependence - CHI '94*, number April, pages 318–322, New York, New York, USA, 1994. ACM Press.

[21] H. Reiterer, G. Tullius, and T. Mann. Insyder: a content-based visual-information-seeking system for the web. *International Journal on Digital Libraries*, pages 25–41, 2005.

[22] F. Ricci, L. Rokach, and B. Shapira. Introduction to recommender systems handbook. In F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, editors, *Recommender Systems Handbook*, pages 1–35. Springer, 2011.

[23] C. Seifert, J. Jurgovsky, and M. Granitzer. FacetScape : A Visualization for Exploring the Search Space. In *Proceedings 18th International Conference on Information Visualzation*, pages 94–101, 2014.

[24] G. Shani and N. Tractinsky. Displaying relevance scores for search results. *Proceedings of the 36th international ACM SIGIR13*, pages 901–904, 2013.

[25] A. Spoerri. Coordinated Views and Tight Coupling to Support Meta Searching. In *Proceedings of Second International Conference on Coordinated and Multiple Views in Exploratory Visualization*, pages 39–48, 2004.

[26] M. Streit and N. Gehlenborg. Bar charts and box plots. *Nature methods*, 11(2):117, Feb. 2014.

[27] K. Swearingen and R. Sinha. Beyond Algorithms Beyond Algorithms : An HCI Perspective on Recommender Systems. *ACM SIGIR 2001 Workshop on Recommender Systems (2001)*, pages 1–11, 2001.

[28] N. Tintarev and J. Masthoff. Evaluating the effectiveness of explanations for recommender systems. *User Modeling and User-Adapted Interaction*, 22(4-5):399–439, Oct. 2012.

[29] K. Verbert, D. Parra, P. Brusilovsky, and E. Duval. Visualizing recommendations to support exploration, transparency and controllability. *Proceedings of the 2013 international conference on Intelligent user interfaces - IUI '13*, page 351, 2013.

[30] M. Q. Wang Baldonado, A. Woodruff, and A. Kuchinsky. Guidelines for using multiple views in information visualization. *Proceedings of the working conference on Advanced visual interfaces (AVI)*, pages 110–119, 2000.

[31] M. O. Ward, G. Grinstein, and D. A. Keim. *Interactive Data Visualization: Foundations, Techniques, and Application*. A. K. Peters, Ltd, May 2010.

[32] C. Ware. *Information visualization: perception for design*. Elsevier, 3rd edition, 2013.

[33] K.-P. Yee, K. Swearingen, K. Li, and M. Hearst. Faceted metadata for image search and browsing. *Proceedings of the conference on Human factors in computing systems - CHI '03*, pages 401–408, 2003.

FutureView: Enhancing Exploratory Image Search

Sayantan Hore, Dorota Głowacka, Ilkka Kosunen, Kumaripaba Athukorala and Giulio Jacucci
Helsinki Institute for Information Technology HIIT, Department of Computer Science, University of Helsinki
first.last@cs.helsinki.fi

ABSTRACT

Search algorithms in image retrieval tend to focus on giving the user more and more similar images based on queries that the user has to explicitly formulate. Implicitly, such systems limit the users exploration of the image space and thus remove the potential for serendipity. As a response, in recent years there has been an increased interest in developing content based image retrieval systems that allow the user to explore the image space without the need to type specific search queries. However, most of the research focuses on designing new algorithms and techniques, while little research has been done in designing interfaces allowing the user to actively engage in directing their image search. We present an interactive FutureView interface that can be easily combined with most existing exploratory image search engines. The interface gives the user a view of possible future search iterations. A task-based user study demonstrates that our interface enhances exploratory image search by providing access to more images without increasing the time required to find a specific image.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous

Keywords

Interactive user interfaces, Content Based Image Retrieval (CBIR), Exploratory search

1. INTRODUCTION

In recent years, image retrieval techniques operating on meta-data, such as textual annotations or tags, have become the industry standard for retrieval from large image collections, e.g. Google Image Search. This approach works well with sufficiently high-quality meta-data, however, with the explosive growth of image collections, it has become apparent that tagging new images quickly and efficiently is not always possible. Secondly, even if instantaneous high-quality image tagging was possible, there are still many instances where image search by query is problematic. It might be easy for a user to define their query if they are looking for an image of a cat but how do they specify that the cat should be of a very particular shade of ginger with sad looking eyes.

A solution to this problem has been content based image retrieval (CBIR) [5, 12] combined with relevance feedback [24]. However, evidence from user studies indicates that relevance feedback can lead to a context trap, where the user has specified their context so strictly that the system is unable to propose anything new, while the user is trapped within the present set of results and can only exploit a limited area of information space [11]. Faceted search [22] was an attempt to solve the problem of context trap by using global features. However, the number of global features can be very large thus forcing the user to select from a large amount of options, which can make the whole process inconvenient and cognitively demanding. Employing various *exploration/exploitation* strategies into relevance feedback has been another attempt at avoiding the context trap. The exploitation step aims at returning to the user the maximum number of relevant images in a local region of the feature space, while the exploration step aims at driving the search towards different areas of the feature space in order to discover not only relevant images but also informative ones. This type of systems control dynamically, at each iteration, the selection of displayed images [18, 7].

However, in spite of the development of new techniques to support queryless exploratory image search, not much attention has been devoted to the development of interfaces to support this type of search [19]. Most research in CBIR interface design concentrates either on faceted search [20, 22] or enabling CBIR through a query image or a group of images [15]. In fact, most of the existing techniques and interfaces rely for exploration on iterative trial-and-error. All of the above techniques provide only limited support for the recent emerging trend of combining interactive search and recommendation [2]. One key question in this respect is how to utilise relevance feedback in optimising not only the narrowing but also the broadening of the scope of the search. We contribute to this problem with FutureView – an interface that supports queryless CBIR image search through more fluid steering of the exploration. This system uses a novel technique that allows users to preemptively explore the impact of the relevance feedback before operating a query iteration. We investigate in an evaluation whether this approach is useful in allowing users to explore more pictures. The evaluation of FutureView is carried out in a comparative user study and we conclude with implications for future development of image search systems that blur interactive search and recommendation.

2. RELATED WORK

Most image search systems still rely on search queries in order to return to the user a set of images associated with a tag related to the search query [1, 19]. There are also a number of alternative in-

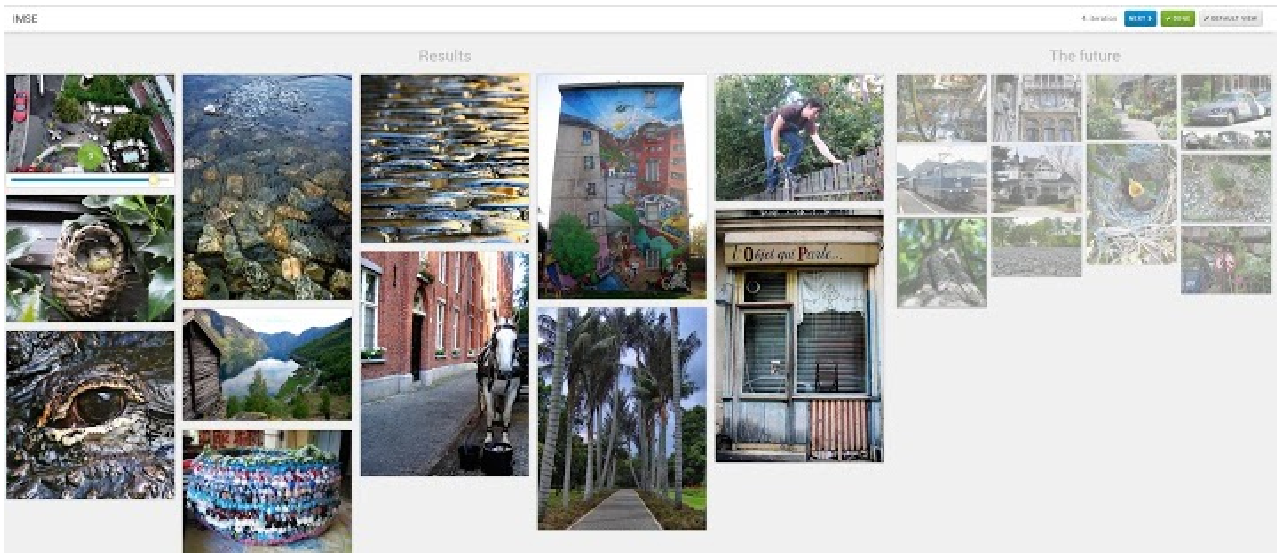


Figure 1: The FutureView interface: users can rate images on the panel on the left-hand side of the screen and the future view of the next iteration is presented on the right-hand side of the screen.

terfaces that group similar images based on various clustering techniques [21], or display similar images close to one another [14, 17, 16, 23]. However, all of these techniques rely on the availability of a dataset of tagged images or an automatic expansion of an initial textual query. Another approach is to rank images based on features extracted from a set of query images provided by the user [4, 6]. Faceted search [22] is another technique applied in CBIR to allow the user to browse through a collection of images using high-level image features, such as colour or texture. However, this approach often leads to a very large number of features, which can make the search process cognitively demanding.

3. OUR APPROACH

The main idea behind interactive interfaces used in most queryless exploratory CBIR systems [3, 13, 18] is that instead of typing queries related to the desired image, the user is presented with a set of images and navigates through the contents by indicating how "close" or "similar" the displayed images are to their ideal image. Typically, the user feedback is given by clicking relevant images or through a sliding bar at the bottom of the image. At the next iteration, the user is presented with a new set of images more relevant to his interest. The search continues until the user is satisfied with the results. Previous studies of CBIR systems show that this type of interface is intuitive and easy to use [3], however, users often feel that the new set of images does not reflect the relevance feedback they provided earlier: users do not feel fully in control of the system.

Our solution to this problem is an interface that provides the user with a "peek into the future". The FutureView interface, illustrated in Figure 1, is divided into two sections. The left-hand part of the screen is similar to a traditional interface, where the user can rate images by using a sliding bar at the bottom of each image. However, after rating one or more images, the user is not taken to the next search iteration but instead presented with the future view of the next iteration on the right-hand side of the screen. This allows the user to "try out" what impact providing feedback to different images will have on future iterations. When the user is satisfied with one of the future views, he clicks the "next" button in the right upper corner of the screen to confirm his choice and then is taken to the next search iteration.

4. EXPERIMENTAL STUDY

We conducted a comparative user study to evaluate the impact of FutureView on three types of image search tasks: target, category and open. The study included two conditions: 1) our FutureView interface; 2) a version of our interface without the future view, which from now on we will refer to as "single view". The same backend system was used with both user interfaces. We used as our backend an existing exploratory image search system, the details of which can be found in [9]. We also recorded the gaze behavior of the participants to determine how much time they spent observing the future during the FutureView condition. Gaze data was recorded during both conditions, and the participants were not informed that only the data in the FutureView condition would be used. We used the Tobii X2-60 eye tracker with sampling rate of 60Hz.

4.1 Participants

We recruited 12 post-graduate students from our university to participate in the study (3 female). The average age of the participants was 24 years (from 20 to 30). Google image search is the most frequently used images search tool by all the participants.

4.2 Design

We used the MIRFLICKR-25000 dataset with three types of features: colour, texture and edge, as described in [10]. We followed the most commonly used categorization of image search to design our tasks[3]:

- *Target search* - the user is looking for a particular image, e.g. a white cat with long hair sitting on a red chair.
- *Category search* - the user does not have a specific image in mind and will be satisfied with any image from a given category, e.g. an image of a cat.
- *Open search* - the user is browsing a collection of images

without knowing what the final target may look like, e.g. looking for an illustration to an essay about "youth".

We used a within subject design so that every participant performed three tasks covering all task types in both systems (six tasks in total = 3 (task types) $\times$ 2 (systems)). We designed two tasks for each category to assign unique task for each system. The subject of the two tasks for target search are: red rose, and tall building. In category search, we asked the participants to find images from the following categories: city by night, seashore. In open search, we asked the participants to imagine they were writing a newspaper article on a given topic and they had to find an image to accompany their article. The topics of the articles were: (1) happiness; (2) gardening. We selected these topics because they are well covered in the MIRFLICKR-2500 dataset. We showed 12 images per iteration in the single view interface and in Futureview. After receiving feedback, FutureView shows the next 12 images on the right-hand side.

4.3 Procedure

At the beginning of the experiment, we briefed the participants as to the procedure and purpose of the experiment before they signed the informed consent form. We then provided them with practice tasks to get them familiar with both systems. The participant would then proceed to perform six search tasks, divided into two groups of three tasks so that each participant would complete each different type of search task once with both systems. Before they started the target search tasks, we presented three example images and a short description of the image that they should look for. We did not provide any example images for category search and open search tasks. We randomized the order of tasks as well as the order of systems. After training, the eye tracker was calibrated.

We instructed the participants to finish each task when they find the target image (in case of target search) or when they feel they found the ideal image for the tasks from category search and open search. In all the tasks, we limited the search to 25 iterations to ensure that the participants did not spend an excessive amount of time on any task. After finishing each task, the participants completed the NASA TLX questionnaire [8]. After the completion all 6 tasks, we conducted a semi-structured interview with every participant to understand their overall satisfaction with the FutureView. A study lasted approximately 45 minutes. We compensated the participants with a movie ticket.

5. FINDINGS

Overall 12 users completed 72 tasks and all the participants completed all the tasks in fewer than 25 iterations. Figure 2 shows the average duration of a search session and the average number of images shown over a search session. On average, category searches were the shortest (104 seconds with single view and 109 seconds with FutureView), while open searches took the longest (145 seconds with single view and 140 seconds with FutureView). The Wilcoxon signed rank test indicates no significant difference in search session duration for any search type with the two interfaces ($p > 0.6$). In spite of the fact that no additional time is required to complete each type of search with FutureView, users are exposed to a much higher number of images – on average three times more than with single view. The Wilcoxon signed rank test shows that this number is significantly higher in open and target searches ($p < 0.05$) and marginally higher ($p = 0.05$) in category search with FutureView. These results indicate that FutureView supports more exploration.

Figure 3 shows the average scores of the NASA TLX questionnaire. In spite of the fact that with FutureView users were exposed to three times as many images as with the single view interface within the same period of time, users did not report feeling hurried, stressed or irritated. Similarly, users did not feel that FutureView made the task more mentally or physically demanding and they did not feel that they had to work any harder to achieve their goal. The Wilcoxon signed rank test indicates that there was not significant difference between the two interfaces in terms of scores for questions 1,2, 4, 5 and 6 ($p > 0.2$). The users, however, felt significantly more successful completing the task with FutureView ($p < 0.04$ according to Wilcoxon signed rank test).

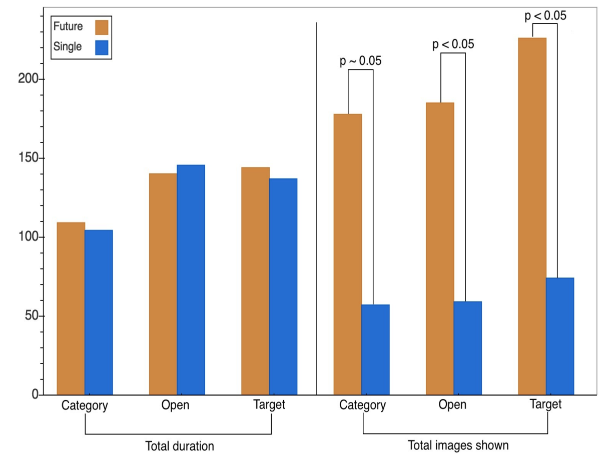


Figure 2: Average duration of a search session (in seconds) and average number of images shown over a search session for the three type of searches with a single view interface and FutureView

The eye tracking results show that the participants spent similar amount of time looking at both the current search results and the future view. Out of the 12 participants, three had excessive amount of errors in the eye tracking data, so only nine participants were considered. On average, the users spent 41.8% of the time looking at the future section of the screen, with standard deviation of 11.8%.

The post-experiment interviews with the participants also indicate that they found the FutureView interface helpful and easy to use. Some of the comments include: "The FutureView is pleasant to use and play with"; "The FutureView helps in reaching target quicker than the single view"; "The FutureView is helpful for people whose job is to search for images". These comments are in striking contrast to the remarks the participants made in the pre-study questionnaire, where they stated that most existing image search engines are tiring and cumbersome to use. The participants also remarked that "Single View can be discouraging as the user has no idea what is coming next", " once deviated from the actual path, there is no way to come back [in single view]".

6. CONCLUSIONS

In this paper, we introduced the FutureView interface for queryless exploratory content based image search. It allows the user to see the effect of the relevance feedback on currently presented images on future iterations, which, in turn, allows the user to direct their search more effectively. Initial experiments show that users take advantage of the FutureView interface and engage in more exploration than in a system with a single view interface.

Our future plans include more extensive user studies with various types of image datasets and various image feature representations.

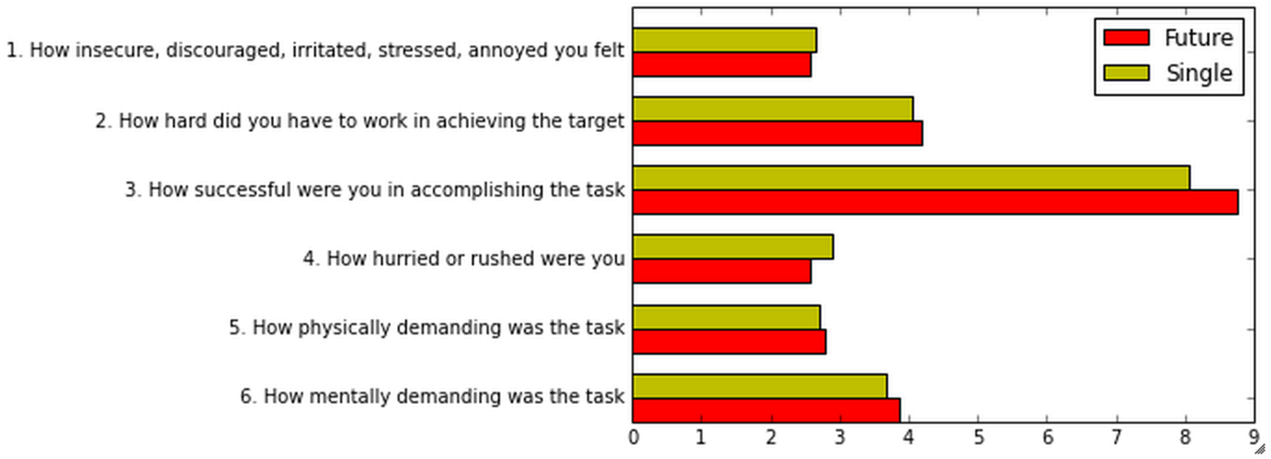


Figure 3: Average score for the NASA TLX questionnaire for tasks conducted with a single view interface and the FutureView.

Currently, the FutureView does not save the search history. We are planning to add this feature to our system to allow the user to branch out their searches using any point in the history as a new starting search point.

7. ACKNOWLEDGEMENTS

This work was supported by The Finnish Funding Agency for Innovation (projects Re:Know and D2I) and the Academy of Finland (the Finnish Centre of Excellence in Computational Inference).

8. REFERENCES

[1] P. André, E. Cutrell, D. S. Tan, and G. Smith. Designing novel image search interfaces by understanding unique characteristics and usage. In *Proc. of INTERACT*, 2009.

[2] E. H. Chi. Blurring of the boundary between interactive search and recommendation. In *Proc. of IUI*, 2015.

[3] I. Cox, M. Miller, T. Minka, T. Papathomas, and P. Yianilos. The bayesian image retrieval system, pichunter: theory, implementation, and psychophysical experiments. *Image Processing*, 9(1):20–37, 2000.

[4] J. Cui, F. Wen, and X. Tang. Real time google and live image search re-ranking. In *Proc. of MM*, 2008.

[5] R. Datta, J. Li, and J. Wang. Content-based image retrieval: approaches and trends of the new age. In *Multimedia information retrieval*, pages 253–262. ACM, 2005.

[6] J. Fogarty, D. Tan, A. Kapoor, and S. Winder. Cueflik: Interactive concept learning in image search. In *Proc. of CHI*, 2008.

[7] D. Głowacka and J. Shawe-Taylor. Content-based image retrieval with multinomial relevance feedback. In *Proc. of ACML*, 2010.

[8] S. G. Hart and L. E. Staveland. Development of nasa-tlx (task load index): Results of empirical and theoretical research. *Advances in psychology*, 52:139–183, 1988.

[9] S. Hore, L. Tervainen, J. Pyykko, and D. Glowacka. A reinforcement learning approach to query-less image retrieval. In *Proc. of Symbiotic*, 2014.

[10] M. J. Huiskes, B. Thomee, and M. S. Lew. New trends and ideas in visual concept detection: The mir flickr retrieval evaluation initiative. In *Proc. of MIR*, 2010.

[11] D. Kelly and X. Fu. Elicitation of term relevance feedback: an investigation of term source and context. In *Proc. of SIGIR*, 2006.

[12] H. Kosch and P. Maier. Content-based image retrieval systems-reviewing and benchmarking. *JDIM*, 8(1):54–64, 2010.

[13] J. Laaksonen, M. Koskela, S. Laakso, and E. Oja. Picsom–content-based image retrieval with self-organizing maps. *Pattern Recognition Letters*, 21(13):1199–1207, 2000.

[14] H. Liu, X. Xie, X. Tang, Z.-W. Li, and W.-Y. Ma. Effective browsing of web image search results. In *Proc. of MIR*, 2004.

[15] M. Nakazato, L. Manola, and T. S. Huang. Group-based interface for content-based image retrieval. In *Proc. of the Working Conference on Advanced Visual Interfaces*, 2002.

[16] N. Quadrianto, K. Kersting, T. Tuytelaars, and W. L. Buntine. Beyond 2d-grids: A dependence maximization view on image browsing. In *Proc. of MIR*, 2010.

[17] G. Strong, E. Hoque, M. Gong, and O. Hoeber. Organizing and browsing image search results based on conceptual and visual similarities. In *Advances in Visual Computing*, pages 481–490. Springer, 2010.

[18] N. Suditu and F. Fleuret. Iterative relevance feedback with adaptive exploration/exploitation trade-off. In *Proc. of CIKM*, 2012.

[19] B. Thomee and M. S. Lew. Interactive search in image retrieval: a survey. *International Journal of Multimedia Information Retrieval*, 1(2):71–86, 2012.

[20] R. Villa, N. Gildea, and J. M. Jose. A faceted interface for multimedia search. In *Proc. of SIGIR*, 2008.

[21] S. Wang, F. Jing, J. He, Q. Du, and L. Zhang. Igroup: Presenting web image search results in semantic clusters. In *Proc. of CHI*, 2007.

[22] K.-P. Yee, K. Swearingen, K. Li, and M. Hearst. Faceted metadata for image search and browsing. In *Proc. of CHI*, 2003.

[23] E. Zavesky, S.-F. Chang, and C.-C. Yang. Visual islands: Intuitive browsing of visual search results. In *Proc. of CIVR*, 2008.

[24] X. Zhou and T. Huang. Relevance feedback in image retrieval: A comprehensive review. *Multimedia systems*, 8(6):536–544, 2003.

An Adaptive Electronic Menu System for Restaurants

Paulo Henrique Azevedo Filho
Aberklar GbR
Helene-Mayer-Ring 7
80809, Munich, Germany
paulo.azevedo@aberklar.com

Wolfgang Wörndl
Technische Universität München
Boltzmannstraße 3
85748, Garching bei München, Germany
woerndl@informatik.tu-muenchen.de

ABSTRACT

This work shows the early stages of the development of a collaborative-filtering-inspired adaptive system to streamline the ordering process at restaurants that use electronic menu systems.

Among other results, the proposed system achieved a reduction of the session duration, while increasing feedback given by restaurant guests.

Categories and Subject Descriptors

H.4.m [**Information Systems Applications**]: Miscellaneous

General Terms

Human Factors, Economics

Keywords

Recommender Systems, Adaptive, Collaborative-Filtering, Electronic Menu, Target Variable

1. INTRODUCTION

There is an increasing trend in the number of tablet-based systems for ordering food, in order to streamline the service, increasing efficiency and profit. This is a great opportunity to make a change: instead of mere replacements for the paper menu and replacement for some of the roles of the waiters, such systems can do much more, such as becoming feedback-gathering devices, or helping guests choose their dishes better and increasing their satisfaction by offering meaningful suggestions.

Those meaningful suggestions may stem from collaborative filtering. Well-established algorithms for collaborative filtering exist, as well as several hybridization techniques blending this with other approaches. However, since users of electronic menus are mostly anonymous (no log-in necessary), and the offer of dishes and drinks varies greatly from restaurant to restaurant, the choice for a recommendation technique has to be further tweaked to address these issues.

The present work is an attempt to embrace this opportunity, building an adaptive recommender system for electronic restaurant menus, to aid guests in their ordering process, based on MenuMate.

MenuMate, developed by the German startup Aberklar[1], is an electronic menu that offers users a picture-centric approach for use at restaurants. Through MenuMate users can place orders, request the bill and provide feedback about their experience. This is the system that was extended with the adaptive system described in this paper. Figure 1 gives an idea of the general structure of the dish overview screen, and the arrows suggest the repositioning of dishes of the proposed method, that will be described soon.

The dish overview screen shows a title bar for each category, followed by thumbnails of each dish in that category. From that screen, the user can go to the dish details screen, with a full screen picture and detailed description, from where the dish can be ordered. After orders have been placed and the tab asked for, the user is invited to provide feedback about a few different variables, such as food, drinks, service and electronic menu system, using a star-based rating.

The rest of this document is organized as follows: Section 2 will put this work in perspective in terms of the electronic menu used for its implementation, as well as of the related work. Section 3 will describe in general terms the proposed algorithm and how it fits in. Section 4 describes some of the performed experiments and displays their results. Finally, section 5 offers a brief summary as well as points some possibilities of further work to be explored.

2. BACKGROUND AND RELATED WORK

To achieve the aforementioned goal, once the current state of MenuMate is known, it is necessary to know what the current state of the art for this specific niche is.

Wasinger et al. have proposed an electronic menu with an embedded recommender system, called Menu Mentor [9]. The authors come from a perspective of highly personalized explicit recommendations, processed on the user's phone, and that must be scrutable, that is, allow the users to know why a given recommendation (be it positive or negative) was

[1]www.aberklar.com/en

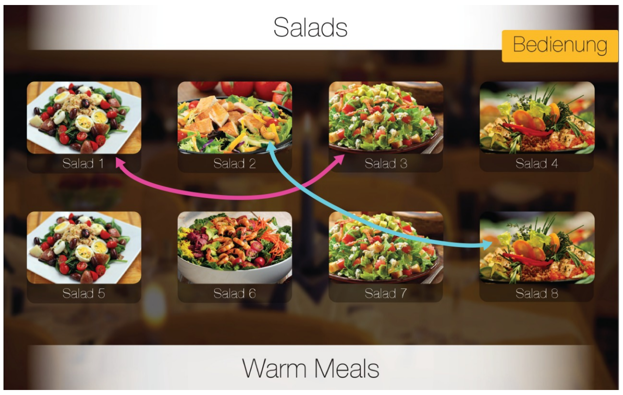


Figure 1: Screen of MenuMate with the overview of the dishes, as well as a suggestion of the dynamic reordering of the menu items.

given, and give them the chance to override it. It does, however, assume explicit user profiles, acquired through usage of the system on the user's smartphone. In order to minimize user setup, restaurants should be able to offer the system and the hardware.

There are psychological studies about how to organize a restaurant menu, with techniques such as placing items with high prices first to "smoothen" the effect of the lower-priced items following, even if they are not actually cheap [6]. There are also studies evaluating guests' gaze and how it correlates to which dishes get ordered, placing dishes restaurateurs want to have ordered in those positions, such as shown at [1]. There are as well other possibilities that go in a similar direction, such as [8] and [10]. Going for this sort of psychological study, however, would require extensive trials after every change to the menu, as well as the usage of the same menu in a different place where cultural background changes. This type of approach is also sensitive to the direction of reading of the mother tongue of the restaurant guest, such as right to left in case of Arabic speakers.

The goal of this work was to develop an electronic menu system that adapts itself according to a certain target variable, such as session duration, but while avoiding the need both for the setup of a user profile and extensive trials after each change. This is what will be presented in the next section.

3. ALGORITHM AND ARCHITECTURE

The proposed idea is rather simple. It consists in reordering the menu items to optimize a quantifiable target variable, that can be either maximized or minimized. For this work, four target variables were studied: tab value, session duration, revenue rate (cents per minute) and feedback rating. Once this definition is set, as different menu sequences are used the value of each target variable is recorded for each session, as well as the position of each dish in the menu. The optimal menu sequence is computed by calculating the correlation coefficient between each dish's position and the performance, and sorting ascending or descending, depending on whether the action is to maximize or minimize, respectively.

Four correlation coefficients were tested: Spearman's [4], Pearson's [5], Goodman and Kruskal's [2] and Kendall's [3]. In fact, the experiments tried all of them in different sessions and made a comparison between them. More coefficients could be used, the only requirement is that they must yield values between -1 and 1, which could imply a normalization step for coefficients that do not yield results in this range.

In order to vary the position of dishes, there is what we called pre-optimization randomization, which will shuffle the dishes before sorting them, giving the chance to dishes that have the same coefficient (within a small delta) to change places, changing not only absolute but also relative order. A bias could, otherwise, arise from the fact that stable sorting algorithms were used, thus keeping items with a similar correlation coefficient always in the same relative order, preventing them from switching places and moving far away from each other. Also, dishes for which there is no previous information always start with coefficient 0.

The basic principle is as follows: after each session, the tablets send the raw data gathered to the system that generates the sequence (there is one such system per restaurant). This system, based on the chosen coefficient and target variable, calculates the value for that variable and the coefficients, generates the menu sequence, and at the beginning of each session the tablets poll this system for the latest sequence to be employed. This system, called Menu Optimizer, is configurable in respect to the target variable, action (maximize or minimize) and correlation coefficient, together with other parameters relevant for the A/B testing performed, that will be explained in the next section.

4. PRELIMINARY USER STUDIES

Since the method proposed does not explicitly recommend a single item or try to predict ratings, some methods usually employed to evaluate recommender systems cannot be employed, such as cross-validation, recall and precision and accuracy.

There are, however, other methods that can be employed directly: the test was a double-blind A/B test, in which neither the user nor the waiter knew which the target variable at the time was, nor the correlation coefficient used. It was also online (in the sense that real users were using the system, generally spending real money through it [7], as was the case in our tests), and measured a few variables.

Since there is no way to directly measure accuracy for this system, there is employment of efficiency. In traditional recommender systems, the goal is to try to predict what the user would like to find and show it to them, shortening the search. Assuming that users find more easily the information they are looking for, they will more promptly take decisions based on that information, namely order the dish they intend to. Assuming this to be true, it would derive that a reduction of session duration would imply increased accuracy in a way. To assess whether this is true, it should be coupled with an increase of the feedback ratings, which would show that the items found satisfied the user. This metric, that as mentioned before, is called efficiency, and was used to assess the system.

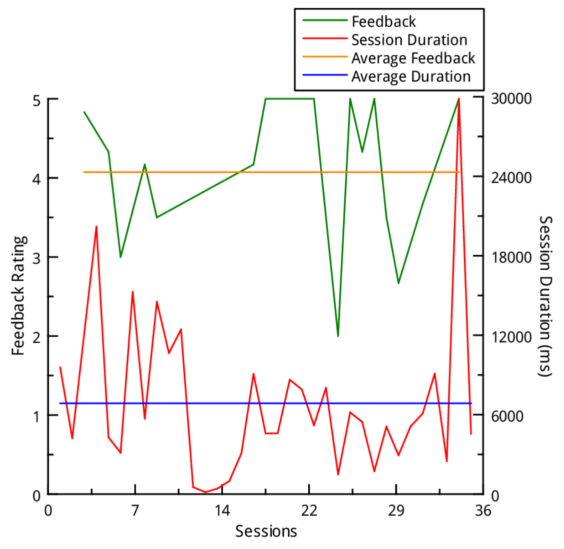


Figure 2: The evolution of feedback ratings and session duration at one of the restaurants, that suggests good efficiency.

Another adaptation made was with regards to serendipity. Instead of measuring it, there was the measurement of catalog coverage: the measurement of how much users tend to order varied dishes, compared to users of plain paper menu. For that, observations were made of tables whose guests did not use MenuMate, and which dishes they ordered, and those were compared to the orders of MenuMate users.

There were two classes of tests performed, the stress tests and the user tests. The stress tests, which will not be displayed in detail here, were used to assess the scalability of the system, which was developed to be run on a Raspberry Pi computer, with an embedded 900MHz ARM processor. It suffices to say that the system could serve between 100 and 160 tablets with unnoticeable performance losses, and that the limits reached were due to the test rig employed, rather than the system itself. Another interesting result on this front is that, on the employed hardware, menu optimization time is increased on average by 3ms for each session stored, which allows prediction of the optimization time based on the size of the history of observations.

The preliminary user studies were performed at two different restaurants in Munich, for about a week in each, time during which the system would gather session data and automatically adapt itself, switching the target variable at regular intervals. The number of observations was rather small due to the reduced number of days allowed for observations. There were 13 sessions observed in one restaurant, called El Patio, and 35 in the other, called Wendlinger. The restaurants had, respectively, 201 and 290 menu items, split in 24 and 34 categories. The original menu sequence was devised by their respective owners.

Figure 2 shows the evolution over time of the efficiency at Wendlinger (the number of sessions to which users gave feedback at El Patio was too low to assume it was meaningful). It suggests that indeed over time, independent of those being the target variables, feedback tends to rise and session duration to reduce, indicating good efficiency.

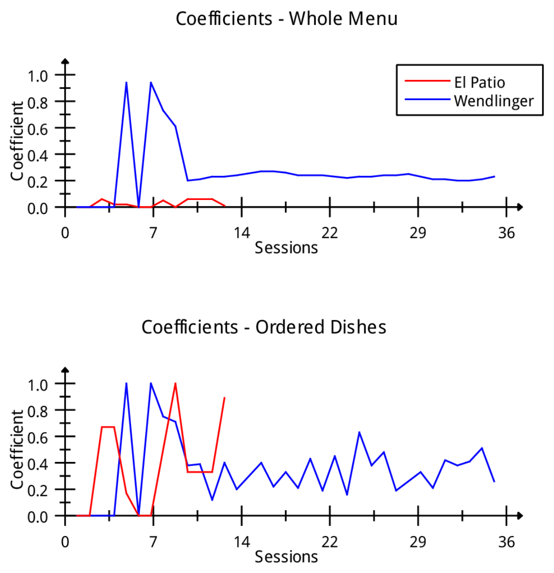


Figure 3: How the values of the correlation coefficients evolved over time at the two restaurants.

Table 1 shows how the choice of a target variable influences the performance of that variable, as well as of other variables. In this table only sessions optimized with Spearman's correlation coefficient are used, because it was the only coefficient with a high enough number of observations from which to derive conclusions (21 sessions in total). The "none" line, represents results for a non-optimized menu. The system did well in optimizing for the target variable, with the best results for both feedback and session duration, a close second for tab value, while not delivering good results for revenue rate, because the higher tab value was not enough to compensate for the roughly halved session duration time achieved when the target variable was the duration. It is also worth noticing that indeed, revenue rate is a direct consequence of both tab value and session duration, but the average revenue rate is not necessarily the same as the revenue rate of the averaged tab values and durations. For clarity, the best results for each variable are displayed in boldface.

Figure 3 shows another facet of the inner workings of the system: although the number of observed sessions was low, it is very fast to converge the internal coefficients, suggesting some stability after approximately 10 sessions. Another interesting effect is that the absolute values of the coefficients tend to be higher for dishes that ended up being ordered, which means the system can in fact predict which dishes will be ordered, by checking the dishes with highest absolute value of the coefficients.

Due to operational constraints, catalog coverage was only measured at El Patio. There, 12 of the 13 sessions were with menu optimization enabled, that will be considered for this measurement. 50 sessions with paper menus were also

Target Variable	Feedback	Value (€)	Duration	Rev. Rate
None	2,0	25,17	12.865,6	20,9
Feedback	**5,0**	13,25	15.779,9	33,8
Tab Value	4,4	38,00	7.499,5	59,1
Session Duration	4,3	36,85	**2.406,9**	**154,9**
Revenue Rate	**5,0**	**40,10**	4.400,6	56,6

Table 1: Cross-references between target variables and the results for all interest variables.

observed. In total, 97 different dishes and drinks were ordered, in different quantities. From those 97, 19 were ordered both with and without MenuMate, 63 only by users of the paper menu and 15 only by users of MenuMate with menu optimization.

It is not easy to extrapolate how those proportions would be in case an equal number of observations was available, and discarding paper menu-based sessions could arbitrarily lead to any results. Assuming, however, that for both systems at each session there is an equal probability of adding not previously ordered items, and that this probability is inherent to either MenuMate or the paper menu, a proportion rule may be followed.

In the case of the paper menu, $19 + 63 = 82$ different items were ordered in 50 sessions, which yields an average of 1,64 new items per session. With MenuMate in use, there were $19 + 15 = 34$ different items ordered in 12 sessions, resulting in 2,83 new items per session. That may indicate that thus, the menu optimizations led to guests ordering a bigger variety of items. Alternatively, it could be that as more sessions would be measured, these sessions would progressively get less "innovative", in terms of the ordered dishes, which could revert this balance. An extended evaluation, with a similar number of sessions in both conditions would help settle this matter.

5. CONCLUSIONS AND FUTURE WORK

The proposed menu optimizer harnesses principles of recommender systems to improve restaurant menus according to an arbitrary interest variable. In fact, it can be used to facilitate user interaction for any system to which access is anonymous and the number of items not overwhelming. Another use that comes to mind is the choice of which presentations to attend at a conference, or main sights to visit in a city.

One of the main contributions is the strong suggestion that this purely statistical method may improve the menu even if the underlying mechanism that drives the change is not understood by the system (i.e. the psychological implications).

The developed system comprises the algorithm for menu optimization, coupled with a robust and scalable implementation of it. It was followed by qualitative and quantitative tests, with real paying users, of which only very few results were presented this time due to page number constraints, but that nevertheless suggest efficacy of the proposed method.

There are, however, some promising improvements to the method. Among which, the highlights are:

- Feature extraction, to allow dishes of different restaurants to be matched and correlation information exchange, possibly improving results;
- Extra variable isolation, that would allow control over influential external variables such as weather, time of the day or season;
- If individual user profiling is done, further personalization, such as filtering out dishes based on allergies or taste preferences, could be done;
- Stochastic exploration of dishes, which could allow for recommendation of the next course based on previously ordered dishes in a session.

6. REFERENCES

[1] J.-G. Choi, B.-W. Lee, and J.-w. Mok. An experiment on psychological gaze motion: a re-examination of item selection behavior of restaurant customers. *Journal of Global Business and Technology*, 6(1):68, 2010.

[2] L. A. Goodman and W. H. Kruskal. Measures of association for cross classifications iii: Approximate sampling theory. *Journal of the American Statistical Association*, 58(302):310–364, 1963.

[3] M. G. Kendall. A new measure of rank correlation. *Biometrika*, pages 81–93, 1938.

[4] J. L. Myers, A. Well, and R. F. Lorch. *Research design and statistical analysis*. Routledge, 2010.

[5] K. Pearson. Notes on regression and inheritance in the case of two parents. In *Proceedings of the Royal Society of London*, volume 58, pages 240–242, 1895.

[6] W. Poundstone. *Priceless: The myth of fair value (and how to take advantage of it)*. Macmillan, 2010.

[7] G. Shani and A. Gunawardana. Evaluating recommendation systems. In *Recommender systems handbook*, pages 257–297. Springer, 2011.

[8] B. Wansink, J. Painter, and K. Van Ittersum. Descriptive menu labels' effect on sales. *The Cornell Hotel and Restaurant Administration Quarterly*, 42(6):68–72, 2001.

[9] R. Wasinger, J. Wallbank, L. Pizzato, J. Kay, B. Kummerfeld, M. Böhmer, and A. Krüger. Scrutable user models and personalised item recommendation in mobile lifestyle applications. In *User Modeling, Adaptation, and Personalization*, pages 77–88. Springer, 2013.

[10] S. S. Yang, M. M. Sessarego, et al. $ or dollars: Effects of menu-price formats on restaurant checks. *Cornell Hospitality Reports*, pages 6–11, 2009.

User Controlled News Recommendations

Jon Espen Ingvaldsen
Norwegian University of Science and Technology, Department of Computer and Information Science, Trondheim
Norway
jonespi@idi.ntnu.no

Jon Atle Gulla
Norwegian University of Science and Technology, Department of Computer and Information Science, Trondheim
Norway
jag@idi.ntnu.no

Özlem Özgöbek
Department of Computer Engineering, Ege University,
Izmir,
Turkey
ozlem.ozgobek@ege.edu.tr

ABSTRACT
The adoption of mobile devices is pushing the Internet into a more personal and context aware space. A common challenge for online news services is to deliver contents that are interesting to read. In this paper, we describe the user interface design of the SmartMedia news recommender prototype. Through deep analysis of textual news contents it is able to deliver local, recent and personalized news experiences, and the user interface is designed to give the users control over the news stream compositions. We will present its innovative user interface and the approach taken to transform raw textual data into well defined and meaning bearing entities.

Categories and Subject Descriptors
H.4.7 [**Information Systems Applications**] Communications Applications – *Information browsers*

General Terms
Algorithms, Design, Experimentation, Human Factors.

Keywords
Recommender system, news, mobile, user interfaces, user control

.

1. INTRODUCTION
The Smartmedia project[1] at NTNU targets construction of context aware news experiences based on deep understanding of text in continuous news streams [4, 9]. The goal of the Smartmedia project is to deliver a mobile and context aware news experience based on deep understanding of textual contents, combining both geo spatial exploration and context aware recommendations. The system is designed with scalability in mind and ability to support multiple languages.

Privacy is an important aspect when engineering recommender systems and exploitation of user interaction and context data. When dealing with personal data and privacy, transparency tools are tools that can provide to the concerned individual clear visibility of aspects relevant to these data and the individual's privacy. The combination of transparency tools and user control yields viable functionality to empower users to protect their privacy [5].

In the Smartmedia project, we want to build transparent news recommender systems where the user can control gathered data and how their news streams are composed based on geo spatial location, personal interest profile and time. When designing user-friendly systems for mobile devices, we need to be careful about the amount of buttons and menu items introduced. In this paper we will describe the news recommender system prototype and its mobile user interface where the users can control their news stream recommendations from three toggleable buttons.

[1] http://research.idi.ntnu.no/SmartMedia

2. IMPLEMENTATION
The backend of the news recommender prototype developed is constructed as a pipeline of operations harvesting and transforming Rich Site Summary (RSS) entries and raw text data into a semantic and searchable representation. The pipeline and its operations are implemented with using Apache Storm[2]. This distributed computing framework enable scalability and ability to handle large amounts of news items from a magnitude of publishers continuously.

As shown in Figure 1, the news processing pipeline consists of five steps. The first step creates an input stream by continuously monitoring a large set of RSS feeds. Whenever a new news item occurs, properties such as the title, lead text and HTML sources are extracted. The HTML sources are parsed and cleaned to extract a representative body text. In the second step, natural language processing operations such as language identification, sentence detection and part-of-speech tagging is applied to extract entity mentions from the textual data. The third step uses supervised models to map entity mentions to referent entities in the WikiData[3] and Geonames[4] knowledge bases. These models combine textual similarities, graph relations and entity frequency and co-occurrence statistics to classify the relevance of multiple referent candidates. First Story Detection (FSD) is applied in the forth step to group news items describing the same news story. In the fifth step this semantic representation is indexed and made searchable. As this backend architecture is stream based, it is able to index and promote recent news items.

WikiData is the community-created knowledge base of Wikipedia [12]. Since its public launch in 2012, the knowledge base has gathered more than 15 millions entities, including more than 34 million statements and over 80 million labels and descriptions in more than 350 languages [3]. Most geographical entities in WikiData provide a reference to Geonames containing more detailed geographical properties. In the implementation of the Smartmedia prototype, the news and entity information including news text, titles, publication timestamps, entity labels and

[2] http://storm.apache.org/

[3] https://www.wikidata.org/

[4] https://www.geonames.org/

geospatial properties are indexed in a Lucene based search index. This index makes the news items and their related entities searchable and creates a foundation for detailed querying.

When a user is opening the news app on the mobile a request containing user id, location and preferences are sent to the backend. Here, a multi factor search query is formed to retrieve relevant news entries from the index.

3. USER INTERFACE

A web-based user interface is developed to make the news stream contents explorable on mobile devices. In this interface, the user is allowed to extract news items that are relevant to the geo special locality context, personal interests and given point of time. These three relevance factors are customizable and the user can select whether or not they should influence the retrieval and ranking of available news items.

To customize the geographical locality, the user specifies a circular relevance region on a map. Figure 2a shows an example of such a relevance region. By default, the relevance region is set to users current GPS location with a 50 km radius. By moving the region or modifying the radius, users can generate a local newspaper for any region of the world. If the location factor is disabled, it means that the system is recommending news from any location in the world and news that are not containing location information.

In the current Smartmedia prototype, we have predefined a handful of user interest profiles. Examples of such profiles are stock trader, soccer fan, technology geek, etc. Each profile consists of a weighted concept vector, where each entry is a WikiData entry associated with an interest score between 0 and 1. By selecting any of these interest profiles, the retrieved news will be influenced and biased towards the interest topics. When the personal interest factor is disabled, the user retrieve a news composition which is general and without such bias.

To customize the time-factor, the user is presented with a calendar where it is possible to move in time and retrieve either recent or historic news items. When, the time-factor is disabled the user will retrieve news solely based on the other relevance factors (location and personal interests).

Figure 2b shows an example of how news stories are presented. Here we see one news article *"Theresa May urges media restraint in coverage of terror suspects"* from the Guardian about politics and terror, followed by another news story from BBC. The three circular buttons on the bottom of the screen allow users to toggle whether their locality, personal interest profile and time setting such influence news story retrieval.

By clicking on a news story, the user gets the ingress of the news story and a list of the most salient entities for the selected news story. Figure 1c shows the ingress and relevant WikiData entities from the news article about Theresa May. As we can see, our news story about politics and terror related to Syria, Theresa May, ISIL and Sky News. By hovering these items, the user is presented with their textual WikiData description. On figure 2c, we can see that the WikiData entity for Theresa May contains the description *"British politician"*.

In general, the three buttons at the bottom of the screen for location, interest profile and time can at any time be activated and de-activated to provide very different recommendation strategies. For example, keeping all buttons active with default parameters means that the system will recommend news articles that have recently takes place in the vicinity of the reader and are consistent with her profile. Figure 3 describes different combinations of recommendation factors and summarizes how the user can control the retrieval and composition of news items.

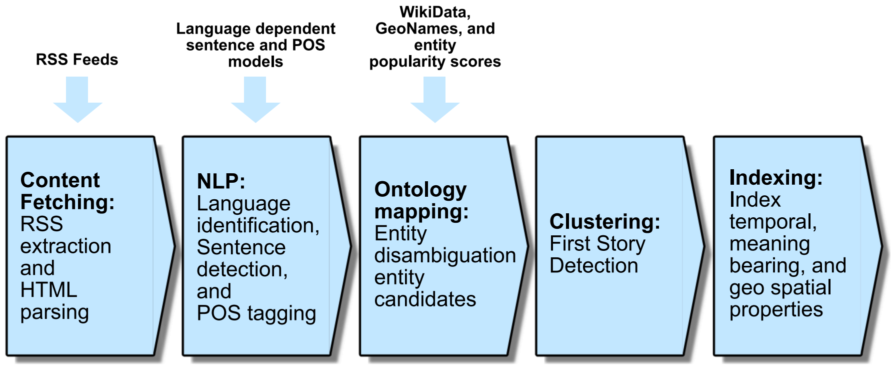
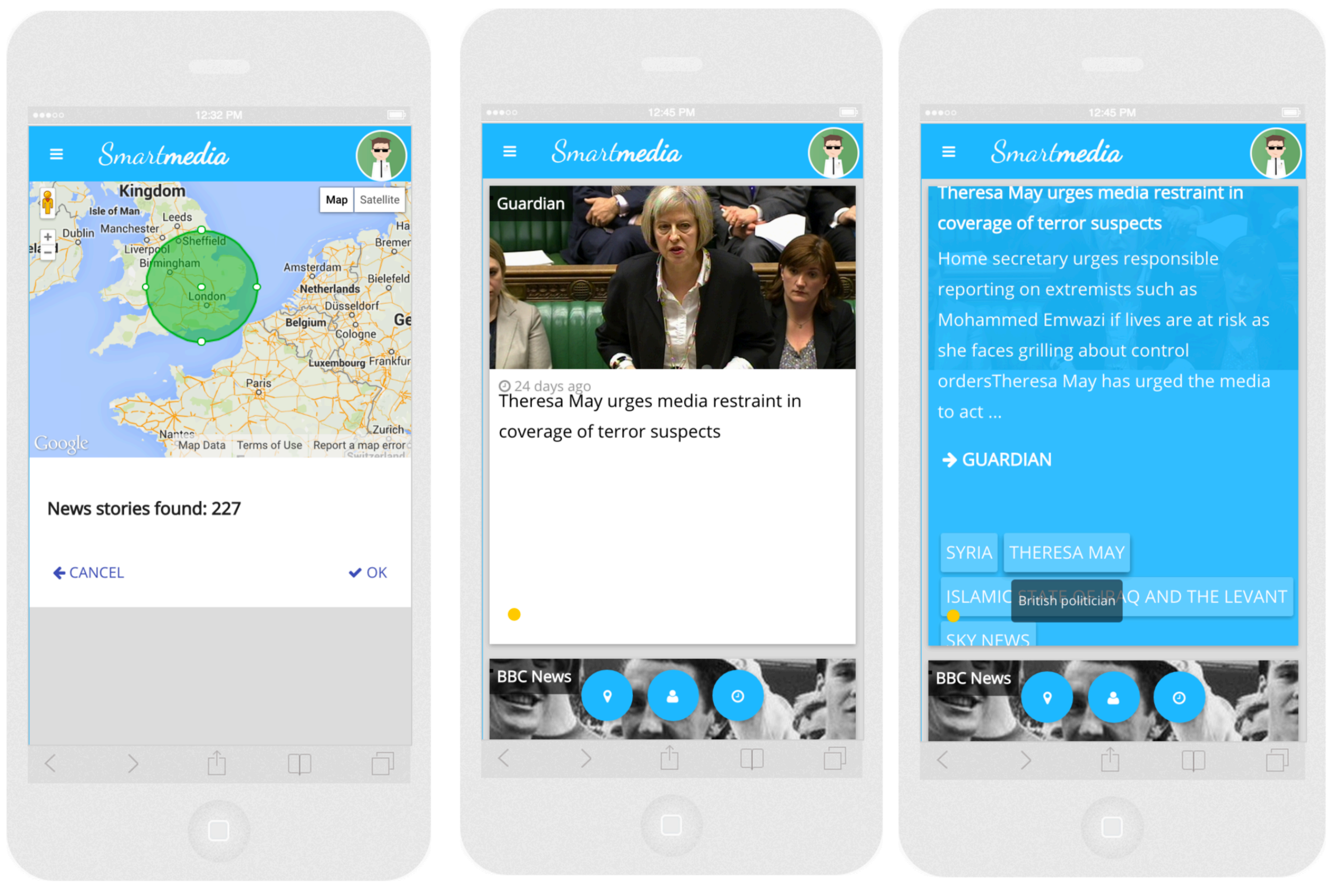

RSS Feeds
Language dependent sentence and POS models
WikiData, GeoNames, and entity popularity scores
Content Fetching: RSS extraction and HTML parsing
NLP: Language identification, Sentence detection, and POS tagging
Ontology mapping: Entity disambiguation entity candidates
Clustering: First Story Detection
Indexing: Index temporal, meaning bearing, and geo spatial properties

Figure 1. Steps of the news stream processing pipeline.



News stories found: 227

Theresa May urges media restraint in coverage of terror suspects

Theresa May urges media restraint in coverage of terror suspects

Home secretary urges responsible reporting on extremists such as Mohammed Emwazi if lives are at risk as she faces grilling about control ordersTheresa May has urged the media to act ...

a) b) c)

Figure 2. Screenshots from the Smartmedia prototype. a) The map query interface. b) Presentation of news stories. c) Presentation of news details.

4. RELATED WORK

People nowadays have access to more worldwide news information than ever before. As Internet services get more information about their users and their context, they can deliver personal and customized contents and user experiences.

The prototype system, described in this paper, share similarities to other academic news applications such as NewsStand [8, 10] and News@Hand [1, 2]. Both these systems map textual news contents to entities defined in a knowledge base.

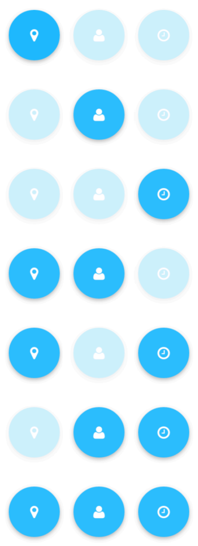
NewsStand targets geo spatial exploration of news. It is an example application of a general framework developed to enable people to search for information using a map query interface. It utilize maps both to explore and find news stories and to visualize and present single news events.

News@hand combines textual features and collaborative information to make news suggestions. It uses Semantic Web technologies to describe the news contents and user preferences. Both news items and user profiles are represented in terms of concepts appearing in domain ontologies, and semantic relations among those concepts are exploited to enrich the above representations, and enhance recommendations.

Both these NewsStand and News@Hand have user interfaces targeting desktops and larger device screens. They both provide user control over the retrieved set of news, either through a map or category based navigation or preferences settings.

Tran and Herder [11] have looked at the studied news event timelines and shown that manually constructed timelines are subjective and often missing important dates or other information. By complementing the timelines with elements extracted algorithmically from multiple sources, it is possible to create more objective and argumentative timelines. However, the manual processing and editing efforts are still needed to enhance the communicative qualities of the timelines, and to adapt it to the needs of the readers

Parra et al. [6, 7] presents SetFusion, a visual user-controllable interface for hybrid recommender system. Their approach enables users explore and control the importance of recommender strategies using an interactive Venn diagram visualization. Their evaluations indicate that this interface had a positive effect on the user experience and improved users engagement. Their idea of using the Venn diagram to explain intersections among recommendation approaches is transferable and valuable to the news domain.



Local
Retrieve news from a geospatial area

Personal
Retrieve news matching the interest profile of the user

Temporal
Retrieve news published after a given date

Local and personal
Retrieve news that both relate to a geospatial area and match users interest profile.

Local and temporal
Retrieve news related to a given geospatial area and published after a given date.

Personal and temporal
Retrieve news matching the interest profiles of the user and published after a given date.

Local, personal and temporal
Retrieve news with relevance for the selected geospatial area and interest profile, and published after a given date.

Figure 3. Combinations of selectable recommendation factors

5. CONCLUSIONS AND FUTURE WORK

The predefined user profiles can be replaced or used in combination with more personal profiles trained on traced interaction logs from the system. As users leave interaction data behind, we can gather knowledge about what the users interests are. However, for new users where no past interaction records exist, we have a cold-start problem where we still benefit on predefined stereotypes.

In future work we plan to use trained personal profiles with predefined stereotypes in combination. We will also gather user feedback and evaluate to which extent users want to control and customize their news presentations and study how their requirements can be met in a mobile user interface design.

Deep understanding of textual contents together with knowledge base structures provides a fundament for innovative and intelligent applications. This paper has described one such innovation from the news domain, and how its mobile user interface allow users to control the composition of news. A screencast video demonstrating the prototype and its user interface is available at: http://vimeo.com/121835936

6. REFERENCES

[1] Cantador, I. et al. 2008. News@ hand: A semantic web approach to recommending news. *Adaptive hypermedia and adaptive web-based systems*. (2008).

[2] Cantador, I. et al. 2008. Ontology-based personalised and context-aware recommendations of news items. *Proceedings of the 2008 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology*. 1, (2008).

[3] Erxleben, F. et al. 2014. Introducing Wikidata to the Linked Data Web. *The Semantic Web–ISWC 2014*. (2014).

[4] Gulla, J.A. et al. 2013. Learning User Profiles in Mobile News Recommendation. *Journal of Print and Media Technology Research*. II, 3 (2013), 183–194.

[5] Hansen, M. 2008. Marrying transparency tools with user-controlled identity management. *The Future of Identity in the Information Society*. (2008).

[6] Parra, D. et al. 2014. See what you want to see. *Proceedings of the 19th international conference on Intelligent User Interfaces - IUI '14* (New York, New York, USA, Feb. 2014), 235–240.

[7] Parra, D. and Brusilovsky, P. 2015. User-controllable personalization: A case study with SetFusion. *International Journal of Human-Computer Studies*. (2015).

[8] Samet, H. et al. 2014. Reading news with maps by exploiting spatial synonyms. *Communications of the ACM*. 57, 10 (Sep. 2014), 64–77.

[9] Tavakolifard, M. et al. 2013. Tailored news in the palm of your hand: a multi-perspective transparent approach to news recommendation. (May 2013), 305–308.

[10] Teitler, B. and Lieberman, M. 2008. NewsStand: A new view on news. *Proceedings of the 16th ACM SIGSPATIAL international conference on Advances in geographic information systems*. (2008).

[11] Tran, G. and Herder, E. 2015. Detecting Filter Bubbles in Ongoing News Stories. *Extended Proc. UMAP 2015*. (2015).

[12] Vrandečić, D. and Krötzsch, M. 2014. Wikidata: a free collaborative knowledgebase. *Communications of the ACM*. (2014).

Interaction Design in a Mobile Food Recommender System

Mehdi Elahi
Politecnico di Milano, Italy
mehdi.elahi@polimi.it

Mouzhi Ge
Free University of Bozen-Bolzano, Italy
mouzhi.ge@unibz.it

Francesco Ricci
Free University of Bozen-Bolzano, Italy
fricci@unibz.it

Ignacio Fernández-Tobías
Universidad Autónoma de Madrid, Spain
ignacio.fernandezt@uam.es

Shlomo Berkovsky
CSIRO, Australia
shlomo.berkovsky@csiro.au

Massimo David
Free University of Bozen-Bolzano, Italy
david.massimo@stud-inf.unibz.it

ABSTRACT

One of the most important steps in building a recommender system is the interaction design process, which defines how the recommender system interacts with a user. It also shapes the experience the user gets, from the point she registers and provides her preferences to the system, to the point she receives recommendations generated by the system. A proper interaction design may improve user experience and hence may result in higher usability of the system, as well as, in higher satisfaction.

In this paper, we focus on the interaction design of a mobile food recommender system that, through a novel interaction process, elicits users' long-term and short-term preferences for recipes. User's long-term preferences are captured by asking the user to rate and tag familiar recipes, while for collecting the short-term preferences, the user is asked to select the ingredients she would like to include in the recipe to be prepared. Based on the combined exploitation of both types of preferences, a set of personalized recommendations is generated. We conducted a user study measuring the usability of the proposed interaction. The results of the study show that the majority of users rates the quality of the recommendations high and the system achieves usability scores above the standard benchmark.

1. INTRODUCTION

Recommender systems are decision support tools that proactively identify and suggest items, which are expected to be interesting for the users. Recommendations are based on the users' previous interactions with the system and the explicitly provided users' preferences [15]. One important and new application domain for recommender systems is food. This application has recently drawn much attention in the research community due to its potential to improve eating behaviour of users and positively influencing their lives [8, 18, 20, 4, 11]. There is a broad spectrum of available information about food, such as recipe data and cooking instructions. Thus, some applications and websites already provide support functions allowing users to browse recipes and related information. However, most applications only offer generic and non-personalized recipe catalogue browsing support, without tailoring it to the tastes and preferences of individual users.

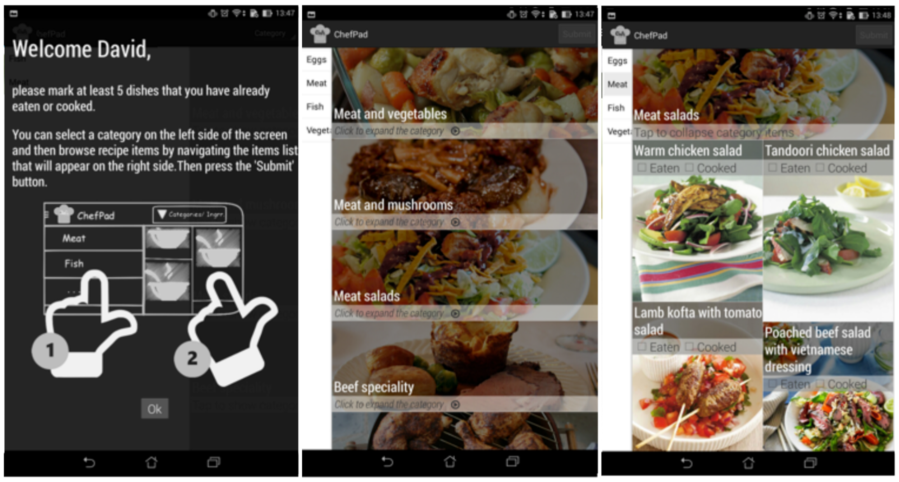
User preference elicitation is a fundamental and necessary step to go beyond this generic support and generate personalized recipe recommendations. More importantly than in other application domains, such as movies or books, recipe recommendations should not only be based on user's long-term tastes, but also fit their ephemeral preferences, such as the available ingredients or current cooking constraints.

In this paper we address this problem by proposing a preference elicitation approach for food recommender systems that obtains user preferences through a novel and effective interaction design. First, it exploits an integrated *Active Learning* algorithm [5, 6] for selecting the recipes to rate and tag that are estimated to be the most useful for the recommender. The active learning algorithm scores a recipe according to its predicted its rating (using transformed matrix of user-recipe) and then selects the highest scoring recipes. This reveals the users' long-term preferences, i.e., what they usually like to eat or cook. Second, when requested to generate recommendations, the system acquires short-term preferences referring to ingredients the user wants to cook or to include in the meal. The acquired preferences are used by a Matrix Factorization (MF) rating prediction model designed to take into account both tags and ratings [11, 13, 7].

In a real user study, we evaluated the proposed preference elicitation interaction and observed that the users have scored the usability of the system between "good" and "excellent" and assessed the presented recommendations, which are generated on the basis of the elicited preferences, to be of high quality.

Thus, the main contributions of our paper are: (a) a novel *interaction design* that is used to elicit long-term (general) and short-term (session-based) user preferences; and (b) an effective *preference elicitation* method that exploits active learning in the food recommendation domain.

.


Welcome David,

please mark at least 5 dishes that you have already eaten or cooked.

You can select a category on the left side of the screen and then browse recipe items by navigating the items list that will appear on the right side.Then press the 'Submit' button.

Figure 1: (a) user instructions, (b) browsing food categories, and (c) selecting eaten or cooked recipes.

2. RELATED WORK

Several recommender systems for the food domain have recently been developed [9, 18, 19, 20]. For example, Freyne and Berkovsky [9] proposed a food recommender that, through an easy-to-use interface, elicits user preferences and provides personalized recommendations Their system transferred the recipe ratings collected by the system to ingredient ratings and then aggregated the ratings of the ingredients used in a recipe to generate rating predictions.

Elahi et al. [4] proposed a food recommendation model that combines the predicted value of a recipe along different dimensions (user food preferences, nutritional indicators, and ingredients costs) to compute a single utility measure of a recipe. The goal is to consider factors influencing the user's food decisions in order to produce more useful and valuable recommendations. In a follow-up work [11], the authors conducted an offline evaluation of the rating prediction algorithm, which extends MF by using, in addition to ratings, the users' tags assigned to recipes. It was shown that this additional source of information about the user preferences allowed the proposed method to outperform other state-of-the-art algorithms, e.g., those proposed in [10].

In general, the user's preferences that are collected and used by a recommender can be either long-term (general preferences) or short-term (session-based and ephemeral). While obtaining both preference types is crucial, many recommender systems do not distinguish between the two. In fact, there are few studies that taken this consideration into account. Ricci and Nguyen proposed in [14] a mobile recommender system in travel domain, which elicits both general long-term preferences (e.g., explicitly defined by users) and short-term preferences in the form of critiques expressing more detailed session-based preferences. More recently, short term preferences were found to depend on the recommendation context and many context-aware approaches have been proposed to better suit the needs of the users [1].

It is worth noting that RSs research often focused on the improvement of the prediction model, by assuming that the preference elicitation process is completed. Hence, they ignore the complete user-system interaction, required for building a real-world recommender system. To address this limitation, this paper focuses on the interaction design, mainly for the preference elicitation: long-term and session-based.

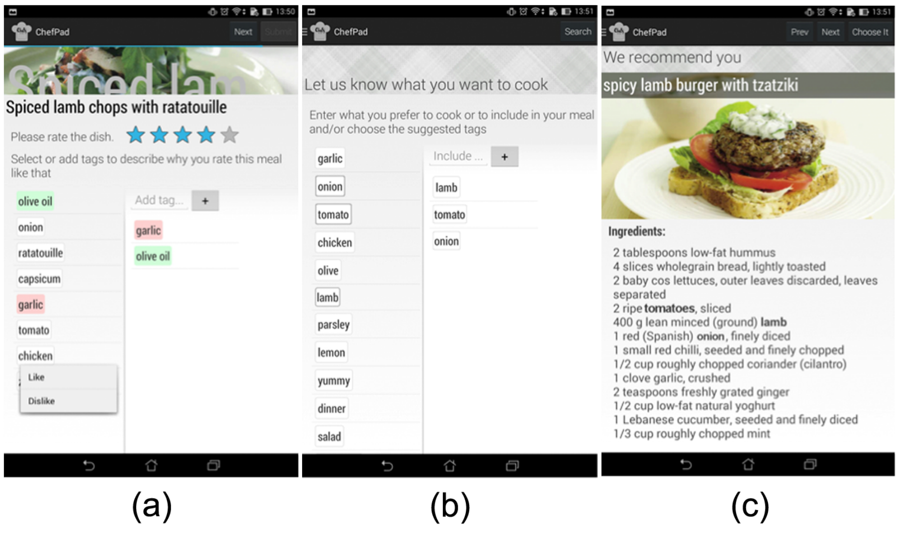
3. USER-RECOMMENDER INTERACTION

We designed a complete human-computer interaction for collecting user preferences, in the form of recipe ratings and tags [4]. An Android-based prototype was developed, in order to implement this interaction. The first step is a general preference elicitation, aimed at collecting the long-term (stable) user preferences, i.e., what she generally likes to cook (or eat). This step includes two stages: (1) the system asks the user to specify the recipes she cooks at home and, (2) the user assigns ratings and tags to the recipes she experienced.

Upon logging in the system, the user can browse the full catalogue of recipes and mark those that she has eaten before (see Figure 1). Users can navigate through the recipe categories and sub-categories in order to find the desired recipe, e.g., 'Beef' → 'Roasted Beef' → 'Roasted Beef with Salad'. Inside each category there is a list of recipes mapped to this category. When the user finds one of them she can mark it as 'Eaten'or 'Cooked' by clicking the check box.

After that, a selection of the recipes that the user marked as eaten or cooked, is presented to the user for rating and tagging. This allows the system to acquire knowledge about the general user preferences. However, the system also needs to deeper explore the user's preferences and it presents additional recipes for the user to rate and tag. These are found by predicting what the user might have eaten, but did not mark in the first step. In order to find such recipes, we use active learning. For this, the rating dataset is transformed into a binary format indicating only whether the user rated an item: null entries are mapped to 0, and not null entries to 1. Then, using a factor model, predictions are computed for all the values mapped to 0, and for each user the items with the highest prediction are shown to the user [5, 6].

Figure 2-a shows the rating and tagging interface. This interface uses the classical 5-star Likert scale. The users are also requested to "explain" the core motivations for their ratings by assigning tags to recipes. Users can either tag a



Spiced lamb chops with ratatouille

Please rate the dish.

Select or add tags to describe why you rate this meal like that

olive oil
onion
ratatouille
capsicum
garlic
tomato
chicken

Like
Dislike

Add tag...

garlic
olive oil

Let us know what you want to cook

Enter what you prefer to cook or to include in your meal and/or choose the suggested tags

garlic
onion
tomato
chicken
olive
lamb
parsley
lemon
yummy
dinner
salad

Include ...

lamb
tomato
onion

We recommend you

spicy lamb burger with tzatziki

Ingredients:

2 tablespoons low-fat hummus
4 slices wholegrain bread, lightly toasted
2 baby cos lettuces, outer leaves discarded, leaves separated
2 ripe **tomatoes**, sliced
400 g lean minced (ground) **lamb**
1 red (Spanish) **onion**, finely diced
1 small red chilli, seeded and finely chopped
1/2 cup roughly chopped coriander (cilantro)
1 clove garlic, crushed
2 teaspoons freshly grated ginger
1/2 cup low-fat natural yoghurt
1 Lebanese cucumber, seeded and finely diced
1/3 cup roughly chopped mint

Figure 2: (a) general preference elicitation, (b) session-based preference elicitation, and (c) recommendation.

recipe with the suggested tags or add their own tags. At the recommendation time, session-specific preference are elicited (see Figure 2-b). The user enters the core ingredient she wants to include in the recipe. This is done by selecting a keyword from the list of suggestions derived from food ingredients and popular tags assigned by other users.

Then, the recommendations leverage both types of the collected user preferences, long-term and session-specific. The long term preferences are exploited by a custom MF rating prediction model [13], which uses the tagging information [7]. Each user is associated with a vector that models her latent features and each recipe is modeled by a vector that contains its latent features. Then, the rating of a user for an item is predicted by computing the inner product of the user and item vectors. To exploit the short-term model, the system post-filters the recommendations according to the current user preferences. The recipes with the highest rating are presented to the user one by one. When the user selects a recommended recipe, the system presents the required ingredients and detailed cooking instructions (see Figure 2-c).

4. USER STUDY

The main goal of the evaluation was to assess whether the system can effectively assist users in finding recipes that suit their preferences. For the user study, we designed a usage scenario a task that was formulated as follows: "You want to avoid everyday routine meals. You can use this application to discover new recipes that suit your taste".

The users were asked to use the mobile application and complete a questionnaire referring to two performance indicators: perceived quality of recommendations quality and system usability. The first part of the questionnaire measured the level of user satisfaction with the recommendations. We used a validated instrument based on a set of questions developed by Knijnenburg et al. [12]. The second part of the questionnaire aimed at collecting the users' impression of the usability of the system. Here, we exploited the System Usability Scale (SUS) questionnaire [17]. The overall usability scores range from 0 to 100 and the bench-

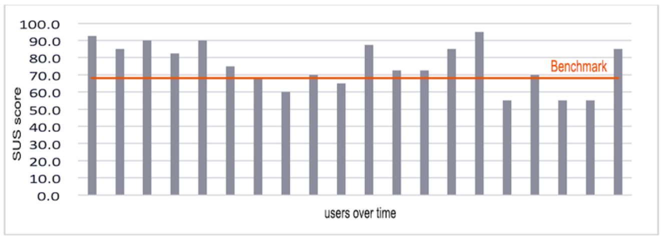


Figure 3: SUS results.

mark value is 68, which is the average SUS score computed over 500 usability studies [16].

In our experiment 20 subjects used the system and completed the questionnaire. They were either computer science researchers or non-academic people. 60% of subjects were male and 40% were female, the age range was 23 to 50, and the ethnical background varied across the subjects (Italy, France, USA, Germany, China, and more).

We first present the perceived recommendation quality results. The survey measures the recommendation quality using 7 questions on a Likert scale from 0 to 4, where 4 is the highest score. Thus, the maximum overall quality score is 28. The average perceived recommendation quality score across the 20 subjects was 19 and the median was 19 (see [3] for more details on the calculation). We observed that the maximal recommendation quality score was 26, and the minimal was 12. Thus, we can conclude that, on average, the users agreed that the recommendations were well-chosen and suited their preferences.

For the SUS usability score, we observed that for 75% of subjects the SUS score was higher than the 68 point benchmark (see Figure 3). The system achieved overall average SUS score of 75.50 and the median was 73.75, which is well above the benchmark. We observed that the minimal usability score of 55, and the maximal was 95. According to these results we can conclude that the system usability was considered between "good" and "excellent" [2].

We have computed the average replies for all the SUS statements and observing the statements with the highest average values, we can report that the users have evaluated the system easy to learn and easy to use. They also believe that various components were well-integrated into the system. On the other hand, by observing the statements with the lowest values we can state that users think that they have to learn a lot before they can use the system properly and they may need technical person for that. Our explanation for this result is that we need to improve further the interface and provide more explanations, so that users can better learn and understand the usage of the components in the system.

5. CONCLUSION AND FUTURE WORK

In this paper, we illustrated the preference elicitation process of a novel food recommender system [11]. Our system generates recommendations by exploiting tags and ratings in a MF algorithm. In our study, we collected user evaluations of the recommendation quality and system usability. Both measurements were found to be positive. This means that the proposed preference elicitation process and system interaction are liked by users.

Considering that this is a preliminary study, this paper has several limitations. First, the evaluation is performed on the whole system rather than on preference elicitation. Since the prediction model was already tested in another study [11], this work mostly focuses on preference elicitation as the main component of user interaction. Second, we have not compared our system with alternative preference elicitation processes. Our current result mostly reflects the users' direct perception of their interaction with the system. Third, we admit the limited number of subjects in the user study. In the future, we plan to increase the number of participants in the study. Also, we plan to extend the recommendation model by considering nutritional factors, e.g., the required calories and proteins, in order to build a health-aware recommender system.

6. REFERENCES

[1] G. Adomavicius and A. Tuzhilin. Context-aware recommender systems. In *Recommender systems handbook*, pages 217–253. Springer, 2011.

[2] A. Bangor, P. Kortum, and J. Miller. Determining what individual sus scores mean: Adding an adjective rating scale. *Journal of usability studies*, 4(3), 2009.

[3] M. Braunhofer, M. Elahi, F. Ricci, and T. Schievenin. Context-aware points of interest suggestion with dynamic weather data management. In *Information and Communication Technologies in Tourism 2014*, pages 87–100. Springer International Publishing, 2014.

[4] M. Elahi, M. Ge, F. Ricci, D. Massimo, and S. Berkovsky. Interactive food recommendation for groups. In *Poster Proceedings of the 8th ACM Conference on Recommender Systems, RecSys 2014, Foster City, Silicon Valley, CA, USA, October 6-10, 2014*. 2014.

[5] M. Elahi, F. Ricci, and N. Rubens. Active learning strategies for rating elicitation in collaborative filtering: A system-wide perspective. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 5(1):13, 2013.

[6] M. Elahi, F. Ricci, and N. Rubens. Active learning in collaborative filtering recommender systems. In *E-Commerce and Web Technologies*, pages 113–124. Springer International Publishing, 2014.

[7] I. Fernández-Tobías and I. Cantador. Exploiting social tags in matrix factorization models for cross-domain collaborative filtering. In *Proceedings of the 1st Workshop on New Trends in Content-based Recommender Systems, Foster City, California, USA*, pages 34–41, 2014.

[8] J. Freyne and S. Berkovsky. Intelligent food planning: personalized recipe recommendation. In *IUI*, pages 321–324. ACM, 2010.

[9] J. Freyne and S. Berkovsky. Intelligent food planning: personalized recipe recommendation. In *IUI*, pages 321–324. ACM, 2010.

[10] J. Freyne and S. Berkovsky. Evaluating recommender systems for supportive technologies. In *User Modeling and Adaptation for Daily Routines*, pages 195–217. Springer, 2013.

[11] M. Ge, M. Elahi, I. Fernaández-Tobías, F. Ricci, and D. Massimo. Using tags and latent factors in a food recommender system. In *Proceedings of the 5th International Conference on Digital Health 2015*, pages 105–112. ACM, 2015.

[12] B. P. Knijnenburg, M. C. Willemsen, Z. Gantner, H. Soncu, and C. Newell. Explaining the user experience of recommender systems. *User Modeling and User-Adapted Interaction*, 22(4-5):441–504, 2012.

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

[14] F. Ricci and Q. N. Nguyen. Acquiring and revising preferences in a critique-based mobile recommender system. *Intelligent Systems, IEEE*, 22(3):22–29, 2007.

[15] F. Ricci, L. Rokach, and B. Shapira. Introduction to recommender systems handbook. In F. Ricci, L. Rokach, B. Shapira, and P. Kantor, editors, *Recommender Systems Handbook*, pages 1–35. Springer Verlag, 2011.

[16] J. Sauro. Measuring usability with the system usability scale (sus). http://www.measuringusability.com/sus.php. Accessed: 2013-01-15.

[17] J. Swarbrooke and S. Horner. *Consumer behaviour in tourism*. Routledge, 2007.

[18] C.-Y. Teng, Y.-R. Lin, and L. A. Adamic. Recipe recommendation using ingredient networks. In *Proceedings of the 4th Annual ACM Web Science Conference*, pages 298–307. ACM, 2012.

[19] M. Trevisiol, L. Chiarandini, and R. Baeza-Yates. Buon appetito: recommending personalized menus. In *Proceedings of the 25th ACM conference on Hypertext and social media*, pages 327–329. ACM, 2014.

[20] R. West, R. W. White, and E. Horvitz. From cookies to cooks: Insights on dietary patterns via analysis of web usage logs. In *Proceedings of the 22nd international conference on World Wide Web*, pages 1399–1410. International World Wide Web Conferences Steering Committee, 2013.

Recommender Systems for the People — Enhancing Personalization in Web Augmentation [*]

Martin Wischenbart
CIS, Johannes Kepler University
Linz, Austria
martin@cis.jku.at

Sergio Firmenich,
Gustavo Rossi
LIFIA, Universidad Nacional de La
Plata and CONICET, Argentina
[firstname].[lastname]
@lifia.info.unlp.edu.ar

Manuel Wimmer
BIG, Vienna University of
Technology, Austria
wimmer@big.tuwien.ac.at

ABSTRACT

Web augmentation techniques allow the adaptation of websites on client side using browser extensions or plug-ins designed to run dedicated user scripts. However, while number and variety of such scripts from publicly available repositories have grown remarkably in recent years, they usually neglect the user's personal profile or individual preferences, and therefore fail to provide enhanced personalized services. At the same time recommender systems have become powerful tools to improve personalization on the Web. Yet, many popular websites lack this functionality, e. g., for missing financial incentive. Therefore, we present a novel approach to empower user script developers to build more personalized augmenters by utilizing collaborative filtering functionality as an external service. Thus, script writers can build recommender systems into arbitrary websites, in fact operating across multiple website domains, while guarding privacy and supplying provenance information. This paper discusses the architecture of the proposed approach, including real-world application scenarios, and presents our tool kit and publicly available prototype. The results show the feasibility of combining Web augmentation with recommender systems, to empower the crowd to build new kinds of applications for a more personalized browsing experience.

1. INTRODUCTION

In recent years web *augmentation* techniques, i. e., the addition of external content or behaviour to Web pages, have become a popular means for *end users* to adapt pages according to their own requirements, with reduced dependency on the website provider. Thereby, advanced users with knowledge of JavaScript, so-called scripters, write (user) scripts to modify web pages, which are then executed within the browser on client side – using dedicated browser plug-ins such as GreaseMonkey[1], and without the inclusion of the sites' webmasters. These user scripts are often publicly shared, and to date there are several large repositories providing a vast amount of various scripts for all kinds of Web pages and modification tasks. For instance, GreaseFork[2] has more than six thousand scripts, some of which are installed more than fifty thousand times. Another well-known repository, called UserScripts[3], hosts more than one hundred thousand scripts. Examples range from layout modification and tweaks (e. g., regarding video player size, video & audio customizations, etc.) on `youtube.com`[4], managing comments on `geocaching.com`[5], to improving navigation on `dropbox.com` by rendering a Tree View panel[6].

Unfortunately, as pointed out by a recent survey [8], and according to our own experience, current technologies for adapting the Web browsing experience still do not sufficiently support individual *personalization*, as it is provided by applications incorporating recommender system functionality. Consequently, with a single Web augmentation artifact (i. e., script) every user has the same experience.

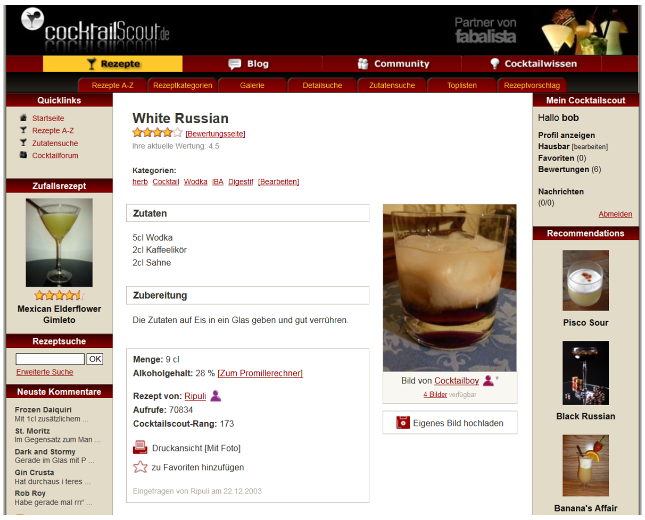
Recommender systems meanwhile have a longer standing history in various domains, such as e-commerce or music recommendations, and have become one of the most popular ways to personalize services and user experience. Commonly, they are classified [17] into content-based, collaborative, knowledge-based, as well as hybrid approaches. Although all these approaches rely on a user model, they differ in how they build this model, and they exploit different kinds of additional information and algorithms for presenting personalized recommendations of items to the user. For instance, collaborative approaches, also known as collaborative filtering or community-based, take into account the opinions of large amounts of users to make predictions about a specific user's preferences for items. Despite their potential, however, oftentimes websites do not implement recommendation services, either because of lack of economic incentives, or simply for lack of know-how on these techniques.

To alleviate these problems, we aim to introduce recommender system functionality for enhancing personalization in Web augmentation, combining the benefits of both approaches. To illustrate the use of collaborative filtering

[*]This work has been funded by grants Mincyt-BMWF AU/14/02, AU/14/09, ÖAD AR13/2015, ICM-2013-06624.

[1]`http://www.greasespot.net` (Firefox)
[2]`http://greasyfork.org`
[3]`http://userscripts-mirror.org`
[4]`http://greasyfork.org/es/scripts/943-youtube-center`
[5]`http://userscripts-mirror.org/scripts/show/75959`
[6]`http://greasyfork.org/es/scripts/4955-dropbox-plus/code`

for realizing Personalization in Web Augmentation Applications (PAA) we exhibit an example: The website `cocktailscout.de` is one of largest German language websites for cocktail recipes[7], having a community of almost 4000 registered users, who can search, rate and comment recipes for drinks. A rating mechanism is used for ranking of recipes, and averages and distributions of ratings for each drink can be viewed. Yet the users' individual ratings are not exploited to give them personalized recipe recommendations. Instead, the site provides a random link to a recipe on each page. Containing more than 1500 recipe items, however, the site is a perfect target for implementing a collaborative filtering recommender system. Based on the tastes of similar users, personalized recommendations for drinks could be presented in one of the sidebars, as shown in Figure 1.



White Russian
[Bewertungsseite]
Ihre aktuelle Wertung: 4.5

Kategorien:
herb Cocktail Wodka IBA Digestif [Bearbeiten]

Zutaten

5cl Wodka
2cl Kaffeelikör
2cl Sahne

Zubereitung

Die Zutaten auf Eis in ein Glas geben und gut verrühren.

Menge: 9 cl
Alkoholgehalt: 28 % [Zum Promillerechner]

Rezept von: Ripuli
Aufrufe: 70834
Cocktailscout-Rang: 173

Eingetragen von Ripuli am 22.12.2003

Bild von Cocktailboy
4 Bilder verfügbar

Frozen Daiquiri
Mit 1cl zusätzlichem ...
St. Moritz
Im Gegensatz zum Man ...
Dark and Stormy
Gerade im Glas mit P ...
Gin Crusta
Hat durchaus i teres ...
Rob Roy
Habe gerade mal rrr' ...

Hallo bob
Hausbar [bearbeiten]
Favoriten (0)
Bewertungen (6)
Nachrichten
(0/0)

Figure 1: Adopted `cocktailscout.de` website with personalized recommendations augmented (right).

Typically, the implementation of such functionality heavily depends on the *website provider*, and in the world of online shopping there is a lot of incentive for service providers to introduce such recommenders for increasing sales and profit. In contrast, for non-commercially oriented websites, or when items and monetization are not related directly, as with cocktail recipes, there is no such incentive for the site provider. For end users, however, recommender system techniques are commonly *beyond* the scope and to complex to be employed. Nevertheless, users of the CocktailScout website who are hobby JavaScript programmers, might be interested, and have the skills to implement a user script for utilizing the ratings and adding personalized recommendations of recipes. They do, however, lack access to a recommender system providing them with item recommendations.

Such a system could theoretically be implemented as a content-based recommender on client-side, given a catalog of items with features (such as a drink's ingredients), and the user's interests (such as preferences for ingredients). If no such item catalog is as available, however, the alternative, namely to collect all possible items manually in the Web augmentation script, seems to be a tedious task. For collaborative filtering, in contrast, such explicit *domain knowledge* is not required, but instead, only explicit or implicit user ratings (i. e., weighted relations between users and items) are required. Therefore, for the PAA approach we propose to share ratings and compute recommendations using collaborative filtering on a dedicated server, not least because of the current trend to cloud services, and protection of users' privacy. Consequently, providing a RESTful API for standard HTTP requests (and a corresponding object-oriented library in JavaScript, the prevailing language in the Web augmentation community), a recommendation service can be offered to the exemplary cocktail-drinking hobby JavaScript programmer. Providing a generic service, the approach has the potential to reach and benefit a large existing web augmentation community, and can be employed for *arbitrary websites*, under control of users and on client-side, and it may even go beyond single domains or the scope of a single site provider. Thereby, providing a simple and clear API and complete documentation is a key requirement, as we also discovered in the context of the composition language for building personalized recommenders and services in our research project TheHiddenYou [19].

In the upcoming Section 2 we discuss related work in the context of our approach. The following Section 3 presents an overview of the approach, including the tasks for scripters. Next, Section 4 introduces several real-world application scenarios, and demonstrates the approach as seen by the users. Section 5 elaborates on the architecture in detail, in particular the server part, and the PAA-API for scripters, which provides the functionality to store and retrieve item ratings, and to retrieve rating predictions and recommendations. In this context important issues are privacy and provenance, issues that are commonly disregarded in the Web augmentation community, but gain importance in the light of a central repository for collecting ratings. Finally, Section 6 discusses our prototype and some practical issues, before Section 7 presents conclusions and future work.

2. RELATED WORK

This section discusses related research in several fields, such as adaptive hypermedia, Web personalization, recommender systems, as well as Web augmentation, and finally compares our approach to similar approaches which are used in practice.

Research on Web personalization has been steadily growing, and in order to satisfy the huge number of end-users, several approaches for personalizing Web content have emerged, e.g. user profiling for personalization [14], or recommender systems [17]. In this context different ways for rating have been studied [24], and classifications of user feedback have been surveyed (including their correlation to ratings) [18].

Although usually recommenders work on server-side, some approaches for client-side personalization have been developed [2, 16]. In such scenarios, since different Web applications can share a single user profile (e.g. managed on client-side using a browser extension), and recommendations may cover different sites. Regardless whether personalization mechanisms work on server or client side, these mechanisms are usually specified by website owners, and they are always limited by the information available on the user profile. Meanwhile browser extensions monitoring user navigation can be used to populate user profiles (with navigation history, bookmarks, keywords, etc.) and thereupon recom-

[7]Highest global Alexa rank (`http://www.alexa.com/`) in a comparison of 13 German language cocktail websites.

mend relevant Web pages to users [10, 12]. Although there are some works aiming to define and extract [25] comprehensive profiles, and analyze their interoperability [5], it is difficult to implement an adaptation mechanism which is broad enough to contemplate every user requirement, especially while protecting privacy [20] and providing provenance information [21].

Web augmentation techniques are another way to achieve personalization; augmentation allows users to customize website user interfaces (UIs) in terms of content and functionality, according to their own requirements [11]. Most Web augmentation approaches are developed as browser extensions, and once installed by the user, they modify loaded Web pages, thus altering what the user perceives. In this way, end users with programming skills are the ones creating Web augmentation artifacts. However, most recent research about Web augmentation do not target personalization, but aim to provide tools (frameworks, or languages) to solve domain-specific adaptations (i.e., support recurrent tasks, automate tasks, improve accessibility, etc.) or raise the abstraction level in order to allow more users (without advanced programming skills) to specify how they want to augment their preferred Web sites. For instance, CSWR [13] aims to improve Web accessibility, and WebMakeUp [9] allows end-users to specify their own augmentations. All these approaches propose a way to customize the Web, but most of them work without an underlying user profile. Web augmentation may be employed for guiding the user through content, whereas the navigation mechanisms are not implemented by the content provider himself. Using a collaborative system for recommending items on the Web, in this context, represents a 'social mechanism' with an 'open corpus of documents'.

A similar kind of systems to adapt existing third-party Web content are intermediaries [3], which intercept the content in a proxy server and not on client-side. From our point of view, and in comparison with intermediaries, Web augmentation approaches are usually more powerful as adaptation mechanisms. Web augmentation tools usually extend the Web Browser, and consequenlty these tools give more information about the users activity than a those systems working on a proxy server.

Some authors have proposed personalization as a service [15], and nowadays there are several companies offering rating[8] and recommendation[9] as Web services. Despite similarities to the proposed PAA approach in terms of the employed technology, these approaches require changes in the "original" Web site, and these changes need to be performed by the provider, for instance, by integrating a JavaScript library. Furthermore, these approaches require an upload of a complete product catalog beforehand[9], thereby causing additional maintenance effort. In our proposed PAA approach, this catalog is built on-the-fly, i.e., product details are pushed to our repository alongside with ratings. Finally, whereas in recommendation as a service, concrete recommendations are rather generic; we argue that a scripter who is an active member from both Web site community and Web augmentation community may have further insight to exploit domain knowledge about human decision making in that community (cf. [6]) for giving item recommendations.

[8]Rating-Widget: `http://rating-widget.com/`
[9]Strands: `http://retail.strands.com/`

3. THE APPROACH IN A NUTSHELL

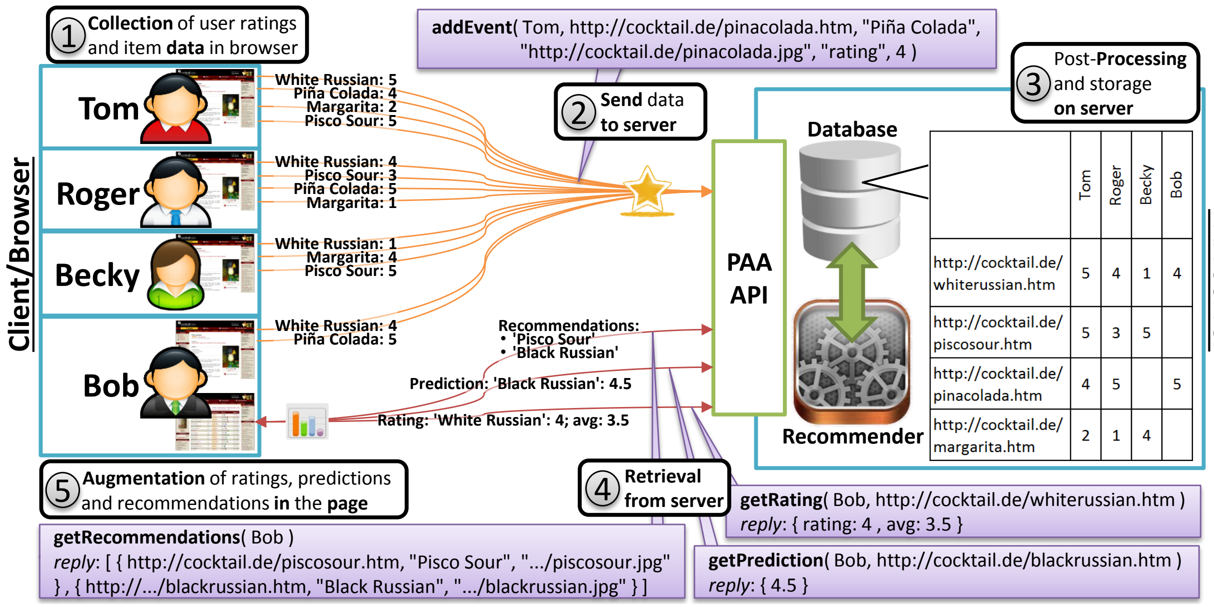
This section gives an overview of the approach, in particular as seen by the script writer, referring to the example outlined in the introduction. The complete architecture and technical details about the server will be explained in Section 5. In short, the PAA approach supports the scripter to make adaptations to the website, as outlined in Figure 2. At first ratings are collected in the browser by the scripter ① and sent to the server using dedicated API methods ②. After processing on the server ③, another set of methods may be used to retrieve previous ratings, rating predictions, as well as recommendations ④, to be finally employed to modify the page ⑤, for instance for link ordering, link hiding, link annotation, or link generation, as classified by Brusilovsky [4]. In the following, the five steps are outlined in detail.

Data Collection ①: Scripters may rely on an existing rating mechanism to measure the user's interest in an item. If no such mechanism exits, it can be implemented by the scripter, either in terms of an explicit rating (e. g., 1-5 stars, like vs. dislike, etc.) or implicitly computing a score (e. g., based on page visit, time spent on the page, activities such as posting comments or uploading pictures). For modeling the user's interest, we rely on *events* relating *users* with *items* (both identified with unique IDs) and including a numeric score for *rating*. For presentation purposes later on (cf. ⑤), here we also require additional features about an item, including a human readable *name*, or *meta-info* such as an image, to be shown as link in the web browser. For the use-case regarding cocktails, this means we will need the drink's URL, its name, an image URL, as well as a user ID and the numeric rating from the page. *Tasks for the scripter:* To extract all this information, the scripter usually reads the Web page's document object model (DOM): 1. Extract a unique user ID from DOM (or rely on user's login on the PAA server's web interface; cf. Sect. 5); 2. Extract a unique item ID from DOM; 3. Extract further information about the item from DOM, such as name (to be used as link text), or links (to be used for image links); 4. Extract the rating value, or compute a numeric rating from collected explicit or implicit user feedback (or rely on default scores of predefined event types; cf. Sect. 5).

Send to Server ②: As a next step, the previously collected data must be *sent* to the server, to ultimately *collect* a large number of such events as basis for the recommender algorithm. *Tasks for the scripter:* Send data to the server via the API, using our provided JavaScript library (parameter string, or object-oriented), or using HTTP POST requests.

Processing on Server ③: On the server events are stored, a *timestamp* is added, and they are processed to be in a format for being used by the recommender (cf. Sect. 5).

Retrieval from Server ④: As a next step, queries may be performed from the script, either for any page on the site or pages representing or containing items (i. e., automatically), or on demand of the user (i. e., manually, cf. 'pull recommendations' [23]). The following data can be retrieved from the server: firstly, previously stored ratings including meta-information, such as average ratings and their distributions; secondly, predictions for user ratings; and finally, recommendations for items, with the latter two being *computed on-the-fly* using a recommender system library on the server. *Tasks for the scripter:* Via several dedicated API methods, the scripter can query this information from the server, again using our provided JavaScript library or HTTP



addEvent(Tom, http://cocktail.de/pinacolada.htm, "Piña Colada", "http://cocktail.de/pinacolada.jpg", "rating", 4)

	Tom	Roger	Becky	Bob
http://cocktail.de/whiterussian.htm	5	4	1	4
http://cocktail.de/piscosour.htm	5	3	5	
http://cocktail.de/pinacolada.htm	4	5		5
http://cocktail.de/margarita.htm	2	1	4	

getRating(Bob, http://cocktail.de/whiterussian.htm)
reply: { rating: 4 , avg: 3.5 }

getRecommendations(Bob)
reply: [{ http://cocktail.de/piscosour.htm, "Pisco Sour", ".../piscosour.jpg" } , { http://.../blackrussian.htm, "Black Russian", ".../blackrussian.jpg" }]

getPrediction(Bob, http://cocktail.de/blackrussian.htm)
reply: { 4.5 }

Figure 2: Overview of the five steps of the proposed PAA approach as seen by end users and script writers, including a simplified representation of the communication between client and server.

POST requests.

Augmentation in Page ⑤: The previously retrieved information may finally be augmented in the page. Previous and average ratings as well as their distributions may be shown as additional information on the item page, or for *link annotation* (cf. [4]), e. g., as popups for all links referring to drinks. Rating predictions may be employed for *re-ordering* items or links on the page, or, if the predicted score is below a certain threshold, for *hiding* a link (cf. [4]). Finally, the provided list of recommendations can be used to *generate* (cf. [4]) personalized links on the page, referring to items the user could be interested in, e. g., as a list of recommended drinks. *Tasks for the scripter:* The desired results are displayed on the page by modifying the DOM. For showing drink recommendations, this can be achieved by using the response from the server to add elements and set their properties. The complete script source code for this example is available online with our prototype (cf. Sect. 6).

4. APPLICATION SCENARIOS

This section elaborates on several user script use-cases that can be realized with our approach, and focuses on benefits for the end user.

Cocktail Recipes – Recommendation Scenario. In the previous sections and example we based on the example of `cocktailscout.de`, a site which has almost 4000 registered users and more than 1500 recipe items, and an existing rating mechanism. Only by exploiting these ratings on the PAA server, personalized drink *item recommendations* can be *generated* for the user.

Bookstore – Rating Mechanism. The Argentinean online bookstore `cuspide.com` provides functionality to comment on books, but does neither offer ratings nor personalized recommendations. In such scenarios without a re-usable rating mechanism, a rating widget may be added to the page (cf. Fig. 3), to show items *annotated* with previous ratings

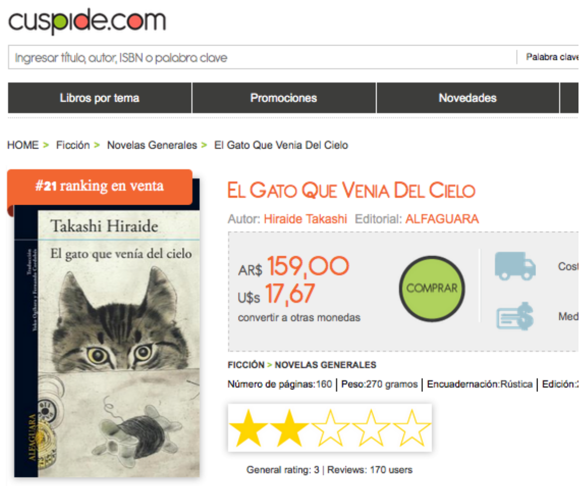
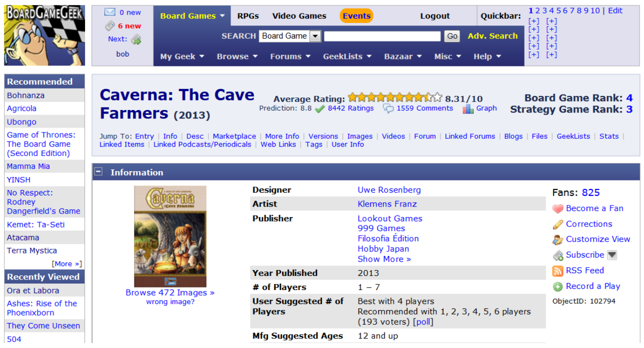

cuspide.com
Ingresar título, autor, ISBN o palabra clave
Palabra clave
Libros por tema
Promociones
Novedades
HOME > Ficción > Novelas Generales > El Gato Que Venia Del Cielo
#21 ranking en venta
Takashi Hiraide
El gato que venía del cielo
ALFAGUARA
EL GATO QUE VENIA DEL CIELO
Autor: Hiraide Takashi Editorial: ALFAGUARA
AR$ 159,00
U$s 17,67
convertir a otras monedas
COMPRAR
FICCIÓN > NOVELAS GENERALES
Número de páginas:160 | Peso:270 gramos | Encuadernación:Rústica | Edición:
General rating: 3 | Reviews: 170 users

Figure 3: Example modified bookstore website with an augmented rating mechanism (bottom).

and statistics, as well as to *generate* personalized recommendations. For the scripter this scenario is slightly more complicated, since it requires the definition of what items can be rated in the first place, i. e., a definition of items to be processed by the user script.

Board Games #1 – Using Predictions. In addition to annotation and generation of links, to improve the visibility of relevant items for the user, *rating predictions* may be exploited for link *hiding* and *re-ordering*. This is particularly useful for sites with a large catalog of items, such as the board gaming community website `boardgamegeek.com`, with around 77.000 games grouped by publishers, artists, as well as various categories and families (and more than one million of registered users). Even though a rating mecha-



Recommended
Bohnanza
Agricola
Ubongo
Game of Thrones: The Board Game (Second Edition)
Mamma Mia
YINSH
No Respect: Rodney Dangerfield's Game
Kemet: Ta-Seti
Atacama
Terra Mystica
[More »]

Recently Viewed
Ora et Labora
Ashes: Rise of the Phoenixborn
They Come Unseen
504

Caverna: The Cave Farmers (2013)

Average Rating: 8.31/10
Prediction: 8.8 ✓ 8442 Ratings 1559 Comments Graph

Board Game Rank: 4
Strategy Game Rank: 3

Jump To: Entry | Info | Desc | Marketplace | More Info | Versions | Images | Videos | Forum | Linked Forums | Blogs | Files | GeekLists | Stats | Linked Items | Linked Podcasts/Periodicals | Web Links | Tags | User Info

Information

Browse 472 Images »
wrong image?

Designer	Uwe Rosenberg
Artist	Klemens Franz
Publisher	Lookout Games 999 Games Filosofia Édition Hobby Japan Show More »
Year Published	2013
# of Players	1 – 7
User Suggested # of Players	Best with 4 players Recommended with 1, 2, 3, 4, 5, 6 players (193 voters) [poll]
Mfg Suggested Ages	12 and up

Fans: 825
Become a Fan
Corrections
Customize View
Subscribe
RSS Feed
Record a Play
ObjectID: 102794

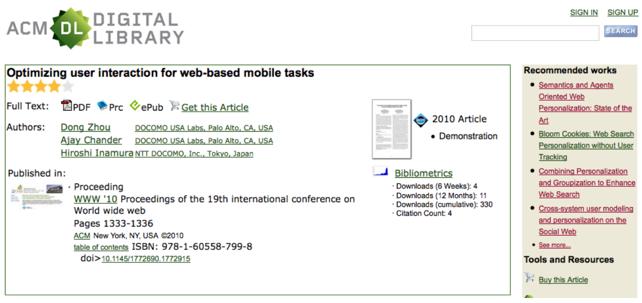
Figure 4: Example modified BoardGameGeek website with augmented recommendations (left) and rating prediction (center/top).

nism exits, personalized recommendations of games are not given to users. Instead, the site displays a list of 50 currently 'hot' games. To personalize this list using the PAA approach, e. g., it can be re-ordered according to rating predictions for these items and users, and games having a particularly bad prediction (e. g., below a certain threshold) can be hidden.

Board Games #2 – Feedback & Event Types. As an alternative to the previous scenarios, where users are required to give explicit ratings, other kinds of user feedback may be exploited, such as explicit feedback without numeric ratings, or implicit user feedback. On `boardgamegeek.com`, besides explicit ratings, users can become fan of a game, subscribe, tag, record plays, add games to collections, and much more. In addition to these explicit events, implicit feedback may be recorded directly in the browser (i. e., behaviors exhibited by the user while using the site; cf. [18] – PAA does not require a special browser for this functionality, but it can be implemented as scripts). Based on this, a script can compute a single numeric rating value to be communicated to the server. Alternatively, to relieve the scripter of this task, we further propose a mechanism to allow putting multiple events to the server, and compute an aggregated single rating value to estimate the user's interest in an item on server-side. Thus, for the scripter the task of computing a rating value is broken down to recording different user actions (i. e., events), and defining how the events should be accumulated to a collective score (sum, average, median, logarithmic, etc.). Details about how this is achieved will be explained in detail below in Section 5.

Scientific Literature – Cross Domain. Finally, since Web augmentation scripts can be defined to run on multiple websites, going beyond the scope of a single content provider, the PAA approach enables personalized link recommendations across multiple domains. This feature is specially useful for those Web sites sharing an underlying domain. For instance, for scientific literature search, a script can track user activities such as page visits and downloads on sites such as ACM DL, Springer, IEEE Xplore, or Science Direct[10]. In every site of this list, the same item (Paper) may be defined with similar properties (url, title, authors, abstract). Based on events collected from these user activities, a recommendations pane can be added to each of these websites to present potentially relevant scientific literature

[10]`http://dl.acm.org/`, `http://link.springer.com/`, `http://ieeexplore.ieee.org/`, `http://www.sciencedirect.com/`



Optimizing user interaction for web-based mobile tasks

Full Text: PDF Prc ePub Get this Article

Authors: Dong Zhou DOCOMO USA Labs, Palo Alto, CA, USA
Ajay Chander DOCOMO USA Labs, Palo Alto, CA, USA
Hiroshi Inamura NTT DOCOMO, Inc., Tokyo, Japan

2010 Article
- Demonstration

Published in:
- Proceeding
  WWW '10 Proceedings of the 19th international conference on World wide web
  Pages 1333-1336
  ACM New York, NY, USA ©2010
  table of contents ISBN: 978-1-60558-799-8
  doi>10.1145/1772690.1772915

Bibliometrics
- Downloads (6 Weeks): 4
- Downloads (12 Months): 11
- Downloads (cumulative): 330
- Citation Count: 4

Recommended works
- Semantics and Agents Oriented Web Personalization: State of the Art
- Bloom Cookies: Web Search Personalization without User Tracking
- Combining Personalization and Groupization to Enhance Web Search
- Cross-system user modeling and personalization on the Social Web
- See more...

Tools and Resources
Buy this Article

Figure 5: Example modified literature search website with augmented recommendations (right).

from any of the sites to the user, as shown exemplary in Figure 5.

5. ARCHITECTURE

As it was explained in the introduction, on client-side dedicated plug-ins enable the execution of user scripts to manipulate the DOM in the browser. Such user scripts, written in JavaScript language, may execute HTTP requests to external RESTful APIs. This is shown in Figure 6, alongside with the proposed PAA architecture and components of the server, which will be explained in more detail in the following. In PAA, requests can be made manually using standard scripting tools (e. g., functions provided by Firefox or GreaseMonkey), or using our JavaScript library, which provides functions for sending parameters concatenated as a single strings (e. g., `userName=Bob&rating=4&...`), or in an object-oriented manner. The latter option is shown exemplary in Figure 7, including code to extract the parameters from the DOM, and the interal library implementation is presented in Figure 8.

Processing of Events. As mentioned in step ③ in Section 3, events that are stored on the server are equipped with a timestamp, before all parameters are checked for validity, and events are stored to the database. Furthermore, provided `userNames` and item IDs (`itemIri`[11]) are mapped to numeric IDs, and stored along with the `rating` and the timestamp in a separate table to be accessed directly by the recommender engine.

Different Types of Events. To distinguish between different types of events, such as explicit numeric ratings and other kinds of feedback, the scripter can choose to supply a custom `eventType` (e. g., `'pictureUpload'`). Except for the distinction, this allows us to provide a default rating value for several pre-defined event types (following a classification of observable user behaviour found in literature [18]). These default ratings, however, are currently assigned in unscholarly manner, and thus can also be overridden by the scripter.

Rating Accumulation. Whereas events represent user actions, for the recommender engine we rely on ratings only. For this, the rating values provided with the events may be used directly. Defined by a parameter `insertAs`, first, events may simply be added as `'new'` ratings (i. e., several ratings per user for a single item are provided to the recommender engine). Second, the rating specified by the event may `'replace'` the previous one. Finally, as an alternative multiple events may be accumulated – in a way that can be configured by the scripter – to compute the rating value to

[11]commonly URIS/IRIS (`http://tools.ietf.org/html/rfc3987`)

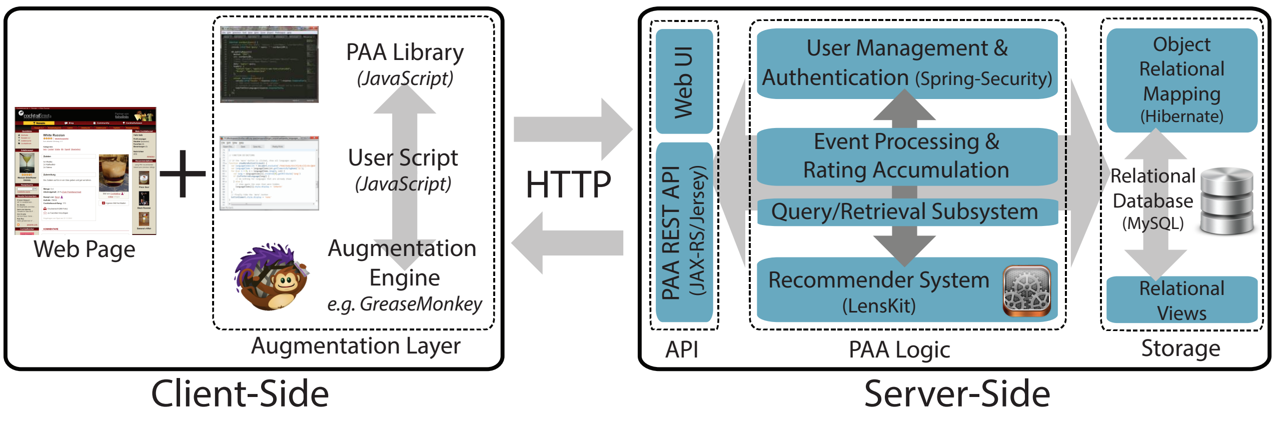


Figure 6: Architecture of the proposed PAA approach: client-side, server-side, communication via HTTP.

```
item_name = document.querySelector(".recipe-detail...")
item_iri = window.location.pathname;
```
Item data obtained from DOM

```
item = paa.item(item_name, item_iri);
item_img = document.querySelector(".recipe-image").src;
item.addMetaData("img",item_img);
```
Item creation and initizialization

```
username = document.querySelector(".mycs_username..."
user = paa.user(username);
```
User creation (username obtained from the DOM)

```
user_rating_value = document.querySelector(".curr..."
user = paa.rating(user_rating_value);
```
Rating creation (value obtained from DOM)

```
event = paa.event(user, item, rating);
paa.sendEvent(event);
```
Event creation and sending to server

```
paa.getRecommendations(user,receiveRecommendations);
```
Obtain recommendations and handle the result with *receiveRecommendations()*

Figure 7: Example JavaScript code for creating and storing an event object, and for retrieval of recommendations using the object-oriented PAA library.

```
getRecommendations:function (user, callback) {
  var pluginName = GM_info.script.name;
  var pluginNamespace = GM_info.script.namespace;
  var domain = window.location.host;
  var recsData = this.serializeRecommendation(user,
                 pluginName,
                 pluginNamespace,
                 domain)
  GM_xmlhttpRequest({
        method: "POST",
        url: this.getRecommendationsURL,
        data: recsData,
        headers: {
          "Content-Type": "application/x-www-form...",
          "Accept": "application/json"
        },
        onload: function(response) {
             callback(response)
        } }); }
```
Define and serialize data (current script, website, user) for requesting recommendations from the server

Perform the request and execute the callback function (defined by the scripter) to handle the server's response (i.e., recommendations)

Figure 8: Example internal JavaScript library function of client-side API to request recommendations.

be used by the recommender. Since user scripts are typically run once per page load, this may be a particularly useful option to track implicit user feedback in terms of potentially many events regarding a single item. For this accumulation, ratings may be derived as '`sum`', '`average`', or '`median`' of event scores. While summing or averaging, weights for older events may be decreased, e. g., linearly or logarithmic. Furthermore, the `aggregationScope` can optionally be limited to a certain number of most recent events, or using a time window. As a result, the rating scale must allow float values, and consequently also the ratings provided from the user script can be continuous, allowing different scales or binary ratings. For the future we are planning extensions, and we aim to make this computation more customizable, for instance, to enable that different event types contribute to the accumulation in different manners.

Authentication & Web UI. User authentication is not required if while making API calls the user script provides a `userName`. Clearly this is security issue, because in this basic form there is no authentication involved. This option was enabled for the sake of simplicity, and for simple scenarios where security is not an issue. Alternatively, instead of providing a user name, the parameter `paaAuthRequired` can be set to `true`, enabling authentication via the PAA website (the current status can be queried using `getLoginStatus()`, to open the login form if necessary). Once logged in, the corresponding cookie is being included by the browser when making calls to the server's API. Via the Web user interface, users can furthermore manage their profile data and view coarse grained provenance information. Thereby, the user may be provided with insights to what data is being stored and how it is used, including information on why specific recommendations were given. This increases transparency and helps to gain the users' trust, and to satisfy users who might normally be reluctant to share personal data with providers of (commercially oriented) personalization services.

Recommender Engine on Server. For adding events, as well as for queries to the recommender engine, every API query must provide the script's `pluginName`, `pluginNamespace`, and the site `domain`. These define the operation domain for the recommender, i. e., the view for ratings that are seen by the recommender engine (also user names must be unique within this operation domain). For computing predictions and recommendations the current generic prototype implementation relies on the open-source framework LensKit[12], configured to perform item-item collaborative filtering for scoring items (cf. [7, 22], LensKit doc.[13]), and directly accessing the database for building models.

Provision to Client. Finally, for answering queries regarding ratings, predictions, and recommendations, again user names and item IDs are mapped to numeric IDs and vice versa, and previously stored `itemNames` and `meta` infor-

[12]`http://lenskit.org/` – Recommender Toolkit
[13]`http://lenskit.org/documentation/algorithms/item-item/`

mation (stored per event) are added where applicable. Example responses are shown online along with the prototype, as discussed in the following section.

6. PROTOTYPE & EVALUATION

The implemented PAA prototype is publicly available online[14], including our object-oriented JavaScript library with example scripts and instructions for their usage. The implementation bases on Jersey[15] for handling RESTful HTTP requests, and employs Spring Security[16] for authentication, which also allows the future integration of identity providers such as OpenID (supported by Google) or Facebook Connect, to lower the barrier for acquiring users. In general, persistence is realized with Hibernate[17], however, using relational views LensKit[12] accesses the underlying database directly.

The prototype shows the feasibility of *combining* Web augmentation with recommender systems. It was not yet optimized for performance (*prediction accuracy* and *response times*), yet it shows that the approach is suitable for empowering scripters to build new kinds of augmentations.

Since we base on established item-item collaborative filtering algorithms, the evaluation of prediction accuracy was not a goal for this paper. However, we do plan to do this in the future, along with an evaluation of our methods for computing the cumulative score from different event types. In this context, an interesting question is how well does the chosen LensKit configuration generalize for all kinds of items, and how can it be made configurable by the scripter.

Concerning response times, on one hand, the approach is limited by the connection to the server through Internet, however, with optimizations possible, for instance by providing API methods to combine several HTTP requests into one (e. g., storage of multiple events as a batch, or combining the storage of an event with a query for recommendations), or by reducing the amount of redundantly transferred event meta information. On the other hand, the computation of recommendations on the server is currently made on demand for each request, but could be made configurable, such that the scripter may define whether recommendations should be computed in advance, or if they have to reincorporate all the latest added events.

As with collaborative filtering recommenders in general, the known cold start problem is an issue to be considered, also for script writers. To mitigate this problem, in situations where ratings are publicly available, the scripter may add them as a batch, e. g., by extracting user ratings from `cocktailscout.de`, or using the XML API of `boardgamegeek.com`. Additionally, to support early development of new scripts, our prototype can be configured to append random items to the list of recommendations, until the recommender algorithm can compute enough actual ones.

Finally, so far our approach does not foresee the removal of items. In an open environment, where every end user can add items, maybe role-based user management, a reputation based approach, or a voting mechanism can be used to decide which items to remove. In this context, it also may happen that websites change their URL structure. While this is something that every Web augmentation approach has to deal with, this may cause additional database maintenance efforts if the IDs of items are affected.

7. CONCLUSIONS & FUTURE WORK

We have shown the *feasibility* for providing collaborative filtering recommender system functionality as a *service* for Web augmentation. By providing a simple API and corresponding object-oriented JavaScript library, *user script writers* may employ ratings, predictions, as well as recommendations of items, to develop *new* kinds of recommender *applications* for arbitrary websites. Thus, end users benefit from personalization of the Web browsing experience through annotation, *re-ordering*, *hiding*, and *generation* of links. Several real world world application scenarios were discussed, and we presented our publicly available server prototype.

To improve and further evaluate the proposed approach, we foresee several lines of potential *future work*. Therefore, we are currently working on an extended prototype, mainly to add some minor features, improve response times, and add further means of configuration for the scripter. In a profound evaluation we aim to cover the issues outlined in the previous section: recommendation relevancy, the server-side computation of scores, response times and their effect on users' browsing experience, and methods to cope with the removal of items or changing URLs. In this kind of systems another very well known issue is the cold start problem, in this sense, we plan to provide scripters with some mechanisms to define how the script adapts the Web page when the available ratings are not enough. A possibility we are studying is to allow a scripter to use a single user profile in several scripts. In this context, we also plan to conduct several user studies, first, with scripters to evaluate the comprehensibility of the API for implementing pre-defined tasks, and how the provided service is being accepted. Second, we intend to evaluate performance, quality of recommendations, as well as scalability, in terms of a larger case study with users, employing experts or a number of end users, e. g., from the `boardgamegeek.com` community. There, publicly available ratings can be exploited to reduce the cold start problem, or to build a dataset for an offline evaluation of the employed recommender algorithm. Finally, with a complete prototype and documentation, downloads of our scripts and usage of the API can be evaluated in long term study.

In terms of functionality, we foresee several *expansions*. Support for different data formats for the API (e. g., JSON, XML, RDF) allows the reduction of technical heterogeneity, and enables simpler integration with arbitrary websites – also to give the scripters more options. Moreover, using RDF facilitates the use of semantic Web technologies and resources, not least to supply further structured information about items from resources such as DBpedia. Together with user profile data and item features collected in the browser, or extracted from online social networks, this information may be exploited by the recommender algorithms, for instance to better determine user-user or item-item similarities, or for improving recommendation results with content-based and hybrid recommender mechanisms on server side. In this context it is particularly interesting how well the current implementation recommends newly added items or items with few ratings, and if the algorithms can be made configurable for end users or scripters, to include items from the *long tail* for increasing serendipity in recommendations. In this context, the recommender system algorithms may

[14]`http://paa.cis.jku.at/`
[15]`http://jersey.java.net/` – JAX-RS Reference Impl.
[16]`http://projects.spring.io/spring-security/`
[17]`http://hibernate.org/` – Object Relational Mapping

also be configured via the API in a more elaborate way, for instance using LensKit's Groovy-based DSL[18]. Furthermore, feedback on recommendations may be exploited to improve recommendations (i. e., when the user follows a recommended link, or rates the corresponding item afterwards), and item recommendations that are constantly ignored may be excluded in the future. The set of items considered by the recommender may also be defined by the scripter, for instance, by providing a specific set or filter criteria via the API, as with a Recommendation Query Language (cf. [1]).

To *decentralize* the approach, giving even more control to the user script writer, we plan to investigate how the server functionality can be hosted by a *generic* "platform as a service" *provider* of cloud computing *services*, and if the API functionality can be provided as a downloadable and configurable *bundle*. Going beyond the server side approach, the question remains whether the architecture can be re-implemented to be independent of a single third party recommendation service provider, with recommendations being computed on client-side, and exchanging anonymized data via a dedicated data exchange server only, or relying on a peer-to-peer architecture.

Finally, we foresee the implementation of a browser plug-in to offer a graphical user interface for configuration and augmentation of recommendations, to be used by scripters or even end-users, based on a DSL to *define extraction and placement* of item and user information in the DOM (cf. [11]).

8. REFERENCES

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

[2] A. Ankolekar and D. Vrandecic. Kalpana - enabling client-side web personalization. In *Proc. of HT'08 - Hypertext 2008*. ACM, 2008.

[3] R. Barrett and P. P. Maglio. Intermediaries: New places for producing and manipulating web content. *Computer Networks*, 30(1-7):509–518, 1998.

[4] P. Brusilovsky. Adaptive navigation support. In *The Adaptive Web*, volume 4321 of *LNCS*. Springer, 2007.

[5] F. Carmagnola, F. Cena, and C. Gena. User model interoperability: a survey. *User Modeling and User-Adapted Interaction*, 21(3), 2011.

[6] L. Chen et al. Human decision making and recommender systems. *ACM Trans. Interact. Intell. Syst.*, 3(3), 2013.

[7] M. Deshpande and G. Karypis. Item-based top-n recommendation algorithms. *ACM Trans. Inf. Syst.*, 22(1), 2004.

[8] O. Díaz and C. Arellano. The augmented web: Rationales, opportunities, and challenges on browser-side transcoding. *ACM Transactions on the Web*, 9(2), 2015.

[9] O. Diaz et al. Towards the personal web: Empowering people to customize web content. In *Web Information Systems Eng.*, volume 8786 of *LNCS*. Springer, 2014.

[10] D. Eynard. Using semantics and user participation to customize personalization. Technical report, HP Labs, 2008.

[11] D. Firmenich, S. Firmenich, J. Rivero, and L. Antonelli. A platform for web augmentation requirements specification. In *Web Engineering*, volume 8541 of *LNCS*. Springer, 2014.

[12] X. Fu, J. Budzik, and K. J. Hammond. Mining navigation history for recommendation. In *Proc. of 5th Int. Conf. on Intelligent User Interfaces*, IUI '00, New York, NY, USA, 2000. ACM.

[13] A. Garrido, S. Firmenich, G. Rossi, J. Grigera, N. Medina-Medina, and I. Harari. Personalized web accessibility using client-side refactoring. *Internet Computing, IEEE*, 17(4), 2013.

[14] S. Gauch, M. Speretta, A. Chandramouli, and A. Micarelli. User profiles for personalized information access. In *The Adaptive Web*, volume 4321 of *LNCS*. Springer, 2007.

[15] H. Guo, J. Chen, W. Wu, and W. Wang. Personalization as a service: The architecture and a case study. In *Proc. of 1st Int. Workshop on Cloud Data Management*, CloudDB '09, New York, NY, USA, 2009. ACM.

[16] Hendry, H. Pramadharma, and R.-C. Chen. Building browser extension to develop website personalization based on adaptive hypermedia system. In *Current Approaches in Applied Artificial Intelligence*, volume 9101 of *LNCS*. Springer, 2015.

[17] D. Jannach, M. Zanker, A. Felfernig, and G. Friedrich. *Recommender Systems: An Introduction*. Cambridge University Press, 2010.

[18] G. Jawaheer, P. Weller, and P. Kostkova. Modeling user preferences in recommender systems: A classification framework for explicit and implicit user feedback. *ACM Trans. Interact. Intell. Syst.*, 4(2), 2014.

[19] G. Kappel et al. TheHiddenYou - A Social Nexus for Privacy-Assured Personalisation Brokerage. In *12th Int. Conf. on Enterprise Information Systems (ICEIS)*, 2010.

[20] A. Kobsa, B. P. Knijnenburg, and B. Livshits. Let's do it at my place instead? attitudinal and behavioral study of privacy in client-side personalization. In *SIGCHI Conference on Human Factors in Computing Systems (CHI 2014)*, Toronto, Canada, 2014.

[21] L. Moreau et al. The open provenance model core specification (v1.1). *Future Generation Computer Systems*, 27(6), 2011.

[22] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Item-based collaborative filtering recommendation algorithms. In *Proc. of 10th Int. Conf. on World Wide Web*, New York, NY, USA, 2001. ACM.

[23] J. Schafer, D. Frankowski, J. Herlocker, and S. Sen. Collaborative filtering recommender systems. In *The Adaptive Web*, volume 4321 of *LNCS*. Springer, 2007.

[24] E. I. Sparling and S. Sen. Rating: How difficult is it? In *Proceedings of the Fifth ACM Conference on Recommender Systems*, RecSys '11, New York, NY, USA, 2011. ACM.

[25] M. Wischenbart et al. Automatic data transformation: Breaching the walled gardens of social network platforms. In *Proc. of APCCM - vol. 143*, Adelaide, Australia, 2013. Australian Computer Society.

[18] http://lenskit.org/documentation/basics/configuration/