Exploiting Reviews to Guide Users' Selections

Nevena Dragovic
Department of Computer Science
Boise State University
Boise, ID, USA
nevenadragovic@u.boisestate.edu

Maria Soledad Pera
Department of Computer Science
Boise State University
Boise, ID, USA
solepera@boisestate.edu

ABSTRACT

We introduce HRS, a recommender that exploits user reviews and identifies the features that are most likely appealing to users. HRS incorporates this knowledge into the recommendation process to generate a list of top-k recommendations, each of which is paired with an explanation that (i) showcases why a particular item was recommended and (ii) helps users decide which items, among the ones recommended, are best tailored towards their individual interests. Empirical studies conducted using the Amazon dataset demonstrate the correctness of the proposed methodology.

Categories and Subject Descriptors

H.3.3 **[Information Storage and Retrieval]**: Clustering, Information Filtering, Retrieval Models, Selection Process.

Keywords

Recommendation Engine, Explanations, Ranking.

1. INTRODUCTION

Recommendation systems aid users in locating items (either product or services) of interest [1]. Regardless of the domain, from shopping websites (e.g. Amazon, e-bay), to news related sites (e.g. Yahoo, CNN), and hotel or restaurant search (e.g. Yelp, hotels.com), recommenders have a huge influence on businesses' success and users' satisfaction. From a commercial standpoint, existing recommenders enable companies and items to get advertised by being offered to potential buyers. From a user prospective, these systems enhance users' experience by assisting them in finding information pertaining to their interests, thus addressing the information overload concerns that web users have to deal with on a daily basis.

Suggestions generated by existing recommenders are not always personalized and diverse enough to expose users to a wide range of items within their realm of interest, not just popular ones [6]. This is due to the fact that a common alternative for generating recommendations is to rely on existing community data. Suggesting the same items to similar users within a community can be very vague and impersonal [2]. Newly-developed strategies take advantage of different users' generated data to better identify user preferences in an attempt to further personalize recommendations [1]. Another challenge faced by recommenders is to get users to trust them. In many cases, users need to see more details related to a suggestion than just "dry" recommendations to increase their perceived trust on the corresponding recommender [3]. Recent research works focus on explaining the generated recommendations [1]. Unfortunately, justifying the reasons why an item has been suggested to a user is not an easy task. Thanks to the growth of online sites that archive user reviews, researchers have suggested examining these reviews to enhance the recommendation process [7]. Nonetheless, the better understanding of the aspects or features of a particular item that appeal the most to an individual user, such as price in the case of restaurants or pacing of the story in the case of a book, is yet to be accomplished.

In this paper, we present the initial research conducted to attempt to solve the issues mentioned above. We created Honest Recommendation System (HRS), a novel recommender system that shows items in their real light. In developing HRS, we focus our efforts in using collected information from users' reviews to generate *personalized suggestions* with their *corresponding explanations*. By incorporating into the recommendation process the feature preferences of an individual user (inferred from his reviews), we can get to know the user better than by simply considering his rating patterns. We strive for the development of a recommender system a user trusts by providing information he is interested in, no matter if it has a positive or negative connotation. Our main contribution is the increased effectiveness and satisfaction on a domain independent recommender. This is accomplished by giving users information they care about, which helps them make the best decision, in terms of selecting the most adequate item among the recommended ones. Users' overall satisfaction with a recommender is related to the perceived quality of its recommendations and explanations [1]. Consequently, users' confidence is also increased.

2. OUR PROPOSED RECOMMENDER

In this section we discuss HRS overall recommendation process. (Parameters used by HRS were empirically determined. However, details are omitted due to page constrains.)

Identify User's Interest on Items. Consider a user *U*, who is a member of a popular site, such as Yelp or Amazon, which archives *U*'s reviews and rating history. Given that we aim to provide *U* with information he values and needs to choose among suggested items, we examine reviews written by *U* and identify the set of features (i.e., traits) that *U* cares the most about. Since we know features are mainly expressed as nouns, we perform semantic analysis on reviews[1] and consider the frequency of occurrence of nouns *U* employs in his reviews. We rely on WordNet-based similarity measures (using WS4J java library, specifically WuPalmer algorithm) to find and cluster similar

[1] Using Stanford CoreNLP http://nlp.stanford.edu/software/.

terms, as different nouns can be used to express similar meaning. Each cluster would contain most frequent terms together with its closest words among the ones *U* uses in reviews. We do this to learn what items' traits *U* most frequently mentions in his reviews and use that knowledge to predict which candidate items would be of *U*'s interest. The top-2 most frequently-used term clusters are treated as *U*'s preferred features. Note that each cluster is labeled using the most representative[2] cluster term.

Generate Candidate Recommendations. We take advantage of *U*'s historical data (i.e. rated items) and employ the well-known matrix factorization strategy [4] based on LensKit implementation to generate a number of candidate suggestions for *U*.

Generate Top-k Recommendations. We examine archived reviews for each candidate item *I* and following the same process defined for identifying features of interest to *U*, we identify the top-2 features most-frequently mentioned in reviews pertaining to *I*. Thereafter, we generate a ranking score for *I*, which shows the degree to which *U*'s preferred feature are addressed in *I*'s reviews. This score is computed by averaging the degree of similarity (defined based on WordNet using the RitaWordnet library) between all the words in the term clusters generated for *U* and *I*. This score represents the level of *U*'s interest in *I* and is used for ranking *U*'s candidate items, such that the top-*k* ranked candidate items are selected as the items to be recommended to *U*.

Generate Explanations. We generate the corresponding explanation for each recommended item *I* by showing why *I* is likely appealing to *U*. We do so by extracting the descriptions other users provided on *U*'s preferred features pertaining to *I* from archived reviews. We identify sentences in reviews pertaining to *I* that include terms exactly-matching (or highly-similar as determined using WordNet) to each of the labels generated for *U*'s clusters. In the explanation of each recommended item, HRS includes 3 sentences for each label. In doing so, HRS provides *U* with sufficient information about the recommendations without overwhelming *U* with too much information to read related to the recommended items. As previously stated, we do not emphasize the sentiment of the features, since our intent is not to make *U* like one option more than another, but save *U*'s time in identifying information important for him.

3. EXPERIMENTAL RESULTS

We conducted initial experiments using the Software[3] domain in the Amazon Review dataset [5], which consists of 68,464 users, 11,234 items, and 95,084 reviews. We evaluated the performance of HRS in terms of Normalized Discounted Cumulative Gain (NDCG), which considers the *correctness* of the recommendations and penalizes relevant recommendations positioned *lower* in the ranking. We compared HRS with a baseline, yet popular, algorithm: Matrix Factorization (SVD). As shown in Table 1, HRS outperforms SVD. The significant NDCG improvement demonstrates that, in general, recommendations provided by HRS are preferred over the ones provided by SVD, which does not consider users' feature preferences.

Table 1. Performance of HRS compared to baseline algorithms

Metrics	**SVD**	**HRS**
NDCG	0.704	0.748

4. CONCLUSION & FUTURE WORK

We developed a new recommendation system that takes advantage of ratings and reviews, to create personalized suggestions. The first version of HRS, which showcases the main idea and purpose of the system, generated promising results, yet there are opportunities to explore in the future that will enhance its performance. Even though HRS did better than SVD, we plan to provide deeper examination and comparisons with other baseline and state-of-the-art recommendation strategies. We will also analyze the effect of considering only candidate items with rating scores above 3, which we anticipate will improve the overall performance of HRS. We will also extend the performance evaluation by conducting online user studies to further verify the fact that HRS helps users in making appropriate choices among provided suggestions. One of the limitations of the current design of our recommender is that only nouns extracted from reviews are treated as features which cause losing rich information from adjectives and verbs. To address this issue, we will conduct more in-depth analysis on part-of-speech and type dependencies on sentences in reviews. We are aware that HRS, in its current state, does not entirely solve the "cold start" problem. We will consider adopting a hybrid recommendation strategy that considers general item metadata, along with the popularity of items, in addition to examining alternative ways to extract information from reviews and further work towards eradicating the cold start problem.

5. REFERENCES

[1] F. Gedikli, D. Jannach, and M. Ge. How Should I Explain? A Comparison of Different Explanation Types for Recommender Systems. *International Journal Human-Computer Studies*, 72:367-382, 2014

[2] N. Good, J. B. Schafer, J. A. Konstan, A. Borchers, B. Sarwar, J. Herlocker and J. Riedl. Combining Collaborative Filtering with Personal Agents for Better Recommendations. In *AAAI/IAAI*, p. 439-446, 1999.

[3] S. Kanetkar, A. Nayak, S. Swamy and G. Bgatia. Web-Based Personalized Hybrid Book Recommendation System. In *ICAETR*, p. 1-5, 2014

[4] Y. Koren, R. Bell and C. Volinsky. Matrix Factorization Techniques for Recommender Systems. *IEEE Computer Society*, 42(8):30-37, 2009.

[5] J.J. McAuley and J. Leskovec. Hiddent Factors and Hidden Topics: Understanding Rating Dimensions with Review Text. In *ACM RecSys*, p. 165-172, 2013.

[6] M.S. Pera. *Using Online Data Sources to Make Recommendations on Reading Materials for K-12 and Advanced Readers*. PhD Dissertation, BYU, 2014.

[7] Y. Zhang, G. Lai, M. Zhang, Y. Zhang, Y. Lui and S. Ma. Explicit Factor Models for Explainable Recommendation Based on Phrase-level Sentiment Analysis. In *ACM SIGIR*, p. 83-92, 2014.

[2] Using WordNet, we generate a list of synonyms for each cluster term, such that the most frequent term among these synonym lists is treated as the corresponding cluster label.

[3] Note that we developed HRS to be a generic recommender so it can be used on items on varied domains, beyond the Software domain we considered only for initial assessment purposes.

One of the features was for the user to be able to select two items and then take her/him to anotherâ€¦Â The documentation first calls to determine the selection key type to use to then build a KeyProvider. You can use whichever type you like but the selection library provides support for three types: Parcelable, String and Long. There are also some guidelines and advice on which type to use depending on your use case. Editorial Reviews. Review. "I highly recommend Physical Database Design by Lightstone, Teorey, and Nadeau.Â " Focuses on physical database design for exploiting B+tree indexing, clustered indexes, multidimensional clustering (MDC), range partitioning, shared nothing partitioning, shared disk data placement, materialized views, bitmap indexes, automated design tools, and more! Read more. See all Editorial Reviews. Product details.Â Query optimization, plan selection and execution aspects are addressed with focus on where the indexing, partitioning and clustering techniques previously discussed fit in and on how physical design can be improved by selecting the right plan. 2015. Exploiting Reviews to Guide Users' Selections. In Poster Proceedings of ACM RecSys 2015. Santiago LarraÃn, Denis Parra, and Alvaro Soto.Â 2013. Rating Support Interfaces to Improve User Experience and Recommender Accuracy. In Proceedings of the Seventh ACM Conference on Recommender Systems (RecSys '13). ACM, New York, NY, USA.