Accessible Discussion Boards

CSE 591: Adaptive Web Project

Piyush (Papreja) Nolastname
School of Computing, Informatics, and Decision Systems Engineering
Arizona State University
Tempe, Arizona
ppapreja@asu.edu

ABSTRACT

This report describes the analysis, and the development done for building a Recommender Engine as a component of a Discussion Board website for the course CSE 591 Fall 2017.

Keywords

Recommender Engines, Collaborative Filtering, Content Based Recommendation

1. INTRODUCTION

The internet was designed to be universally accessible. The inventor of the World Wide Web, Tim Berner-Lee has said that the power of the Internet is in its universality. However, owing to the over-reliance on standard forms of input such as keyboards, and screen-readers [8], *coupled with un-manageable and non-personalized content*, accessibility is far from being achieved. Whilst the first two problems are primarily design features shortcomings, the third problem is down to improper content management. This problem can be mitigated by employing recommendation engines. A recommendation system is a subclass of information filtering system that seeks to predict the "rating" or "preference" that a user would give to an item [1]. These systems take as input, some form of historical data of the user, whose interests and preferences they try to guess. Recommendation engines have become ubiquitous today. From Amazon [2] to Netflix [3], from Facebook to Google, the entire internet is starting to get built on recommendation engines, to handle the problem of content management for users, showing them content which is considered to be most useful to them.

1.1 Motivation

The motivation for this project is two fold:

1. Bringing the users with disabilities on par with the rest of the population so that they can access the internet as easily as the rest of the population.
2. Improving the accessibility of the content for everyone by incorporating certain design features, along with the recommender engines,

Table 1: Data-set statistics

Users	12602
Posts	9194
Questions	3524
Comments	13128
Users with Posts	2837
Users with Comments	1716

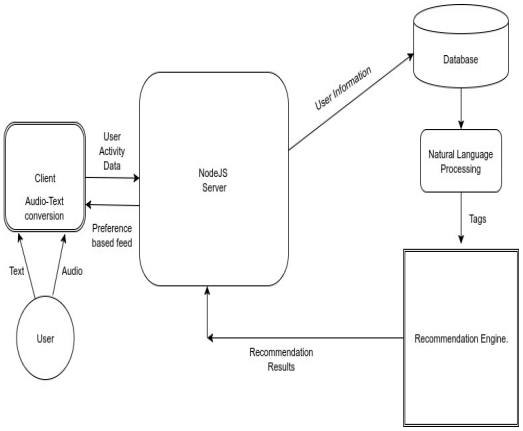

Figure 1: Overview of system architecture

2. SYSTEM DESIGN

2.1 The Data-set

For this project, data-set from **Robotics.stack-exchange.com** [13] has been used. The data-set stats are shown in Table 1.

2.2 Recommendation goal

We wish to recommend posts to users, relevant to them, based on their website activity. Almost all websites based on the principle of crowd-sourced content have the concept of home-feed, where users can see the most relevant information on their home page. We will also be displaying the recommended content to the user on their home page.

2.3 The system

As shown in figure 1, The system consists of 4 components primarily:

1. The Client Side, built using HTML/Javascript, where the user interacts with the system and data is displayed to the user.
2. The Server Side, built using nodeJS web framework, which handles the requests coming from the user provides the content to the client side to be rendered.
3. The Database, which handles the persistence of the entire data. Redis has been used for cache management and MYSQL for persistent storage.
4. The recommender engine, which lies on the server side and makes use of the data to provide recommendations to the user. Surprise [6], a python package has been used as the recommender system library.

2.4 Application Flow

The user interacts with the website on the client side using the audio-text/text input. If the user uses the audio input feature, audio-to-text conversion happens on the client side using Javascript's Web Speech API [9]. The user query is then routed to the server side, which directs that to the recommender engine, that produces recommendations. The recommendations produced are then returned to the server, which in turns renders them onto the client side, where the user can see results on his home screen.

3. METHODOLOGY

3.1 Improving the Data

As seen in table 1, we have a lot of data sparsity in our data, we will create simulated browsing logs for all users. Since there are two types of users, ones who have some kind of activity (posts or comments) associated with their profile and the ones who don't, we will adapt our algorithm for the creation of browsing activity for the two types

For users of type 1(some activity users), we pick tags from these users' profile, sample posts from those tags and associate those to the user profile. For users of type 2(no activity users), we first pick random tags, then sample posts from these tags and associate these to the user profile.

3.2 Recommendation techniques

For our system, we are using a combination of the following two recommendation techniques:

1. Content Based Recommendation Techniques [4]
2. Collaborative Filtering [5]

3.3 Content Based Recommendation System

3.3.1 User Profiling

In section 3.1, we created simulated browsing activity records for all the users. Next step is to extract important keywords from each post. We can do this by using the technique tf-idf [10]. Tf-idf is a technique using which we can get the list of important keywords from each document. It looks at the relative importance of a keyword in a document, as well as the entire corpus. Having extracted keywords from all the posts, and also having posts associated with the browsing history of all the users, now, we can associate the important keywords (along with the tag associated with the posts) found in those posts with the user.

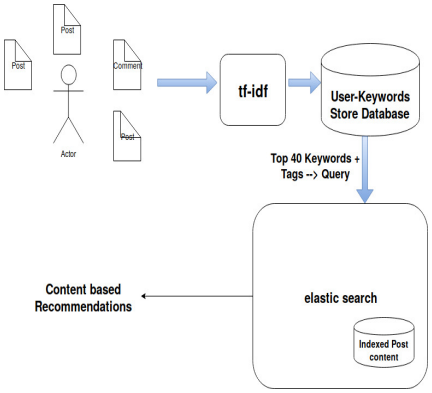

Figure 2: Application of Content Based Recommendation

1. Extract keywords from posts.
2. Find posts for users.
3. Associate keywords to users.

3.3.2 Corpus Profiling

For indexing the corpus content, we use the Elasticsearch [7] full text search engine. The Elasticsearch indexes the whole dataset, and thus can be used to fetch posts for any query.

3.3.3 Finding Posts via Content Based Technique

Now, the information extracted via user profiling becomes the query to the Elasticsearch engine, which now is used to return relevant posts on the basis of keywords from the user profile.

3.4 Collaborative Filtering System

For collaborative filtering , we require a user-item matrix, which is nothing but a rating for an item by a user. The item is our case is the post, which we are recommending to the user. But how do we calculate the rating? There are 3 kinds of activities a user can engage in on the platform; he could ask a question, answer a question or post a comment. We assign a score of 1 for asking a question, a score of 2 for answering a question, and 1.5 for posting a comment. The intuition behind this rating system is that, answering a question implies 2 times greater interest in the topic than just asking a question. This way, we get our user item matrix.

3.4.1 The Algorithm

For collaborative filtering, we are not considering the browsing activity of the users, meaning that the problem of sparsity still remains. Fortunately, there are matrix factorization algorithms like SVD (Singular Value Decomposition) [11] [3], NMF (Non negative Matrix Factorization) [12] that find solutions for sparse problems. In our case, we use the SVD algorithm to get the collaborative filtering based recommendations.

Figure 3: Visual Recommenders: Tuning Tags

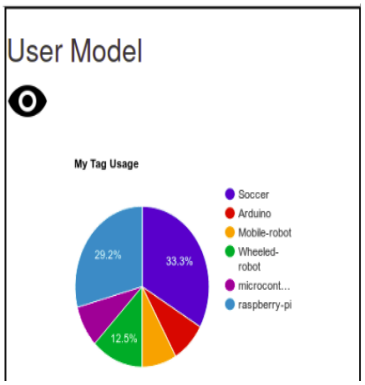

Figure 4: Open User Model

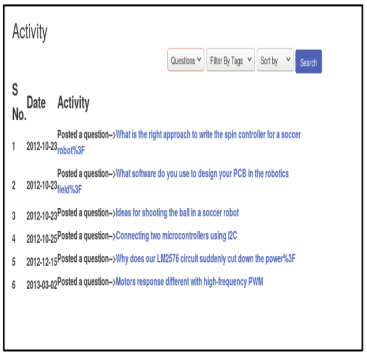

Figure 5: User Activity

3.5 Combining the recommendation techniques

Now that we have implemented both the content based and collaborative filtering recommendation engines, we combine the results from both techniques and render a combination of both on the client side for the user.

4. MANAGING THE COLD START PROBLEM

Even though we do not rely on the user's profile alone to provide him with the recommendations, which often runs the risk of cold start problem. In our case, on the home screen, we have 3 tabs:**Trending tab, Latest tab and the Recommended for you tab**. The Latest tab displays the latest posts ordered by date time, in reverse chronological order. The Trending tab displays the latest posts which are also the most popular, based on the number of views. So, even if a new user logs in on the platform, he can still be shown results that are popular and hence somewhat relevant.

5. RECOMMENDER TUNING

Recommender tuning is important because it helps users understand how the recommendations are working, what factors are considered by the recommender engines for giving them the results, and how those can be utilized maximally by the users.

In our case, among many others, there are primarily two factors considered by the system for providing users with the recommendations:

1. Tags associated with the users profile via Content Based Filtering.

2. Influence of others users via collaborative filtering

5.0.1 *Tuning Tags*

The user should have the option to tune the recommender according to his preferences in a way that is he wants to see posts of a particular topic more than of other topics, he should have the freedom to do so.

The way we handle this is that users are provided with filters for tags associated with their profile. The filter values are mapped to the boost paramater for that particular tag for the query to elasticsearch [7]. By letting users adjust the filter values according to their preference, we can change the value of boost for the query for a tag to elasticsearch, hence getting weighted results based on the eboost value of the tag.

5.0.2 *Tuning the extent of Collaborative Filtering*

Also, the user should have the freedom to decide to what extent does he want other users to influence his recommendation results. If the user doesn't want other users to influence his results, he should have the freedom to opt out of that option. The way we handle this is by providing users with filters to tune the ratio of content based vs collaborative filtering. This is accommodated in the system by giving users the weighted results from a combination of content based and collaborative filtering.

6. OPEN USER MODELING

Open user modeling helps bring a sense of transparency in the system. The user should be aware of his activities on the platform, what patterns are emerging from his activities and ultimately how the recommender engine is viewing the user. We have done a simple open user modeling for our system. We show the user his overall tag distribution, and profile activity.

7. EVALUATION PLAN

Although recommender evaluation generally requires explicit user feedback, such as up-vote, down-vote etc. How-

Table 2: Collaborative Filtering Results for Q-A-C rating 1-2-1.5

Q-A-C	Threshold	Precision	Recall
1-2-1.5	2	0.65	0.76
1-2-1.5	2.5	0.97	0.68
1-2-1.5	3	0.99	0.73
1-2-1.5	3.5	1	0.82

Table 3: Collaborative Filtering Results for Q-A-C rating 2-4-3

Q-A-C	Threshold	Precision	Recall
2-4-3	3	0.72	0.98
2-4-3	4	0.70	0.72
2-4-3	4.5	0.91	0.72
2-4-3	5	0.98	0.68

ever, the thing that can be evaluated purely on the basis of numbers is the performance of collaborative filtering. There are various parameters to evaluate CF (collaborative filtering) such as:

1. **Root Mean Square Error**: For our system, we get a value of 0.64 for RMSE.
2. **Precision and Recall**: For our system, we get a precision value of 0.97274 and 0.68 for recall for k=10 and threshold = 2.5. Here k stands for number of results for which prevision and recall is calculated.

8. CONCLUSION AND FUTURE WORK

This project gave an insight into the world of recommendation engines, as well their workings. It was interesting to know how content model and collaboration model can be combined to form different types of (hybrid) recommendation models. Evaluation of the recommendation engines, as seen in section 7, is a pain point in systems employing recommender engines. There's a lot of scope with regards to future work for this application such as user feedback, transparency of recomendation engine, search engine implementation, and use of deep learning for recommender system modeling.

9. MY CONTRIBUTION

Since I was the only person working on this project, the whole project is my undertaking. I designed the user interface and integrated it with the web framework. I also used MYSQL for persistent storage and Redis for in-memory storage. Lastly, I worked on the recommendation engine part of the system, which included using elasticsearch for implementing content based recommendation and Surprise [6], for, collaborative filtering. I also worked on the integration of the complete system, so that its working from end-to-end.

10. LEARNING

I learned a lot about the intricacies and limitations of various recommendation techniques learned in class. I understood how completely knowing how the system is working, which includes understanding the application of evaluation matrices and their results, makes all the difference between working and non-working recommender models The importance of data exploration and analysis also became clear while working on this project as recommendation solutions always have to be customized according to the problem, and cant be expected to work out of box, thus making the understanding of domain as well as data very significant.

11. REFERENCES

[1] https://www.wikiwand.com/en/Recommender$_system$

[2] Linden, G., B. Smith, and J. York. Amazon.com Recommendations: Item-to-Item Collaborative Filtering. IEEE Internet Computing, Jan.-Feb. 2003.

[3] C. A. Gomez-Uribe and N. Hunt. The Netflix Recommender System: Algorithms, Business Value, and Innovation. ACM TMIS, 6(4):13:1âĂŞ13:19, 2015.

[4] S. Langer and J. Beel. Apache Lucene as Content-Based-Filtering Recommender System: 3 Lessons Learned. In Proc. of the 5th Workshop on Bibliometric-enhanced Information Retrieval (BIR 2017), pages 85âĂŞ92. CEUR-WS.org, 2017.

[5] J. Lee, M. Sun, and G. Lebanon, âĂIJA Comparative Study of Collaborative Filtering Algorithms,âĂİ [Online] arXiv: 1205.3193, 2012.

[6] http://surpriselib.com/

[7] https://www.elastic.co/

[8] http://www.tandfonline.com/doi/full/10.3109/074346109035616

[9] https://developer.mozilla.org/en-US/docs/Web/API/Web$_Speec$

[10] J. Ramos. Using TF-IDF to Determine Word Relevance in Document Queries. Technical report, Department of Computer Science, Rutgers University, 2003

[11] Sarwar, Badrul, et al. Application of dimensionality reduction in recommender system-a case study. No. TR-00-043. Minnesota Univ Minneapolis Dept of Computer Science, 2000

[12] Lee, Daniel D., and H. Sebastian Seung. "Algorithms for non-negative matrix factorization." Advances in neural information processing systems. 2001.

[13] https://archive.org/download/stackexchange/robotics.stackexc

12. ACKNOWLEDGEMENT

I would like to thank Dr.Hsiao for providing me with the opportunity and knowledge to understand and apply recommendation techniques in practice.