Accessible Discussion Boards

CSE 591: Adaptive Web Project

Piyush (Papreja) Nolastname
Arizona State University
Tempe, Arizona
ppapreja@asu.edu

ABSTRACT

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 the internet is not as inclusive as it might sound. While people with disabilities such as visual impairments, motor impairments face problems due to lack of features to facilitate accessibility for them, people without any disabilities face problems because of the cluttered design and un-managed content on the websites. *The problem of un-manageable and non-personalizd content* can be solved by using the recommendation engines. Online communities, especially with crowd-sourced content such Question and Answer sites, discussion boards are a significant part of the internet, and fixing websites like these can go a long way in making the internet accessible for the whole population.

Keywords

Recommendation Engines, Collaborative Filtering, Content Based Recommendation

1. INTRODUCTION

Some of the major problems in online discussion boards and question-answer websites, with regards to accessibility are:

1. Over-reliance on the standard forms of input with limited capabilities, such as keyboards, mouse and screen-readers [8].
2. Limited capabilities of Augmentative and Alternative Communication.
3. Un-manageable and non-personalized content.

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 recommender system or 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. 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.

Table 1: Data-set statistics

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

2. MOTIVATION

The motivation for this project is two fold.

1. Bring 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. By incorporating certain design paradigms along with the recommender engines, improving the accessibility of the content for everyone.

The first point deals with improving the design of website, that can then be leveraged by the recommender engine as well. By adding certain features such as extraction of links from the page and displaying them to the user in a separate window, of giving the user option to highlight relevant text on website, which then can be shared with everyone, accessibility of the content of the website can be greatly improved. These features can also be leveraged by the recommender engines, by modifying the display of these according to the preferences of the user.

The second point deals with the management of the content and personalization according to the user. This can only be handled by the recommender engines.

3. SYSTEM DESIGN

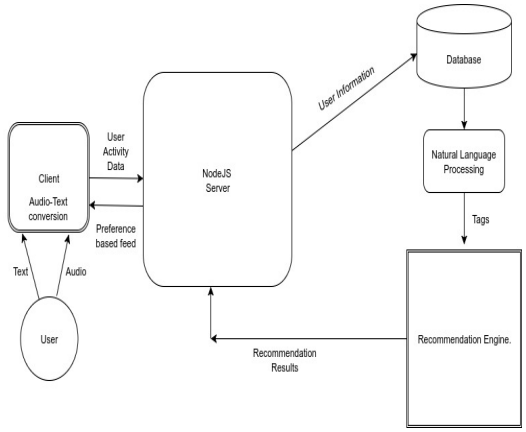

Figure 1: Overview of system architecture

3.1 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. In the data-set, we are provided with detailed information about Posts such as Post creation, modification dates, associated tags, text, title, body content etc, and User information such as Posts created, website usage statistics such as reputation points, total up votes, down votes etc.

3.2 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.

3.3 The system

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

1. The Client Side, where the user interacts with the system and data is displayed to the user.
2. The Server Side, 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
4. The recommender engine, which lies on the server side and makes use of the data to provide recommendations to the user.

3.4 Technical Implementation Details

The system technical stack looks like this:

1. The Client Side has been implemented using HTML and Javascript.
2. The Server side has been implemented using NodeJS.
3. The database used for this application is MYSQL.
4. For cache implementation, redis has been used for this application.
5. For scripting purpose, python 2.7 has been used.
6. For the collaborative filtering recommender libraries, Surprise [6] package of python has been used.

3.5 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 which makes use of this query content as well as the corpus information, stored in the database to produce 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.

4. METHODOLOGY

4.1 Improving the Data

As seen in table 1, out of 12062 users, only 3500 users are the ones who have posted either any kinds of posts (questions or answers) or comments. Which means there are 8500 users without any kind of activity associated with their profile. One more thing to notice here is that we don't have the browsing information for any user, nor do we have any information for up-votes or down-votes for any user, which greatly limits the extent to which we can produce our recommendations. Also, from the perspective of platform, for which we are designing our recommender engine, it is okay to assume that they have the access to this information for all of their platform users.

So we will create simulated browsing activity for all users. Since there are two types of users, ones who have some kind of activity associated to their profile and the ones who don't, we will adapt our algorithm for creation of browsing activity for the two types

Algorithm for creating browsing activity records for users
1. For each user
1.a If user has associated activity:
tags = pick user tags from user activity records
Else
tags = pick random tags
1. b For selected tags
samples posts from tags
Add posts to user's profile.

As a result now, we have browsing activity records for all the users in the dataset, which can be put to use in producing recommendations.

4.2 Recommendation techniques

There are primarily two types of recommendation techniques:

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

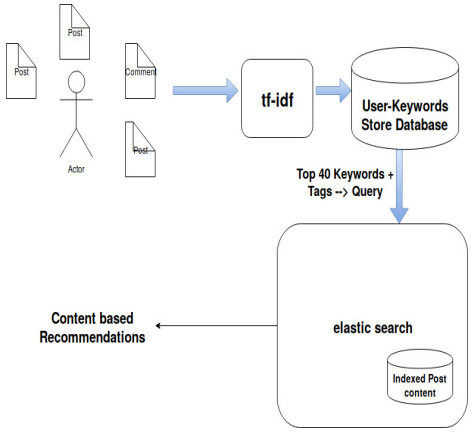

Figure 2: Application of Content Based Recommendation

4.2.1 *Content Based*

Content Based Recomender Systems work by trying see the similarity between the user profile with the corpus and then recommending items from the corpus that best match the user profile. Information taken from the user profile can be of two kinds:

1. Core User Profiling
2. Extended User Profiling

Core user Profiling deals with the user's activity on the platform such as browsing activity, preferences, posts created, accessed, up-votes, down-votes etc. Extended User Profiling takes into account the information that the user enters about himself on the platform such as age, profession, gender, bio etc. Important information is extracted by user profiling in the form of keywords and then those keywords are then searched in the corpus to get the items most closely related to the keywords of the user.

Content based recommendation techniques work well in short term, but are limited in their capabilities in capturing complex relationship between users and items.

4.2.2 *Collaborative Filtering*

Collaborative filtering is another type of recommendation technique that takes into account the relationship between similar users (or/and similar items). It works on the principle of collaborative, where similar users (inferred on the basis of ratings given by them to items) influence each others recommendations. In its simplest form, it can be reduced to the following steps:

1. Similar users are found on the basis of shared ratings of common items.
2. Predictions are calculated for users for items that they have not rated before based on the ratings given by users similar to them.

4.3 Application of Content Based Recommendation

4.3.1 *User Profiling*

In section 4.1, we created simulated browsing activity records for all the users. Our 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 for the entire corpus. 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 , we also have posts associated with the browsing history of all the users. Now by taking into account all the posts associated with a user, we can associate the important keywords (along with the tags associated with the posts) found in those posts via tf-idf technique to the user.

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

4.3.2 *Corpus Profiling*

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

4.3.3 *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.

4.4 Application of Collaborative Filtering

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.

4.4.1 *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 user the SVD algorithm to get the collaborative filtering based recommendations.

4.5 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.

5. 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

Figure 3: Visual Recommenders: Tuning Tags

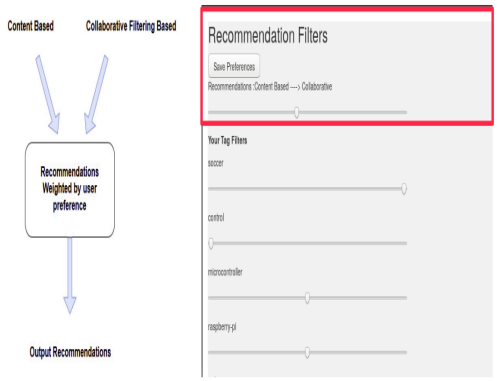

Figure 4: Visual Recommenders: Tuning Collaboration

risk of cold start problem, a problem that arises when the recommender engine does not have enough/any information on the basis of which it can provide recommendations to the user, there can still be scenarios where the recommendation engine would not have anything to go by for providing recommendations. In those scenarios, there should be a way so that user sees some results, instead of blank screen. **One such way is the most popular posts/products. 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.

6. RECOMMENDER TUNING

One of the major issues in recommender engines is that platforms seldom provide users the option to tune the recommenders according to their preference. 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

6.0.1 *Tuning Tags*

For content based recommendation, keywords and tags associated with the browsing activity of the user are considered and fed to the elasticsearch [7] engine to get the recommendations. 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 parameter for that particular tag for the query to elasticsearch. 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 boost value of the tag.

6.0.2 *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.

7. OPEN USER MODELING

Like Recommender Tuning, Open User Modeling is equally important in a recommendation system. What Open user modeling does is that it 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 :

1. His overall tag distribution.
2. His profile activity filtered by tags, date and sort order

8. EVALUATION PLAN

Evaluating a recommender engine is never an easy task because:

1. In most cases, it requires a user study to get the accurate evaluation..
2. It requires an explicit user feedback, because content based filtering fails to capture complex relations between users and items and hence, the recommendations may sometimes appear trivial or completely irrelevant.

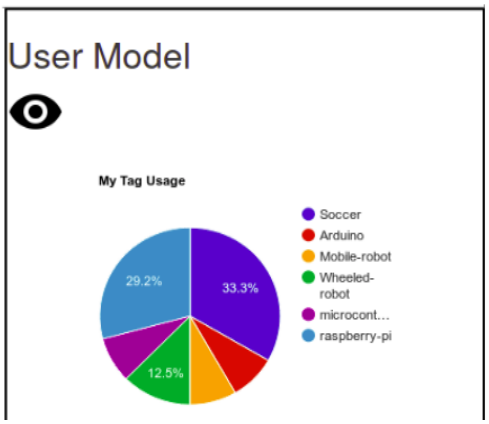

Figure 5: Open User Model

Figure 6: User Activity

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 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

3. The answer is never objective. There is just one screw that can be just tightened to make the system work.

The thing that can be evaluated 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
2. Precision
3. Recall

Root mean Square Error (RMSE): The root-mean-square error (RMSE) is a frequently used measure of the differences between values (sample and population values) predicted by a model or an estimator and the values actually observed. Since in collaborative filtering, we are ultimately predicting the rating of an unrated item by a user based on other items rated by that user, we can compare the values predicted by the model vs the actual rating values. For our model, we were getting the RMSE value of 0.647 for our model.

Precision and Recall: In pattern recognition, information retrieval and binary classification, precision (also called positive predictive value) is the fraction of relevant instances among the retrieved instances, while recall (also known as sensitivity) is the fraction of relevant instances that have been retrieved over the total amount of relevant instances. Both precision and recall are therefore based on an understanding and measure of relevance.

For our model, we were getting a precision value of 0.97274 and 0.68 for recall for k=10 and threshold = 2.5.

8.0.1 Evaluation Plan: Future Work

There can be a lot of ways in which the application can be evaluated:

1. Explicit like/dislike feedback given by the user.
2. Time spent on each post
3. Clicks on posts on home feed

Explicit like/dislike feedback: Explcit feedback given by the user is the clearest form of evaluation for whether the recommended posts are appreciated by the user. The model can be built to modify itself by taking into account

this feedback, hence learning over time according to user activity and user feedback.

Time spent: Time spent on a post or a section of posts is a great way to get insight into user preferences. If the user spends very little time on a post, most of the time it means the user is not interested in the post. On the other hand, if the user is spending considerable period of time on a post, it means considerable interest in that topic and post.

Clicks: Clicks are also a great way to get insight into user behavior and see if the user is actually clicking and accessing posts recommended to him by the model. If the user isn't clicking on a section of posts for a continual period of time, it means he might not be interested in that section.

9. DISCUSSION AND FUTURE WORK

This project was an interesting 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. Some of the points that can be implemented in future are:

1. **Incorporating explicit like/dislike feedback into the model learning.**

   As discussed before, this is a very effective way to keep out model on a continual learning path, based on user activity and direct feedback.

2. **Transparency**: Transparency is a very important component of a recommender engine, which is often overlooked. By providing user with the information about the source and reason for each recommended post, we can make the system more transparent to the user, hence giving him the power to use the system to its maximum extent.

3. **Search**: Implementation of search engine into the platform is yet another feature that can be implemented in the future for this application. Adding search functionality to the platform would make the posts more accessible to the user which is the ultimate goal of this application.

4. **Trying new Models such as deep learning**: Deep learning has been gaining quite a momentum off late and there have been some implementations in the field of recommendation engines also. This is one area which could be looked in the future.

5. **Incorporating design features of the platform into the recommender engine model** Improving design features to improve the accessibility of the platform is one of the primary goals of this project. There are two design features that could be leveraged for recommender engine model:

   (a) **Highlight** feature, by which users can highlight text on the platform, which are then displayed in a separate window This design enables creation of a whole new fragment, which can be then customized according to the preference of the user.

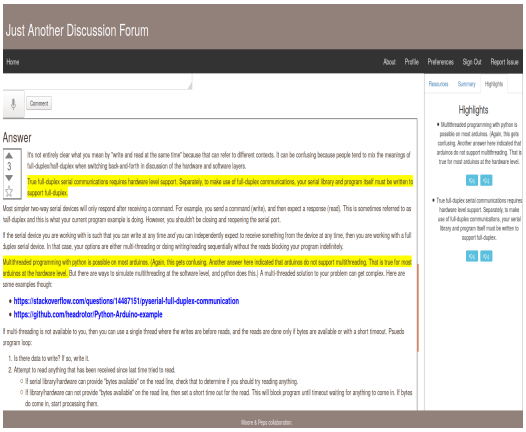

Figure 7: Highlights feature

Figure 8: Resources Panel

   (b) **Resources Panel** Another design feature of the platform is the resources panel, which is a separate window in which extracted links from the page are displayed. This is yet another fragment, which can be customized according to the user.

10. 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$_S peech_A PI/Using_t he_W eb_S peech_A PI$

[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.stackexchange.com.7z