An Open Source Discussion Group Recommendation System

SJSU SAN JOSÉ STATE UNIVERSITY

Advisor
Dr. Chris Pollett
Committee Members
Dr. Katerina Potika
Dr. Leonard Wesley

By
Sarika Padmashali

SJSU SAN JOSÉ STATE UNIVERSITY

Agenda

- Purpose and Introduction
- Background about Yioop and Problem Statement
- Collaborative Filtering Techniques
- Preliminary Work
- Classic Baseline Predictor
- Term Frequency Inverse Document Frequency
- Recommendation Job
- Conclusion and Future Work
- Questions

SJSU SAN JOSÉ STATE UNIVERSITY

Purpose

- In this project we have added a new feature to Yioop, a search and discussion board portal, which recommends the Yioop users with exciting groups and threads.

- Yioop Recommendation System uses two techniques to make recommendation – Collaborative filtering baseline predictor and Term Frequency – Inverse Document Frequency

- This system makes use of the users impressions on the Yioop site and tries to recommend relevant threads and groups.

SJSU SAN JOSÉ STATE UNIVERSITY

Introduction to recommendation

- People take recommendations from others in the form of spoken words, letters, documentaries, surveys, articles, blogs, etc.

- A Recommendation System gives these suggestions to users to browse through the internet, surf through articles, movies, restaurants, books, discussions, groups, groceries, etc. and find the most relevant and exciting information to them.

SJSU SAN JOSÉ STATE UNIVERSITY

About Yioop

- Yioop is an open source search engine and discussion group software
- Yioop provides many features of large search portals such as search results, media services, social groups, blogs and wikis
- Yioop allows users to create discussion groups and start a new thread on those groups where they can discuss with other Yioop users

Problem Statement

- Instead of letting users browse through a bunch of threads and groups. We have improved the users experience by recommending threads or groups which might be of interest to them based on the threads or groups a user has viewed in the past

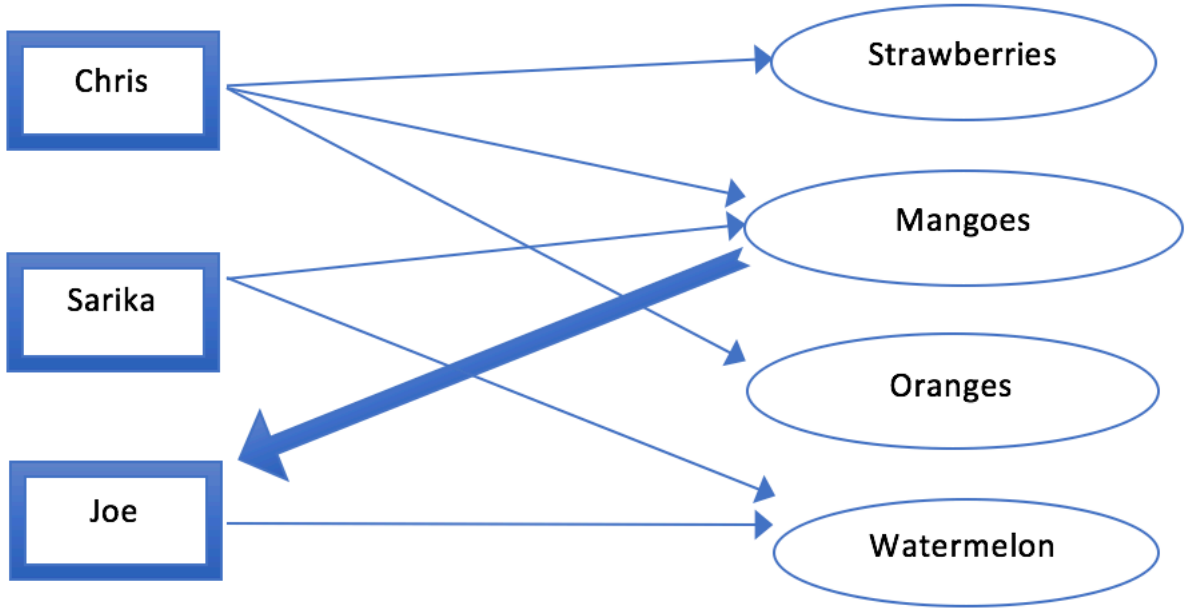
- Our recommendation system suggests the users with two items:
  1. Threads
  2. Groups

- Suggest trending and popular threads to users using the classic baseline predictor model.

- Suggest the threads or groups to users based on the words they have liked to view in the past.

Collaborative Filtering Techniques

- The foundation of collaborative filtering techniques is based on the simple idea that users who rate items similarly are likely to rate other items similarly.



Collaborative Filtering Technique

Preliminary Work Summary

- Recommendation systems can be built by a variety of algorithms.
- User based and item based recommendations are very intuitive and simple to implement, matrix factorization techniques are sophisticated algorithms which helps us to discover latent features between the interactions of users and items.
- Matrix factorization is a simple tool which helps us to discover the information hidden under data.
- As the name implies matrix factorization intends to factorize the matrix i.e., find the two or more matrices which when multiplied will give us back the original matrix
- The basic idea behind latent matrix factorization is the fact that if two or more users are giving high rating to an item, it is because they like some latent (hidden) feature behind that item

SJSU SAN JOSÉ STATE UNIVERSITY

Preliminary Work Summary (Contd.)

- Now if we have a matrix R which will be a matrix of users and item ids.
- Suppose we have m users and *n* items, we now want to find k latent features.
- We must find the two matrices P and Q such that their product is approximately R.
- Thus, matrix P would be a *m* by *k* matrix and matrix Q would be a *k* by *n* matrix.

$$R \approx P \,.\, Q^T = \hat{R}$$

Problems with other techniques

- Building a matrix requires some measure which associates the users with the items such as ‘rating’.
- Some numerical measure to build the matrix
- Yioop did not have a rating option which allowed users to give numerical rating to a thread or a group.
- Problem with pages belonging to very large categories
- Hidden features would be very large
- Computationally expensive

Classic Baseline Predictor

- We have used parts of the contribution to the “Bell-Kor’s Pragmatic Chaos” final solution, which won the Netflix Grand Prize to recommend threads or groups to users
- Collaborative filtering model try to capture the interaction between the users and the items that produce ratings differently
- There are some biases between the users and the items which the baseline predictors try to leverage
- Some users have a tendency of rating generously whereas some users have a tendency of rating poorly

Math behind Classic Baseline Predictor

- Let r denote the total average rating. A baseline predictor for an unknown rating $r_{u,i}$ is denoted by the $b_{u,i}$ and accounts for the user and item effects

$$b_{ui} = r + b_u + b_i$$

- Now for each user i in the corpus we first calculate the user bias for i by summing all the ratings which the user has rated and divide it by the number of items rated by the user. We now subtract total average rating from this computed measure which shows us how much a user is deviating from the average rating *r*

- For example, if a user A has rated 3 movies on 5 as 3, 2, 1 respectively; the overall average rating for all the users is 3.5; then the bias computed fir that user would be -1.5. This clearly states that the user A has a systematic tendency of rating movies, poorly i.e., below the average rating.

Math behind Baseline Predictor(Contd.)

- Similarly, for the items, there are certain items which are very popular and are rated highly whereas some items are rated poorly. This metric shows a lot about the quality of items.

- The metrics b_u and b_i represent how far the user is deviated from the average rating r. Suppose we want to predict what user “Sarika" would rate the restaurant "In-n-Out Burger" and the overall average rating = 3.7, the users and the restaurants bias comes out to be 1.7 and - 1.2 respectively then the predicted rating would be 3.7 + 1.7 -1.2 = 4.2

- We have considered the kind of rating a user gives, the quality of the items and the overall average rating. This way we get an insight about the users taste towards an item

Tweak Baseline Predictor for Yioop

- One of the challenges while building a baseline predictor for a dataset is the absence of ratings.
- The matrix of users and their ratings for items is sparse. Most of the time users don't rate; few users rate a lot of items while some rate very few items.
- The baseline predictors help in reducing these biases as the computation sees how a user rates an item.

SJSU SAN JOSÉ STATE UNIVERSITY

Implementing the baseline predictor

- For implementing the baseline predictors in Yioop, we required the 'rating' attribute - how a user rated a group or a thread
- Yioop which does have a voting system - thumbs up or thumbs down
- The number of times a user views a thread was logged in the ITEM_IMPRESSION table in the Yioop database. We made use of this corpus of data to build the 'rating' attribute.

Mathematically the user and item bias was calculated as below:

$$Average\ Rating\ r = \frac{Total\ number\ of\ threads\ viewed}{Total\ number\ of\ distinct\ threads}$$

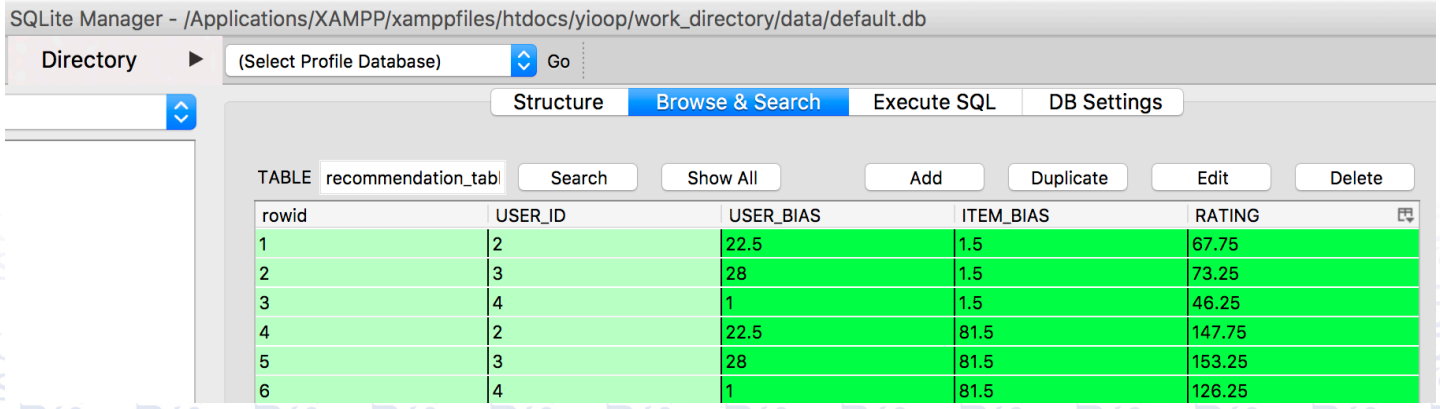
$$User\ bias = u_i = \frac{Number\ of\ views\ of\ a\ user\ for\ all\ threads}{Number\ of\ distinct\ threads\ a\ user\ has\ viewed} - average\ rating$$

$$Thread\ bias = t_i = \frac{Total\ Number\ of\ views\ a\ thread\ receives}{Number\ of\ distinct\ users\ viewing\ that\ thread} - average\ rating$$

Implementing baseline predictor (Contd.)

Mathematically,

$$Predicted\ user\ rating\ for\ a\ thread = \hat{r_{it}} = user\ bias\ u_i + thread\ bias\ t_i + average\ rating\ r.$$



rowid	USER_ID	USER_BIAS	ITEM_BIAS	RATING
1	2	22.5	1.5	67.75
2	3	28	1.5	73.25
3	4	1	1.5	46.25
4	2	22.5	81.5	147.75
5	3	28	81.5	153.25
6	4	1	81.5	126.25

Problem with the classic baseline predictor

- The Baseline Predictor ended up showing the most popular threads
- Baseline Predictor for Yioop made use of the number of times a user views a thread and the number of times a thread was viewed to calculate the user and the item bias
- Hence, we tried to improve the model and make more accurate recommendations by delving deep into the users impressions.
- We were leveraging the user and the thread biases. However, we had to add some contextual analysis to improve the model

Information Retrieval in Search Engines

- The task of retrieving data from a user's behavior has become common and necessary
- Information retrieval is searching a collection of data from text documents, databases, log searches, etc. and gaining insights about that data
- There are many approaches available to retrieve information from, in the search engine such as a probabilistic model incorporating the user's frame of mind while the user entered a query and enhancing the approximations to make better suggestions
- There is Latent Semantic Indexing (LSI) algorithm that builds a reduced dimensional vector space which represents the n-dimensional representation of a set of documents.

Term Frequency – Inverse Document Frequency

- Usually whenever a person queries, if a document contains the terms in the query then we score that document as being relevant.
- However, the number of occurrences of a word in a document is not taken into consideration while weighing the relevance of a document
- TF – IDF tries to weigh the documents such that the number of occurrences of the word is taken into consideration
- Words with a high TF-IDF value in a document implies a strong relationship of the word with the document

SJSU SAN JOSÉ STATE UNIVERSITY

TF - IDF

- We will first describe the problem followed by the mathematical background of the algorithm and examine the behavior of various variables in the following sections.
- Let us suppose we have a set of documents D: $d_1, d_2, d_3, \ldots, d_n$ and a user query $q = w_1, w_2, w_3$ for a set of words w_i. We want to return the subset of D documents which are the most relevant documents in the corpus based on the user's query.

Term Frequency Calculation with an example

- As an example, with three documents, try to understand the mathematics behind the term frequency calculation.

  Document 1: Flume is an Apache Hadoop project. Flume is a distributed, reliable service.

  Document 2: Hadoop has an ecosystem project called Spark which works in memory. Spark is an apache hadoop project

  Document 3: PHP is a good web server side programming language

- Now let us assume that a user has entered a query q: Hadoop project

SJSU SAN JOSÉ STATE UNIVERSITY

TF Calculation (Contd.)

Term Frequency Table for document 1

words	flume	is	an	apache	hadoop	project	distributed	reliable	service
frequency	2	2	1	1	1	1	1	1	1

$$TF = log(f_{t,d}) + 1 \; if \; f_{t,d} > 0 \; and \; 0 \; otherwise$$

words	flume	is	an	apache	hadoop	project	distributed	reliable	service
frequency	0.4	0.4	1	1	1	1	1	1	1

Also, there are few words which occur a lot of time in a big file.
To reduce the effect of such words we introduce the logarithm.

Inverse Document Frequency (IDF)

- While calculating the term frequency we are considering that all the words are equally important in a document
- However, it does not take into consideration the effect of a few words which occur in most of the documents
- IDF helps to leverage that bias:

Mathematically, IDF is calculated as below:

$$IDF(t) = \log\left(\frac{N}{N_t}\right)$$

where 'N' is the total number of documents in the corpus and N_t is the number of documents containing the term t.

IDF

- Let us compute the IDF for the word 'hadoop'.
- The total number of documents for our example is, $N = 3$
- The number of documents which contains the term Hadoop is, $N_t = 2$

$$IDF(hadoop) = \log\left(\frac{3}{2}\right) = 0.17$$

Sample IDF Table for document 1

Words	IDF
flume	0.47
is	0
a	0
hadoop	0.17
project	0.17

Weights TF * IDF

- By ***multiplying the term frequency and the inverse document frequency we can get the frequently occurring words in a document*** and we can also propagate the effect of the frequency that word occurring in the rest of the documents

- For example, in document 1 the word Hadoop has the normalized term frequency 1 and the IDF for Hadoop is 0.17 so the weight will be 0.17

Weights for query terms in the document

Query terms	Document 1	Document 2	Document 3
hadoop	0.17	0.17	0
project	0.17	0.17	0

Cosine Similarity

The cosine similarity gives us a measure of how important a document is to a user.

Mathematically,

$$Cosine\ similarity, CS(d_1, d_2) = (d_1 \bullet d_2) / (||d_1|| \cdot ||d_2||)$$

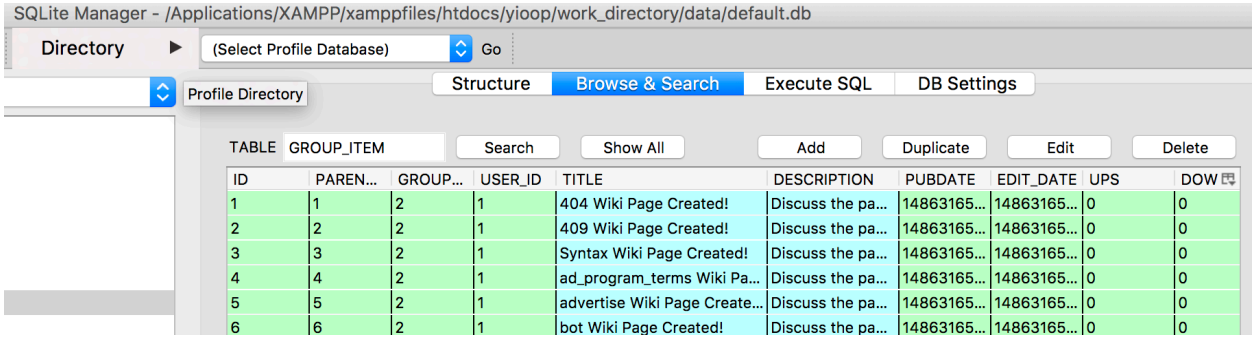
$$Dot\ product(d_1 \bullet d_2) =$$

$$d_1[0] \cdot d_2[0] \ldots d_n[0] + \ldots + d_1[n] \cdot \ldots \cdot d_n[n]$$

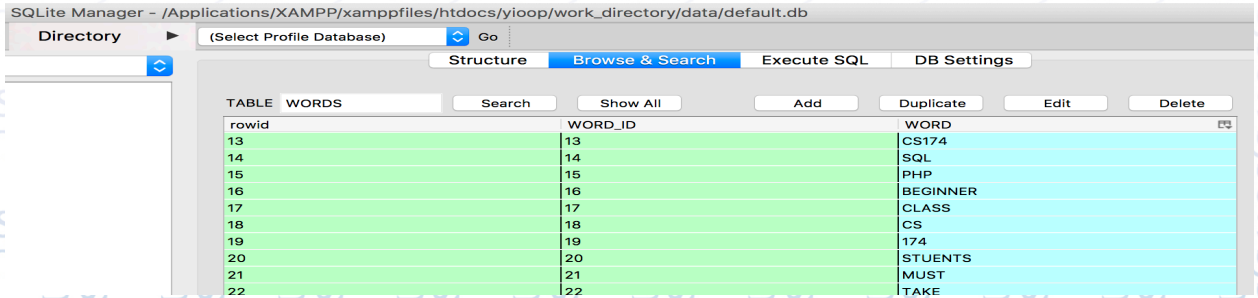
$$||d_n|| = \sqrt{d_n[0]^2 + d_n[1]^2 + \cdots + d_n[n]^2}$$

Tweaking TF – IDF for Yioop

- Every thread in Yioop has a title and description associated with it. We made use attributes to create the bag of words for us.
- Adding filters to remove the wiki pages



- Create bag of words based on the meta data - title and description



SJSU SAN JOSÉ STATE UNIVERSITY

Calculating TF for each word

- The GROUP_ITEM table in Yioop had all the description of a thread and had attributes like title, description, date published, group admin, the id of the group it belonged to, etc
- We scanned through each thread extracting the title and the description, followed by tokenizing the words.
- We made use of the number of times a thread is viewed and then summed the term frequency count for that word using the ITEM_WORD_FREQUENCY table to give us the measure of how many times a user has seen a word

$$\text{Term Frequency TF(items)} = log(f_{t,d}) + 1 \; if \; f_{t,d} > 0 \; and \; 0 \; otherwise$$

```
SELECT COUNT (*) AS FREQUENCY, USER_ID AS UID, WORD_ID AS WID FROM
ITEM_WORD_FREQUENCY IWF INNER JOIN ITEM_IMPRESSION II WHERE IWF.ITEM_ID =
II. ITEM_ID GROUP BY USER_ID, WORD_ID
```

IDF

- We now calculated the Inverse Document Frequency IDF for threads by scanning through the bag of words and for each word we measured the number of times it appeared in the threads as compared to the entire corpus.

  We calculated it using the formula below:

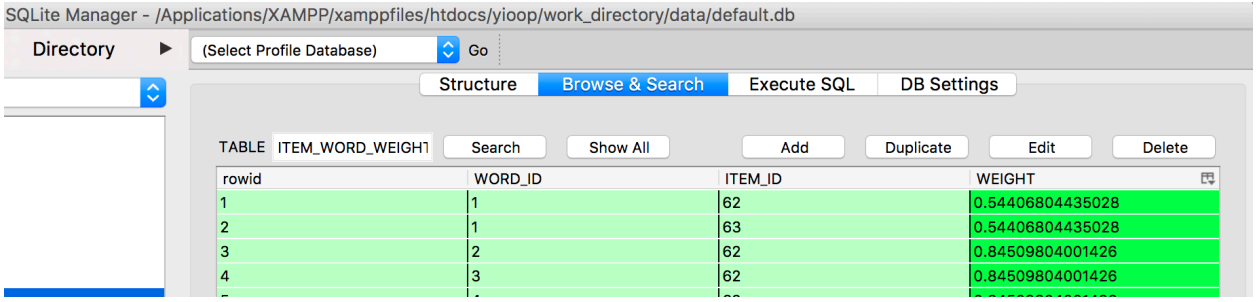
$$IDF_{w,t} = \log\left(\frac{Total\ number\ of\ threads}{Number\ of\ threads\ containing\ the\ word\ (w)}\right)$$

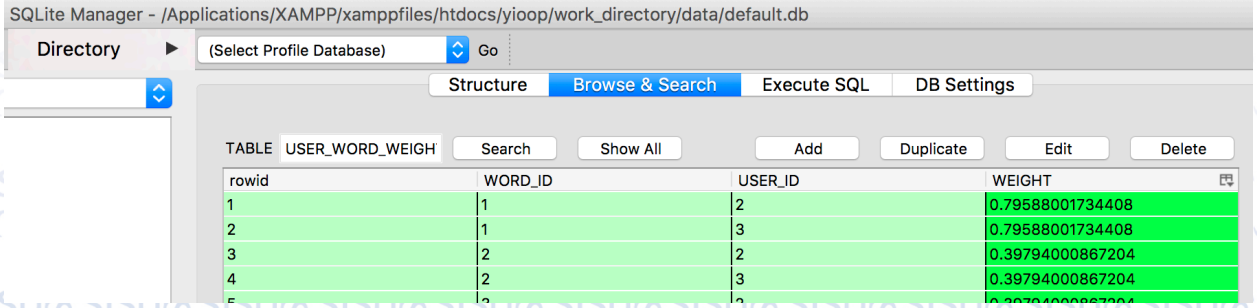
$$IDF_{w,u} = \log\left(\frac{Total\ number\ of\ users}{Number\ of\ users\ who\ viewed\ the\ word\ (w) + 1}\right)$$

We add 1 while calculating IDF for users to consider a case where no user has seen the word (thread)

SJSU SAN JOSÉ STATE UNIVERSITY

TF *IDF for Yioop





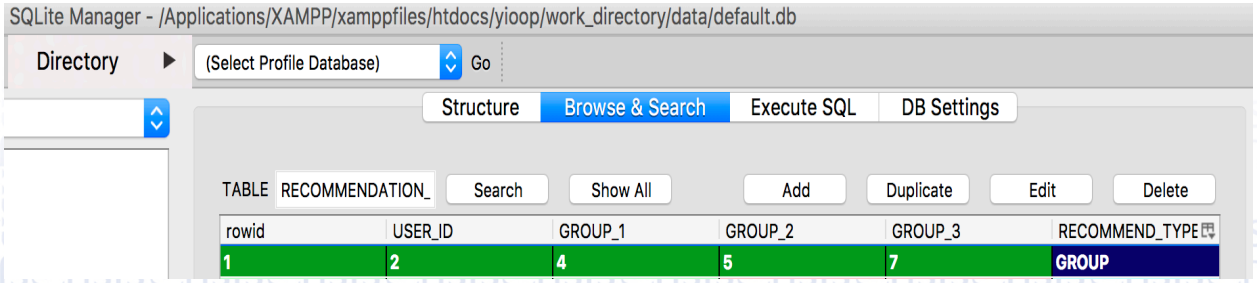
SJSU SAN JOSÉ STATE UNIVERSITY

Recommending threads and groups

- We calculate the cosine similarity between each user and the threads and try to find the threads which are most like the users taste.
- We recommend the top similar threads to users
- Groups in Yioop are made of threads. Group names in Yioop are very generic and does not really describe the content of what the group is about.
- So, we decided to make use of the threads title and description which makes up a group to recommend groups
- For each group, we first extracted all the threads belonging to a group and then summed up their cosine similarity measure for a group.

Advantages of TF – IDF for recommendation in Yioop

- The method is very intuitive as we are trying to find out how much relevant a word is to a user based on the number of times a user has viewed a word.
- We are trying to find out how important a word is to a thread based on the frequency of the word in the thread



Running the recommendation Job

- We were dealing with a lot of data as each user, thread/group logs were being examined

- If we ran a recommendation job in the background all the time based on the new logs received, it would slow down the computing power and there are chances that the computer might crash

- For faster execution and to recommend threads/groups to users instantly we had two parts to our job implementation:
  1. Offline Job – Computation Part
  2. Online recommendations – Rendering Part

SJSU SAN JOSÉ STATE UNIVERSITY

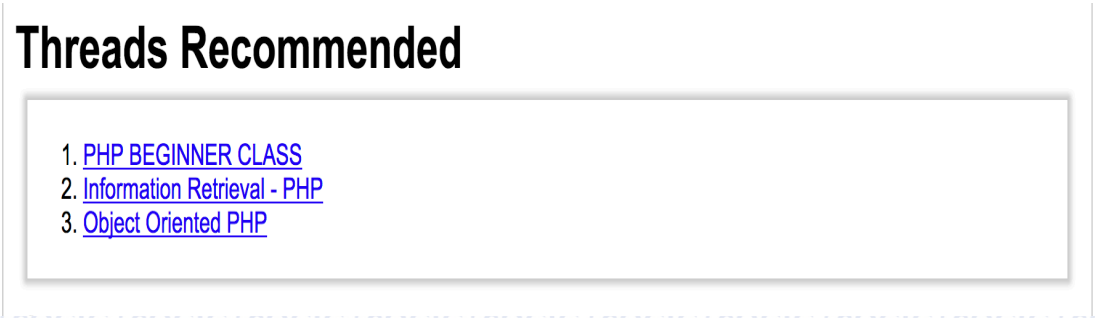
Offline Job – Computation Part

- A batched job is a job which runs for a data for a frame of time
- Yioop has a scalable framework media job which schedules job and runs based on when it is scheduled
- We made an offline job for the recommendation system which makes the Yioop system learn about the log
- The recommended threads are stored in RECOMMENDATION_LIST_THREADS_* and groups are stored in RECOMMENDATION_LIST_GROUPS_* tables. The * tells us which table is live. Here * takes the value of 1 or 2 depending on which table is live.
- The table which is live is not affected at all while updating.

SJSU SAN JOSÉ STATE UNIVERSITY

Online Recommendation (Rendering Part)

- The user is recommended with three items:
  1. Three most trending groups based on the classic baseline predictor model we built
  2.Three most interesting threads based on the words which he/she has viewed in the past
  3.Three most interesting groups based on the words which he/she has viewed in the past



Incremental Job

- Running a job from scratch increases the running time of the system and it does not improve the recommendations by much
- Having the recommendation engine run once a week is effective but we also want to take the current logs into consideration
- The incremental model tweaks the term frequencies of the users and threads and approximates its effect in the TF*IDF weights
- Our system captures the effects of the latest logs and tweaks the old weights calculated accordingly reducing the running time

$$TF * IDF_{U,W} = \frac{Total\ number\ of\ old\ users}{Total\ number\ of\ old\ and\ users + 1} * OLD\left(TF * IDF_{U,W}\right) + \frac{Total\ number\ of\ new\ users + 1}{Total\ number\ of\ old\ and\ new\ users + 1} * NEW\left(TF * IDF_{U,W}\right)$$

SJSU SAN JOSÉ STATE UNIVERSITY

User Feedback and Testing

- We checked the effectives of this system by asking some people
- We tested our model by asking people who were users of Yioop as well as some new users. 90 out of 100 people who tested this model had good reviews and had a better website experience
- Cold start problem was one of the issues we resolved because of user feedback
- We also did some unit testing on the model using Yioop's Testing Framework to check for correctness of the words extraction and frequency count created.

SJSU SAN JOSÉ STATE UNIVERSITY

User Interface

Threads Recommended

1. PHP BEGINNER CLASS
2. Information Retrieval - PHP
3. Object Oriented PHP

Groups Recommended

1. PHP ASSIGNMENT
2. INFORMATION RETREIVAL PHP
3. PHP CLASS

Trending Groups

1. PHP BEGINNERS TUTORIAL
2. Hadoop Architecture
3. Advanced PHP

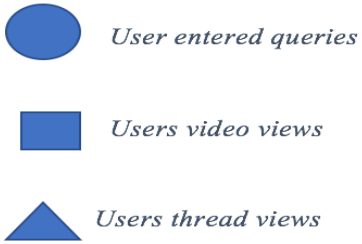
SJSU SAN JOSÉ STATE UNIVERSITY

Challenges while building the recommendation system

- Dealing with a lot of data
- Usually recommendation systems are built with some ratings data
- One of the challenges of this project was to figure out a way to rate a thread as no user could give a numeric rating to a thread/group.
- We made use users viewing history to convert it into a rating a user would give to a thread.
- For the term frequency inverse document frequency, we made use of the fact of the number of times a user views a word to calculate the metrics.

Areas of Improvement

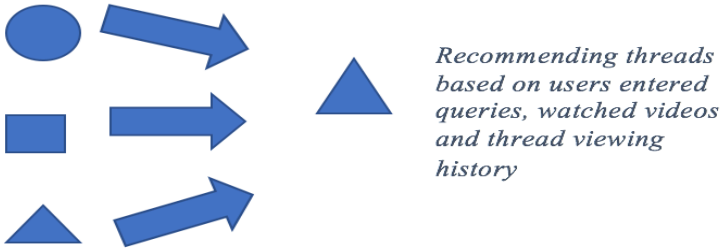
User Behavior



Recommendations

Now,

Multimodal recommendation system,



SJSU SAN JOSÉ STATE UNIVERSITY

Conclusion

- Built a recommendation system for Yioop using the baseline predictor
- We improved our recommendation system by incorporating the term frequency and the inverse document frequency in to our model
- We implemented the recommendation system and integrated it with Yioop to recommend the most popular threads/groups and the most relevant threads/groups based on users viewing history.
- To test the effectiveness, we asked several users to try our recommendation system and give their opinion. We also did some unit testing on the system

SJSU SAN JOSÉ STATE UNIVERSITY

References

[1] P. Resnick and H. R. Varian, “Recommender systems,” Communications of the ACM, vol. 40, no. 3, pp. 56–58, 1997.

[2] Ramos, Juan. "Using TF-IDF to Determine Word Relevance in Document Queries." [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.121.1424&rep=rep1&type=pdf [Accessed: 02 - May- 2017].

[3] Koren, Yehuda. The BellKor Solution to the Netflix Grand Prize (2009). [Online]. Available: http://www.netflixprize.com/assets/GrandPrize2009_BPC_BellKor.pdf [Accessed: 02 - May- 2016].

[4] "Netflix Prize." Wikipedia. Wikimedia Foundation. [Online]. Available: https://en.wikipedia.org/wiki/Netflix_Prize [Accessed: 02 - May- 2017].

SJSU SAN JOSÉ STATE UNIVERSITY

References (Contd.)

[5] "Term Frequency and Weighting." Term Frequency and Weighting. Cambridge University Press, 2008. [Online]. Available: https://nlp.stanford.edu/IR-book/html/htmledition/term-frequency-and-weighting-1.html [Accessed: 02 - May- 2017].

[6] Seekquarry.com, “Resources,” 2015. [Online]. Available: https://www.seekquarry.com/p/Resources. [Accessed: 02 - May- 2017].

[7] Dunning, Ted, and B. Ellen Friedman. Practical Machine Learning: Innovations in Recommendation. Beijing: O'Reilly, 2014. Print.

[8] Langville, A. N., & Meyer, C. D. (2012). Who's #1? The science of rating and ranking. Princeton: Princeton University Press.

[9] Cormen, T. H., Leiserson, C. E., & Rivest, R. L. (1990). Introduction to algorithms. Cambridge, MA: MIT Press.

SJSU SAN JOSÉ STATE UNIVERSITY

References (Contd.)

[10] Gábor Takács , István Pilászy , Bottyán Németh , Domonkos Tikk, Major components of the gravity recommendation system, ACM SIGKDD Explorations Newsletter, v.9 n.2, December 2007 [doi>10.1145/1345448.1345466]

Demo and Questions

SAN JOSÉ STATE UNIVERSITY *powering* SILICON VALLEY

Thank You!