RECOSMART: Knowledge Based Recommendation on Online Articles

Sricharan Pamuru (spk2f@mail.umkc.edu)

Prudhvi Raj Mudunuri (pm6y6@mail.umkc.edu)

Bhargav Krishna Velagapudi (bvg33@mail.umkc.edu)

Snehal Vantashala(svnq9@mail.umkc.edu)

Under the guidance of: Dr Yugyung Lee and Mayanka Chandrashekar, Computer science department, University of Missouri Kansas City

Nowadays the use of Information Retrieval has been profound. Recommendation systems act as part of Information Retrieval. It retrieving information from a collection of information resources. In this article we will present an overview and working of a novel smart recommendation system called RECOSMART which recommends the user with small set of sport articles based on his previous articles read by him.

1. INTRODUCTION

We have implemented a recommendation system which is a hybrid version based on Collaborative filtering and knowledge graph based technique. Content based filtering a sub concept in collaborative filtering where we feature vector for the given articles and user preferences to the Recommendation component to get the recommendation list. This concept is combined with knowledge models to get better results.

The basic workflow starts with exploiting the content using content model created using TF-IDF which contains the top TF-IDF words which add meaning to the article which runs using cosine similarity. For classification we have options to use the famous algorithms like K-Nearest Neighbors, Decision Trees, Naïve Bayes, Artificial Neural Networks or Support Vector Machines. As we are using text data Naïve Bayes is proven to be the best classification algorithm.

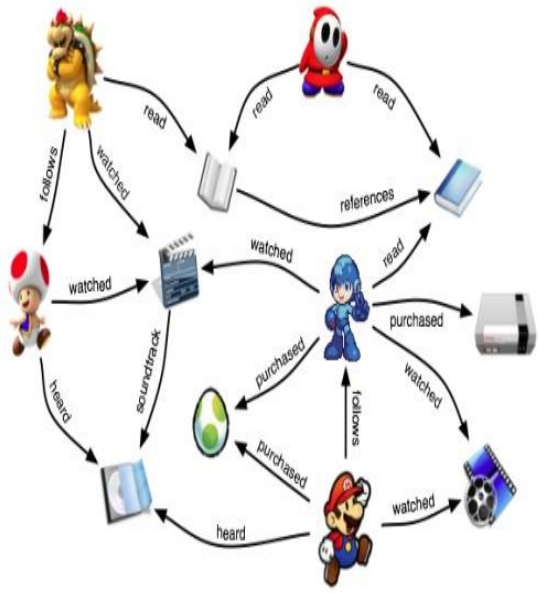

Figure 1: Graph based Recommendation

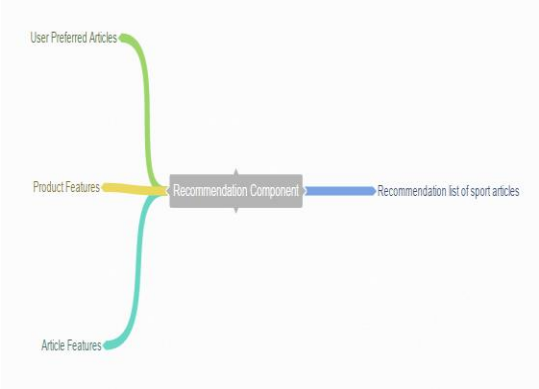

Figure 2: concept implementation

The figure represents the concept implementation of the RECOSMART developed. The recommendations are emitted as a result of querying the ontology generated from the dataset.

We have tested RECOSMART using BBCsport article dataset which has categories of sports which are correctly categorized using the LDA algorithm we have used.

The motivation of building a recommendation system for articles is an aspect of providing some kind of intuitive outcome for people who read a specific post or blog. In our case, the thought of binding different sources altogether in order to obtain some sort of output which will yield to dynamic data fed in the form of RSS, has been an ideological aspect that we have considered significant to understanding how the data trends and topics of approach have to be brought together. This combination will enable a user to stream into topics of interest across more than one domain. The machine would be able to return a suggestive article in the list through various technologies that we plan on implementing in this project

The goal of our project is to come up with a recommendation system for articles in different domain areas to bring the user a specific search blog or engine. It will help the user to understand how the scalability and necessity of the article is considered. The page rank would be used as a measure to understand the sequence of recommendations put forward [1]. Based upon this, the articles returned would focus on terms that have a similarity index, matching the score of the previously iterated indices. There would be a relative rank associated that would be used to figure out the magnanimity of the article which would have to be returned.

2. RELATED WORK

Recommendation engine is a platform for the machine to be trained and tested. Simulation of such technology actually occurs by rapidly searching the inner context. Big data mining tools focus on scalability and larger data sets are compared in order to retrieve information which is adaptive to the respective target, which is the articles. The NLP tools will help us extract a relation and naming the entities that is crucial to construction of feature vectors. They are combined with machine learning algorithms and tools to form an ontology. The ontologies learned will altogether contribute in constructing the recommendation system, which is the ultimate motivation of our project.

3. PROPOSED SOLUTION

Our recommendation engine would return the articles to be recommended in a list along with the associated hyperlink. This would help the user view topics of similar interest. Machine learning algorithms run in the background to bring a result yielding a knowledge graph where there is a proper connection between the elements, which in our case is articles, dynamically.

The outcome of the project would be to bring out articles to be recommended in a list which will help the user view similarity index based articles. Our

project is very useful for people who read blogs and articles. It would be an interactive web page where similar index based documents have a value returned which match the preexisting ones.

4. IMPLEMENTATION

Our project aims at fetching real time data in the form of RSS(Rich Site Summary) which will be dynamic. The recommendation engine will use collaborative filtering and it serves as a branch of deep learning. It will enable the machine to undergo a deep analysis and filter the data collection that is appropriate for mining. Technology has developed immensely and it is required for the machine to be able to understand each fragment in order to return a better perspective of how the inner mechanism works. Neural networks would basically iterate through every fragment and in our case the articles have to be searched and read through to understand how the term frequency and inverse document frequency matches. (TF-IDF) is a key constraint that will enable information retrieval to be scaled. We would obtain the respective important term that will differentiate a specific document when compared with another.

Github URL:
https://github.com/SricharanPamuru/KDM-Summer-2016

Youtube URL:

https://www.youtube.com/watch?v=EENdGUg-4V8&feature=youtu.be

Moreover, the information retrieval can be characteristic of the artificial intelligence based approach that occurs in the learning of in gram to retrieve a continuous set of words which will bind the data and integrate the search. The recommendation system would then focus on setting up the link between the APIs and return the set of article. In order to validate the scope of the project we have included a set of static data to showcase the output. Further analysis can make the scope to be extended and consist of dynamic streaming data. SparkQL queries have been run in Apache Jena Fuseki to showcase the data analysis which focuses mainly on mining.

5. ALGORITHM USED

The recommendation system uses the techniques of collaborative filtering, which in turn is formed as a result of the Alternating Least squares method.

$$P_{aj} = \bar{r}_a + \frac{\sum_{i \in NS_a} sim(a,i) * (r_{ij} - \bar{r}_i)}{\sum_{i \in NS_a} |sim(a,i)|}$$

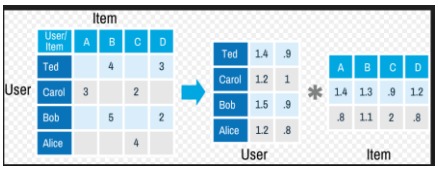

Figure 3: Collaborative Filtering

User is able to obtain a recommended article, based on the one he chooses. It is in turn computed based on the trained data set. A confusion matrix is generated and it would create a parameter that serves as the factor in determining which article has to be recommended.

6. WEB BASED UI

As part of the frontend we have developed a web based application which consists of all the privileges the like login, logout facility and the user can read all the articles from the database and

stores articles of his interest and code also contains socket programming in which the request is sent form the client to the recommendation engine i.e. spark streaming and generated recommendations are sent back to the client. All this is achieved using socket programming and we have used java for developing servlet based application.

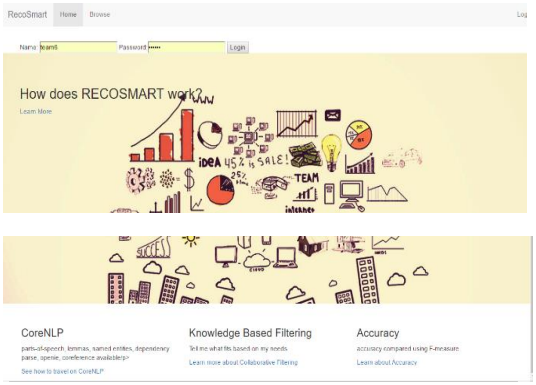

Figure 4: Login page of the application

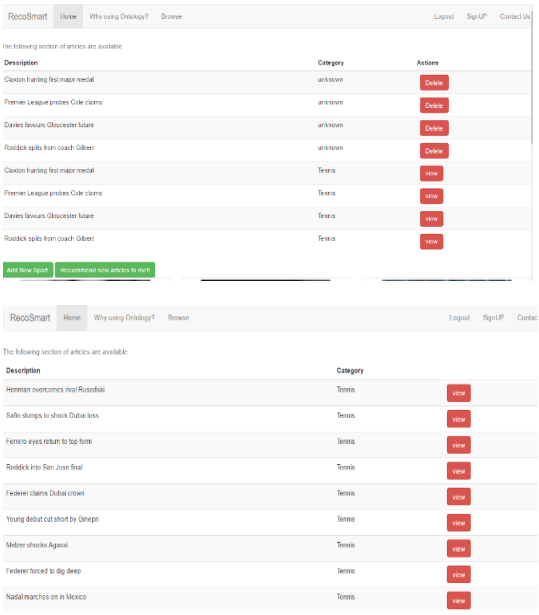

Figure 5: Webpage design

7. SPARK PROCESSING

Once the socket connection is set to listen to the server, the specific ratings of the recommended movies are listed as follows which actually are filtered using the Alternating Least squares algorithm to ensure the implementation of the collaborative filtering model.

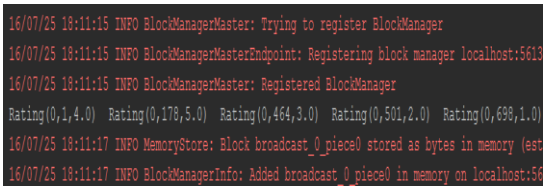

Figure 6: Rating

The recommendation of articles return the mapping of confusion matrix as follows where a sample gist of articles have been presented to ensure the compatibility of data and accuracy formed.

```
Articles recommended for you:
Rating(0,428,3.143457488953163)
 1: Barwick installed as new FA boss
Rating(0,140,2.937588803690186)
 2: Bashar delighted after series win
Rating(0,204,3.407346477927328)
 3: Holding slams Twenty20 'rubbish'
Rating(0,228,2.6127240641799485)
 4: Moyes U-turn on Beattie dismissal
Rating(0,492,2.378373674485209)
 5: Vickery out of Six Nations
Rating(0,192,2.8283972681346135)
 6: Kenya cricket boss fights sacking
Rating(0,160,2.551279143418181)
 7: Hayden ruled out of Kiwi showdown
Rating(0,460,2.2106880990473723)
 8: City tie on as weather hits games
Rating(0,548,2.8798062994769347)
 9: Charvis set to lose fitness bid
Rating(0,696,2.2839642600434904)
10: Clijsters could play Aussie Open
Rating(0,528,2.866233824091595)
11: Woodward eyes Brennan for Lions
Rating(0,196,2.996065109301677)
12: Kallis keen on Glamorgan return
Rating(0,292,3.2972681470212004)
13: Coach Ranieri sacked by Valencia
Rating(0,344,3.181312886932587)
```

Fig 7: Articles recommended based on rating

The final recommendation system has the server return the following articles and it forms the generation of output based on the machine learning techniques for training and testing the data model synthesized.

Figure 8: 5 Articles recommended by the system

The ontology generated from the processing can be run in WebVowl and the following schema can be generated as a result of that and this shows how the different categories have been linked.

Implementation outcomes for visualization:

7.1 **WebVowl**

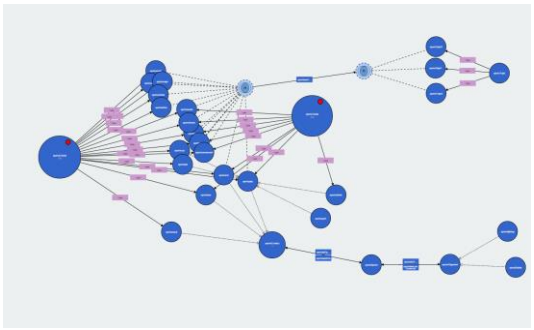

Figure 9: ONTOLOGY generated

SparkQL queries can be run in either Protégé or Apache Jena Fuseki to display the data modules and the following can be generated:

7.2 SparkQL

```
SELECT ?subject ?predicate ?object
WHERE {
  ?subject ?predicate ?object
}
LIMIT 25
```

Figure 10: sample query on ontology

7.3 Apache Jena Fuseki:

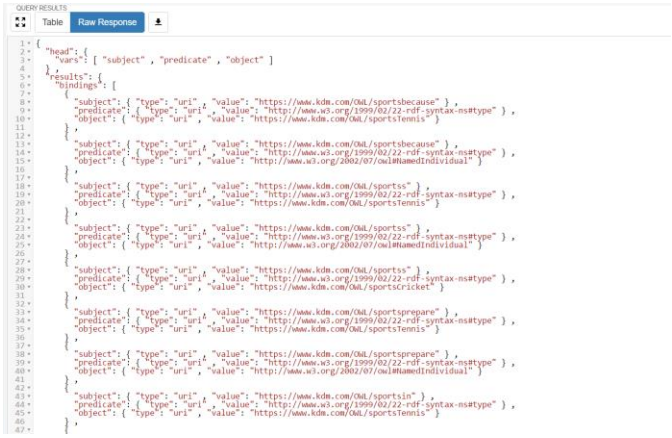

Figure 11:Queries run

The query run in Protégé to display the table based format can be generated as such:

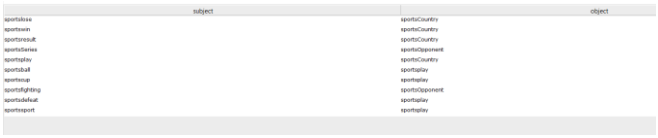

Figure 12: Table simulation formed

7.4 **OntoGraf in Protégé**:

The OntoGraph has been formed in Protégé through the OwlSports.owl file formed as a result of execution. This ensures that the data is linked and forms a complete knowledge base.

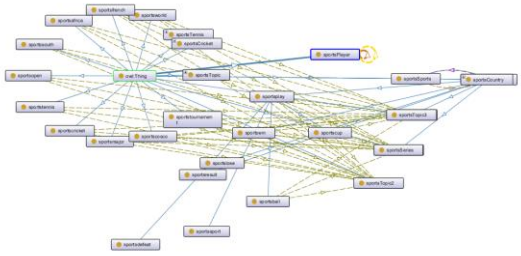

Figure 13: OntoGraf

8 SOFTWARE ARCHITECTURE

The web application will show the user a set of articles. A socket connection has been coded to run the Spark server and the program will run for streaming which would involve collaborative filtering and the MLlib will fetch the libraries and recommend the system. Ontologies are generated and from here information is retrieved. SparQL will perform queries on data. Information is retrieved and the data is linked based on these factors.

The server will then return the data to the web application which will show the user a set of recommended articles. In order to extend the application to dynamic feed such as RSS, it would be required to perform Spark streaming and this is a future area of expansion.

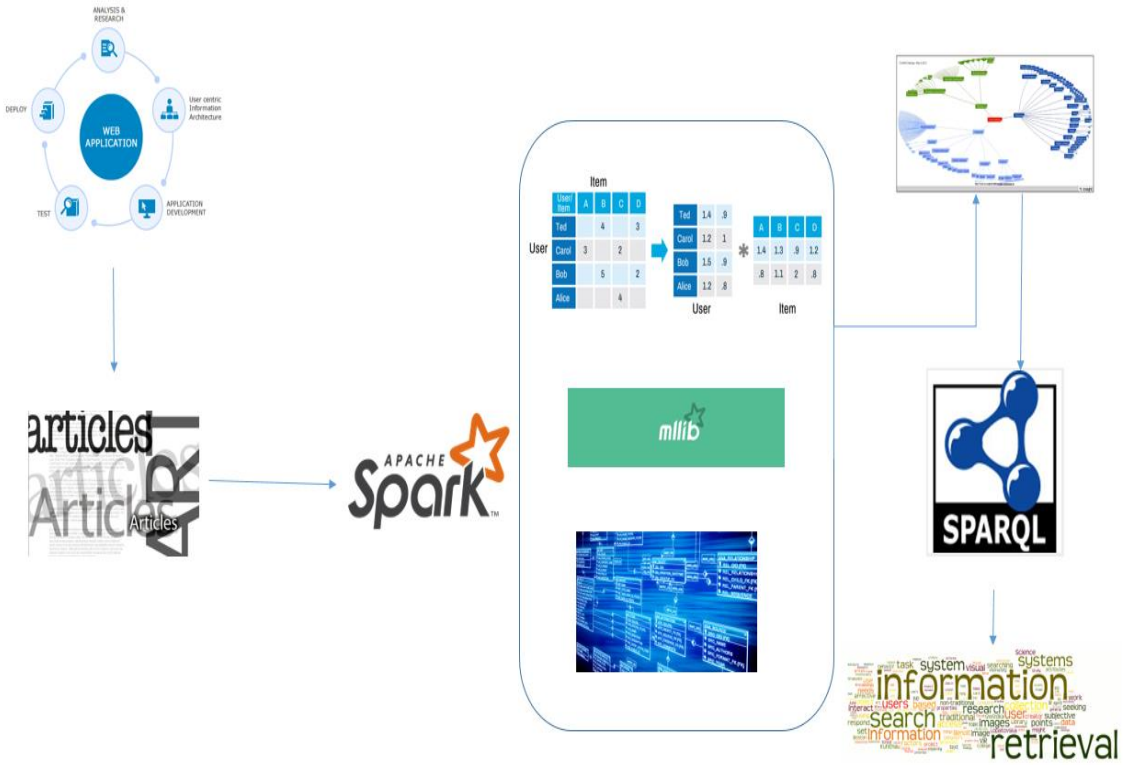

Figure 12: Architecture of Model

The architecture of the model has been explained in Figure 12 and the workflow to simulate the working of the model based on the various constraints has been explained depending upon the specific output that is categorized depending upon the queries and the set of information to be retrieved.

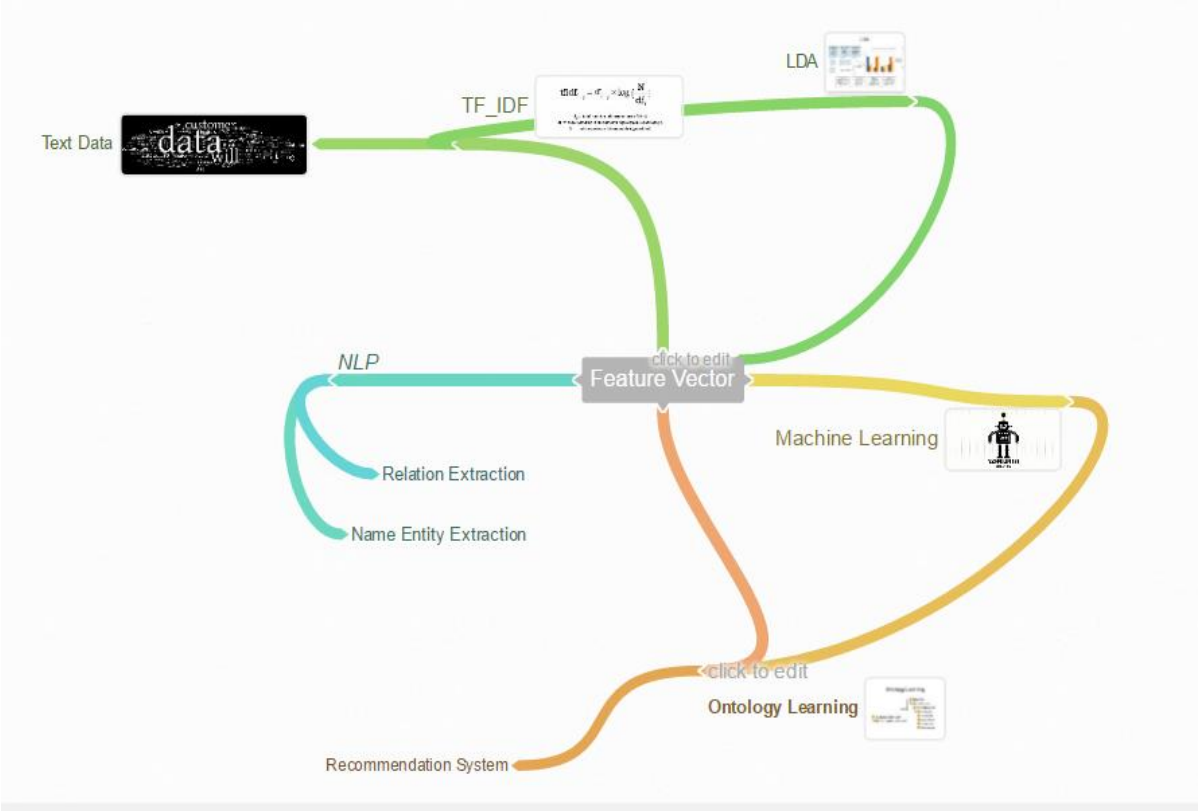

Figure 15: Workflow of the model

The user visits an article once then data from the article is extracted and send for NLP processing where different Computational Linguistics are carried out. Data is given to the NLP goes through sentence segmentation; Tokenization; Stemming; Part-of-speech tagging; parsing; Named Entity Recognition; Co-reference Resolution. Data is passed further to tf-idf class where the information retrieval process is done based on the cosine similarity and words that provide meaning to the document are retrieved and sent to the New York Article search API. The New York Article search API which takes the keywords from the tf-idf process and these keywords are given as the input to the API as a query. API writes top articles and their hits in different social networking sites.

9 Existing Services used

New York Times articles search API enables users to search New York times articles from 1851. Using this api users/ developers can extract data based on keyword, it also offers facet searching(multidimensional).
Information retrieval contains headline, lead paragraphs, related docs, abstract, time specific details can also be retrieved etc.

10 New feature implemented

We intend to develop a content based recommendation system based on collaborative filtering and ranking the recommendation with scores. Our developed model can tackle cold start. Cold start is a state of the machine when there are no previous train data. We intend to get the ontology graph from the

content through machine learning algorithms and recommend based on the first article the user reads. Available recommendation systems struggle during the cold start phase.

Our developed model takes user interest to train the model by asking the user to rate the rate the article before he leaves. If the user has given a negative feed back then then the particular article will not show up in other recommendation for that particular user even when the content matches his query. The user will get a recommendation of article based upon the training data set, and as more hits are obtained the user will look to implement a machine learned recommendation system.

In our model, the specific ontology is generated based on the recommendation system and we have used Protégé to visualize the data model. Apache Jena Fuseki can also be used to run the SparkQL queries which will generate the required data mined. This mainly happens to help access the data and forms a complete knowledge graph.

11 RESULTS AND EVAULATION

i. *Accuracy*: Selection of only most relevant records (Precision), (recall) all relevant ones. The accuracy of the developed model is calculated as 27%.

ii. *A) Recall:* Success in retrieving all correct documents.

B) Precision: Success in retrieving the most relevant documents.

```
Articles recommended for user 0:
Rating(1,572,5.491741224940043)
 1: Williams stays on despite dispute
Rating(1,3233,4.801842767943785)
 2: Imran Khan scores debut century in Test Match
Rating(1,527,4.618497973400906)
 3: France v Wales (Sat)
Rating(1,318,4.597945640980962)
 4: Celtic make late bid for Bellamy
Rating(1,2776,4.510523058351275)
 5: Djokovic wins French Open
```

The articles have been observed to return the matrix that is mapped depending upon the machine learning algorithms run.

iii. *Performance:*

It is slow for processing and requires arguments to be passed in order to observe data patterns and perform information retrieval fast. Our project can face issues when it comes to deployment dynamically as we have to obtain the RSS feed which serves as the main constraint to show the update of the feed data. Moreover, the program would serve differently for various users to suit his preferences.

The machine would not be able to scale exponentially and provide incremental gains over due course. It would limit the user's self-search mechanism as the recommendations are showed based on preferences.

The time taken for the training is more and hence the machine learning algorithms need to be trained first in order to run on the system. User's preferences can also be obtained through the union and intersection of various terms.

The Cold Start Problem is a major issue in recommendation systems.Moreover, passing arguments to enhance the speed of the respective processing is time

consuming. It is also an area of concern and further speculation into this project might lead to a more stipulated mechanism to ensure that the augmentation and processing frame can be developed.

12 CONCLUSION

The specific model has been developed to be a recommendation system of articles. However, it can be expanded in order to inculcate the RSS features which was tried but it was difficult to link both the constraints and try to bring about significant changes.

This model is trained depending upon the user and varies as per their perception.

13 FUTURE WORK

The project can be expanded in various domains individually such as news articles, journals, and scientific data and there are still certain areas of research. Efficient algorithms can be computed. Probably, the cold start problem is something that future researchers would look into so that there is no issue when it comes to certain issues existing over time. It would also be a prospective area of research to increase the processing time and there is a dynamic data implementation which would create representation to understand how RSS feed can be involved in building a more accurate data set to ensure the high level of accuracy.

13 BIBLOGRAPHY

1. Spangher, Alexander. "Building the Next New York Times Recommendation Engine." Open Blog. New York Times, 11 Aug. 2015. Web. 25 June 2016.
2. Jain, Aarshay. "Quick Guide to Build a Recommendation Engine in Python." Analytics Vidhya. N.p., 02 June 2016. Web. 25 June 2016.
3. Ridwan, Mahmud. "Predicting Likes: Inside A Simple Recommendation Engine's Algorithms." Toptal Engineering Blog. N.p., 12 Mar. 2015. Web. 25 June 2016
4. Filloux, Frederic. "DeepMind Could Bring The Best News Recommendation Engine." Medium. Monday Note, 20 Mar. 2016. Web. 25 June 2016.
5. Das, A., Datar, M., Garg, A, and Rajaram, S. Google news personalization: scalable online collaborative filtering. In Proceedings of WWW (2007).
6. Davidson, J., Liebald, B., Liu, J., Nandy, P. Van Vleet, T., Gargi, U., Gupta, S., He, Y., Lambert, M., Livingston, B. and Sampath, D. The youtube video recommendation system. In ACM Conference on Recommender Systems (2010). ACM Press, New York, 293–296.
7. Agarwal, D. and Chen, B.-C. Regression-based latent factor models. In Proceedings of KDD (2009). ACM Press, New York, 19–28.
8. Li, L., Chu, W., Langford, J. and Schapire, R. A contextual-bandit approach to personalized news article recommendation. In Proceedings of the 19th International Conference on World Wide Web. ACM Press, New York (2010) 661–670.
9. Linden, G., Smith, B. and York, J. Amazon.com recommendations: Item-to-item collaborative filtering. IEEE Internet Computing 7, 1 (2003), 76–80.
10. Stern, D., Herbrich, R. and Graepel, T. Matchbox: Large scale online bayesian recommendations. In

Proceedings of WWW (2009). ACM Press, New York, 111–120.