

DONE BY:

1.SRICHARAN PAMURU(28)

2.PRUDHVI RAJ MUDUNURI (22)

3.SNEHAL VANTASHALA (41)

4. BHARGAV KRISHNA VELAGAPUDI (42)

KNOWLEDGE DISCOVERY AND MANAGEMENT

PROJECT

SUMMER 2016

RECOSMART

Contents

[1. Project Proposal: 2](#_Toc454575922)

[a. Motivation: 2](#_Toc454575923)

[b. Objectives: 2](#_Toc454575924)

[c. Expected Outcomes: 2](#_Toc454575925)

[2. Domain: 3](#_Toc454575926)

[3. Data Collection: 4](#_Toc454575927)

[4. Report: 5](#_Toc454575928)

[a. TF-IDF: 5](#_Toc454575929)

[b. NER: 7](#_Toc454575930)

[5. Design of features 21](#_Toc454575931)

[a. Software Architecture: 21](#_Toc454575932)

[i. Architecture of Model: 21](#_Toc454575933)

[ii. UML Modeling: 22](#_Toc454575934)

[b. Workflow: 26](#_Toc454575935)

[c. Existing Services used: 27](#_Toc454575936)

[d. New feature implemented: 28](#_Toc454575937)

[6. Project Management: 29](#_Toc454575938)

[a. Individual Contribution: 30](#_Toc454575939)

[b. Zenhub/Github Screenshots: 31](#_Toc454575940)

[i. Full timeline: 31](#_Toc454575941)

[ii. Report 1: 32](#_Toc454575942)

[iii. Report 2: 32](#_Toc454575943)

[iv. Report 3: 33](#_Toc454575944)

[v. Report 4: 33](#_Toc454575945)

[vi. Burndown chart: 34](#_Toc454575946)

[vii. Pulse: **Error! Bookmark not defined.**](#_Toc454575947)

[viii. Traffic: 34](#_Toc454575948)

[ix. Commits: 35](#_Toc454575949)

[x. Code Frequency: 36](#_Toc454575950)

[c. Concerns/Issue: 36](#_Toc454575951)

[d. Future Work: 37](#_Toc454575952)

[7. Bibliography: 38](#_Toc454575953)

# Project Proposal:

## Motivation:

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

## Objectives:

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. 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.

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.

## Expected Outcomes:

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.

# Domain:

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.

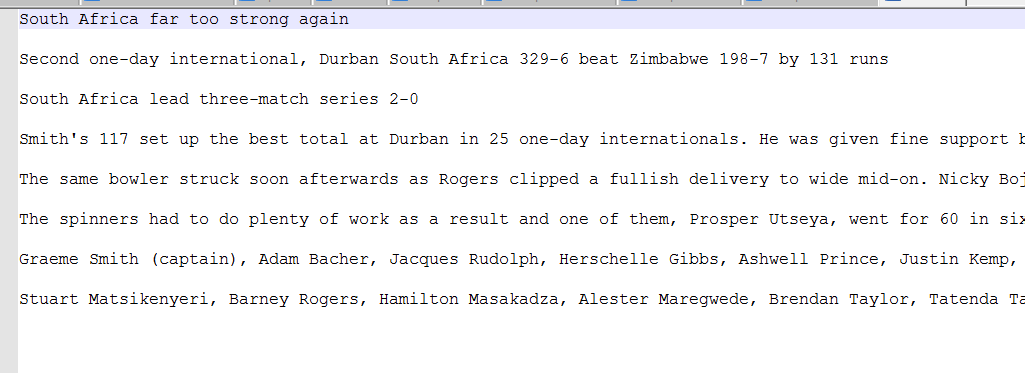
Recommendation engine is a platform for the machine to be trained and tested. Simulation of such technology actually occurs by rapidly searching the inner context. Big data mining tools focus on scalability and larger data sets are compared in order to retrieve information which is adaptive to the respective 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.

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.

# Data Collection:

The data collection for this specific project has been generated from the BBC news sites where the various articles sections have been formed. It has helped us understand how the articles are generated and their category mapping.

Sample article under Sports Category:



Similarly, the data collection is static right now and we have used the BBC datasets as a training set. The overall outcome would be to dynamically obtain data in the form of RSS feed. The RSS feed would use the standard web feed formats in order to publish updates of information that can be categorized as blogs, headlines and also includes audio, video files. The metadata would explain about the specific data content and focus on publish dates, time and author information.

We intend to pull the required information and accordingly sort it as per the user’s search. Dynamically, it would be feasible to categorize it based on the XML file format generated. The RSS reader would check how the user checks feed for information and the user interface will serve as a front-end for interpreting such information. The back end would include the process of fetching data and perform the NLP operations to understand the grammar and contextual meaning.

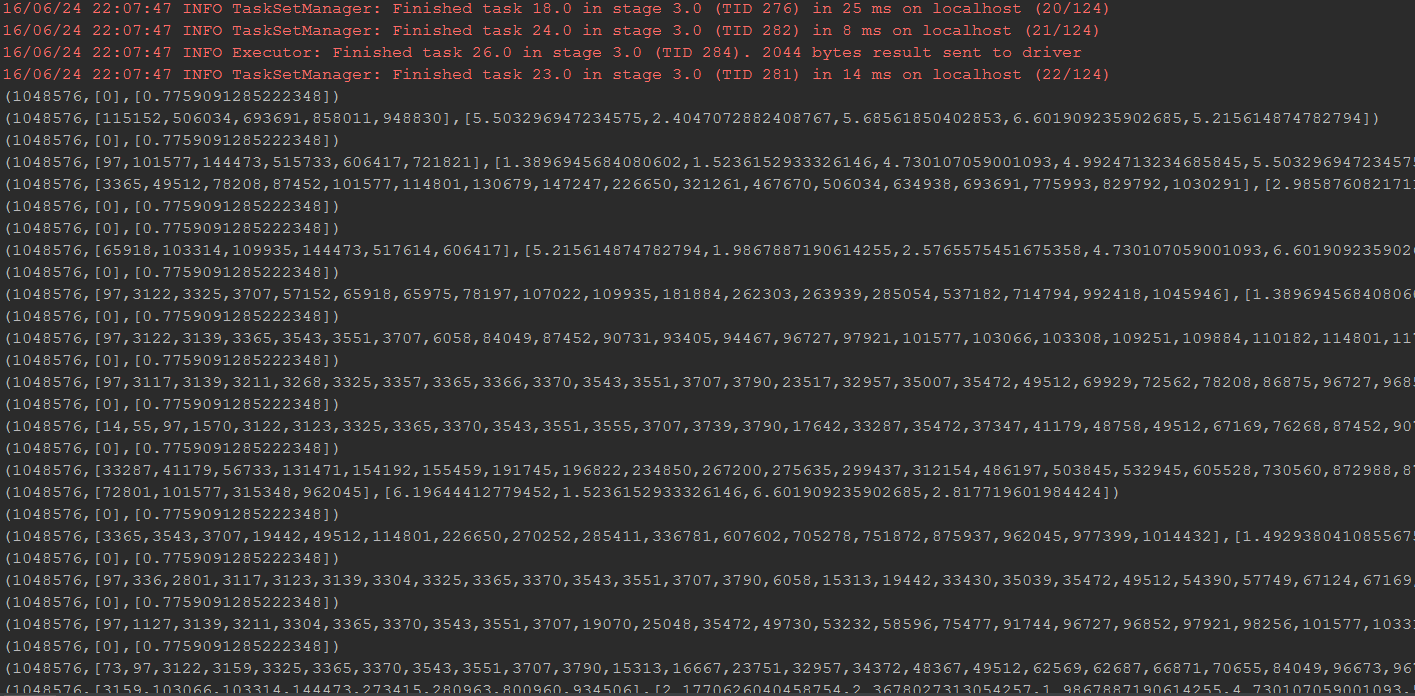
URL: <http://mlg.ucd.ie/datasets/bbc.html>

# Reports:

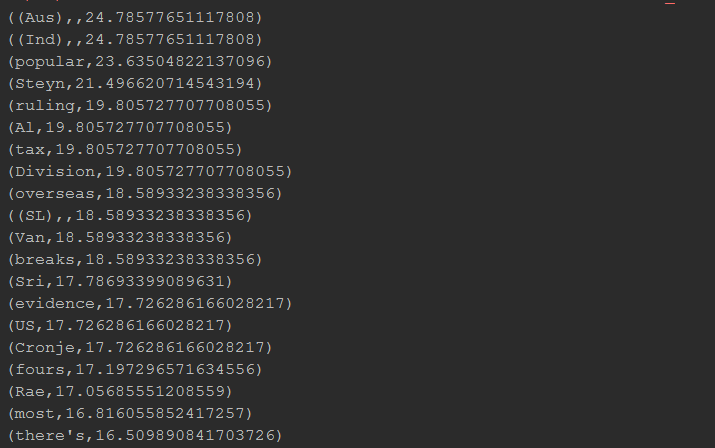
## Project Report 1:

### TF-IDF:

In the first phase of the increment we collected a set of news articles from BBC and have executed TF-IDF from where we obtained the following output which provides the most repeated term. The program filters the files specific in the directory and the term sets are returned along with the frequency count. The vectors are obtained and from here the knowledge base will be crucial in implementation of the knowledge graph.

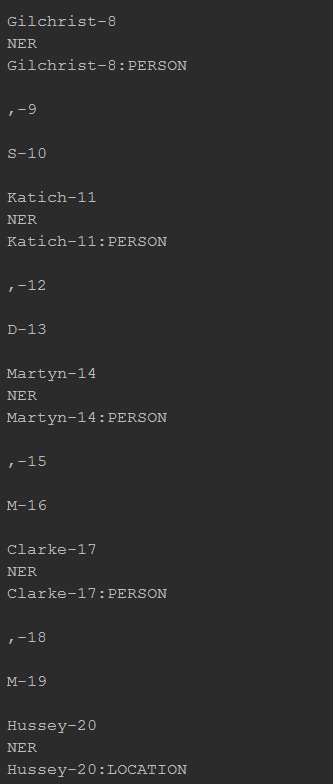
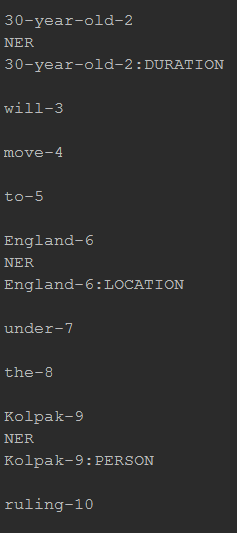


This was followed by determination of the term frequency and inverse document frequency as a measure of how the most important terms are obtained. The top terms are taken and there is an integration of top constraints to obtain a specific link.



The files used here are the cricket directories that contain content and the top most words are filtered to return a set of terms. The words are returned. Stop words have been filtered and accordingly the TF-IDF table can be executed. The code is scalable

### NER:

The Named Entity recognition module was run to return the specific naming conventions. The knowledge graph is extracted based on this and we can show the interlinking of an article through this. The conventional data format is used to show how the data can be superimposed can be seen as the manner for showing the format and class to which it belongs.

## Project Report 2

This increment focuses on detecting the topic of the document using LDA.

The recommendation systems have to be accurate on suggesting the content to the user. We have implemented the topic detection in our project as the recommendation system we build is going to suggest the user with online articles of his interest we have to collect the user interest based on the topics of the articles the user is reading.

Generally the another functioning of the recommendation system can be like recommending the articles with highest click through rate but the article is of no interest to the user. To avoid such unrequired suggestions in our model we have included the LDA to detect the user interest and get the model up and running such that all the recommendations emitted as the output will be of use to the user.

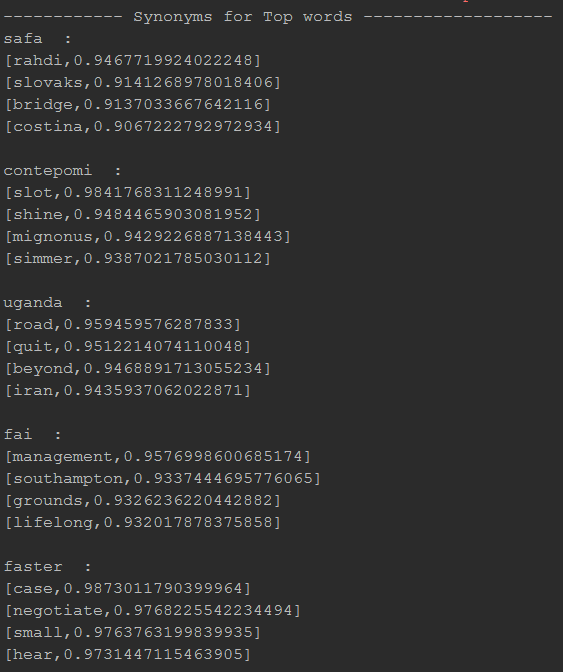
Machine learning algorithms such as collaborative filtering uses the ratings given by the user to find the similarity of choices and uses them as the knowledge base to recommend the user different online articles. In some cases there are latent values like one article which no one has ever rated before. This article might be a good article but no one has visited it till now. In collaborative filtering this article has the least chance of popping up as a recommendation.

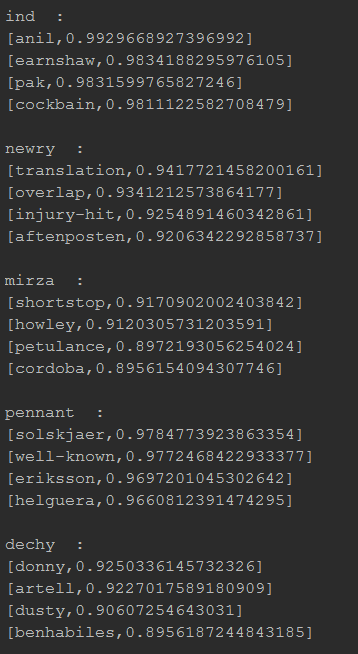
Using our approach as discussed will populate the articles using collaborative filtering but with additional layer of filtering using the LDA for the topic detection. This way the user is given suggestions which contain articles with the same topic of interest and all the suggestions will be of the same topic.

The only drawback which we are considering is that, if all the suggestions belong to the same topic then we are just hindering the user from discovering other interests which is also damaging for the recommendation systems as user preferences keep changing from time to time.

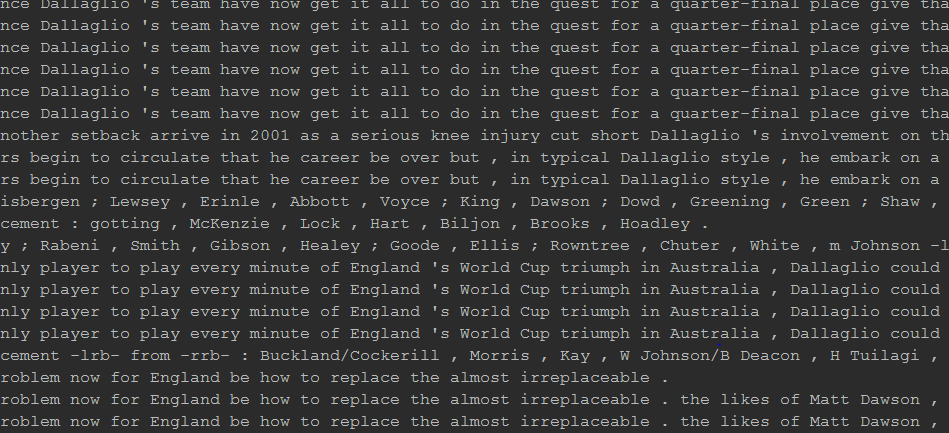
In-efficient recommendation systems neglect the topic discovery and go with only collaborative filtering.

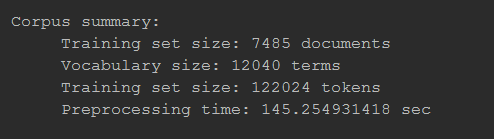
### Results: -





The specific topics are mined together and the frequency can be determined. This will help enhance the similarity and observe patterns which provide an interlinking of features. Feature vectors observe the relation. The lexical dictionary that has been used here is WordNet which provides correlation and the specific meaning is returned. The synonyms and exact referencing has been done over here. The synonyms provide us an insight into the overall mechanism of the algorithm. The words are searched through the lexical frame and the observation of top words will return the output associated with TF-IDF processing.

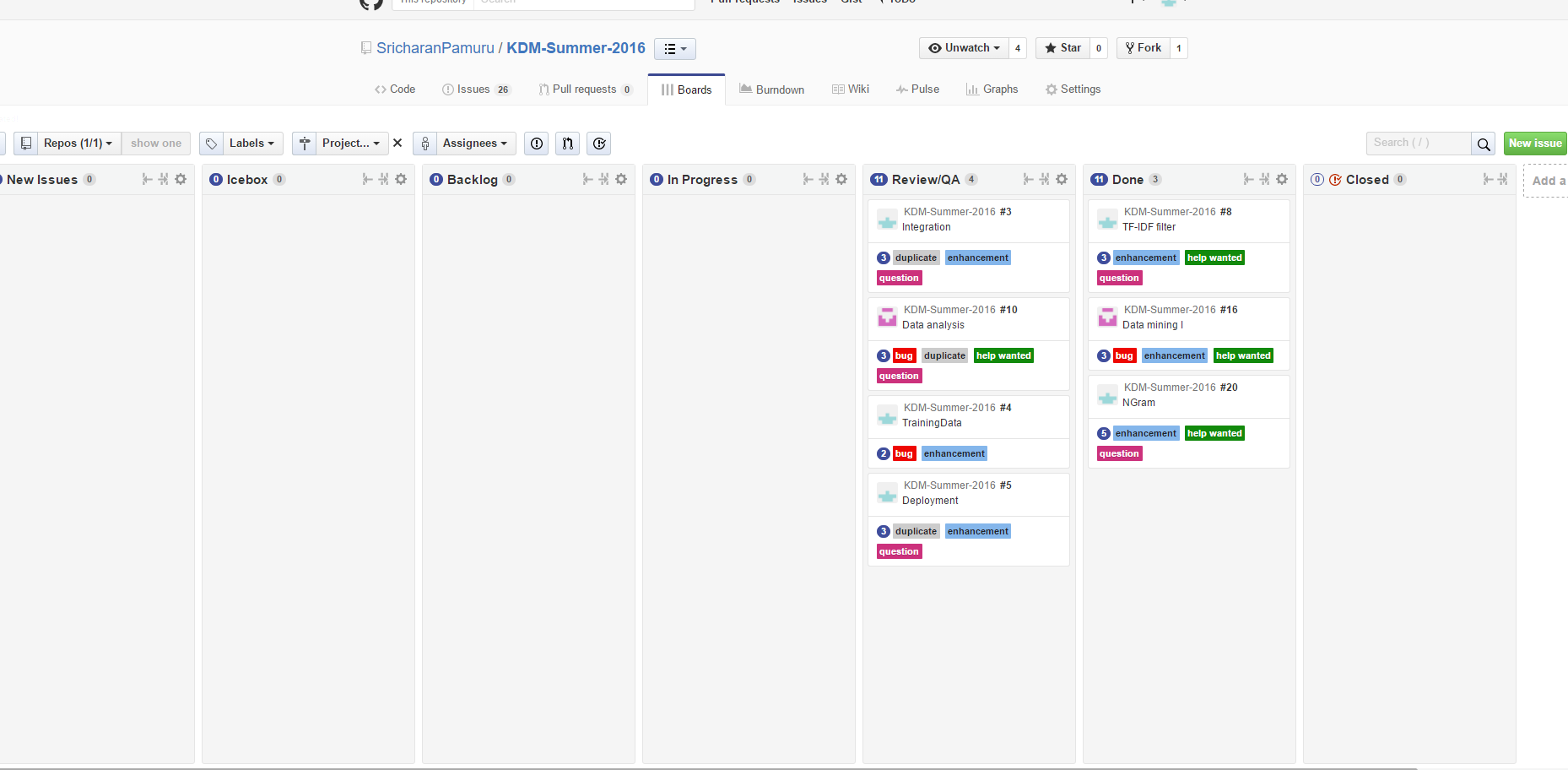




We have use Word2vec to get the synonyms for the top-idf words so that the synonyms can be matched against the topics obtained from LDA and suggest the user with articles which also contain the synonyms of the user interest like if the user likes soccer and is interested in reading articles on soccer then soccer is called football in other countries. This issue can be resolved using the synonyms.

### 2. ZenHub screenshots:

#### Issues of Project development:



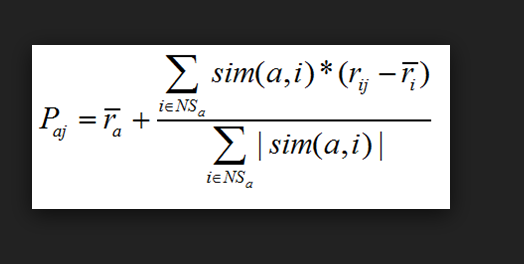
#### Contribution:

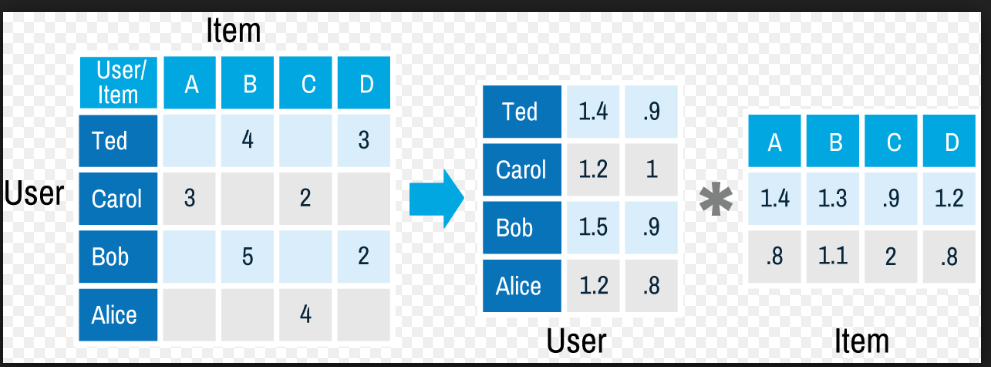


## Final Project Report:

### 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.

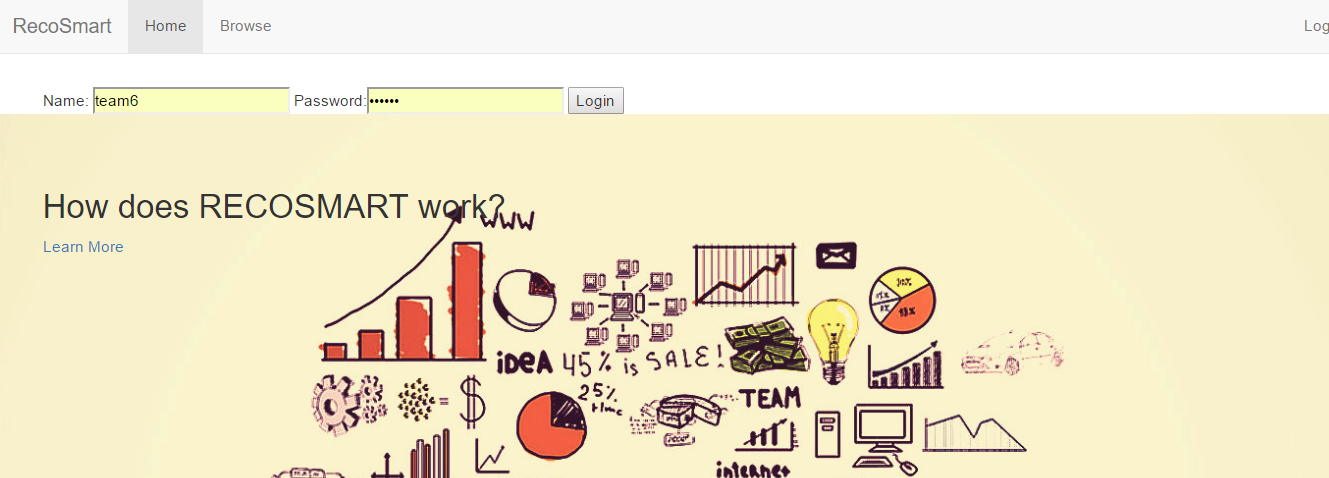


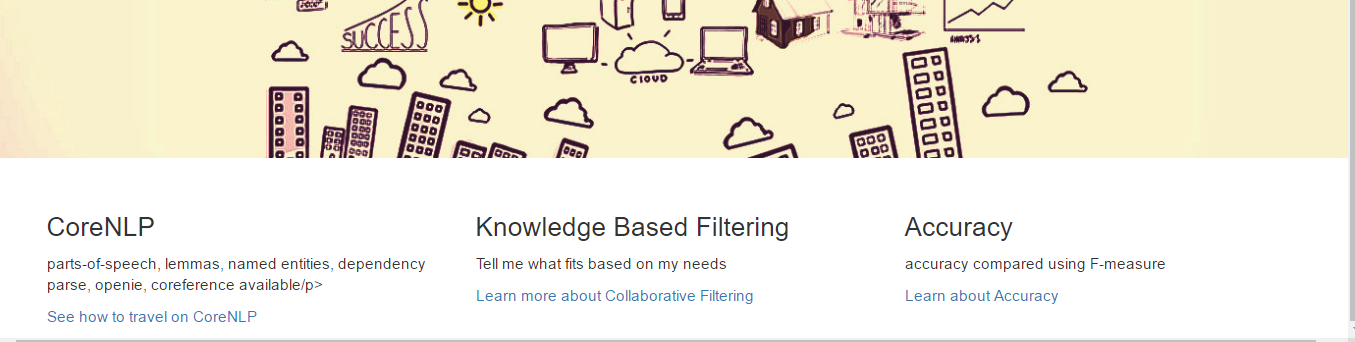


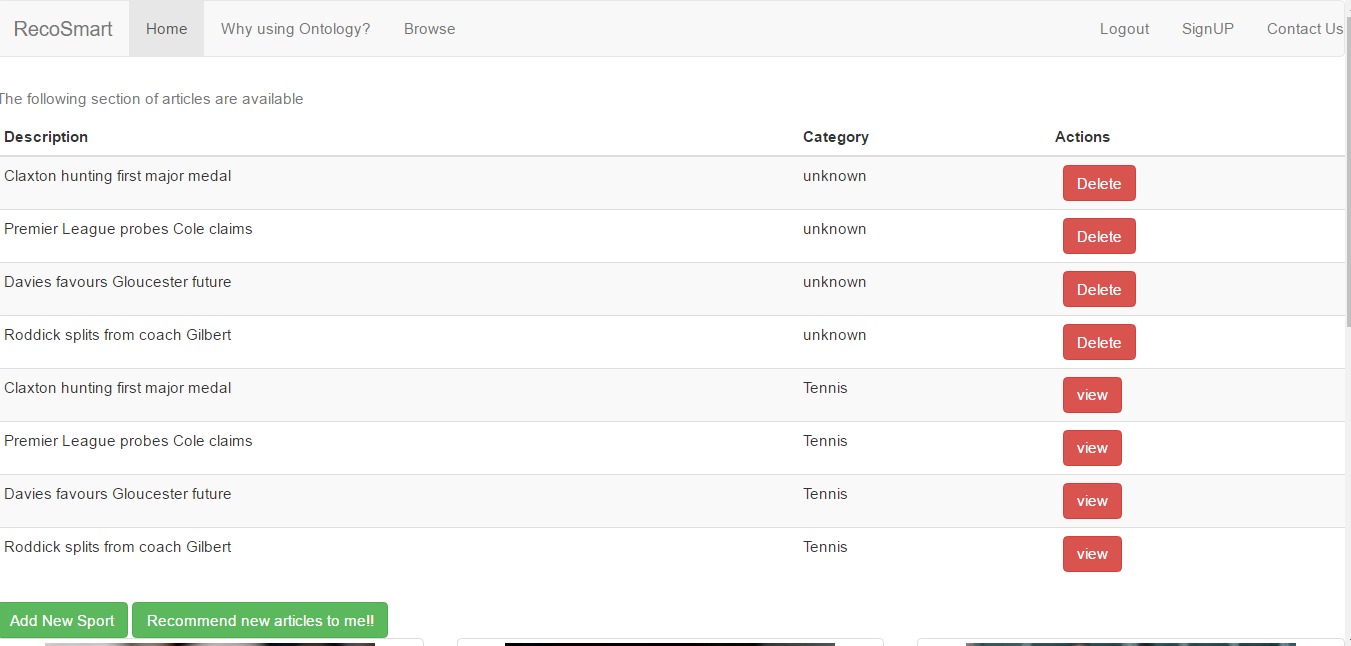
User is able to obtain a recommended article, based on the one he chooses. It is in turn computed based on the trained data set. A confusion matrix is generated and it would create a parameter that serves as the factor in determining which article has to be recommended.

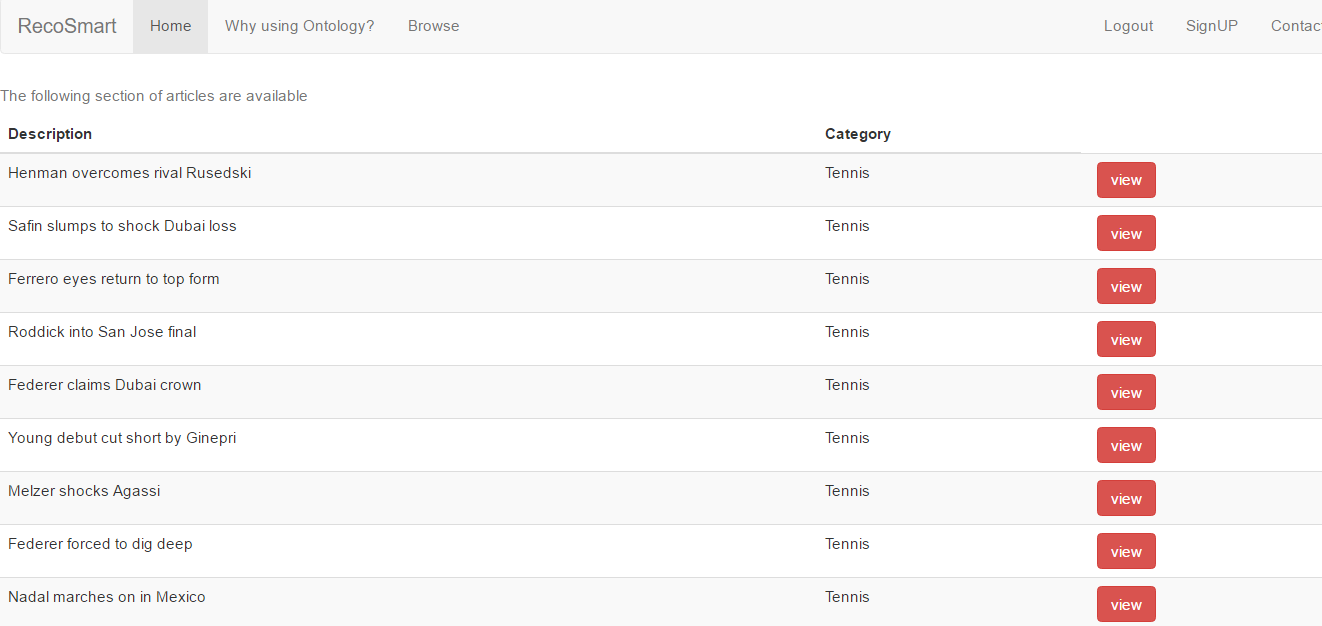
### Web Based UI :

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.



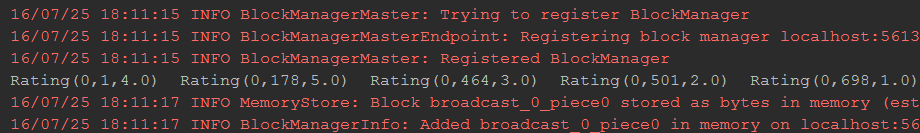




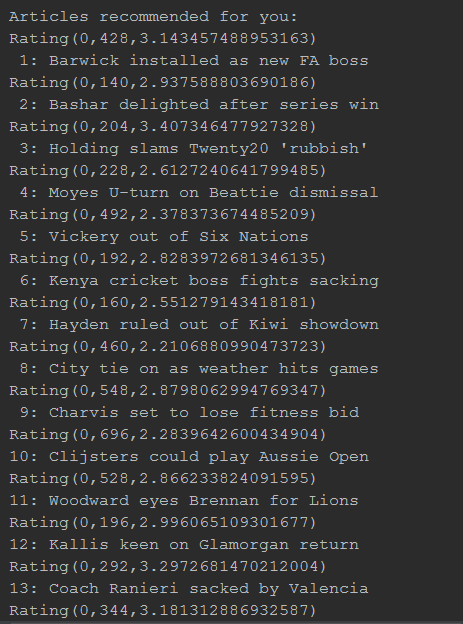


### Spark Processing:

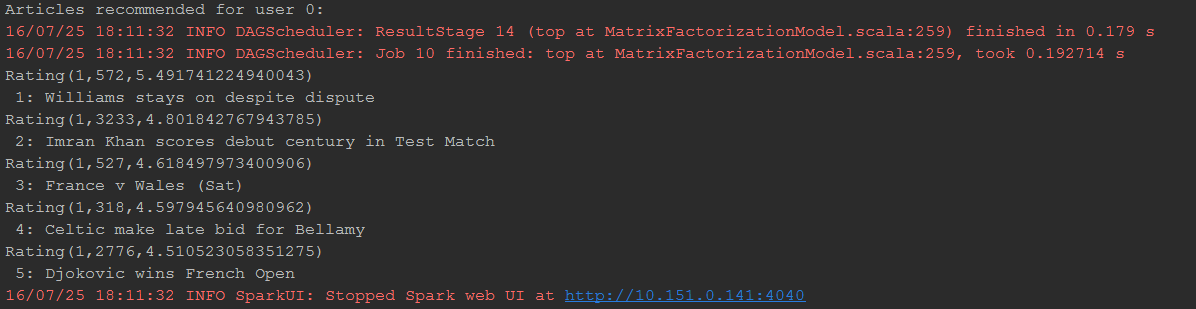
Once the socket connection is set to listen to the server, the specific ratings of the recommended movies are listed as follows which actually are filtered using the Alternating Least squares algorithm to ensure the implementation of the collaborative filtering model.



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.

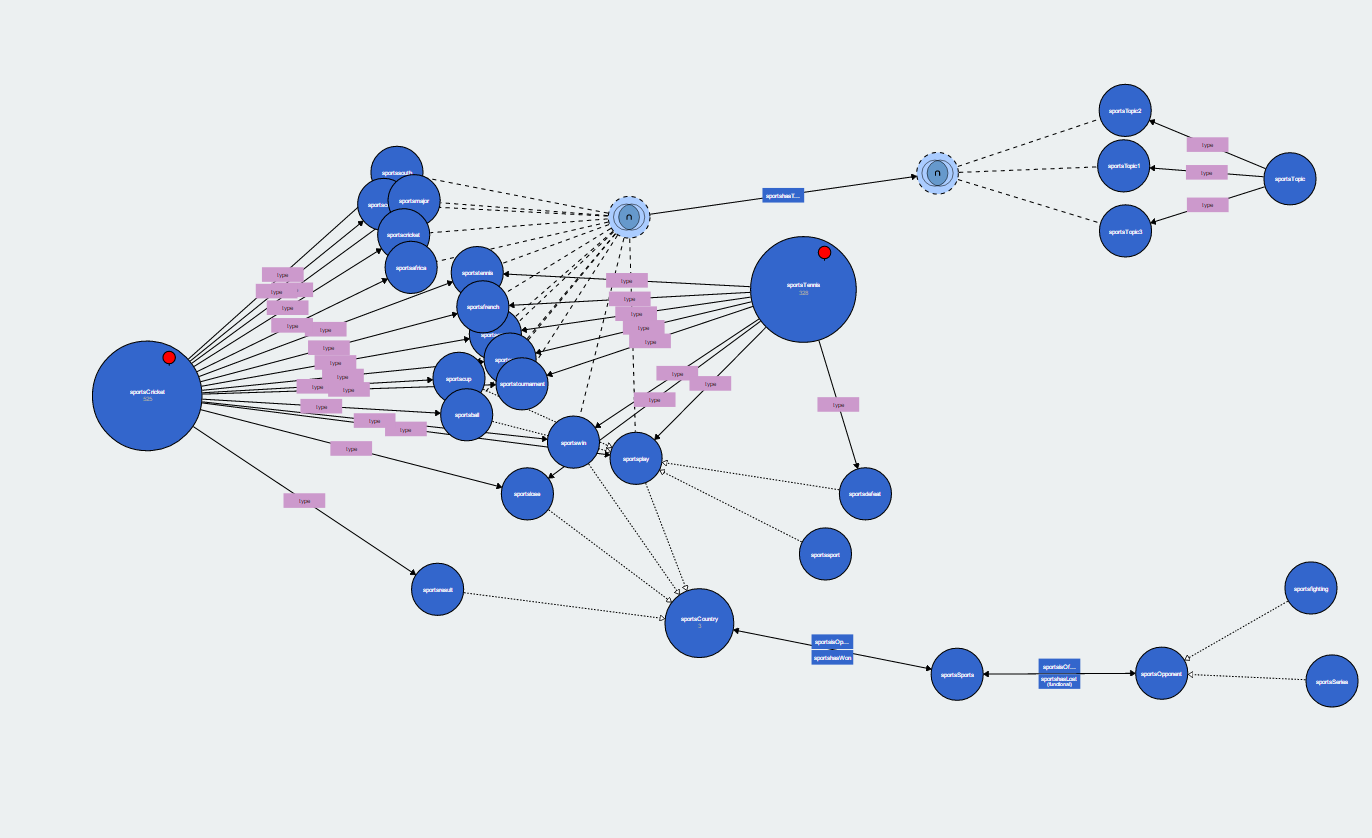


The final recommendation system has the server return the following articles and it forms the generation of output based on the machine learning techniques for training and testing the data model synthesized.



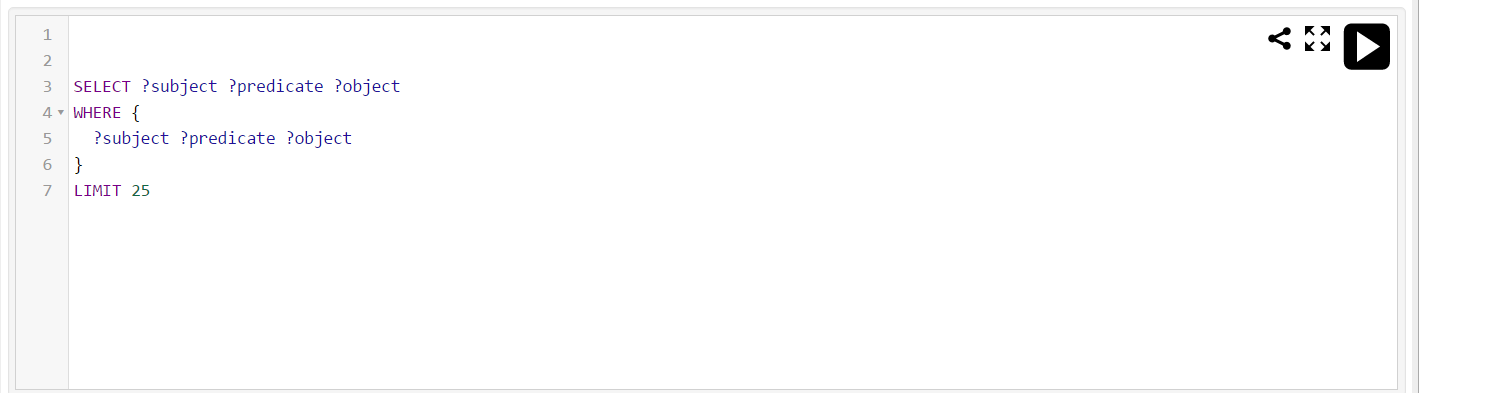
The ontology generated from the processing can be run in WebVowl and the following schema can be generated as a result of that and this shows how the different categories have been linked.

### WebVowl:

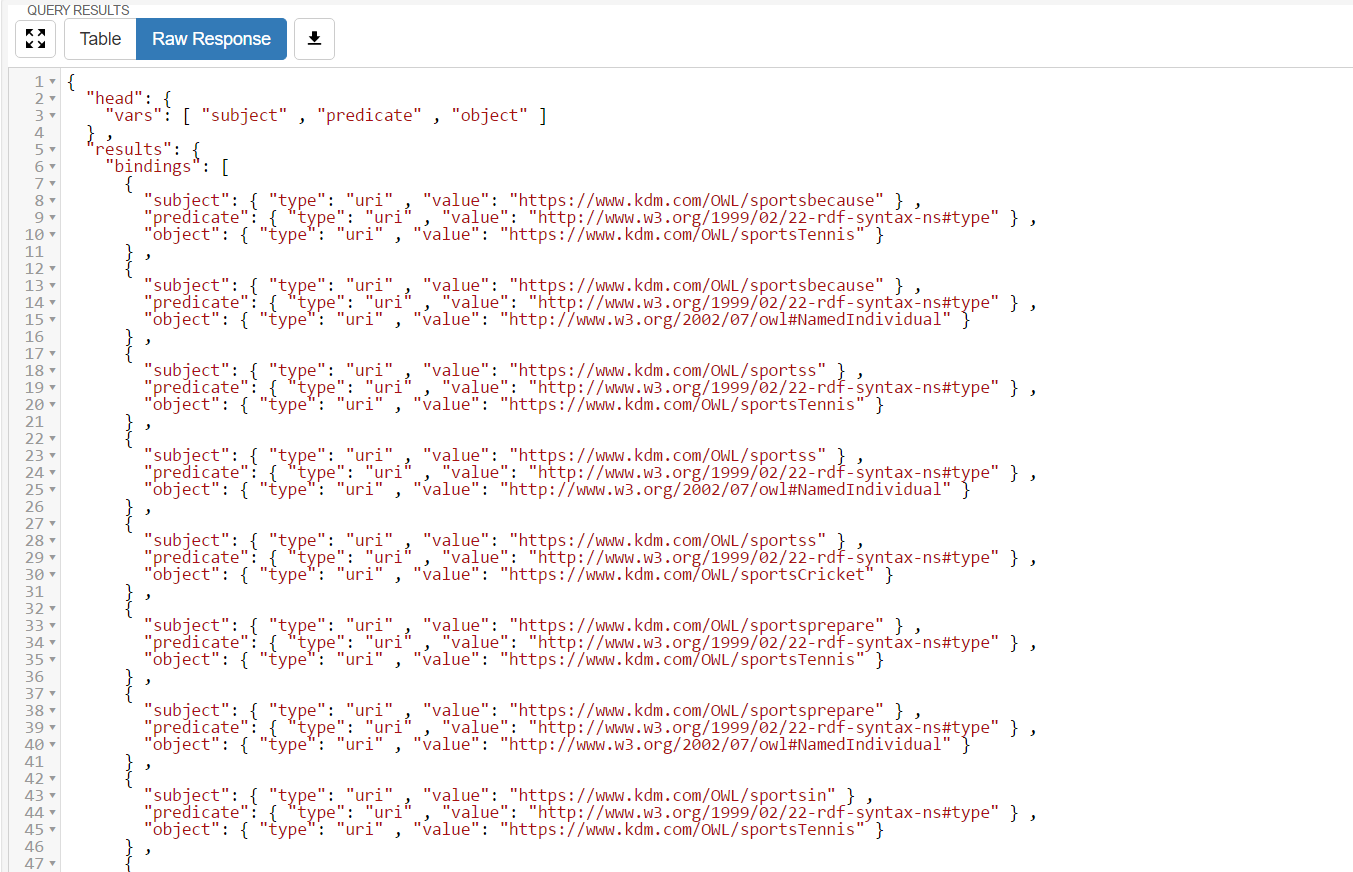


SparkQL queries can be run in either Protégé or Apache Jena Fuseki to display the data modules and the following can be generated:

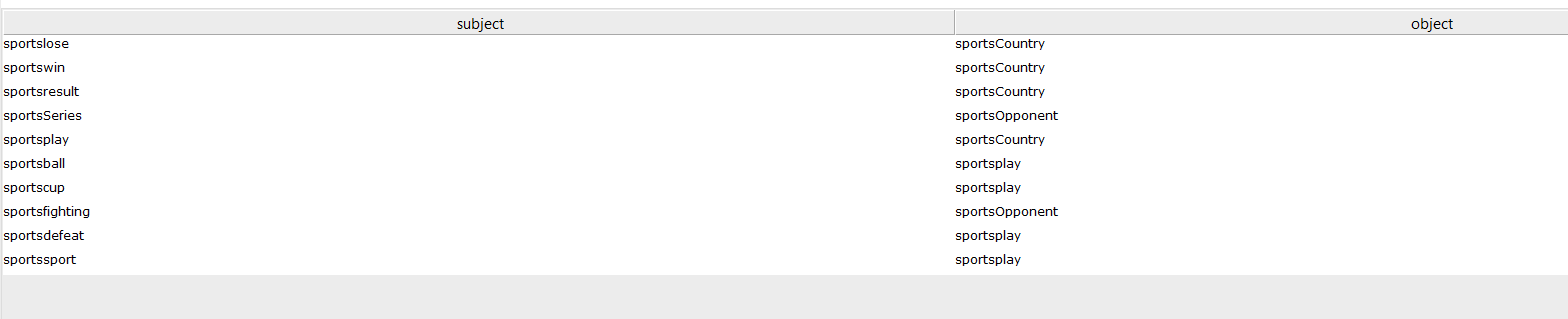
### SparkQL:



### Apache Jena Fuseki:

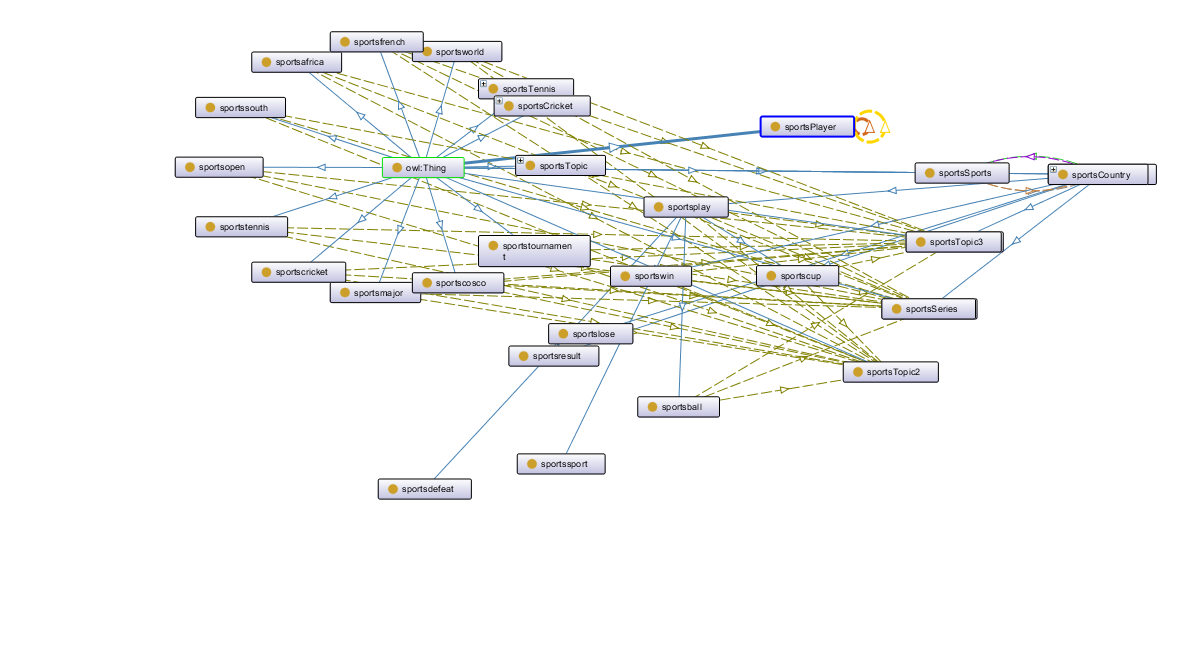


The query run in Protégé to display the table based format can be generated as such:



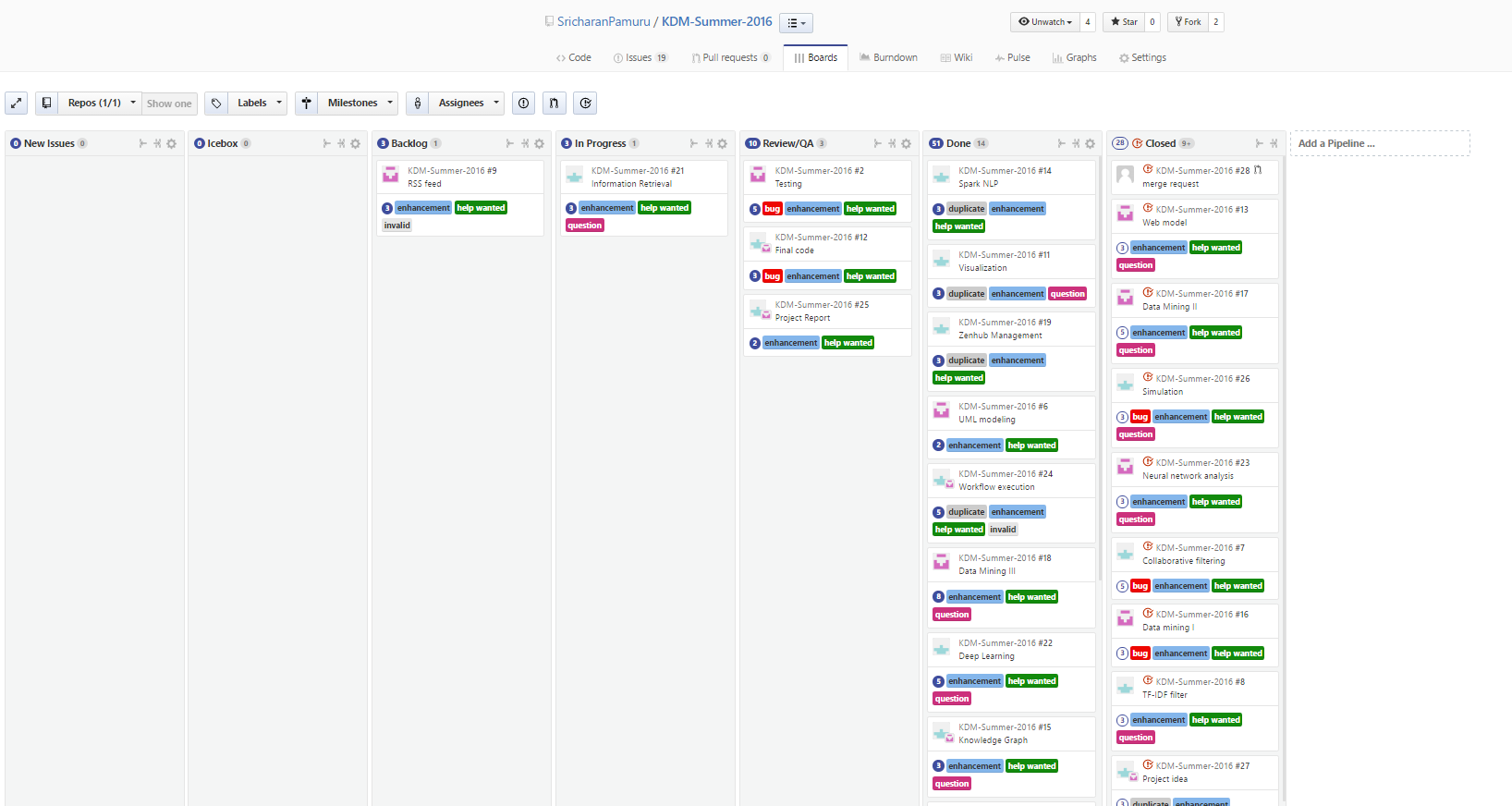
### OntoGraf in Protégé:

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.

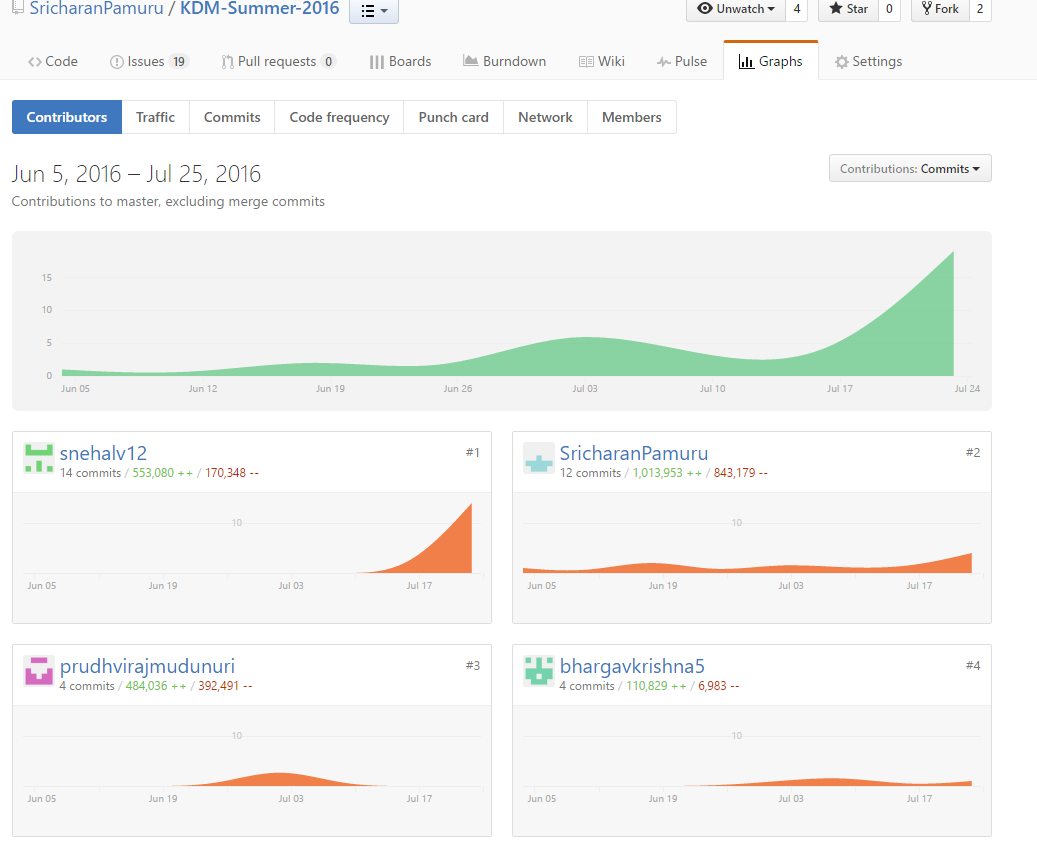


### ZenHub Screenshots :

#### Issues Resolved:



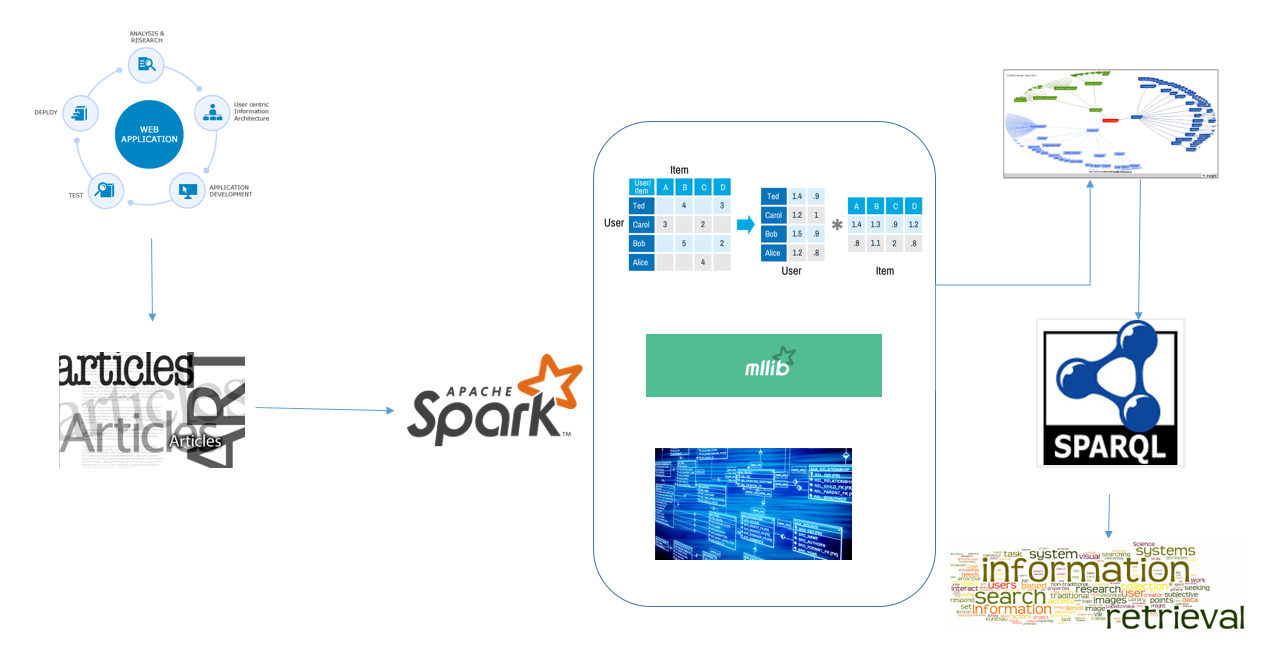
#### Code contribution:



# Design of features

## Software Architecture:

### Architecture of Model:

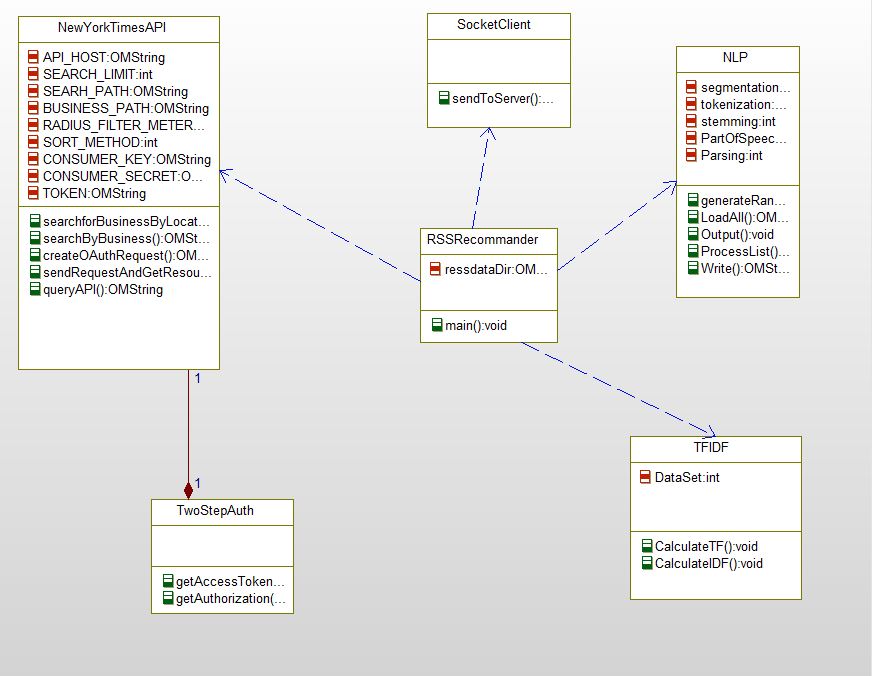


The web application will show the user a set of articles. A socket connection has been coded to run the Spark server and the program will run for streaming which would involve collaborative filtering and the MLlib will fetch the libraries and recommend the system. Ontologies are generated and from here 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.

### UML Modeling:

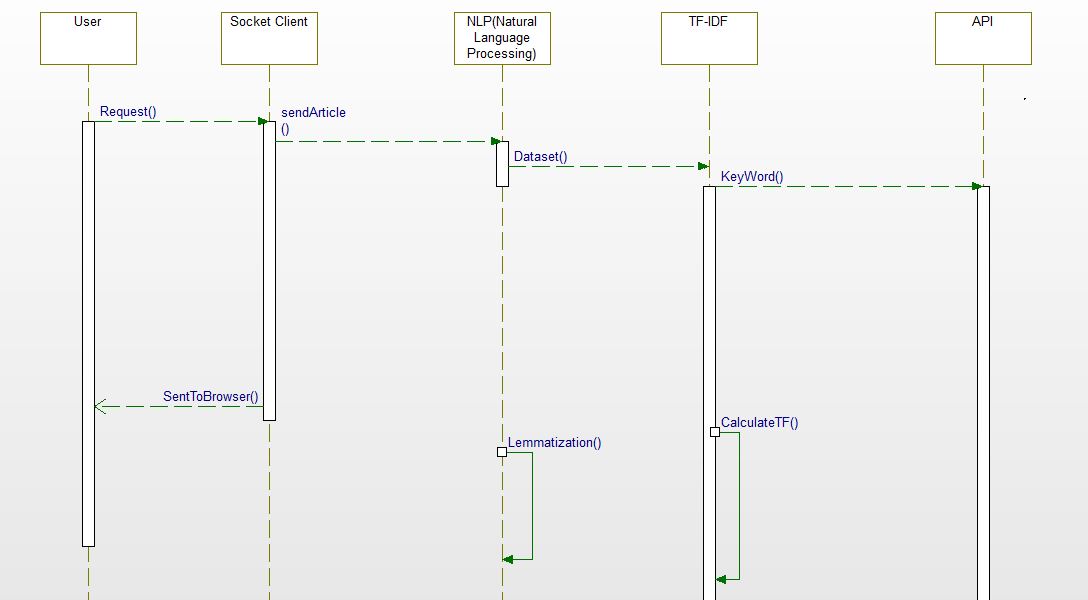
#### Class Diagram:



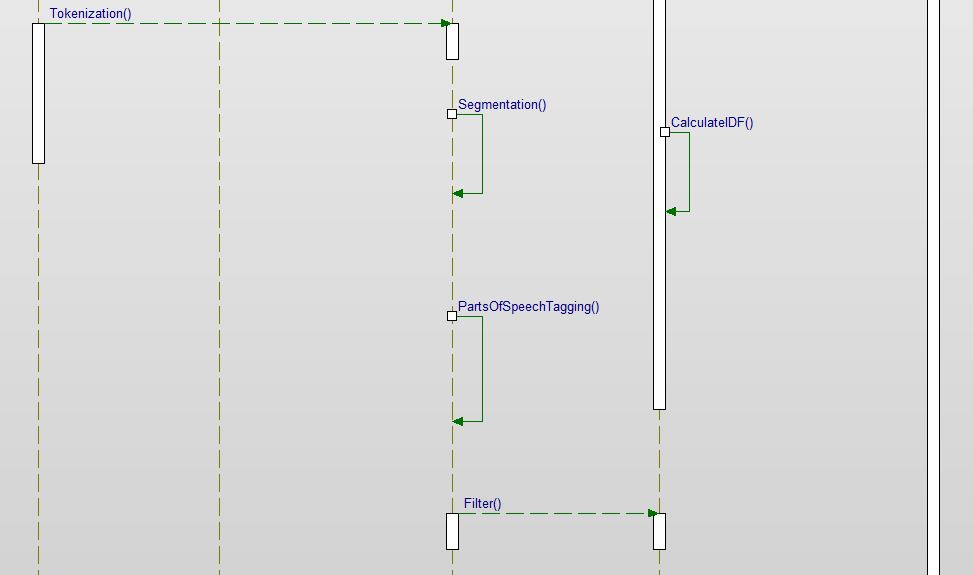
This diagram showcases the different classes that we have used in our program as a key component of the execution and the workflow can be depicted that will serve as base for the knowledge graph eventually processed.

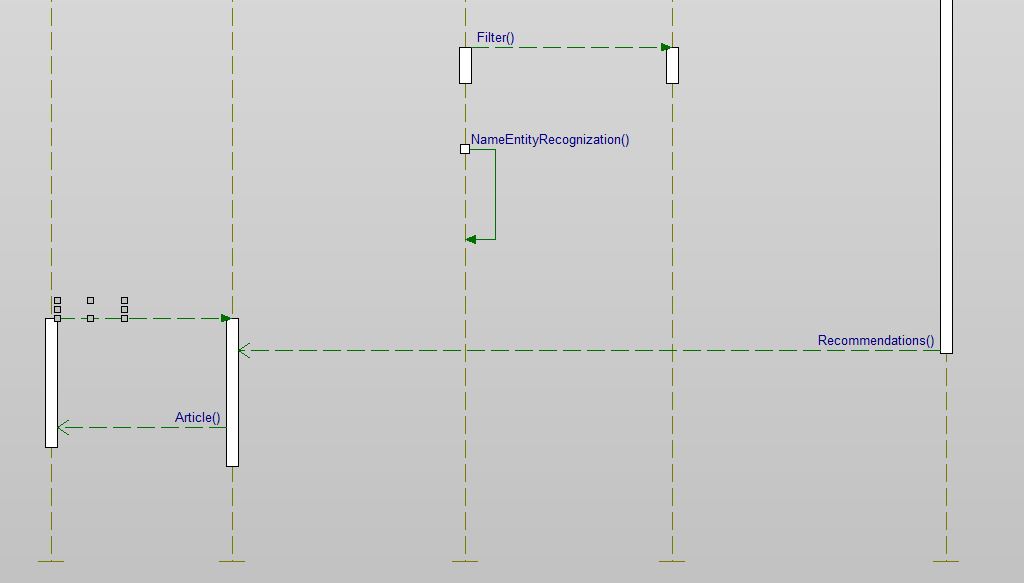
#### Sequence Diagram:

The sequence of execution for each of the classes is shown below. The user would use the present client side application to read articles. The client side program would then perform the various NLP operations on it to filter the text and observe a relative extraction mechanism.



The TF-IDF runs and the keywords are sent to the API. The NLP processes would include lemmatization, Named Entity recognition and stemming that forms the key part of our project. The fragment of code would then be run to check for the relative frequency of words present in the current article. After the program is executed, the recommendation of articles is returned based on process such as collaborative filtering and information retrieval occurs. The entire portion of the code would be executed dynamically using RSS feed.

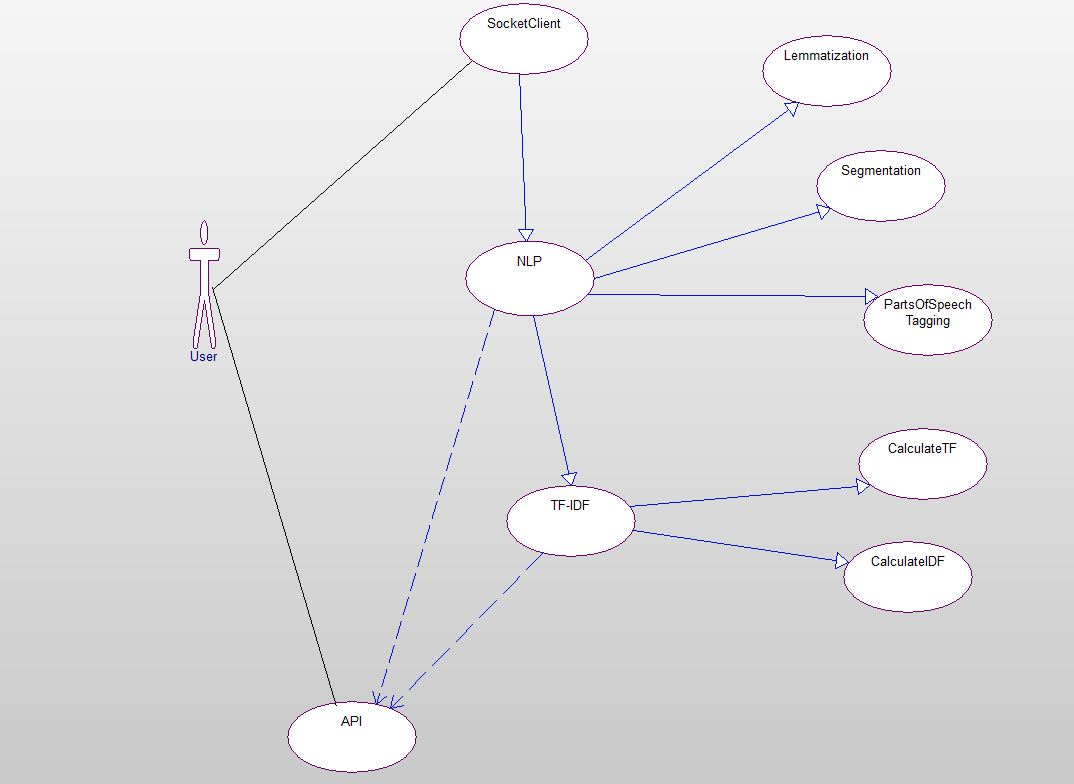




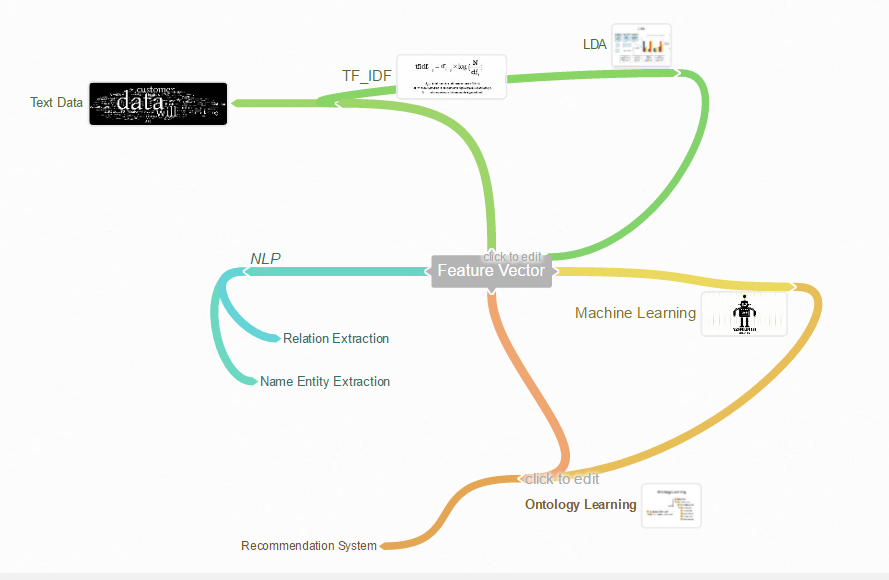
The entire sequence would then return the server side run program which would contain the list of articles which would navigate as per the machine learning algorithms.

#### Use Case Diagram:

The use case diagram would then show how the user is linked through the various NLP operations which will do the different API based handling where the client program will send a request to the server and focus on handling the various terms that can bring about a certain kind of linkage to draw the knowledge graph that will bind the different APIs. The final recommendation engine is built based on the machine learning algorithms that are executed.



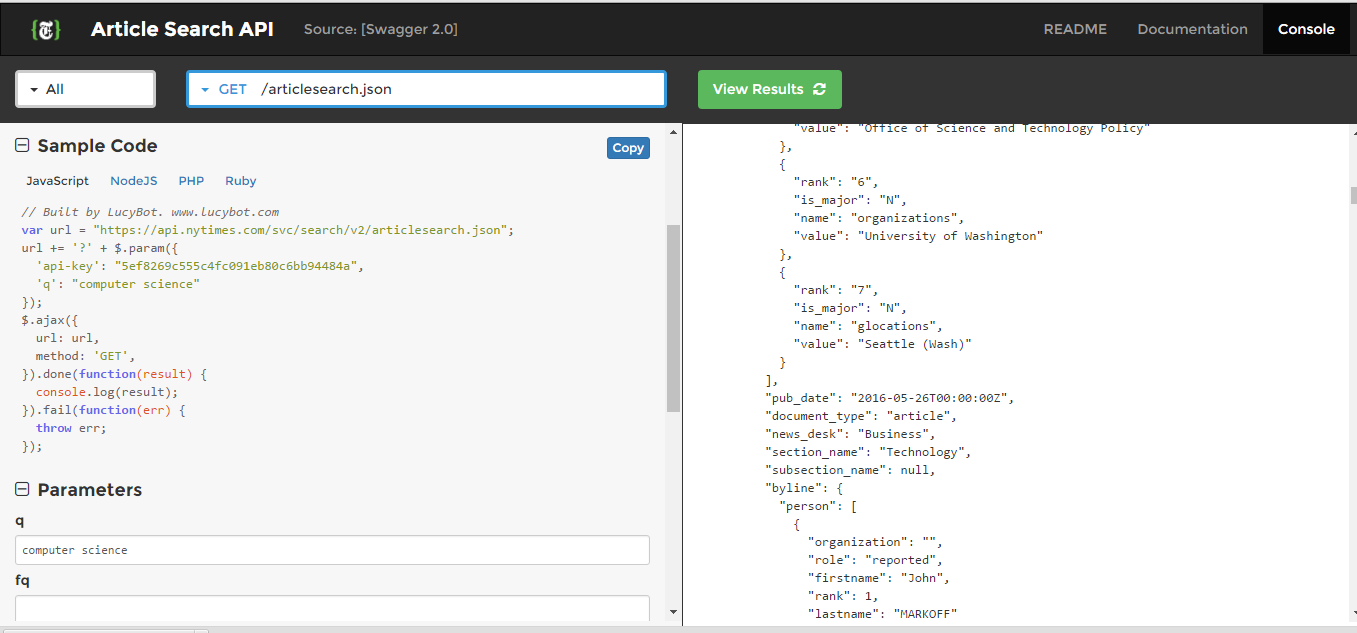
## Workflow:



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.

## 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.





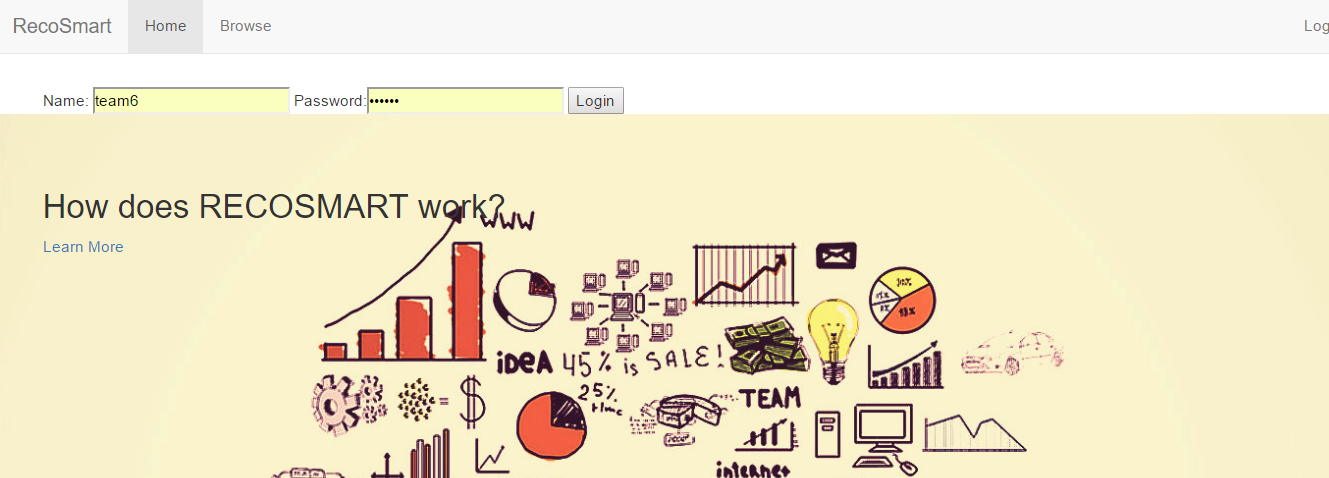
## New feature implemented:

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..

## Web Application:



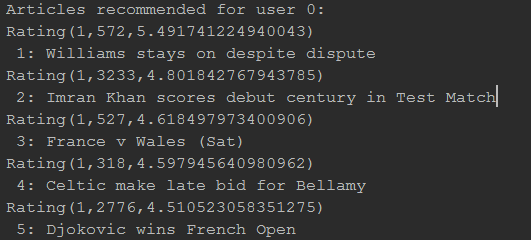
The web application serves as an interface to connect both the front and back end and displays the output based on the recommendation system that is executed.

The Web application is RECOSMART.

# Results and Evaluation:

1. Accuracy:

Precision & Recall:



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

1. Performance:

It is slow for processing and requires arguments to be passed in order to observe data patterns and perform information retrieval fast.

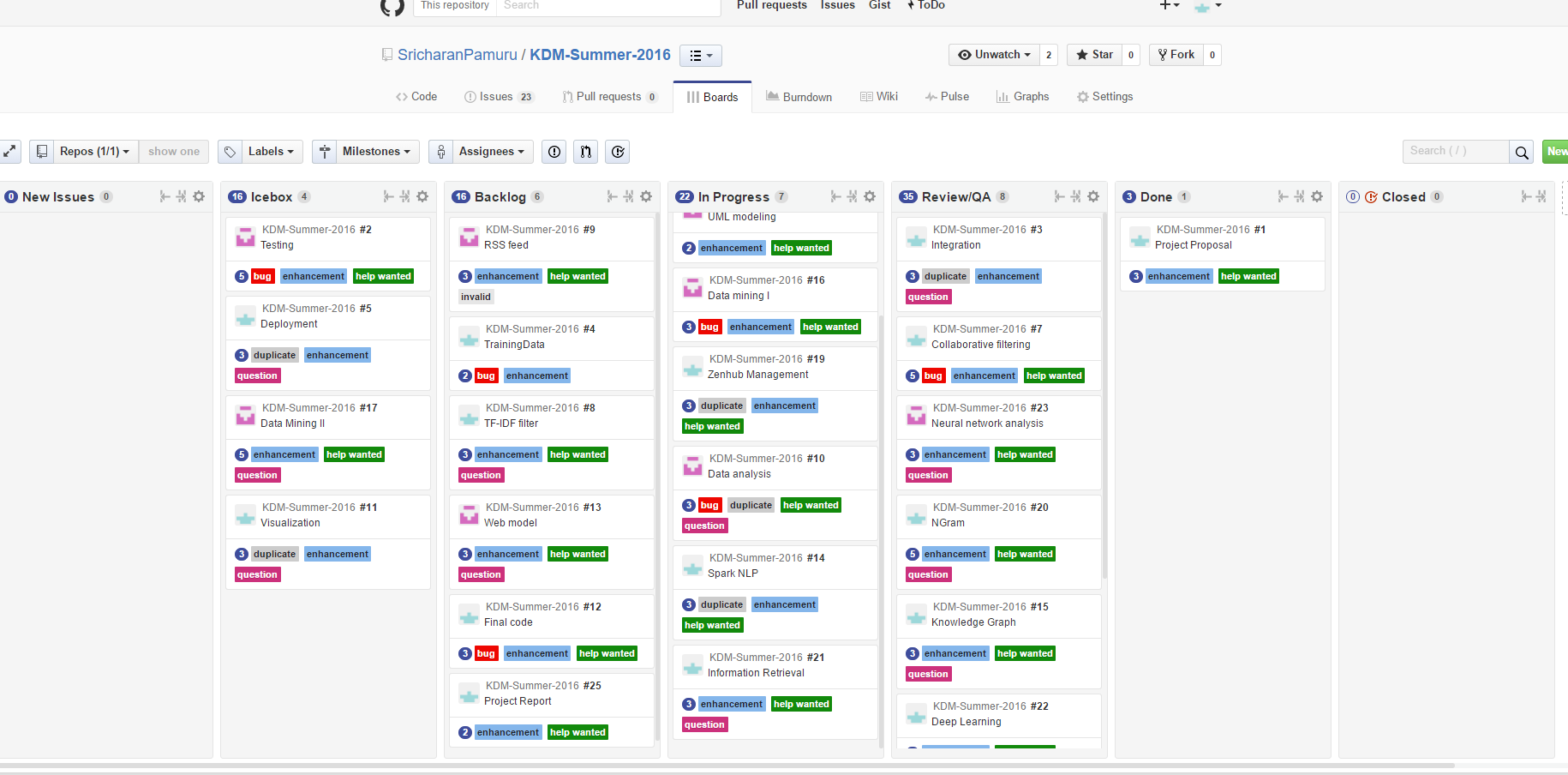
# Project Management:

## Individual Contribution:

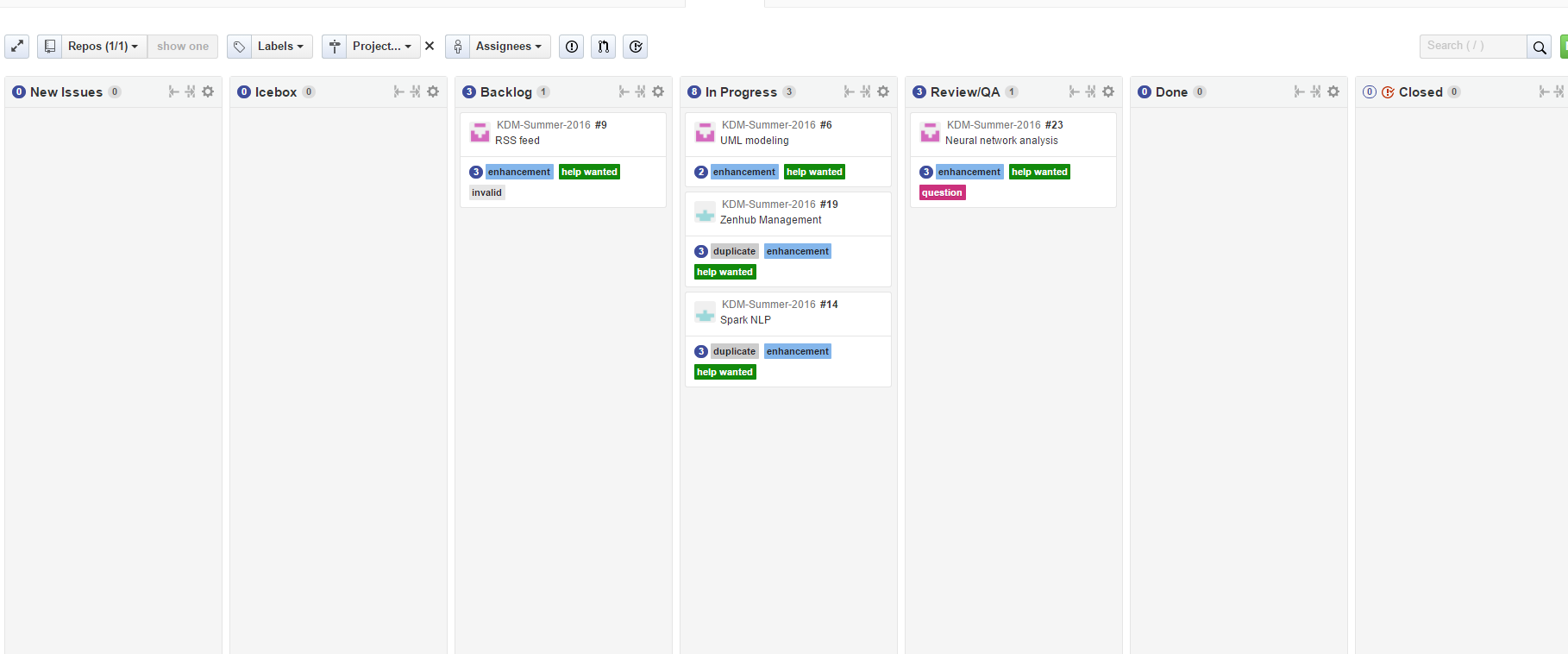
|  |  |  |
| --- | --- | --- |
| Name  (CLASS ID) | Contribution | Number of hours |
| SRICHARAN PAMURU  (20) | Collaborative filtering and building recommendation system | 60 |
| PRUDHVI RAJ MUDUNURI  (22) | Web application development and deployment of LDA | 60 |
| SNEHAL VANTASHALA  (41) | Ontology generation and visualization | 60 |
| BHARGAV KRISHNA VELAGAPUDI  (42) | Data management and extraction of features. Training and testing system | 60 |

## Zenhub/Github Screenshots:

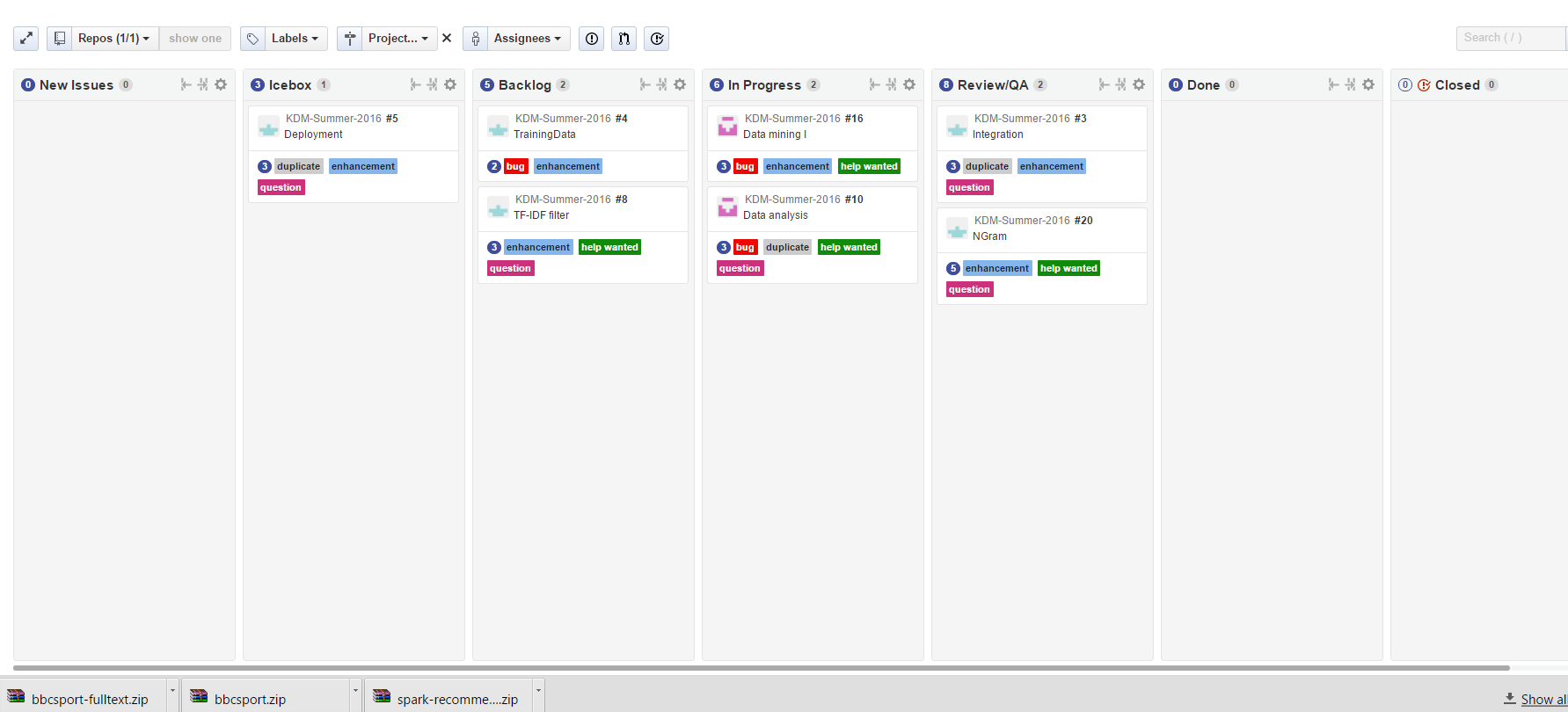
### Full timeline:



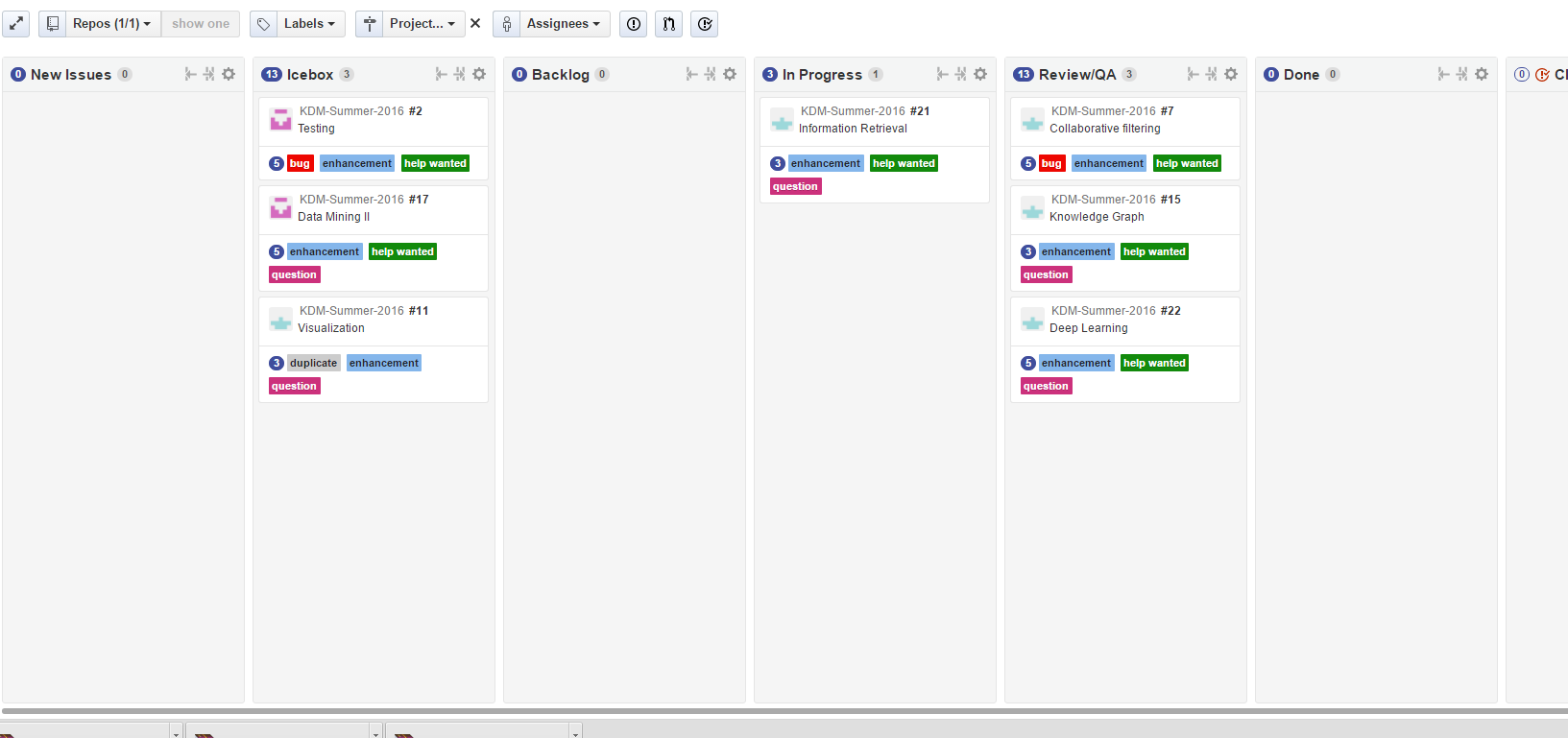
### Report 1:



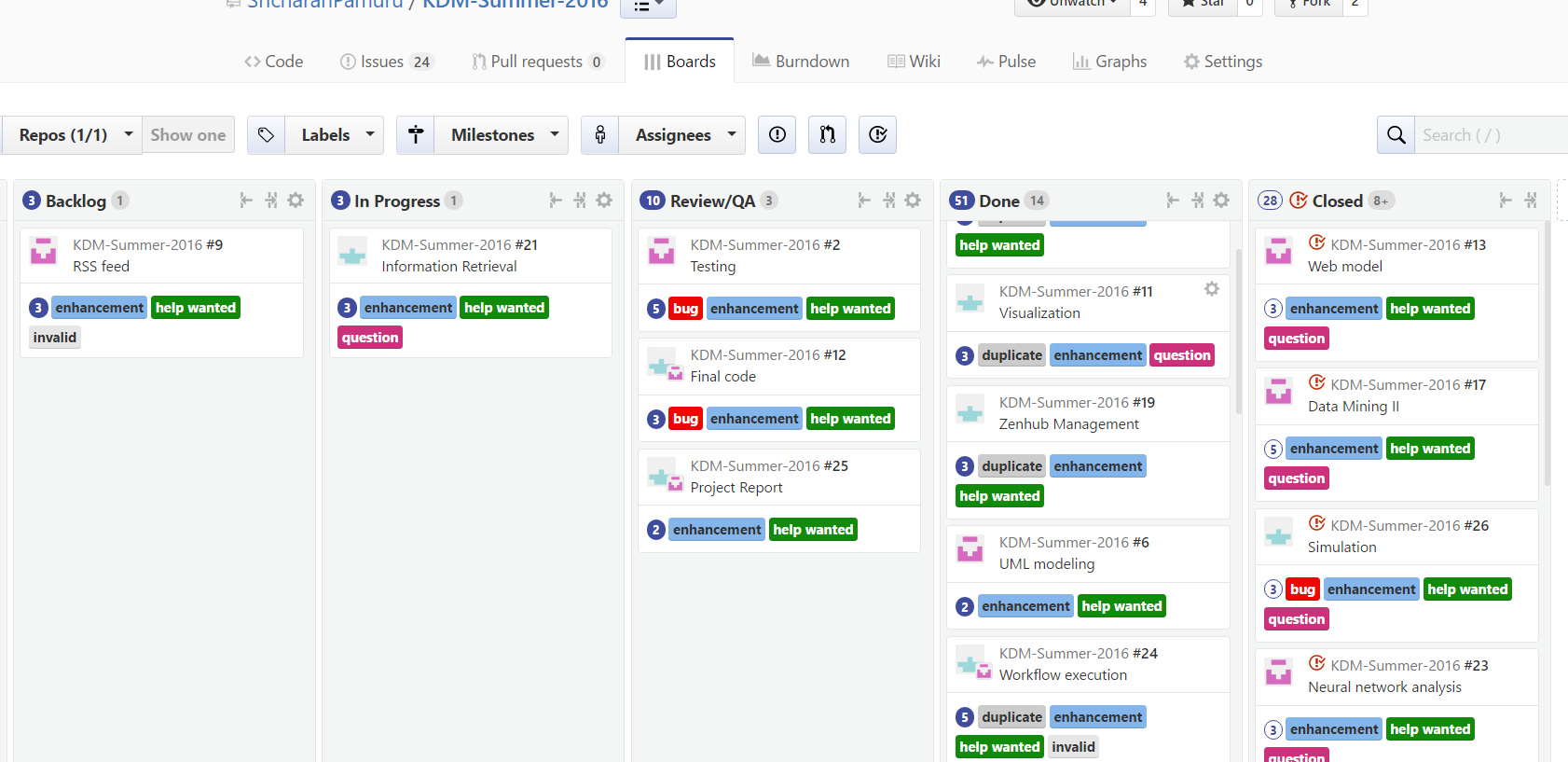
### Report 2:



### Report 3:



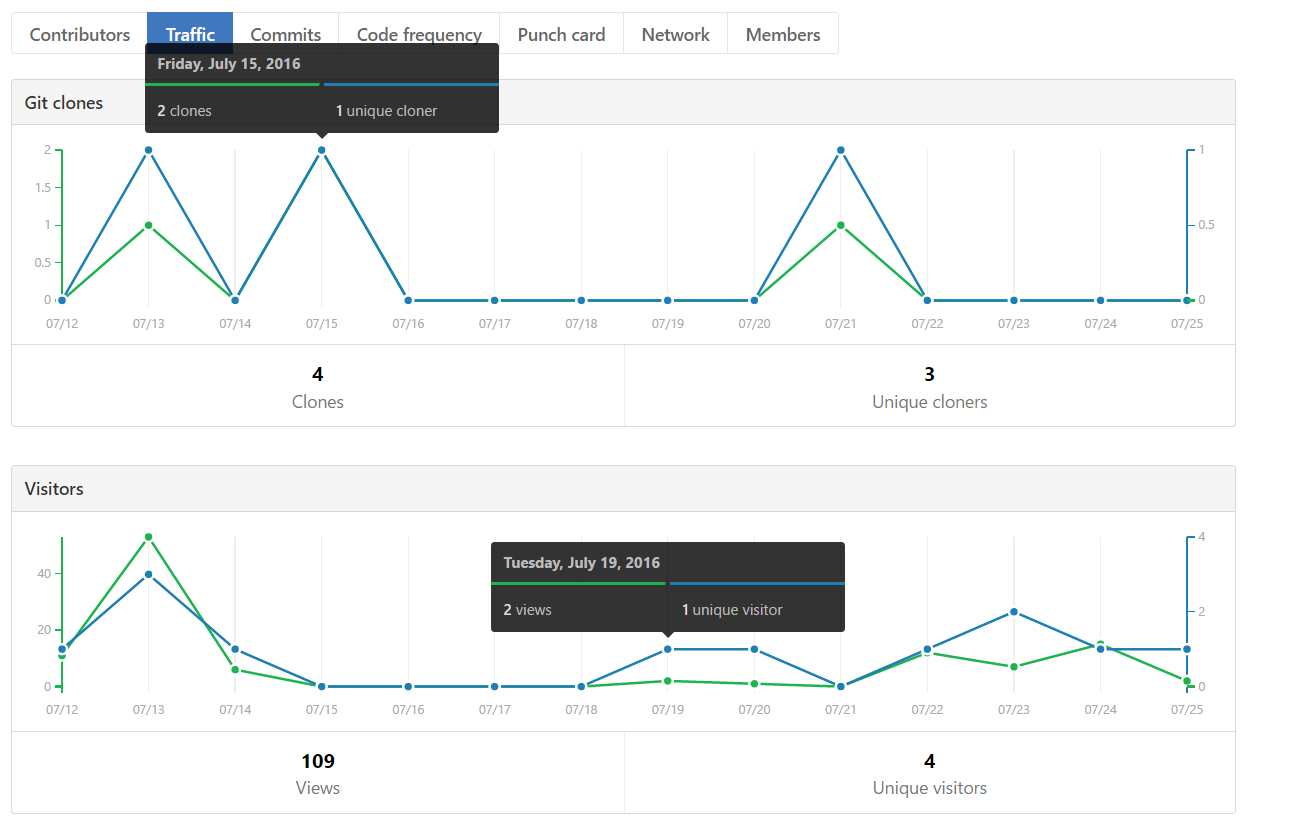
### Final Report:



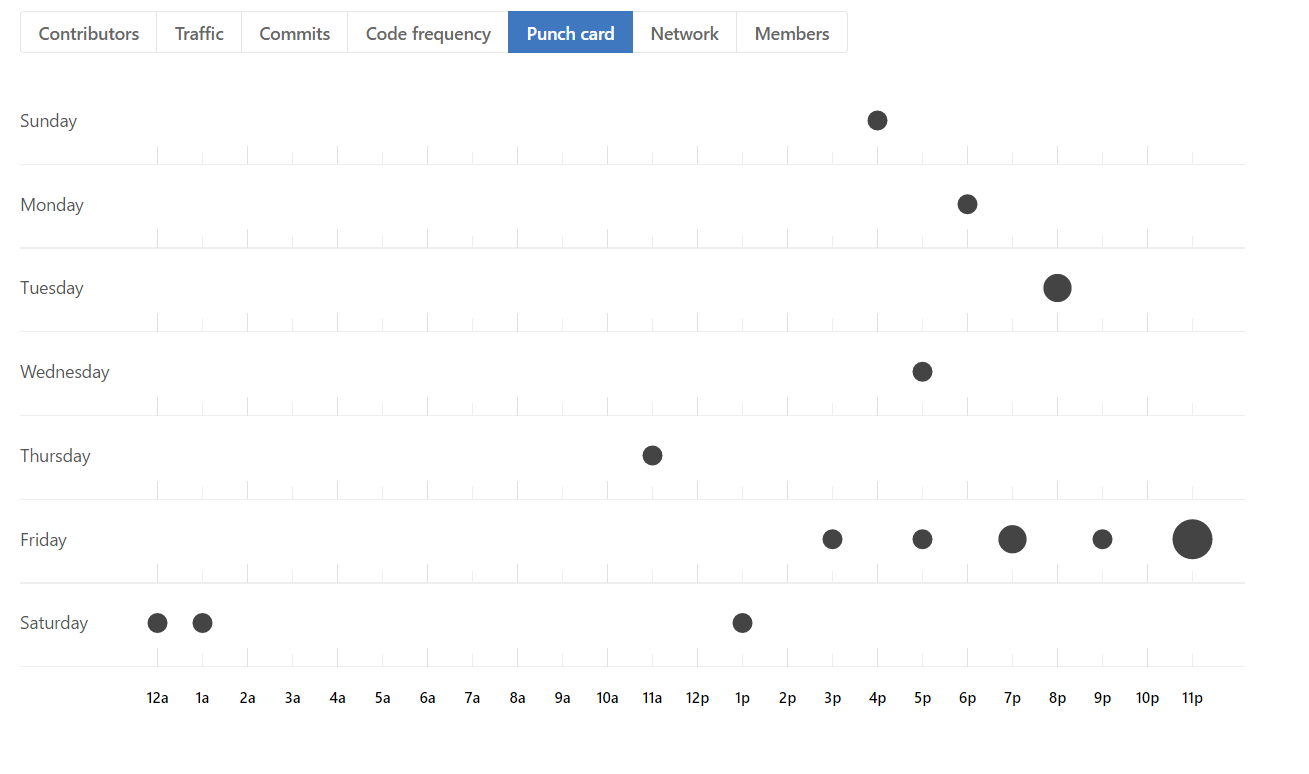
### Burndown chart:



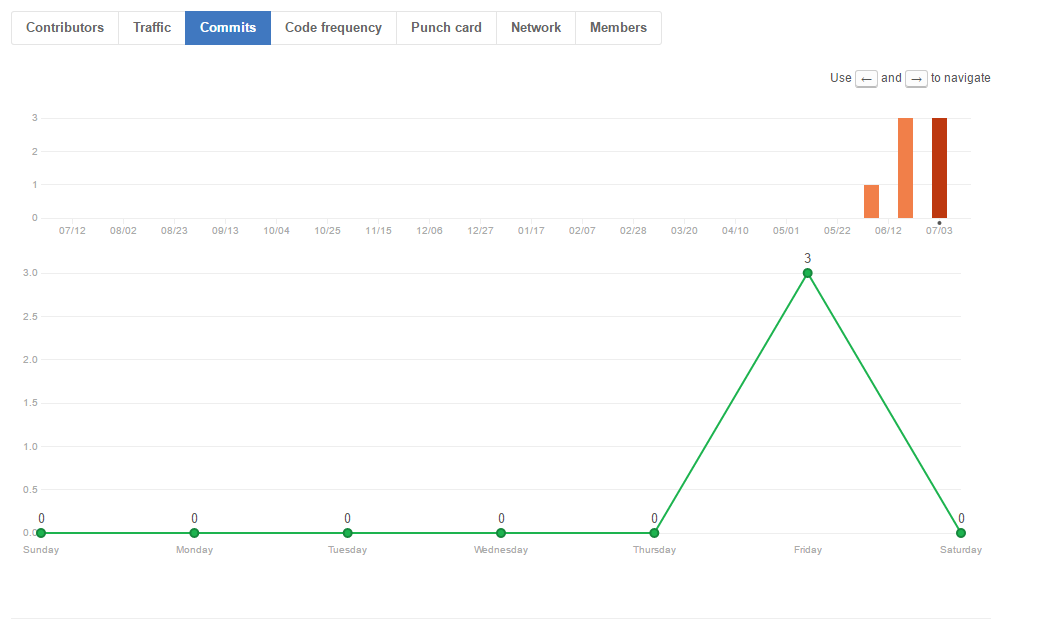
### Traffic:



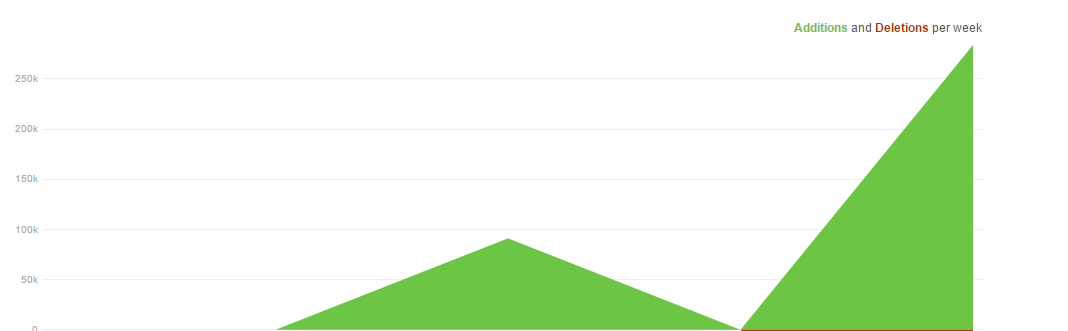
### Punch Card:



### Commits:



### Code Frequency:



## Concerns/Issue:

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.

## 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.

# Bibliography:

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.