KNOWLEDGE DISCOVERY AND MANAGEMENT

PROJECT

SUMMER 2016

DONE BY:

1.SRICHARAN PAMURU(28)
2.PRUDHVI RAJ MUDUNURI (22)
3.SNEHAL VANTASHALA (41)
4. BHARGAV KRISHNA VELAGAPUDI (42)

Contents

1. Project Proposal:

a. 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

b. 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.

c. 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.

2. 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.

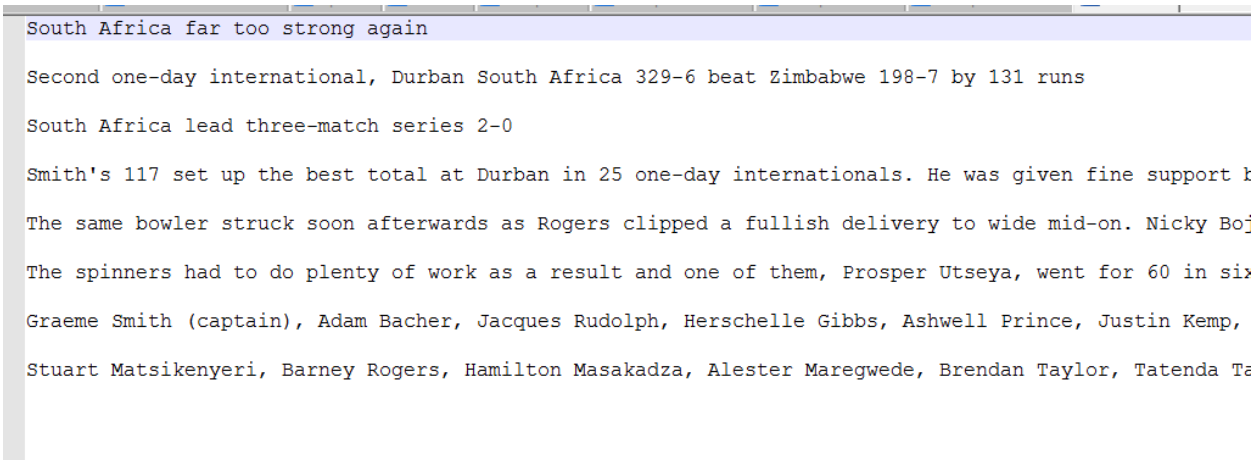
Recommendation engine is a platform for the machine to be trained and tested. Simulation of such technology actually occurs by rapidly searching the inner context. Big data mining tools focus on scalability and larger data sets are compared in order to retrieve information which is adaptive to the respective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

3. Data Collection:

The data collection for this specific increment 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.

4. Report:

a. 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.

```
16/06/24 22:07:47 INFO TaskSetManager: Finished task 18.0 in stage 3.0 (TID 276) in 25 ms on localhost (20/124)
16/06/24 22:07:47 INFO TaskSetManager: Finished task 24.0 in stage 3.0 (TID 282) in 8 ms on localhost (21/124)
16/06/24 22:07:47 INFO Executor: Finished task 26.0 in stage 3.0 (TID 284). 2044 bytes result sent to driver
16/06/24 22:07:47 INFO TaskSetManager: Finished task 23.0 in stage 3.0 (TID 281) in 14 ms on localhost (22/124)
(1048576,[0],[0.7759091285222348])
(1048576,[115152,506034,693691,858011,948830],[5.503296947234575,2.4047072882408767,5.68561850402853,6.601909235902685,5.215614874782794])
(1048576,[0],[0.7759091285222348])
(1048576,[97,101577,144473,515733,606417,721821],[1.3896945684080602,1.5236152933326146,4.730107059001093,4.9924713234685845,5.503296947234575
(1048576,[3365,49512,78208,87452,101577,114801,130679,147247,226650,321261,467670,506034,634938,693691,775993,829792,1030291],[2.985876082171
(1048576,[0],[0.7759091285222348])
(1048576,[0],[0.7759091285222348])
(1048576,[65918,103314,109935,144473,517614,606417],[5.215614874782794,1.9867887190614255,2.5765575451675358,4.730107059001093,6.601909235902
(1048576,[0],[0.7759091285222348])
(1048576,[97,3122,3325,3707,57152,65918,65975,78197,107022,109935,181884,262303,263939,285054,537182,714794,992418,1045946],[1.389694568408066
(1048576,[0],[0.7759091285222348])
(1048576,[97,3122,3139,3365,3543,3551,3707,6058,84049,87452,90731,93405,94467,96727,97921,101577,103066,103308,109251,109884,110182,114801,11
(1048576,[0],[0.7759091285222348])
(1048576,[97,3117,3139,3211,3268,3325,3357,3365,3366,3370,3543,3551,3707,3790,23517,32957,35007,35472,49512,69929,72562,78208,86875,96727,968
(1048576,[0],[0.7759091285222348])
(1048576,[14,55,97,1570,3122,3123,3325,3365,3370,3543,3551,3555,3707,3739,3790,17642,33287,35472,37347,41179,48758,49512,67169,76268,87452,90
(1048576,[0],[0.7759091285222348])
(1048576,[33287,41179,56733,131471,154192,155459,191745,196822,234850,267200,275635,299437,312154,486197,503845,532945,605528,730560,872988,8
(1048576,[72801,101577,315348,962045],[6.19644412779452,1.5236152933326146,6.601909235902685,2.817719601984424])
(1048576,[0],[0.7759091285222348])
(1048576,[3365,3543,3707,19442,49512,114801,226650,270252,285411,336781,607602,705278,751872,875937,962045,977399,1014432],[1.4929380410855675
(1048576,[0],[0.7759091285222348])
(1048576,[97,336,2801,3117,3123,3139,3304,3325,3365,3370,3543,3551,3707,3790,6058,15313,19442,33430,35039,35472,49512,54390,57749,67124,67169
(1048576,[0],[0.7759091285222348])
(1048576,[97,1127,3139,3211,3304,3365,3370,3543,3551,3707,19070,25048,35472,49730,53232,58596,75477,91744,96727,96852,97921,98256,101577,1033
(1048576,[0],[0.7759091285222348])
(1048576,[73,97,3122,3159,3325,3365,3370,3543,3551,3707,3790,15313,16667,23751,32957,34372,48367,49512,62569,62687,66871,70655,84049,96673,96
(1048576,[3159,103066,103314,144473,273415,280963,800960,9345061,[2.1770626040458754,2.3678027313054257,1.9867887190614255,4.730107059001093,
```

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.

```
((Aus),,24.78577651117808)
((Ind),,24.78577651117808)
(popular,23.63504822137096)
(Steyn,21.496620714543194)
(ruling,19.805727707708055)
(Al,19.805727707708055)
(tax,19.805727707708055)
(Division,19.805727707708055)
(overseas,18.589332383383 56)
((SL),,18.589332383383 56)
(Van,18.589332383383 56)
(breaks,18.589332383383 56)
(Sri,17.78693399089631)
(evidence,17.726286166028217)
(US,17.726286166028217)
(Cronje,17.726286166028217)
(fours,17.197296571634556)
(Rae,17.05685551208559)
(most,16.816055852417257)
(there's,16.509890841703726)
```

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

b. NER:

```
Gilchrist-8
NER
Gilchrist-8:PERSON

,-9

S-10

Katich-11
NER
Katich-11:PERSON

,-12

D-13

Martyn-14
NER
Martyn-14:PERSON

,-15

M-16

Clarke-17
NER
Clarke-17:PERSON

,-18

M-19

Hussey-20
NER
Hussey-20:LOCATION
```

```
30-year-old-2
NER
30-year-old-2:DURATION

will-3

move-4

to-5

England-6
NER
England-6:LOCATION

under-7

the-8

Kolpak-9
NER
Kolpak-9:PERSON

ruling-10
```

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.

5. Design of features

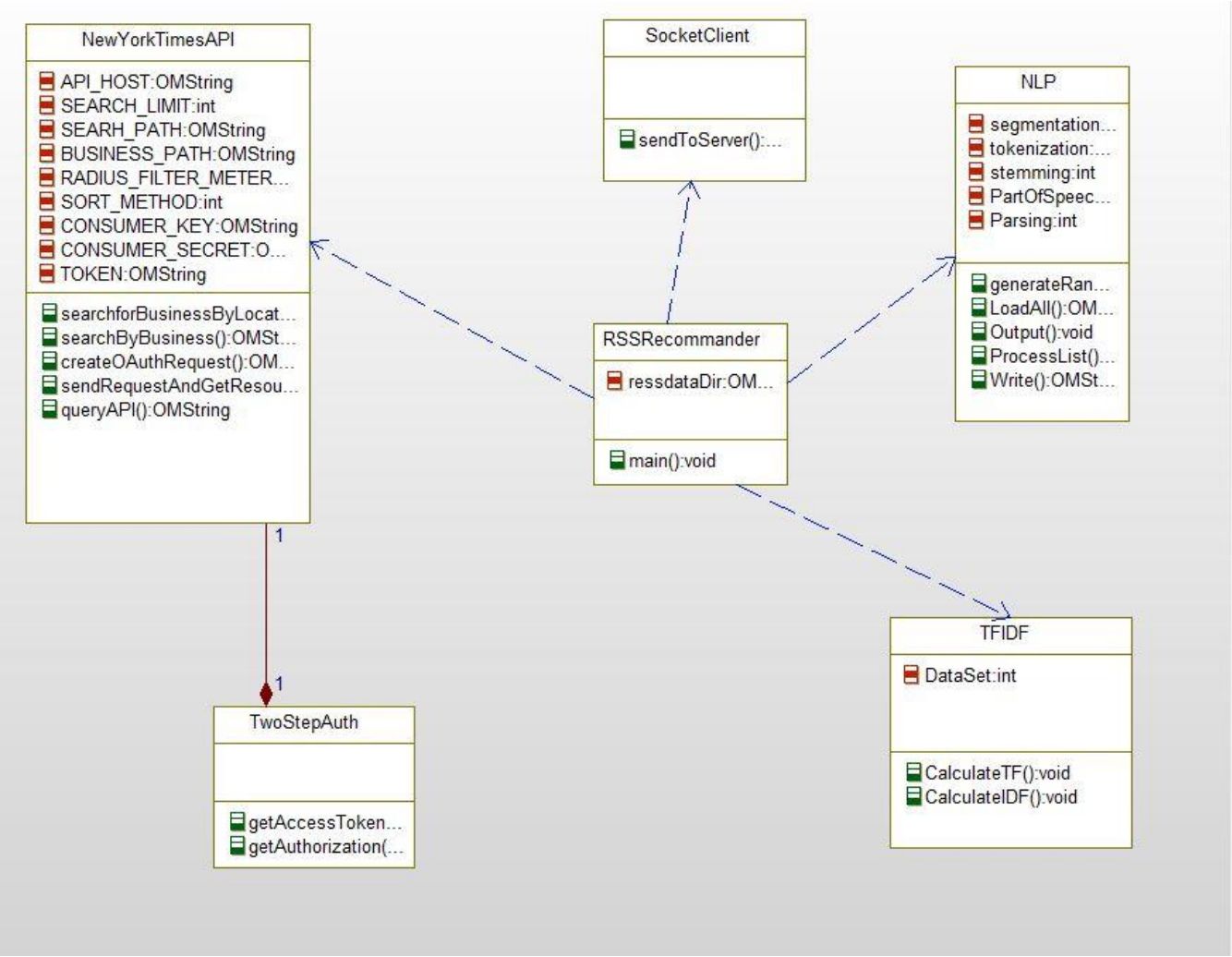
a. Software Architecture:

i. Architecture of Model:

The Articles are brought in online and the different RSS feed is present. The web would then combine with Spark where the collaborative filtering and NLP techniques are processed. The MLLib would serve as the library where the different algorithms are run. Machine learning techniques are being processed to indicate how the NLP parser are used. The articles have their key terms obtained and a recommendation is sent based on the user's search. This recommendation system serves as the main output for the project.

ii. UML Modeling:

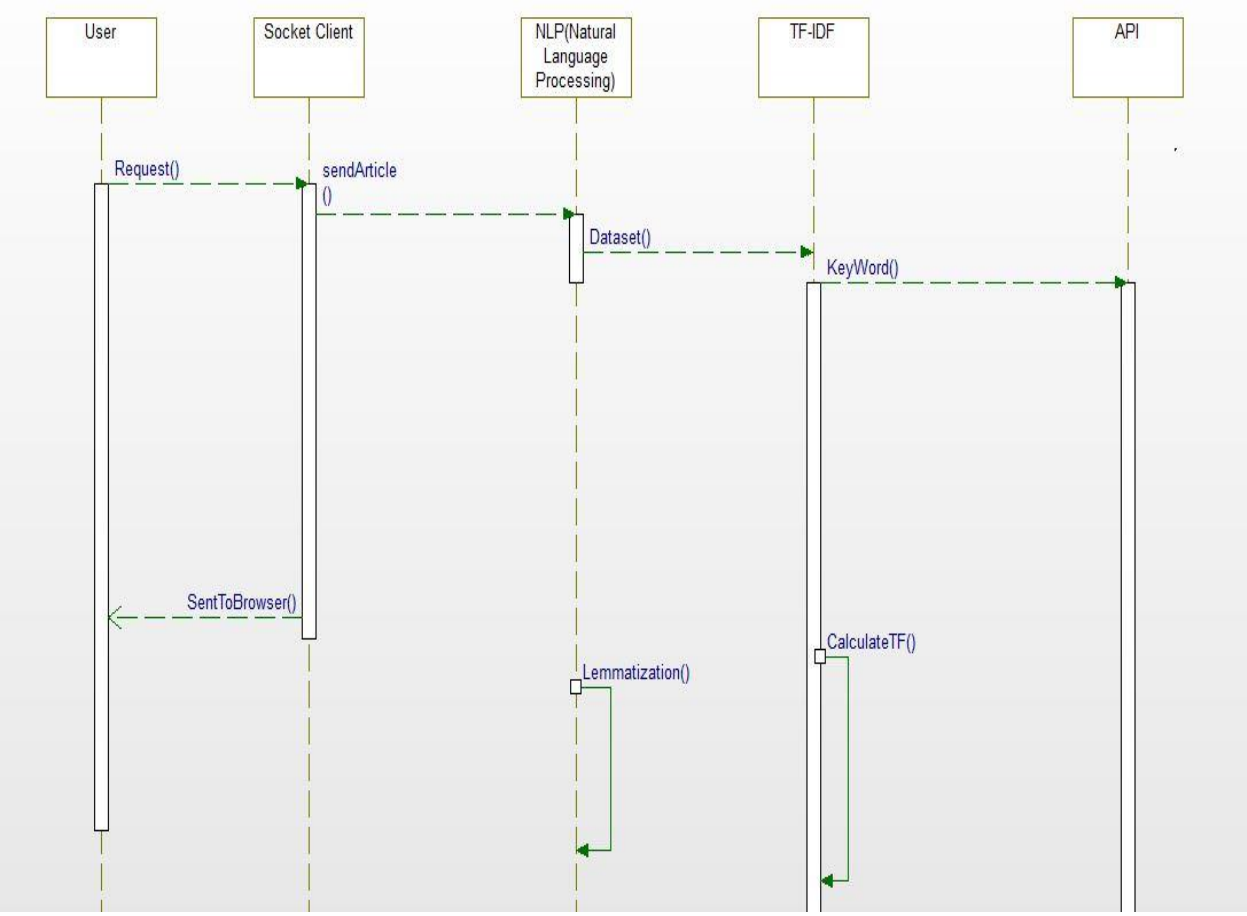
1. Class Diagram:



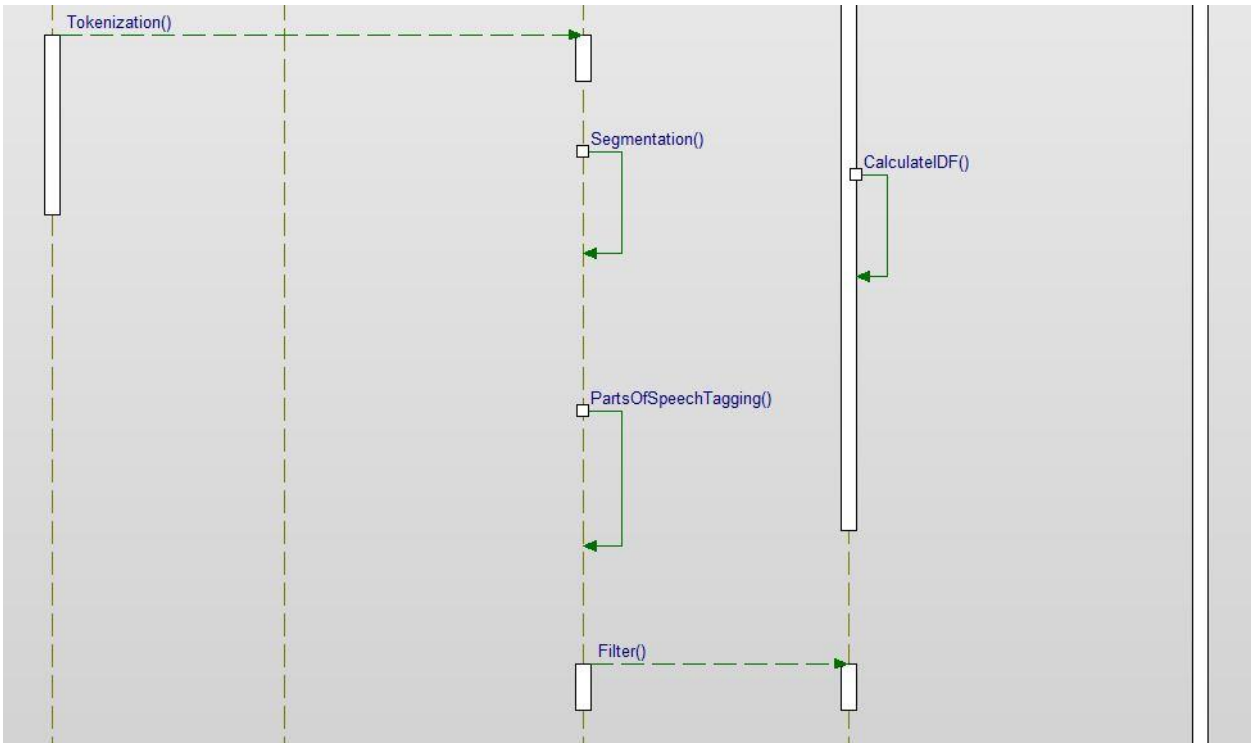
This diagram showcases the different classes that we have used in our program as a key component of the execution and the workflow can be depicted that will serve as base for the knowledge graph eventually processed.

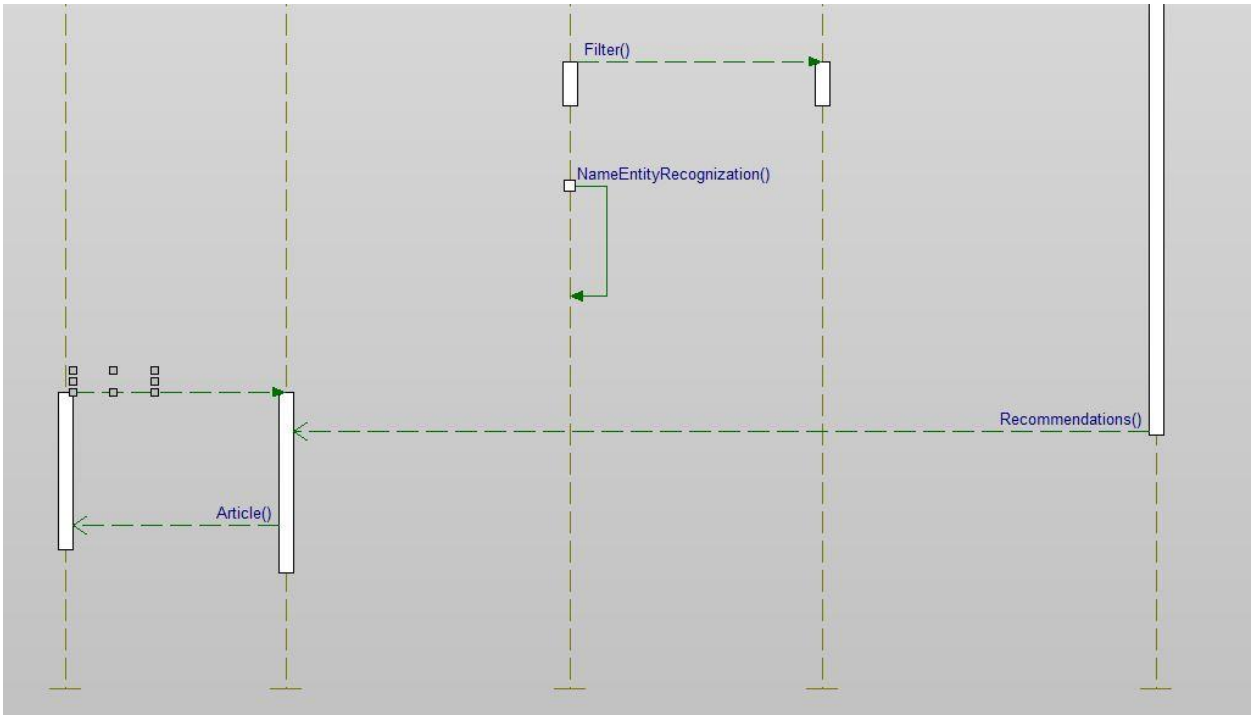
2. Sequence Diagram:

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.

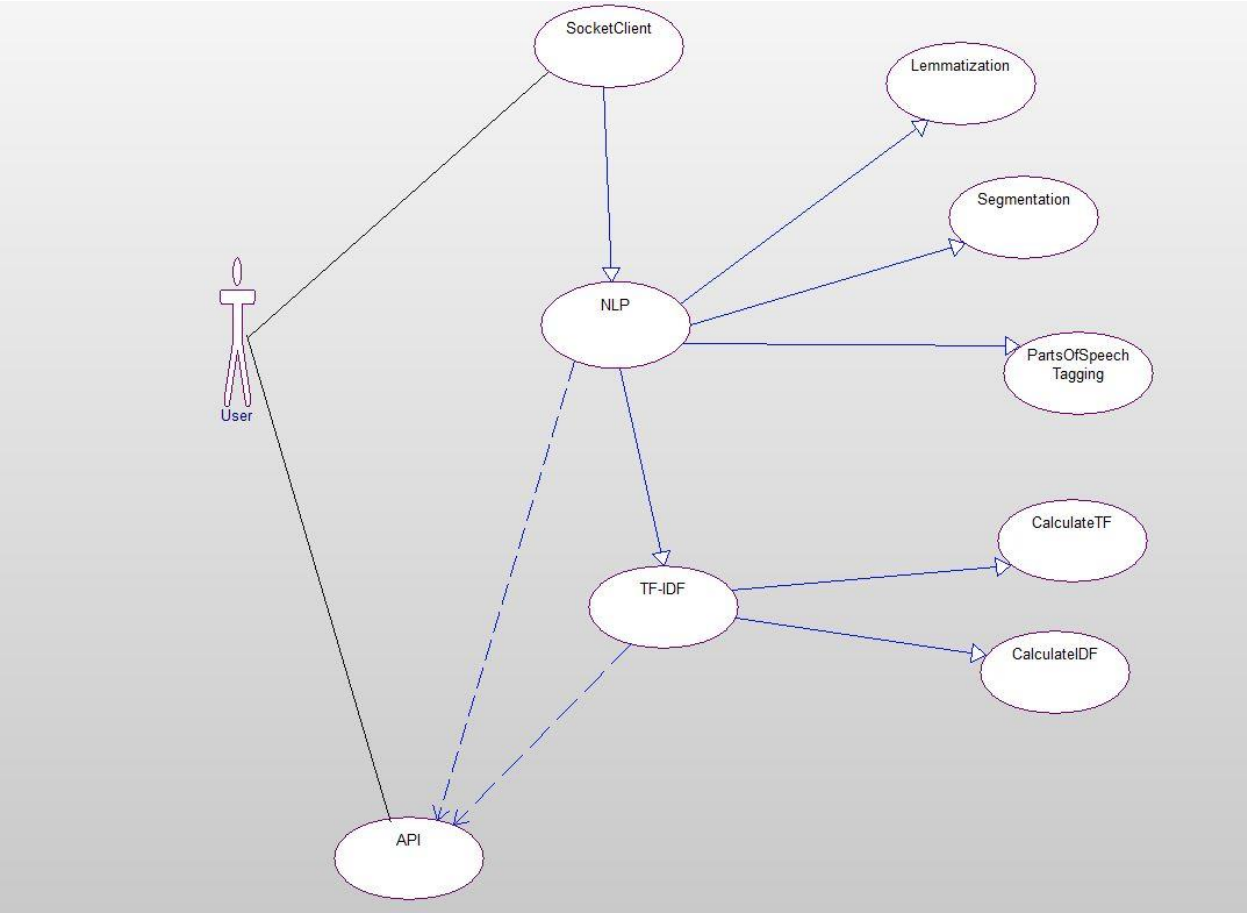




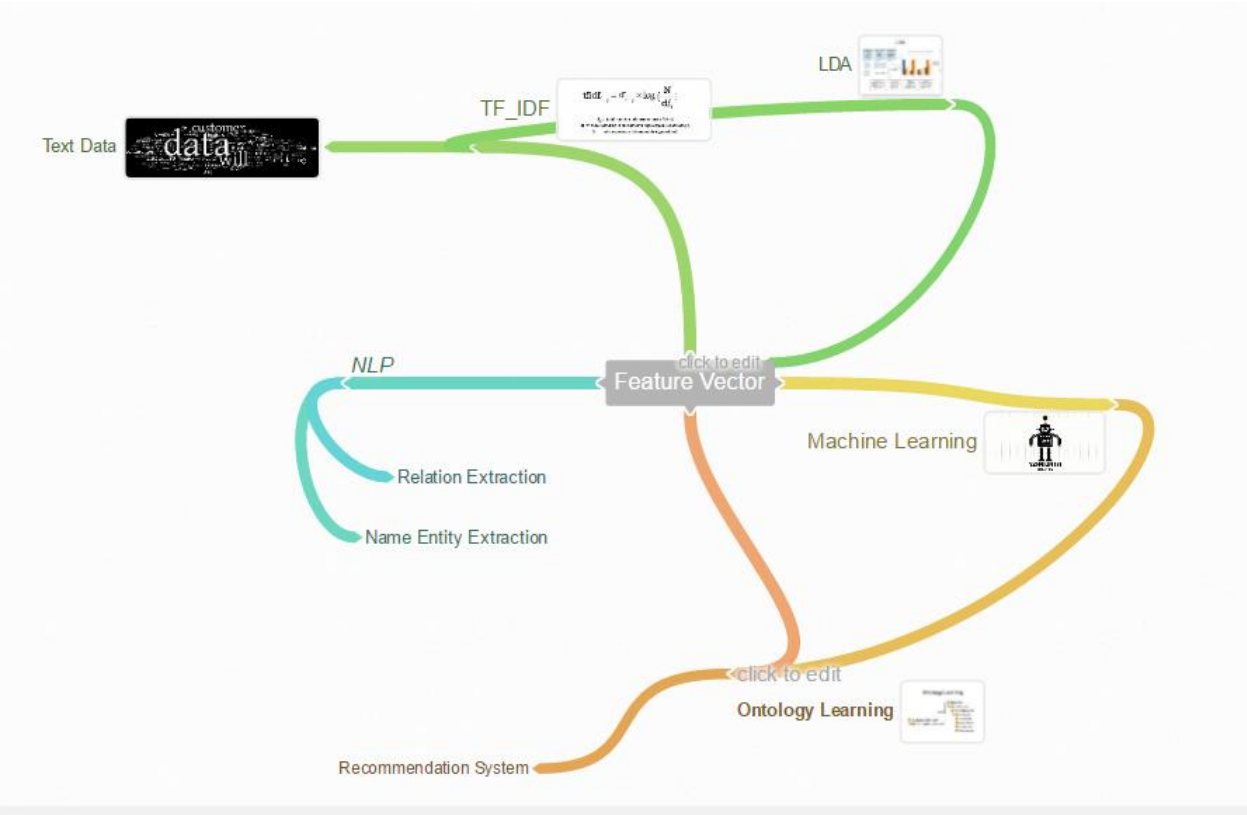
The entire sequence would then return the server side run program which would contain the list of articles which would navigate as per the machine learning algorithms.

3. 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.



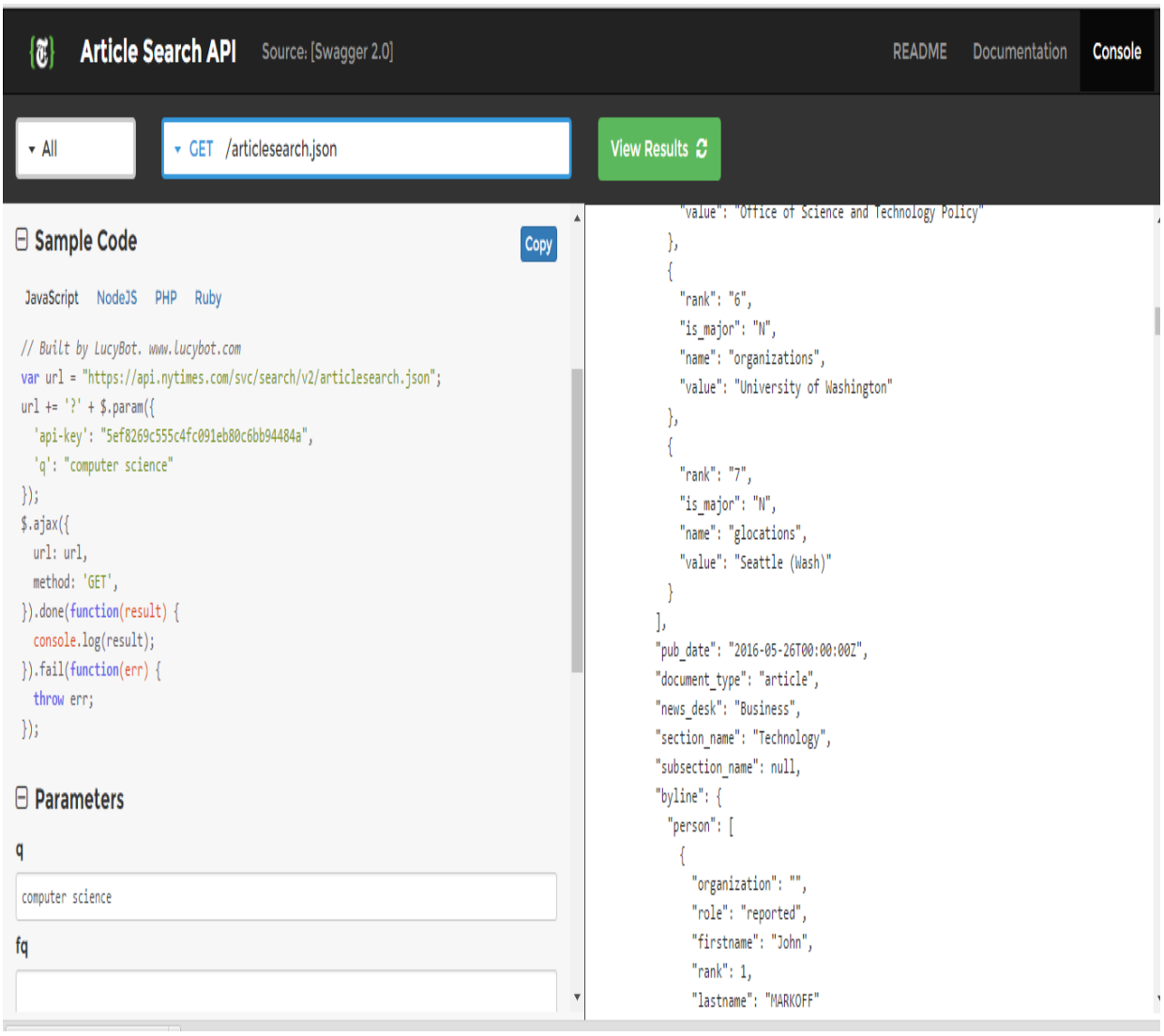
b. Workflow:



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.

c. 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.



Field	Behavior
body	Multiple tokens
body.search	Left-edge n-grams
creative_works	Single token
creative_works.contains	Multiple tokens
day_of_week	Single token
document_type	Case-sensitive exact match
glocations	Single token
glocations.contains	Multiple tokens
headline	Multiple tokens
headline.search	Left-edge n-grams
kicker	Single token
kicker.contains	Multiple tokens
news_desk	Single token
news_desk.contains	Multiple tokens
organizations	Single token
organizations.contains	Multiple tokens
persons	Single token
persons.contains	Multiple tokens
pub_date	Timestamp (YYYY-MM-DD)
pub_year	Integer

d. 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.

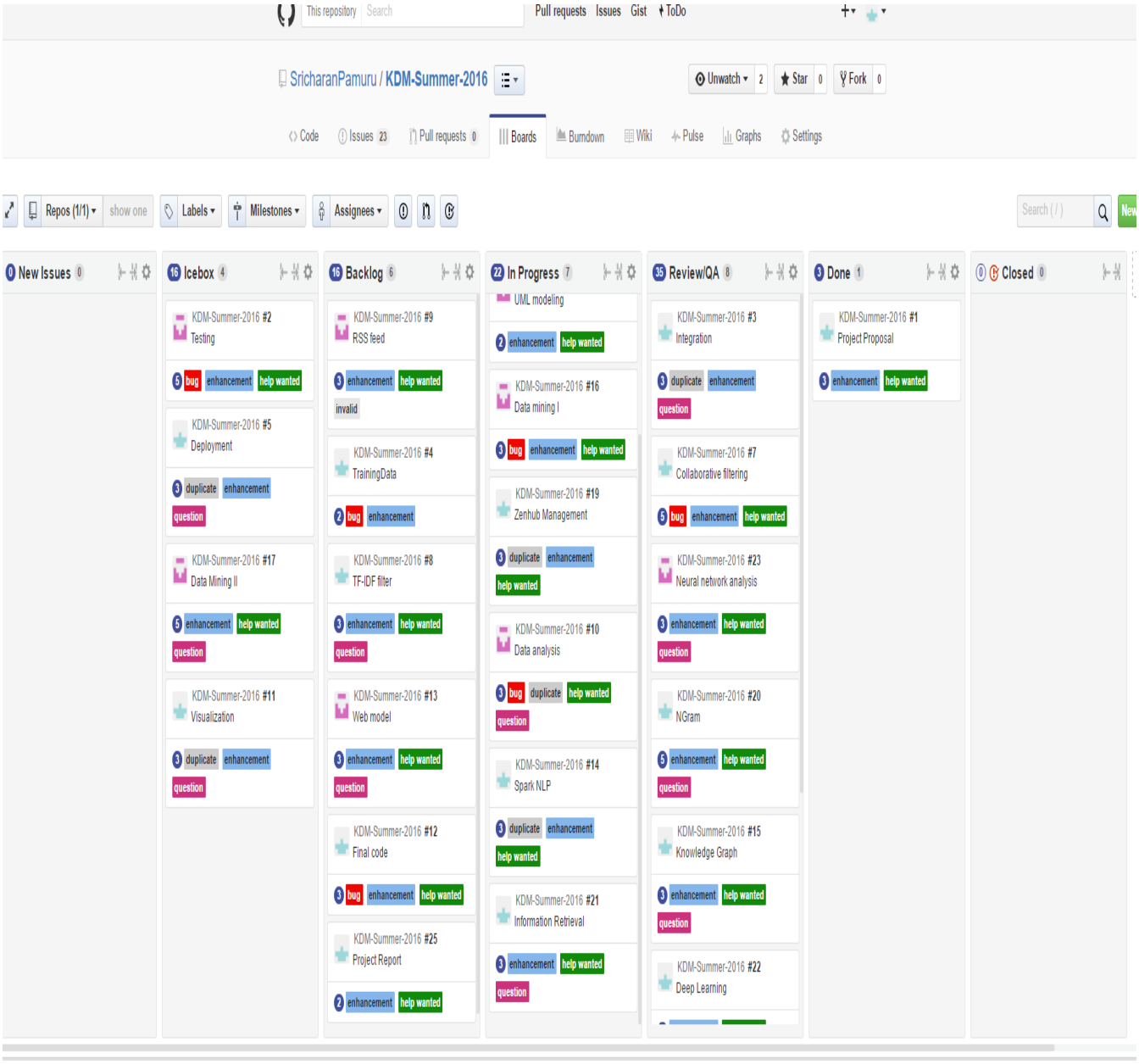
6. Project Management:

a. Individual Contribution:

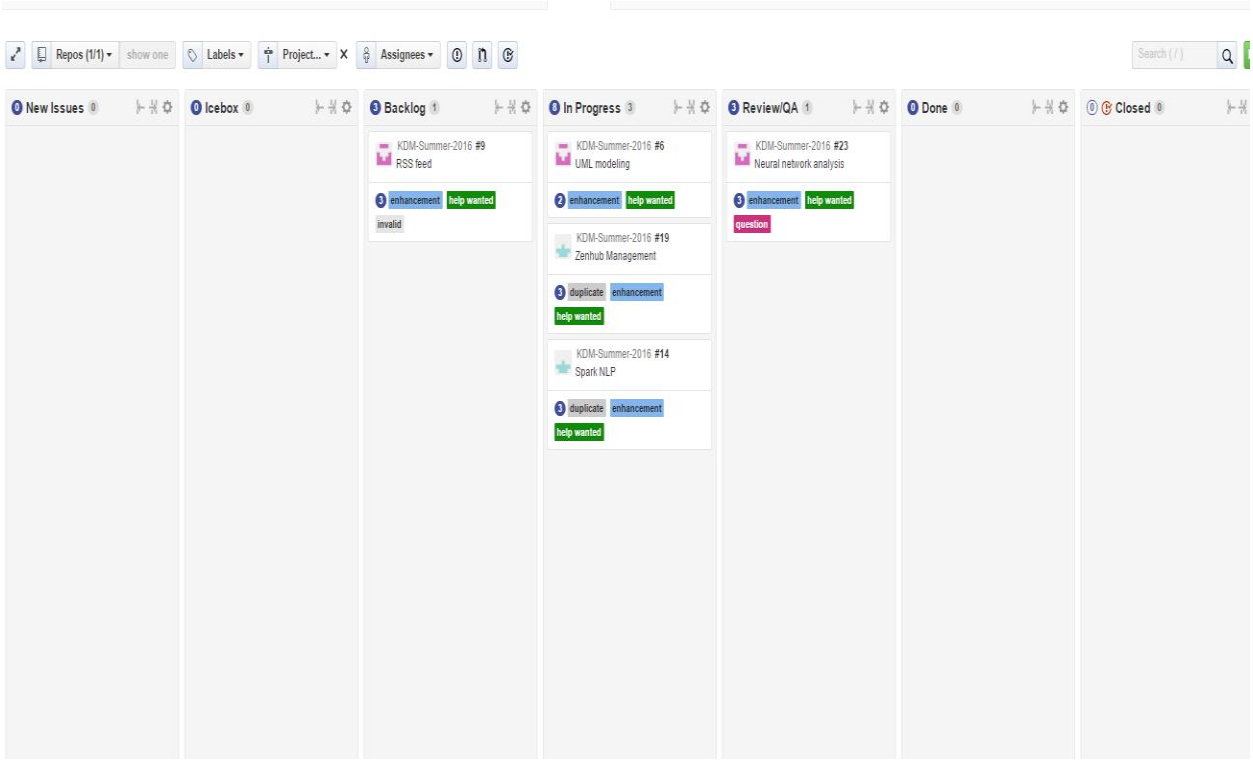
Name (CLASS ID)	Contribution	Number of hours
SRICHARAN PAMURU (20)	CODE EXECUTION FOR TF-IDF AND DATA STREAMING	20
PRUDHVI RAJ MUDUNURI (22)	WORKFLOW DESIGN AND NLP PROCESSING ON DOCUMENTS	20
SNEHAL VANTASHALA (41)	UML MODELING AND ARCHITECTURE DESIGN	20
BHARGAV KRISHNA VELAGAPUDI (42)	DATA COLLECTION AND ARTICLES RESEARCH	20

b. Zenhub/Github Screenshots:

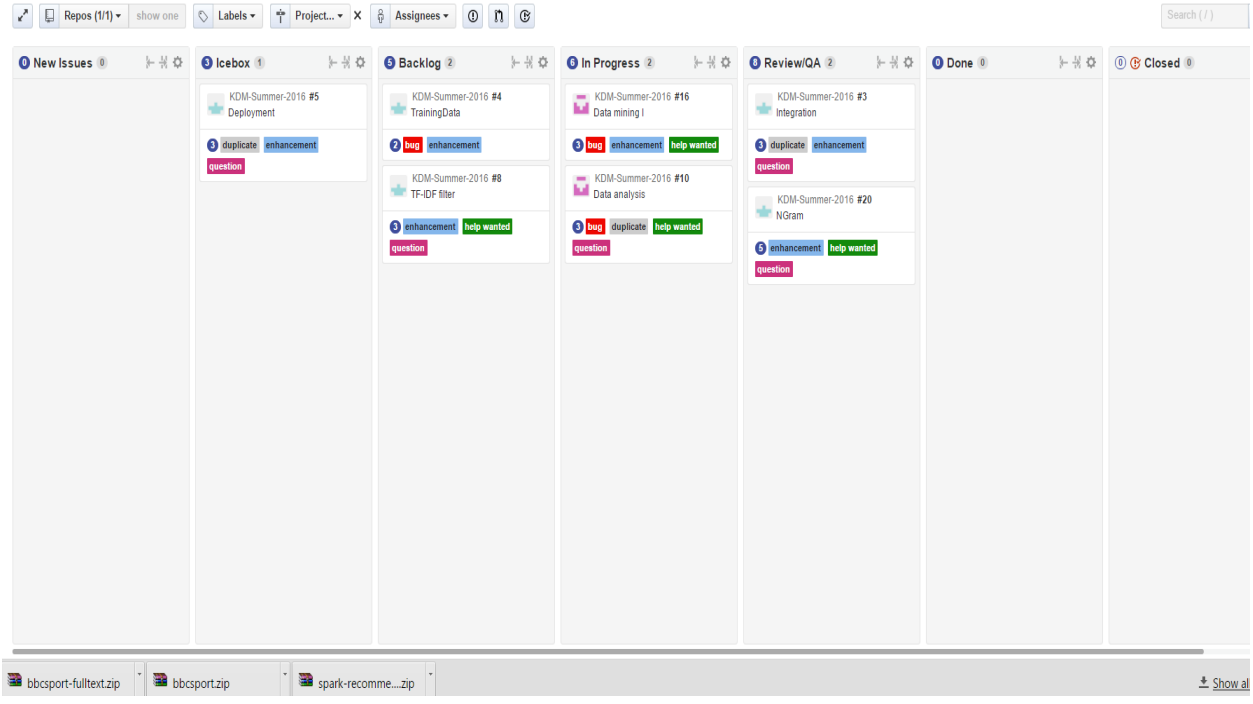
i. Full timeline:



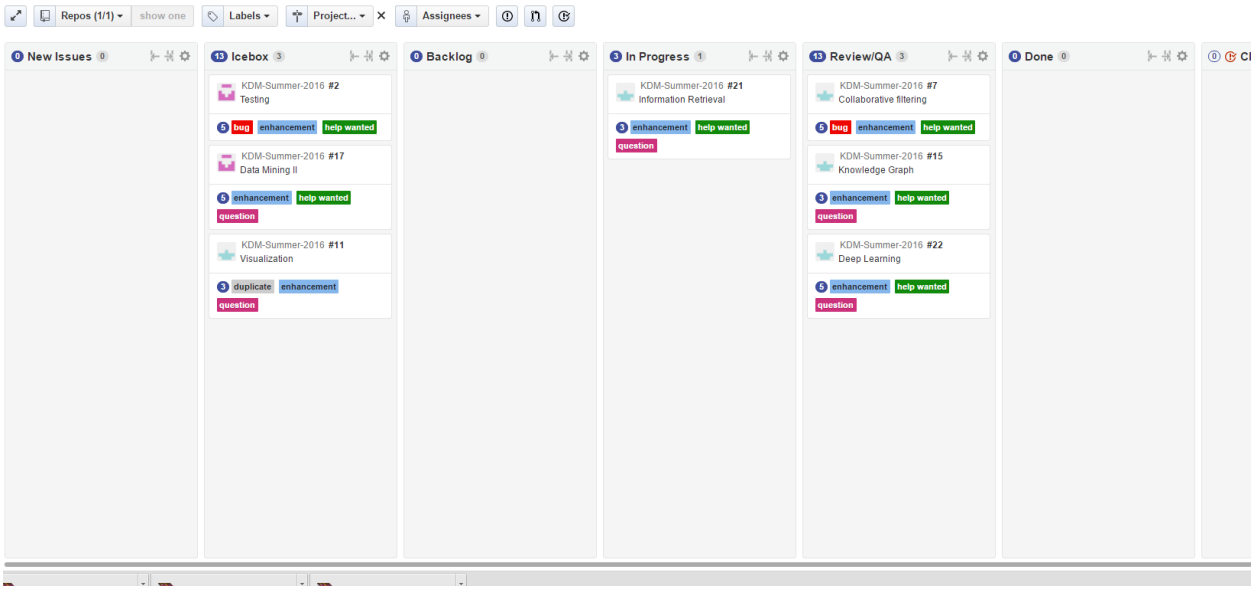
ii. Report 1:



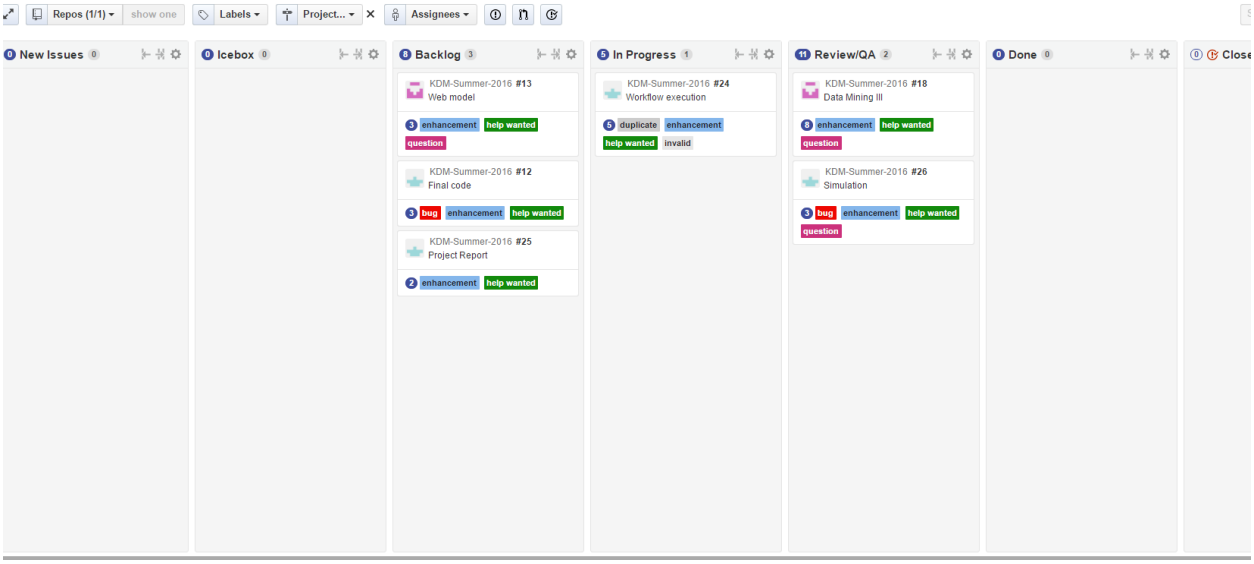
iii. Report 2:



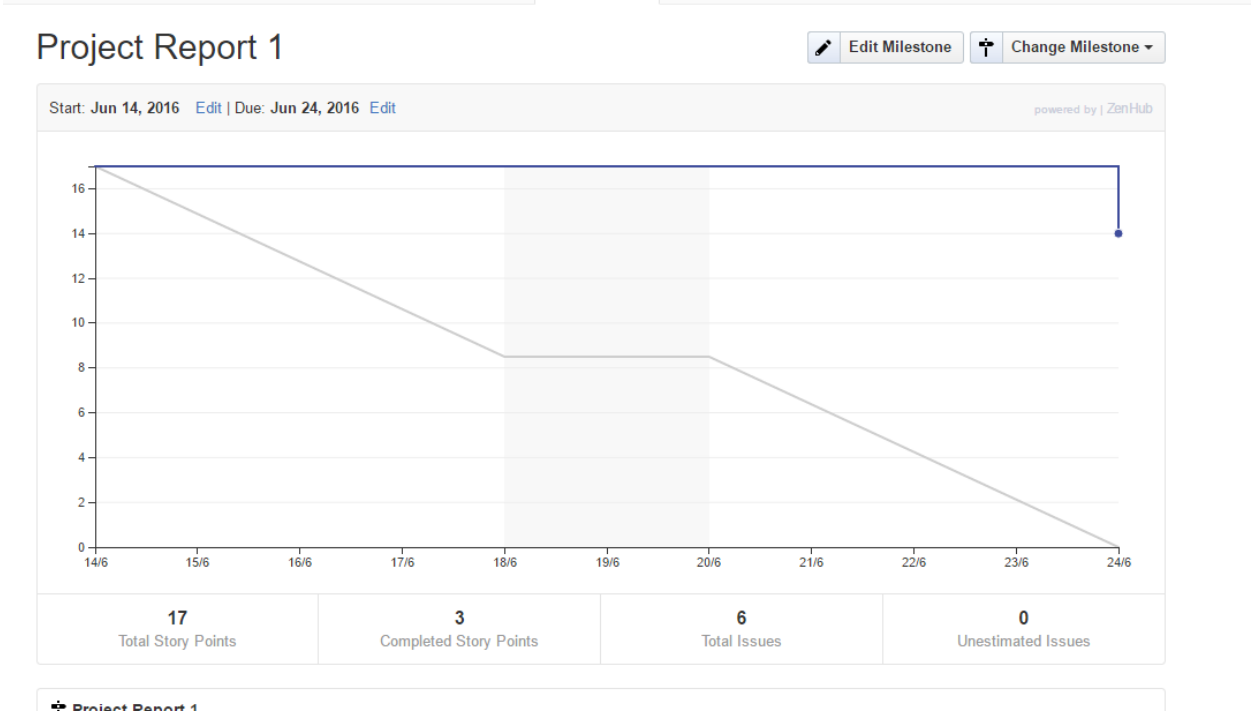
iv. Report 3:



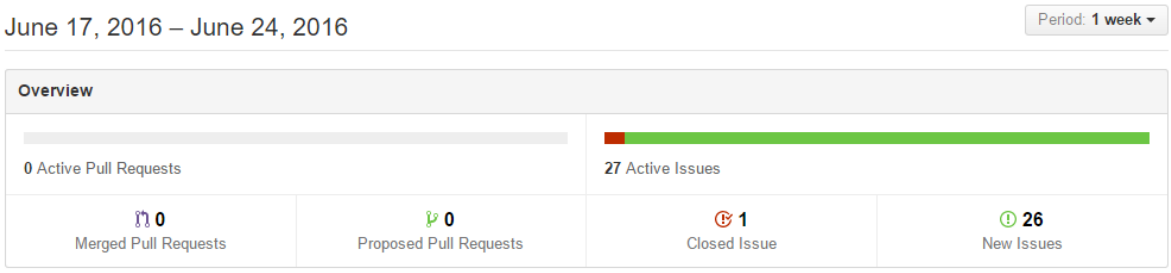
v. Report 4:



vi. Burndown chart:



vii. Pulse:



viii. Traffic:

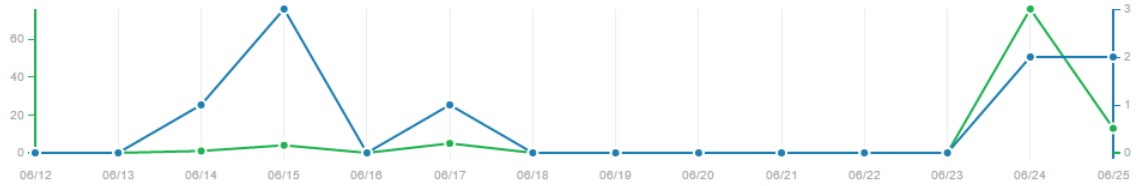
Contributors | **Traffic** | Commits | Code frequency | Punch card | Network | Members

Git clones

We don't have enough data to show anything useful.
It usually takes about a week to populate this graph.

-	-
Clones	Unique cloners

Visitors



99	5
Views	Unique visitors

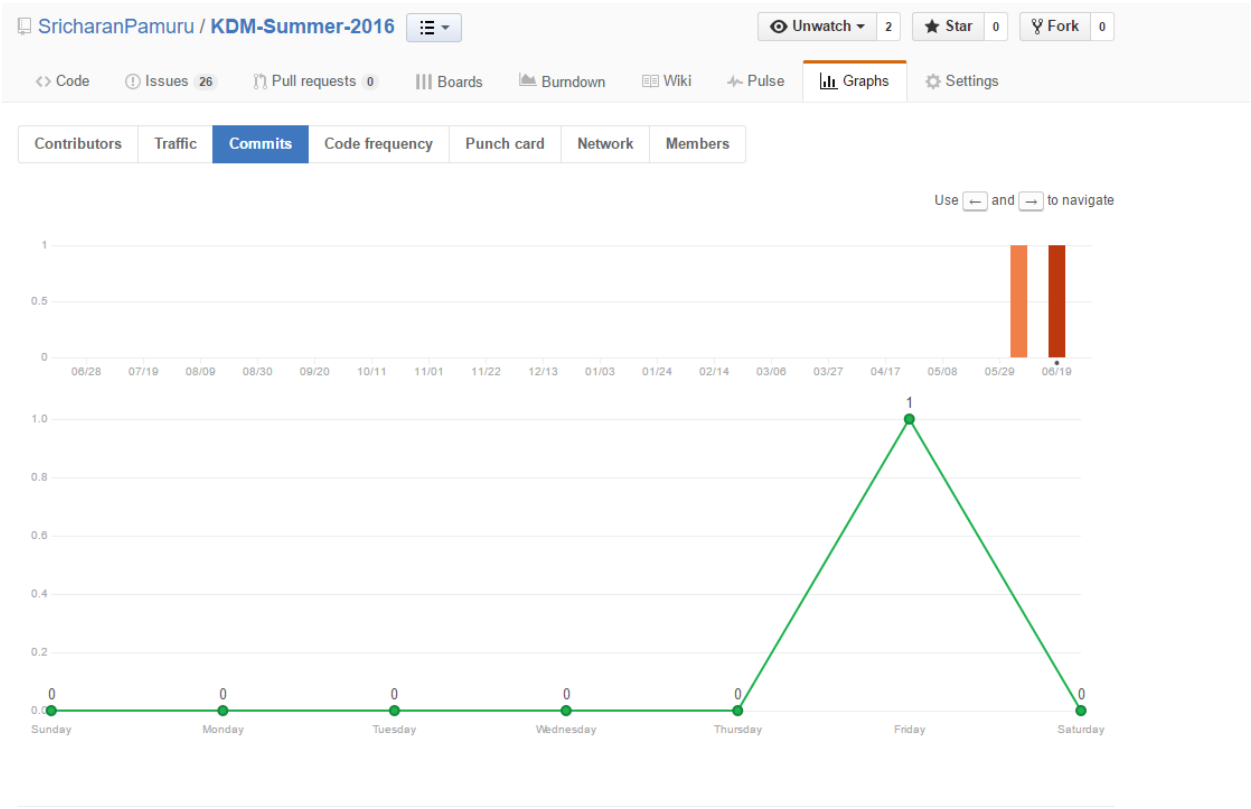
Referring sites

Site	Views	Unique visitors
github.com	9	1

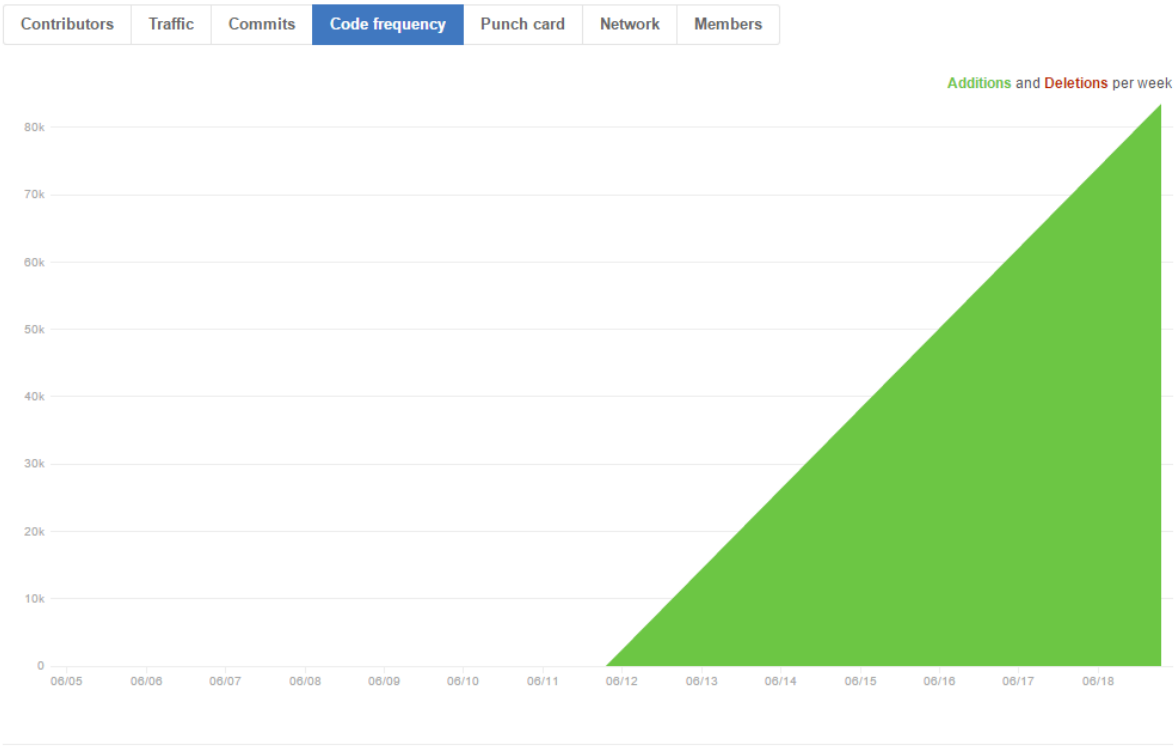
Popular content

Content	Views	Unique visitors
New Issue	21	1
Issues	11	1
SricharanPamuru/KDM-Summer-2016	10	5
Milestones - SricharanPamuru/KDM-Su...	5	1

ix. Commits:


SricharanPamuru / KDM-Summer-2016
Unwatch 2
Star 0
Fork 0
Code
Issues 26
Pull requests 0
Boards
Burndown
Wiki
Pulse
Graphs
Settings
Contributors
Traffic
Commits
Code frequency
Punch card
Network
Members
Use ← and → to navigate
1
0.5
0
06/28
07/19
08/09
08/30
09/20
10/11
11/01
11/22
12/13
01/03
01/24
02/14
03/06
03/27
04/17
05/08
05/29
06/19
1.0
0.8
0.6
0.4
0.2
0.0
0
0
0
0
0
1
0
Sunday
Monday
Tuesday
Wednesday
Thursday
Friday
Saturday

x. Code Frequency:



c. 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.

d. Future Work:

The project can be expanded in various domains individually such as news articles, journals, 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.

7. 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.