SCENE BASED MOVIE RECOMMENDATION SYSTEM

Submitted by:- Shashank Garg(sg4437), Parteek Singh Khushdil(psk287)
YouTube link:- https://youtu.be/dzYKOqaBQwM

Abstract:-

The project aims at developing a recommender system for movie streaming website. We tend to utilize the information provided about the users like what movies each user has watched, his favorite genres, actors etc. and the information about movie tags, like genres, moods, keywords etc. We use two main algorithms, collaborative filtering to find similar users and connected components in a graph to find similar movies. We take in consideration what type of user is required to be recommended and for each use case we recommend movies accordingly.

Introduction:-

The recommender systems are getting popular due to tremendous increase in choices in almost every commercial category. This calls for a need of a recommender systems that can effectively make recommendations to users and reduce their efforts in scanning through products which they might not be interested in. This also is used by many e-commerce companies like Amazon, EBay etc. and also by multimedia streaming companies like Netflix, YouTube etc. to recommend products to users to increase sales. To address this problem many ways have been developed in past. Collaborative filtering is one of them which aims at finding similar users. Other forms of recommenders include YouTube type recommendations which recommends similar content based on your previous search history. We in our project, desire to effectively combine both the techniques and come up with recommendations, some of which are based on collaborative filtering and others on content similarity.

Architecture:-

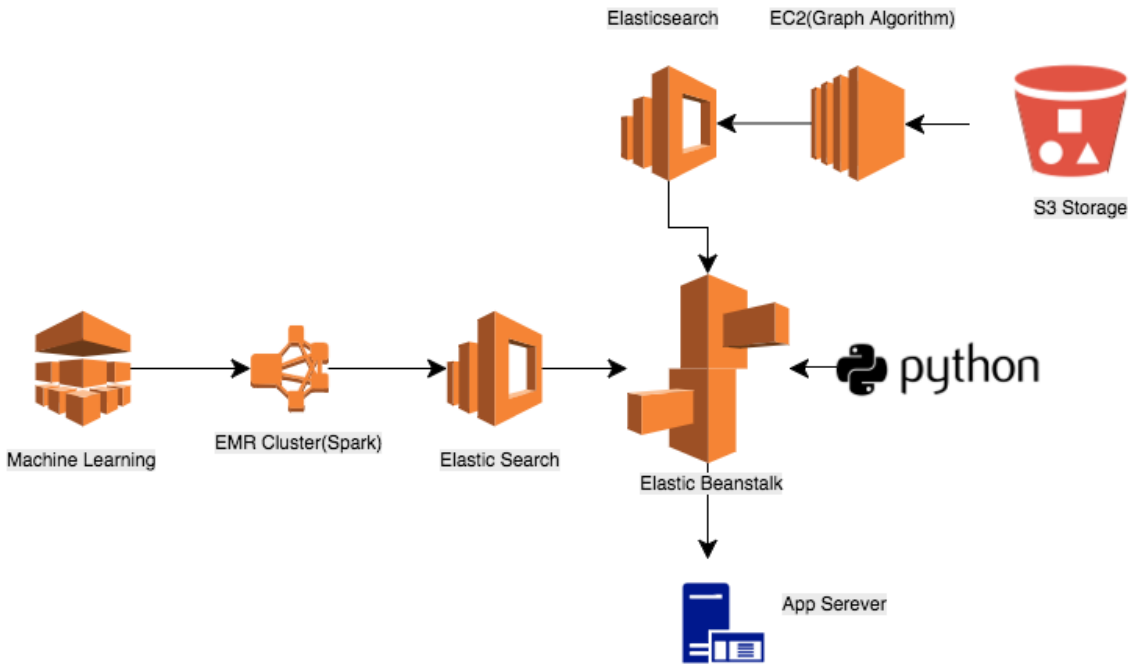


Figure 1: Architecture

Figure 1. explains the architecture of the project. We use EMR cluster to run the collaborative filtering on the clustered user data. The collaborative filtering result generated is then stored in the ElasticSearch. The EMR cluster is run once a day when the audience tags and audience data are updated. Since the content data is mostly static we run the Graph compute EC2 instance whenever required, which fetches the content data from S3 and then stored the computed graph in ElasticSearch. The two ElasticSearch act as endpoints to the server that then finally communicates with the frontend app server.

Implementation:-

Implementation of the project involves configuring the cloud resources as per our requirements and then finally running the application on those resources. To implement collaborative filtering, we chose to not run it on all the users. Rather we decided to run CF on each cluster. This can be explained as follows. Let's say a user belongs to cluster '2' based on his previous watched movies. Here we assume that the clustered user data is available. Now it won't be beneficial to run the CF on all the users since by already clustering users we have a shorter subset which is more relevant to each user. So, we would rather run the CF on each cluster. This reduces the computation time and makes more meaningful recommendations. In the next subsection, we explain how we implement the CF on the user data.

1. **Collaborative filtering on cluster**

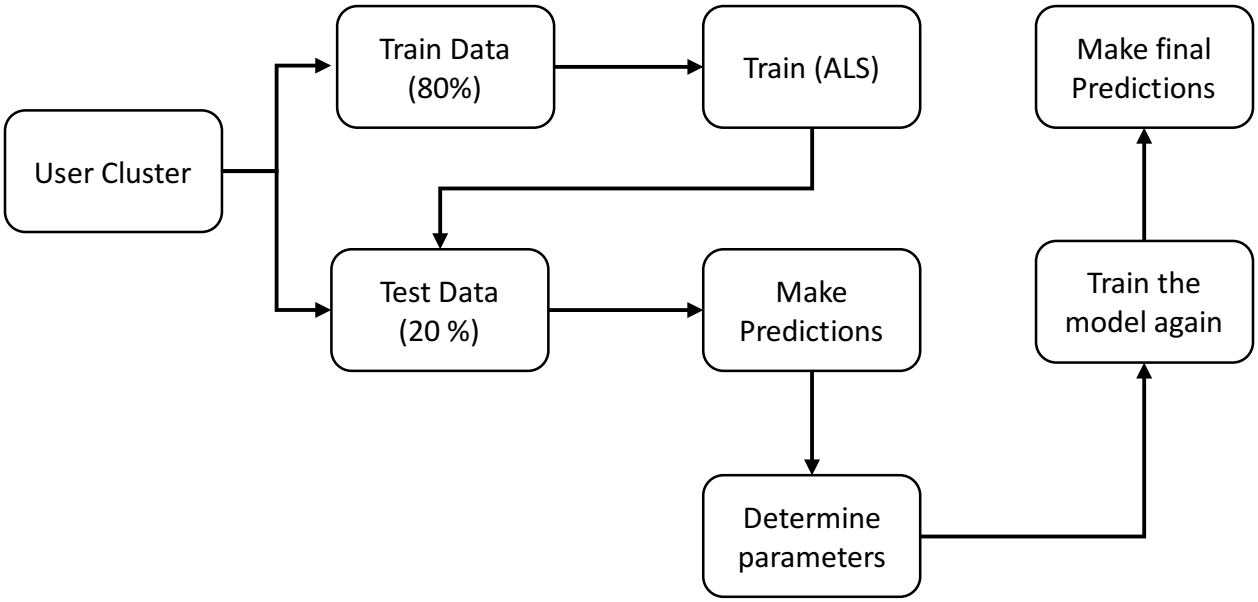


Figure 2: Algorithm flow for Collaborative Filtering

To implement CF we first divide the user data belonging to each cluster into train and test data. We choose 80% of the data as training data and 20% of the data as test data. The input to the training model is the movies watched by all the users. Then as the next step, Alternating Least square algorithm is run on the training data to fit the CF model. To predict the errors for the trained model we use the test data on different parameters of the model. This determines the best combination of parameters and then those parameters are finally used to predict the movies which the users are most likely will watch next.

2. Connected components in Graph

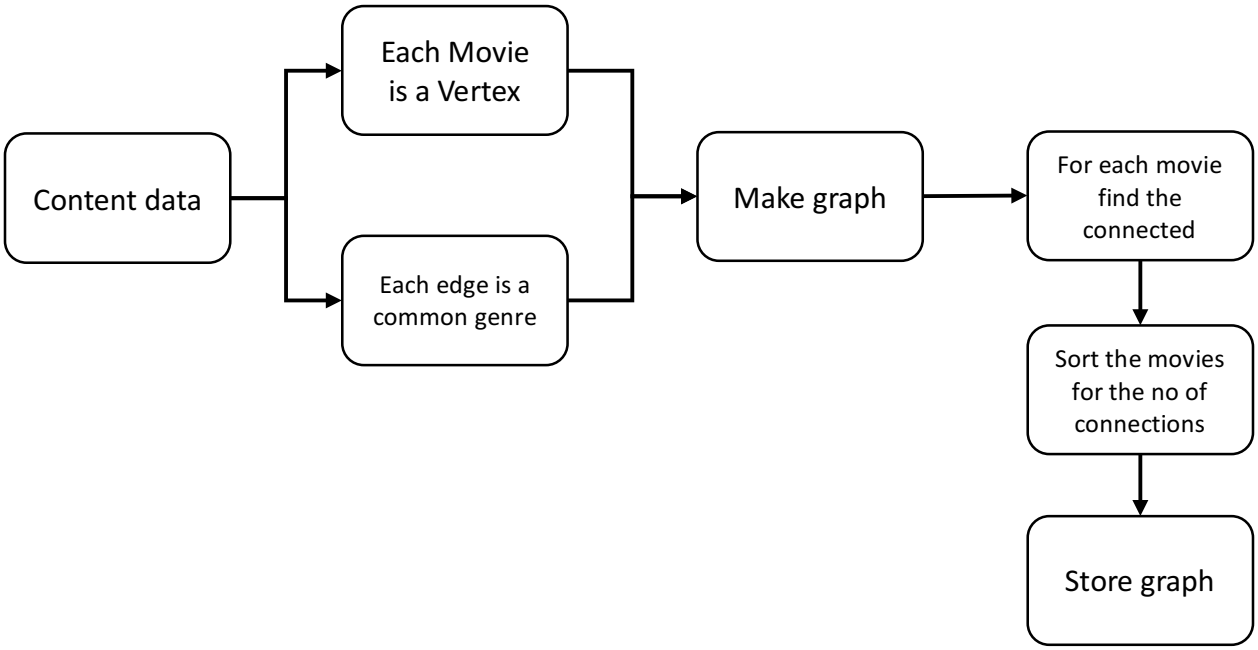
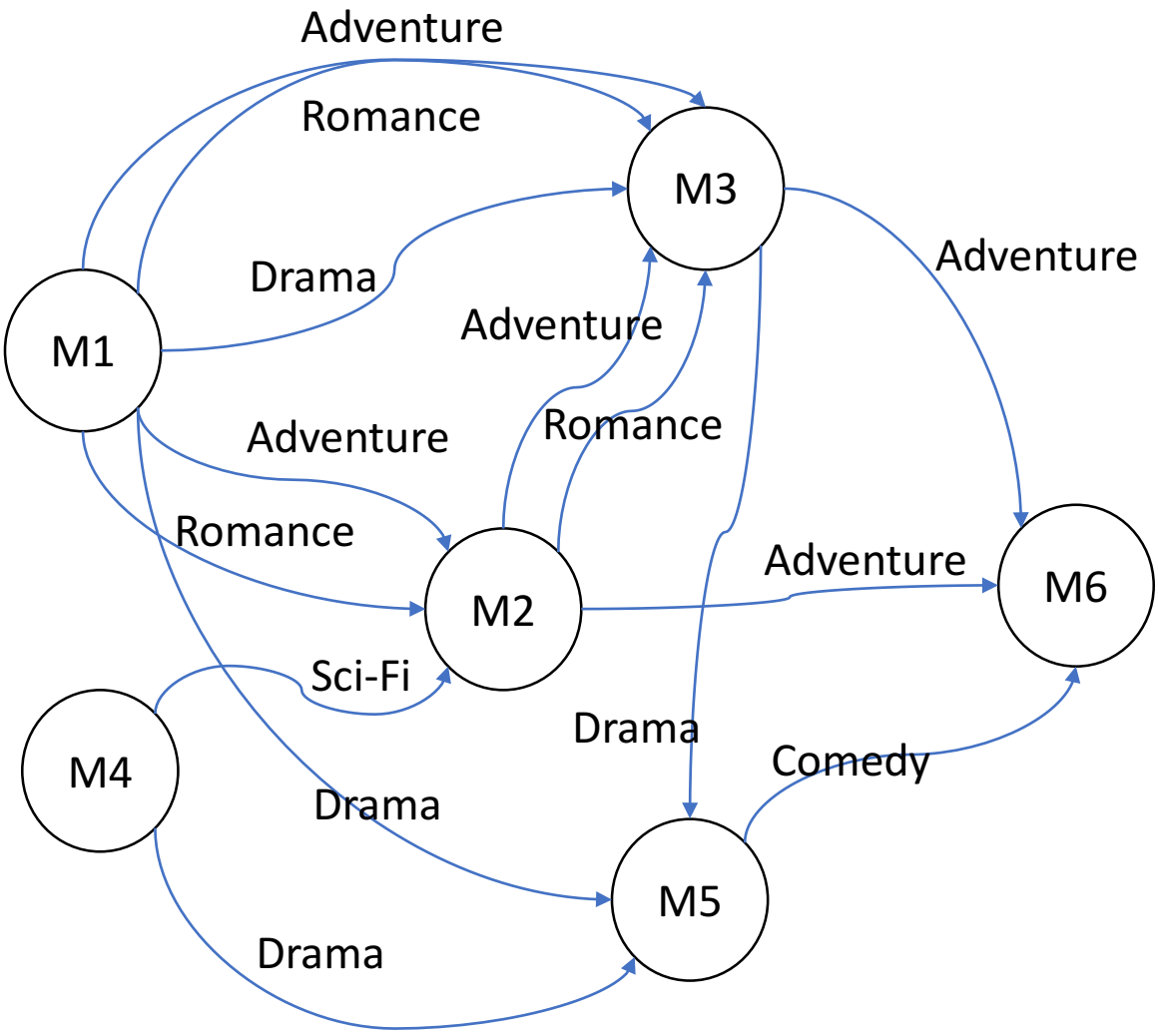


Figure 3: Algorithm flow for Graph formation

Connected components in graph algorithm was used to find the similar content and recommend those to the users. The algorithm works as follows. First step in any graph algorithm is to build the graph. As we know to build the graph we would need edges and vertices, we need to somehow convert our data into a form so that graph can be formed. So we take each movie as a vertex and each edge as a common genre between two movies. Fig below gives an example of how the graph can be formed. After the graph has been formed then we can count the n of edges between each pair of movies and save the connected component data in ElasticSearch.



3. Communicating with the App server

We also hosted our own server that can communicate with the front-end app server. The task of this server was to receive the call from front end server requesting the recommended movies whenever user clicks a movie, a genre or signs in. We considered all different use cases possible depending on whether the user is a first time login or an old user. We have the following use cases.

a. When a new user signs in for the first time:- we show him the top rated movies for the genres he selected.
b. When a new user clicks on the genre:- we show him the top rated movies for the genre he clicked.
c. When a new user clicks on a movie:- we show him the movies connected to those movies.
d. When an old user signs in: We show him the movies based on his score developed due to his previous search history.
e. When an old user clicks on a movie:- we show him the some movies based on the collaborative filtering and some movies based on the graph developed.

f. When an old user clicks on a genre:- we show him top rated movies of that genre.

4. Github Link:-

https://github.com/parteekkhushdil/MovieRecommendation

Results:-

Snippets of the ElsticSearch:


```
"hits" : {
  "total" : 2811,
  "max_score" : 1.0,
  "hits" : [ {
    "_index" : "graph_data_new",
    "_type" : "connected_movies",
    "_id" : "AVkyMcTPhP00PKTu3GwS",
    "_score" : 1.0,
    "_source" : {
      "movie" : {
        "reaper-2014" : [ "don'tbreathe-2016", "infini-2015", "self/less-2015",
"harbingerdown-2015", "somethingwicked-2014", "darkshadows-2012", "hottubtimemachine2-
2015", "thevoices-2014", "bloodsuckingbastards-2015", "dersamurai-2014" ]
      }
    }
```


Figure 4: Snippet of Elastic Search storing Graph

ES Link:- http://search-tweetymap-gurowqxu56ejw6kii5afincr3y.us-east-1.es.amazonaws.com/graph_data_new/


```
    "_index" : "cf_data_new",
    "_type" : "recommendations",
    "_id" : "AVkobgdzhP00PKTu3GwF",
    "_score" : 1.0,
    "_source" : {
      "110001" : [ "earthtoecho-2014", "madmax:furyroad-2015", "AVidhOYgAX7qjWCYvOPq",
"levelup-2016", "acinderellastory:iftheshoefits-2016" ]
    }
  }, {
    "_index" : "cf_data_new",
    "_type" : "recommendations",
    "_id" : "AVkobid4hP00PKTu3GwG",
    "_score" : 1.0,
    "_source" : {
      "110002" : [ "dawn-2014", "AVidhRNMAX7qjWCYvOUR", "AVidhMbsAX7qjWCYvOL3",
"AVidhLBvAX7qjWCYvOJk", "insideout-2015" ]
    }
  }, {
```


Figure 5: Snippet of ElasticSearch storing Collaborative Filtering

ES Link:- http://search-tweetymap-gurowqxu56ejw6kii5afincr3y.us-east-1.es.amazonaws.com/cf_data_new/

Screenshots of App showing the recommendations:

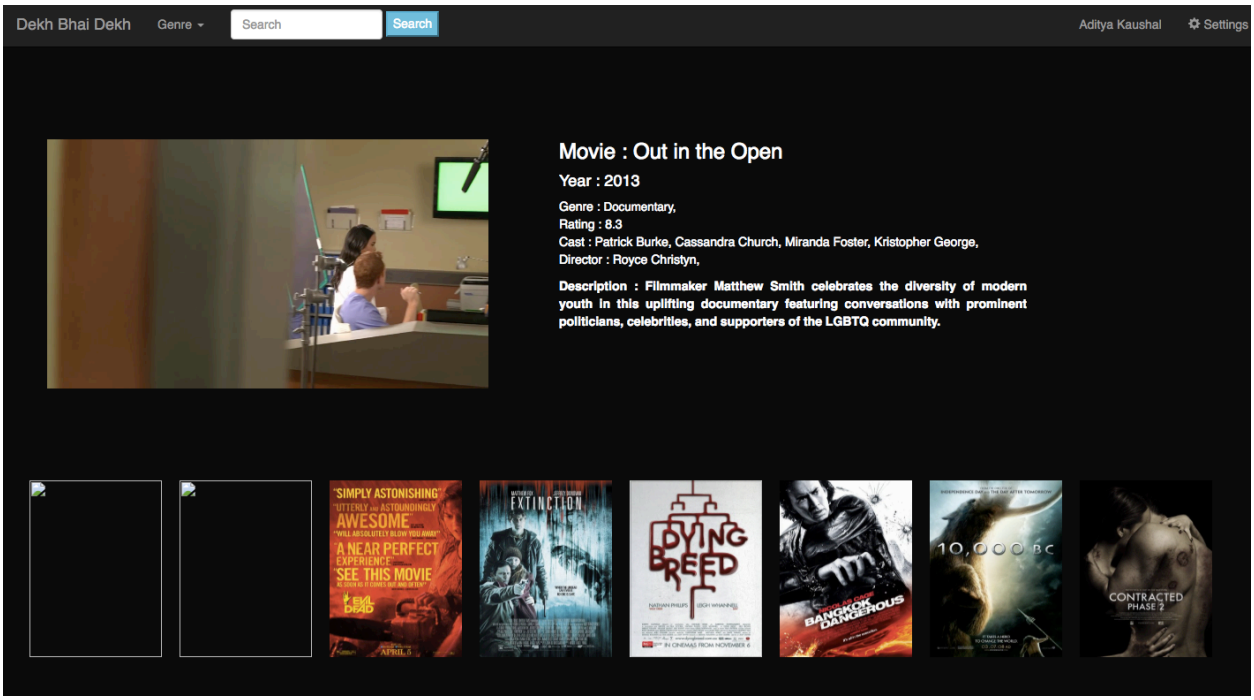


Figure 6: User clicks on Genre "Horror"

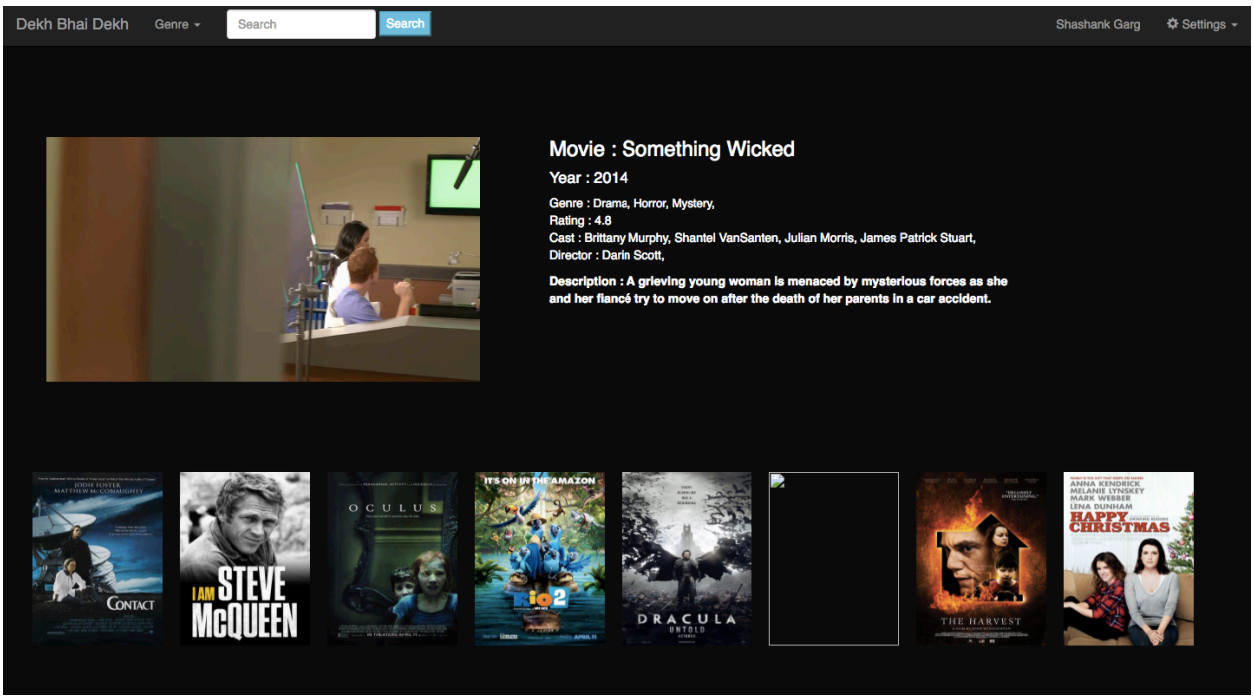


Figure 7: When new user clicks on one of the recommended movies from the suggestions

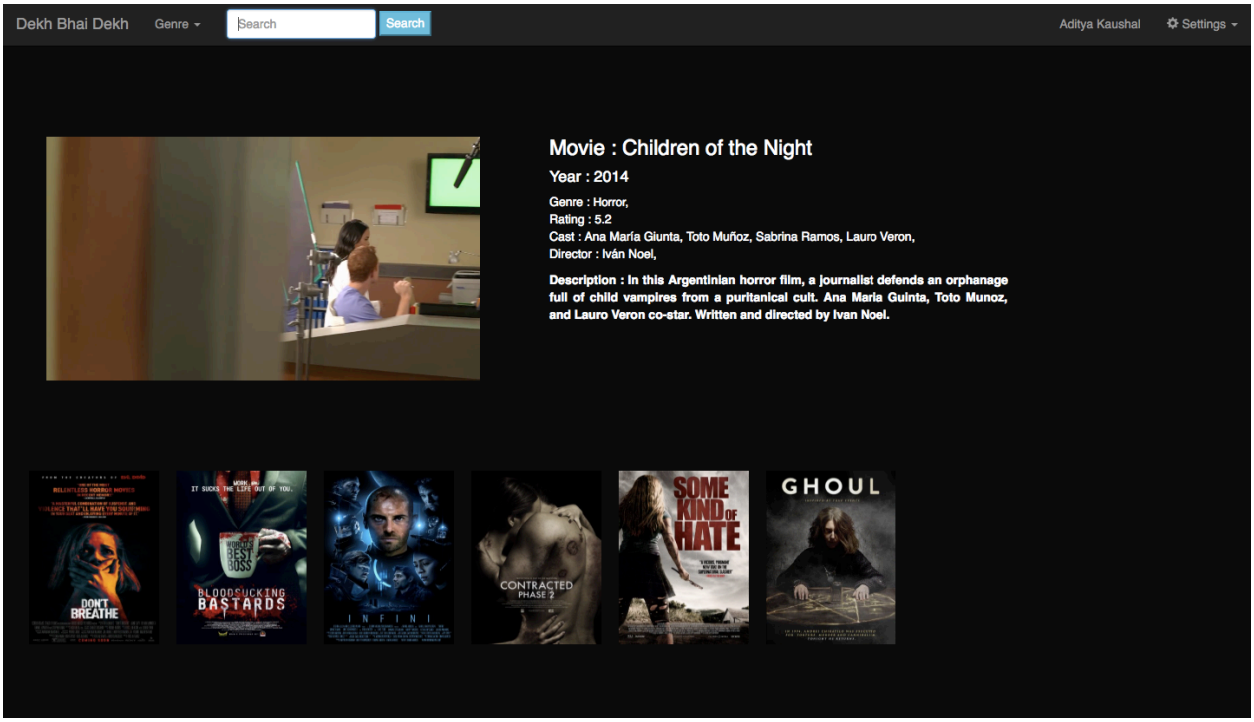


Figure 8: When user clicks on another movie

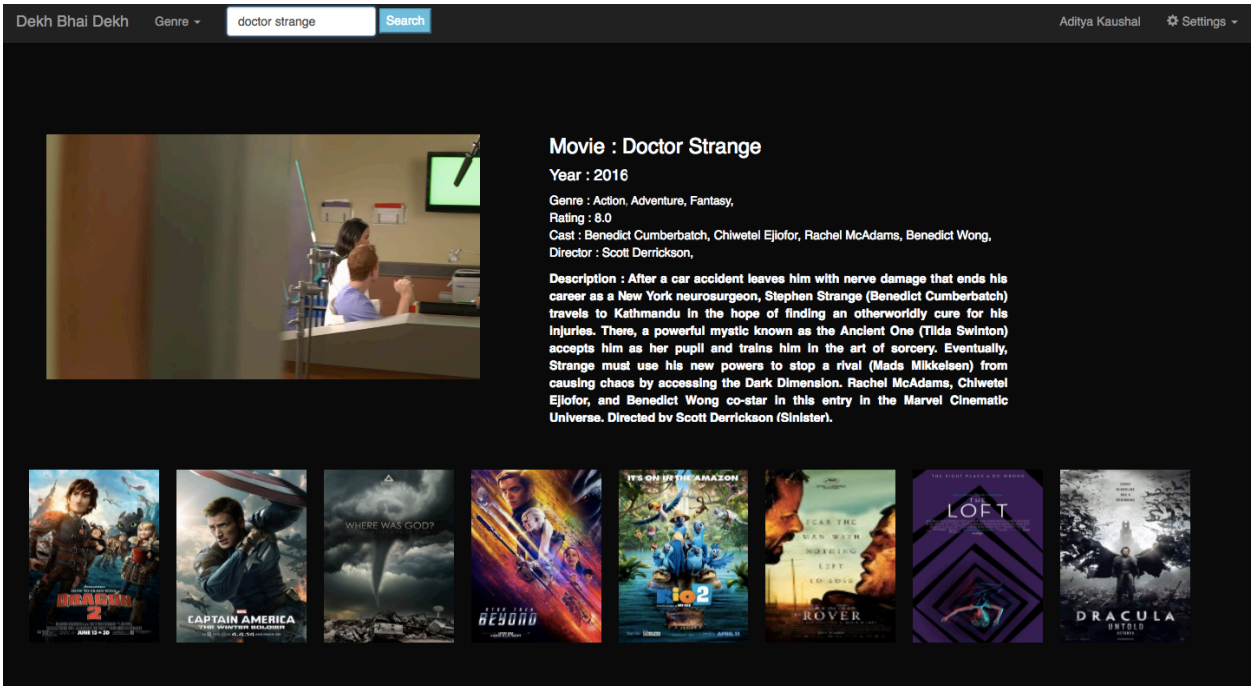


Figure 9: When an old user clicks on a movie "Doctor Strange"

Conclusion:-

Recommendation system for a movie streaming website we developed that recommends based on similar users(collaborative filtering) as well as similar content(connected components in a graph). The built model was successfully implemented.