News Article Recommendation System

Abstract

This report documents and presents the Recommendation System required by JhakaasNewsVala. A Hybrid Recommendation System which is a combination of Content Based System and User Based Collaborative Filtering was used. At the beginning no user data was available and click stream data was to be generated. Assumptions based on research of user behaviour towards web pages/articles were made during the generation of click stream data. Included in the report are the detailed descriptions of the work done.

What is a Recommendation System?

A recommender system is a technology that is deployed in the environment where items *(*products, movies, events, articles*)* are to be recommended to users *(*customers, visitors, app users, readers*)* or the opposite. Recommender systems are tools designed for interacting with large and complex information spaces and prioritizing items in these spaces that are likely to be of interest to the user.

In simpler terms, A Recommender System refers to a system that is capable of predicting the future preference of a set of items for a user, and recommend the top items.

Goal

The goal is quite simple, as a user I don't want to scroll through hundreds or thousands of songs to find out the ones I like or use various filters to narrow down my search. I would want to have the content or the product presented to me that I know I'm going to enjoy.

Need

- Information Overload - system in modern society is that people have too much options to use from due to the prevalence of Internet.
- User Experience - speeds up searches and make it easier for users to access content they're interested in, and surprise them with offers they would have never searched for.
- Revenue - Helps in increasing sales as a result of very personalized offers and an enhanced customer experience.

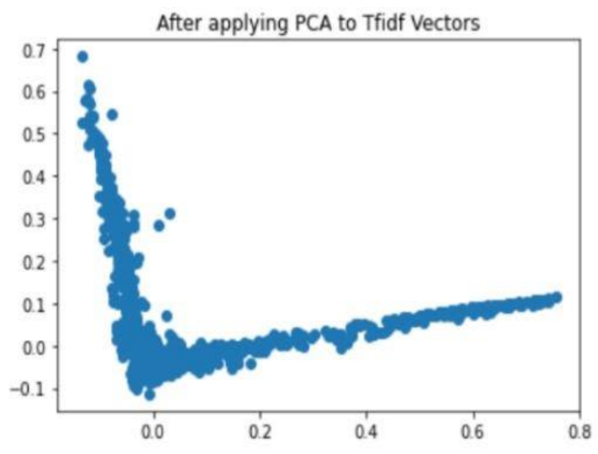
Examples

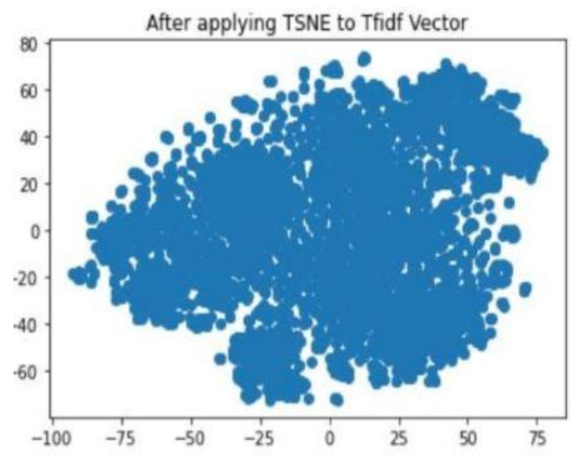
Some of the examples of recommendations systems are:

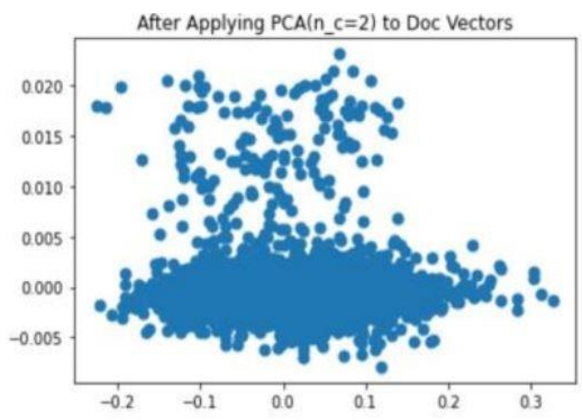
- Facebook – 'People you may know'
- Netflix – 'Movies you may enjoy'
- YouTube – 'Recommended Videos'
- LinkedIn – 'Jobs you may be interested in'

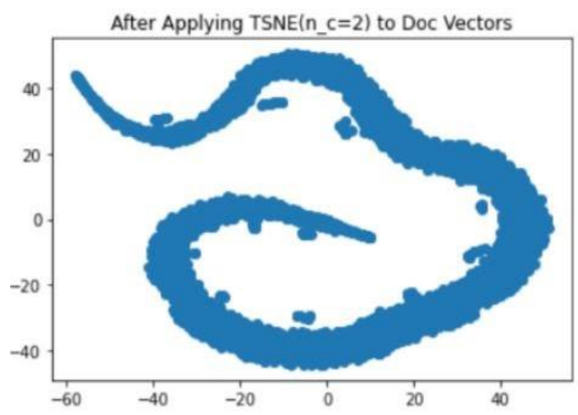
Exploratory Data Analysis (EDA)

Converting text to vector with the help of TFIDF and Word Embeddings. We tried to use both and then plot our results in 2D space by reducing the dimensionality using PCA and TSNE. The purpose of this was to see which technique separates our documents into different categories the best.









Implementation Plan

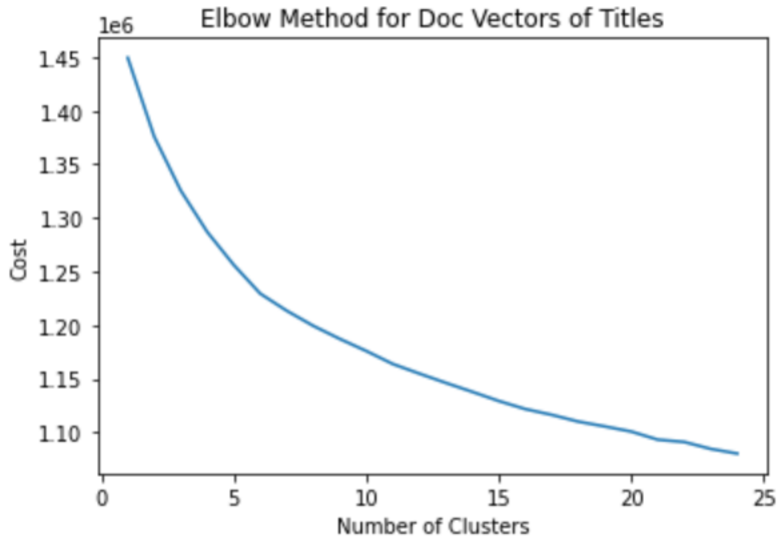
1. Step 1 will be to perform text clustering on the given corpus of the data. We will use both TF-IDF vectorizer and Doc2Vec for vectorizing the text and see which of these gives us a better result. We also plan on using topic modelling or Latent Dirichlet Allocation
2. Step 2 will be to implement collaborative filtering where we will try to predict the time the user will spend on an article which hasn't been recommended to the user yet. This will be predicted using the clickstream data of the users.
3. In step 3 we will try to introduce a bias to handle the cases of articles being recommended to the user but not being read by the user.
4. Final step will be to test our recommender and see if we get better results by changing the ratios of recommendations decided in the strategy

Clustering

Two strategies were used to convert text to vector. First was TFIDF and second was Doc Vectors using Word Vector. It was observed that Doc Vector strategy gave better results at clustering. So, Doc Vectors strategy was used to convert text to vector.

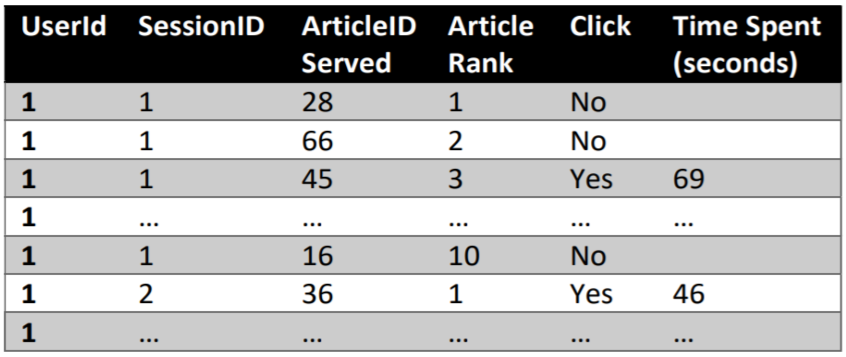
In this strategy doc vectors were created using Google's Word2Vec model. Doc Vectors were created by averaging the word vectors of all the words of title of the articles.

After creating the doc vectors K-Means clustering was used to perform clustering. To determine the value of K i.e. number of clusters, we used elbow method.



As you can see from the above image, the reduce in cost after k > 5 is less as compared to reduce in cost from k=0 to k=5. This is why K = 5 was chosen to perform the clustering of news articles.

Click Stream Data



UserId	SessionID	ArticleID Served	Article Rank	Click	Time Spent (seconds)
1	1	28	1	No	
1	1	66	2	No	
1	1	45	3	Yes	69
1	…	…	…	…	…
1	1	16	10	No	
1	2	36	1	Yes	46
1	…	…	…	…	…

No user information was available and click stream data of the above form was to be generated. Assumptions based on research of user behaviour towards web pages/articles were made during the generation of click stream data. Before we talk about the assumptions lets first have a look at our strategy to create the click stream data.

- Step 1 was to generate number of sessions of the user.
- Step 2 was to select 10 articles which will be recommended.
- Step 3 was to generate number of articles clicked by the user.
- Step 4 was to decide which articles will be clicked.
- Step 5 was to generate time spent on a particular article by the user.

Assumptions we made

- It was assumed that 35% users actually come back to the app after the first session. To generate number of sessions Poisson probability distribution with lambda = 1 was used.
- Two random articles from each of the 5 clusters were selected and recommended to the user. This was done for each session.
- Average number of pages per session were found to be 2.8 [1]. To generate number of clicks we used normal probability distribution with mean=2.8 and std=1. After this we added 0.5 to the number generated and converted it to an integer.
- To find the positions of the article that were clicked we used the data from Chitika Insights [2]. The data shows the percentage of search traffic based on rank of search results on Google. Here it was assumed that user's behaviour towards rank of articles is same as user's behaviour towards rank of search results.
- To calculate time spent on an article our assumptions were based on the data published by Chartbeat o times.com [3]. According to them 29% of users leave the article page within 15 seconds and 71% of the users go on to reading the article. Out of those that go on to read the article, fewer than one-third will read beyond the first one-third of the article. These prior probabilities were used to first select whether the user will stay for more than 15 seconds or not and then if he/she will stay then will they read more than one third of the article or not. After this we used average speed of the user (200 – 250 words/minute) and length of the article to determine how much time the user will spend on the article.

Using these assumptions, we created data for 50,000 users.

Generating Recommendations

Our recommendation strategy depends on the number of sessions of the user.

- For the first session we are selecting two random articles from each cluster and then shuffling them before recommending to the user.
- For the next few sessions (until session number 3) we are selecting two random unseen articles from each cluster.
- For the next sessions we are using user-based collaborating filtering method to generate recommendations. We're using the adjusted cosine similarity formula and adjusted time prediction formula to predict the time that a user will spend on an unseen article. After this we are selecting 7 articles on which the user will spend the most time and for the rest 3 articles we're basically selecting randomly from the unseen articles.

Future Work

- After observing the given articles dataset, it can be seen that articles are mostly from these 5 categories – Sports, Tech, Bollywood/Entertainment, Politics, Crime/Other. We had hoped that our clustering technique would separate these articles into different clusters. But it wasn't the case. Articles belonging to Sport, Entertainment and Politics section were mixing with each other. In future we would like to improve clustering of the articles.
- We would like to implement Matrix Factorization technique and see how does that affect our recommendations.
- To see if we can introduce a negative bias to handle the case of articles that were recommended to the user but the user did not consume it.

References

1. https://www.littledata.io/average/pages-per-session-(all-devices)
2. https://www.searchenginewatch.com/2013/06/20/no-1-position-in-google-gets-33-of-search-traffic-study/
3. https://time.com/12933/what-you-think-you-know-about-the-web-is-wrong/