Natural Language Processing

Recommender System



Submitted by

Mohana Sai Pavan Kumar-18002032C203

P. Virat Chowdary -1800327C203

N. Ragvandhara Nag Varma1800241C203

S. Akhil Ram – 1800262C203

Submitted to

Dr. Sarabjot Singh

Professor in

BML Munjal University

Objective:

                To build a news recommender system

Abstract:

                The world is getting digitalizing and that creates so much data (products, articles, or any other objects) and opportunities. It is hard to find what we want. This is the place where Recommender systems are useful. It sorts the items in a manner that based on one or more user’s interaction and interests typically depends upon the type of recommender system. In this project, we build we used hybrid recommender systems in combination with content-based and memory-based collaborative filtering. In memory-based, we created manually a dataset that consists of the user interactions on the news articles. We also used the popularity model.

Modules used:

Python, Flask, HTML & CSS

Tensor flow

Pandas, seaborn and NumPy

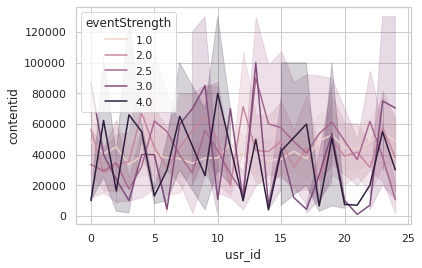
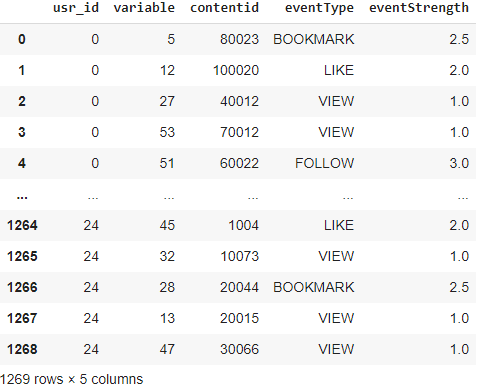
 Implementation: what we did in the project: -

  Dataset:

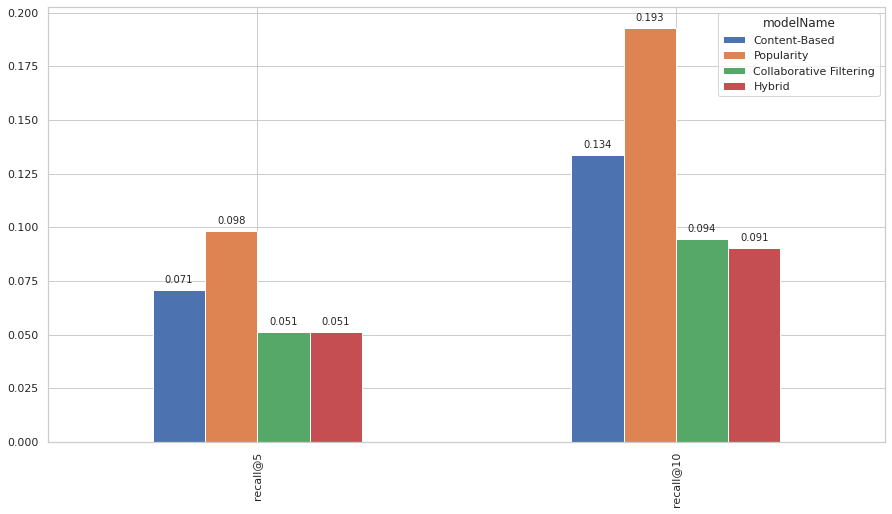
                               In our project there are two datasets. one primary, which consists of a news article data that is scrapped from various internet news websites and another secondary dataset that consists of user interaction and ratings or reviews (categorical). At first, we thought of summarizing the news article text data and we did nicely but luckily, we scraped the data from in Inshortly and labelled it as description. To avoid the cold start problem, we manually created a secondary user interaction dataset that represents what the articles read by user and we made a categorical label called event strength which indicates one value float value for each categorical value as in the below.

 Recommender system:

                                There is a column labelled as ‘contentid' which is common in both the datasets and relates to each other. We created a user rating system. Here we manually given some rating based on the view, like, bookmark or comment label to each article from the user who interacted with it and then we aggregate all the interactions the user has performed in an article by a weighted sum of interaction type strength and apply a log transformation to smooth the distribution. Like this we normalize the rating so that we can minimize the bias.

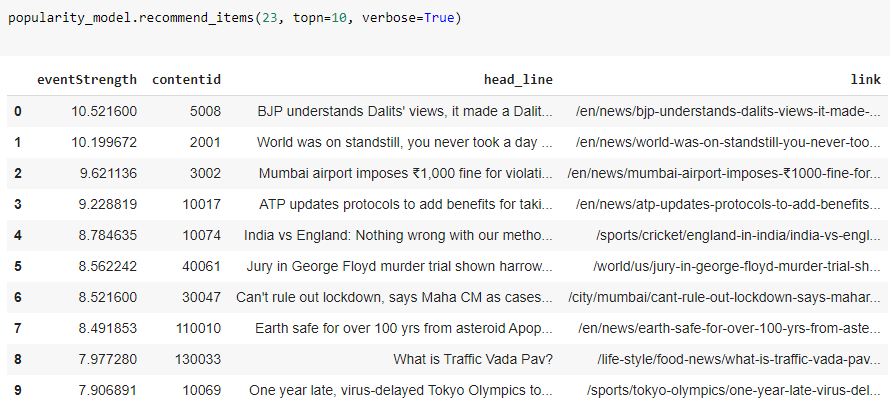


                                In our project you can see four types of recommender systems. There are popularity, content-based, collaborative filtering, and hybrid models. We will discuss briefly as we go on through this report. The main objective of a recommender system is to leverage the long-tail items to the users with very specific interests, which goes far beyond this simple technique.



Popularity based recommender systems:

The popularity model ignores the user profile rather than the user interactions. This model is not actually personalized. it simply recommends to a user the most popular items that the user has not previously consumed. In our recommender system interface, we have two user interfaces. One does not require to sign in and another require signing in. for non-login users (guests) we implemented the popularity model since the guest has no profile so based on ratings, we measured the popularity of each article (even strength) and made recommendations in the order of highest event strength.



Content-based model:

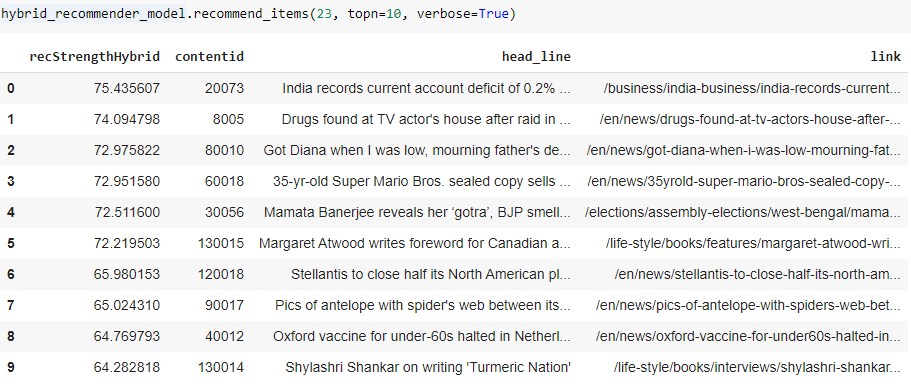
                This model mainly focuses on the user profile and based on interactions and profile (history and choices) it recommends the user but it's not much use when the user data is low. The user interactions we created are robust to avoid the cold start problem. it is simple to use the raw text to build item profiles and user profiles. Here we are using a very popular technique in information retrieval (search engines) named [TF-IDF](https://en.wikipedia.org/wiki/Tf%E2%80%93idf). This technique converts unstructured text into a vector structure, where each word is represented by a position in the vector, and the value measures how relevant a given word is for an article. As all items will be represented in the same [Vector Space Model](https://en.wikipedia.org/wiki/Vector_space_model) is to compute the similarity between articles. To model the user profile, we take all the articles the user has interacted with and average them. The average is weighted by the interaction strength, in other words, the articles the user has interacted with the most (eg. liked or commented) will have a higher strength in the final user profile.

Collaborative filtering model:

                There are two types, one model-based and another are memory-based. Here we used a memory-based collaborative model. This uses the data of previous users’ interactions to compute users’ similarities based on items they've interacted with (user-based approach) or compute items similarities based on the users that have interacted with them (item-based approach). And matrix factorization we used. One advantage of using this approach is that it reduces the high dimensionality to lower since those consist of many missing numbers where the user does not have interacted. We used the sparse matrix to reduce its dimensionality and then after the factorization, we try to reconstruct the original matrix by multiplying its factors. The resulting matrix is not sparse anymore. It was generated predictions for items the user has not yet interacted with, which we will exploit for recommendations.

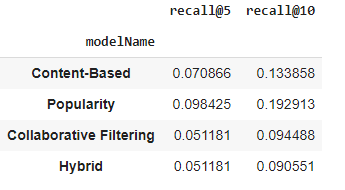
Hybrid recommender systems:

            Here we are summoning both the content-based and collaborative filtering pros and build a hybrid model. We are combing both the user profile based (from content-based) and item-based similarities (from collaborative filtering)



Evaluation:

                        For evaluation we are using recall as the evaluator to evaluate the user-article interaction and user-user similarity. The recall is used to measure the no. of hits that an article is a probability that how much an article is likely to be recommended. We have evaluated for each recommender model with recall that measure which model is better. We chose to work with Top-N accuracy metrics, which evaluates the accuracy of the top recommendations provided to a user, comparing to the items the user has interacted with within the test set.



Interface:

            Here is the interface wex created with flask.

Graphical user interface, text, application

Description automatically generated

Non-login recommendations

Graphical user interface, text, application, email

Description automatically generated

 After log in recommendations