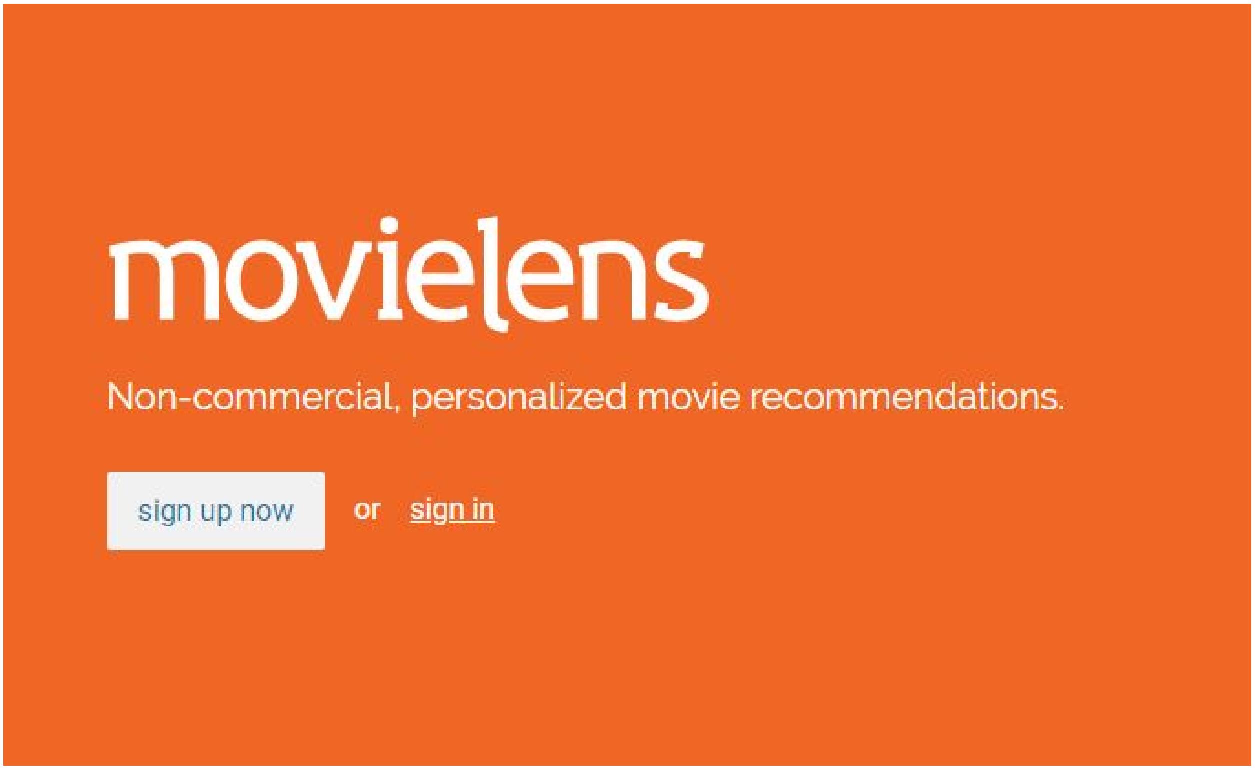


Movie Recommender Systems

05.16.2019

Dhaval Patel
COMSC 415: Machine Learning
Professor Cates

Overview

In this assignment I will create a recommender system for movies and use it to make movie recommendations for yourself and your friends using the MovieLens database.

Goals

1. Create a Content-Based Recommendation system.
2. Create a Collaborative Filtering recommendation System.

Discussion/Methods

1. For the content based recommendation system, I decided to start by researching similarity measures and ways to vectorize the tags give. In the process however, I realized that the tags data wasn't complete and it would make for an incomplete recommendation system. My goal was to combine the tags and genres for every movie as to create keywords to use for the recommender.

   During research, I read about the different similarity measures and decided to try out the ones I thought would be the most relevant for the task. I chose Cosine Similarity, Euclidean Distances and Manhattan Distances. The euclidean distances are good for dense matrices and because of this, I used a count-vectorizer and converted it to a dense matrix. I also used TF-IDF vectors for term-weighting but it did not give the best results for movies with higher amount of keywords. However, it performed better in movies without tags and just genres. This is most likely because of the inverse frequency weighting that TFIDF does. For movies with tags and genres, the count vectorizer performed slightly better.

   The similarity measure I used for content-based recommendations was cosine similarity as it performed better overall. This is because magnitude is not a big factor in the case of keywords. I also decided to use the tfidf vectors instead of count vectors because cosine similarity performs better with vectors bound by [0,1].

   Movies recommended by my chosen method were very good according to me. As seen in the test runs for the content-based recommendations below, the movies recommended for ‘Batman Forever’ were other superhero movies. The results therefore are reasonable and I would enjoy all the movies recommended to me here.

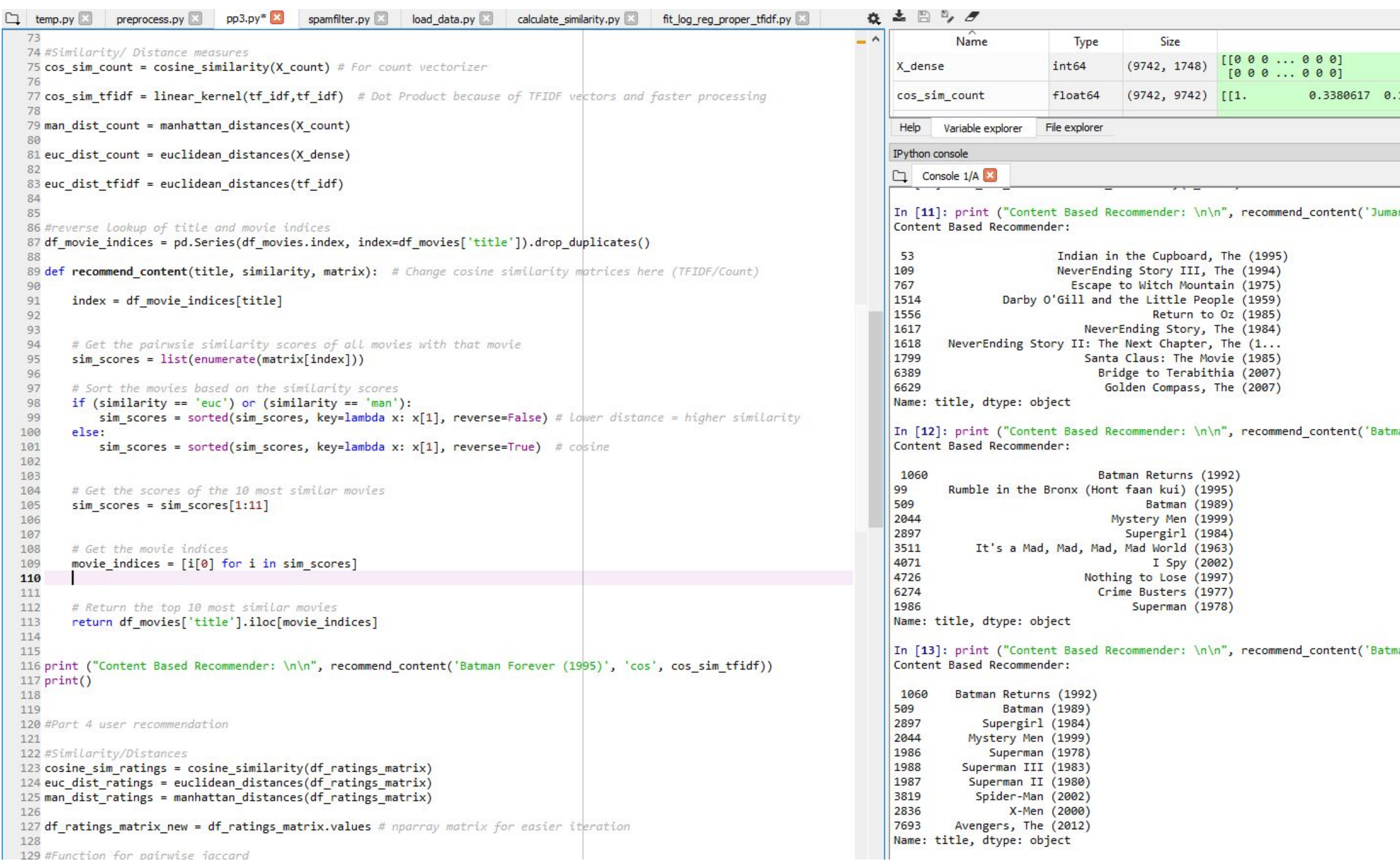
2. Collaborative Filtering requires similarity measures of user ratings. The ratings matrix created was used as a starting point and Jaccard Distances was tested an additional similarity measure. The reason I tried Jaccard similarity in addition to the other measures is initially I thought that similar users would have common movies rated. However, I gave it more thought after tests were not giving me good results. The reason I think this similarity measure doesn’t work is because it doesn’t account for the ratings themselves and rather just considers what movies the users had rated in common.

   The measure I decided to use was euclidean because euclidean distances take into account the magnitude and therefore the ratings themselves. For the purposes of the task of finding similar users, this method worked the best. After getting the most similar user, the sorted list of movies rated by the user was presented as recommendations.

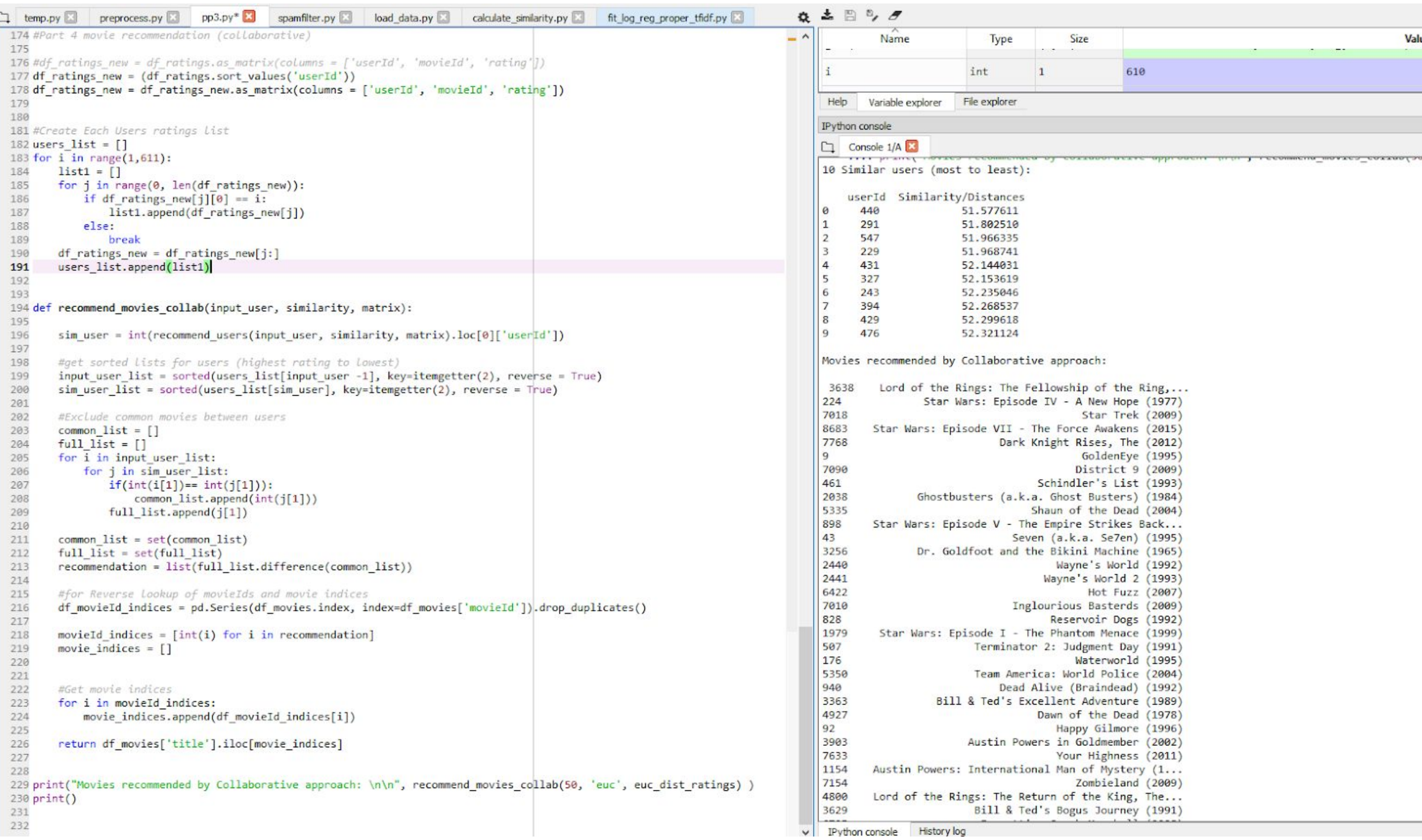
Overall, the recommender systems performed well. Anecdotally, the movies recommended make sense knowing how the recommendations were gathered. For evaluation of the models I think looking at other more complex systems like the movielens website is a good way to judge if there are similarities between the system recommendations. Another good way to judge the recommender system is to use recall and precision values. In general, recall would be what the ratio of movies that a user likes were recommended and precision would be how many movies the user liked out of all the movies recommended. We would want to maximize both of these in order to get the best recommender system we can.

Results/Test Runs

I. Content-Based Recommendations Output



II. Collaborative Filtering Recommendations Output



Sources

Andersen, Kagemand AndersenKagemand. “Compute Jaccard Distances on Sparse Matrix.” *Stack Overflow*, stackoverflow.com/questions/32805916/compute-jaccard-distances-on-sparse-matrix.

“Building A Recommender System on User-User Collaborative Filtering (MovieLens Dataset).” *Code for Thought*, 29 Dec. 2016, acodeforthought.wordpress.com/2016/12/29/building-a-recommender-system-on-user-user-collaborative-filtering-movielens-dataset/.

Jain, Aarshay. “Quick Guide to Build a Recommendation Engine in Python.” *Analytics Vidhya*, 22 Feb. 2019, www.analyticsvidhya.com/blog/2016/06/quick-guide-build-recommendation-engine-python/.

“Recommender Systems in Python Tutorial.” *DataCamp Community*, www.datacamp.com/community/tutorials/recommender-systems-python.

“Recommender Systems through Collaborative Filtering.” *Data Science Blog by Domino*, blog.dominodatalab.com/recommender-systems-collaborative-filtering/.