Recommender Systems White Paper

Personalized Food Recommendations Article:

<https://pdfs.semanticscholar.org/0b1e/4850bf48699fed7b2b3ad970f211d711a1da.pdf>

Being focused more on rating-based recommendations, the system uses **Collaborative filtering**and**Content-based algorithms**.

**Features that compose a recipe are**: category, region, restaurant ID and ingredients. Context features are also considered in the moment of the recommendation, these are: temperature, period of the day, season of the year, and meal’s cost. Each feature has a specific location attributed to it in the recipe and user profile sparse vectors.

The decomposition of recipes into ingredients implemented in this experiment is simplistic: ingredient scores were computed by averaging the ratings of recipes in which they occur.

**Recommendations** are generated by comparing the restaurant’s recipes’ features with the user profile using the cosine similarity measure. The recommended recipes are ordered from most to least similar. In this case instead of referring recipes as vectors of words, recipes are represented by vectors of different features.

**Rocchio’s Algorithm** is widely used relevance feedback method that operates in the vector space model. It uses feature weights to build the prototype vectors, representing the user’s preferences. The weight attributed can be computed using the TF-IDF (Term Frequency-Inverse Document Frequency) scheme. Using relevance feedback, recipes’ feature vectors of positive and negative examples are combined into a prototype vector for each class *c*. These prototype vectors represent the learning process in this algorithm. New recipes’ features are then classified according to the similarity between the prototype vector of each class and the corresponding user’s profile vector, using for example the well-known cosine similarity metric. The algorithm returns a similarity value between the recipe features vector and the user profile vector.

**Summary:**

Recipes are broken down to a number of features with relative weights (determined by TF-IDF) and are used to build a prototype vector. Then, the prototype vector is compared to corresponding user’s profile vector to return a similarity value and suggest a new recipe.

The idea of considering ingredients in a recipe as similar to words in a document lead to the variation of TF-IDF weights. This work presented good results in retrieving the user’s favorite ingredients.

06.24.2020

Algorithms we consider implementing on datasets below:

Cosine Similarity – each attribute is a dimension. If an entity contains that attribute, it gets 1 in its corresponding dimension.

K-Nearest Neighbors – Takes K movies with highest similarity scores to a movie, then recommends them based on their rating.​

Mise-en-scene – uses data from scenes in a movie as an attribute​

Datasets from Kaggle we could consider working with:

<https://www.kaggle.com/datafiniti/fast-food-restaurants> (all fast food restaurants across US)

<https://www.kaggle.com/damienbeneschi/krakow-ta-restaurans-data-raw> (TripAdviser’s European restaurants EXTENDED to 31 cities updated 2years ago)

<https://www.kaggle.com/makingtheworldbetter/tripadviser-restaurants> (same but updated 3months ago)

06.29.2020

Summary of Meeting on July 30

We need to get as much work done by August 24 as possible, because we will be much busier during the semester. Additionally, there are going to be conferences between August and November where we will be able to present our research.

Good coordination and communication within a team is the key to doing research.

We are researching recommender systems. More specifically, we are researching how to recommend food based on a person’s preferences. There are different types of food that different people enjoy. For example, there is Italian food, sushi, Mexican food, spicy food, etc. We are interested in how people make decisions in a restaurant more so than in a grocery store.

To accomplish this there are two steps: First, we will study recommender systems. Second, we will create tests/experiments to learn something new.

We are still learning how recommender systems work. Once we have a solid understanding, we will gather data and conduct experiments.

Work Completed/Notes between July 30 and August6

Recommendation domain:

Our domain is recommending food in a restaurant setting​. The first step is to recommend a restaurant. The second step is to recommend food within a restaurant.

Restaurant recommendations​:

The first step is to recommend restaurants​. In order to recommend restaurants that users will like, we need to predict what they will like, based on user tastes and past purchases ​. Using data/features about the restaurant in combination with features about the user is the best way to recommend a restaurant​. In our research, we have created a list of features of restaurants​. We believe this is a good first step in building a recommender system

Food Recommendations​:

The next step is to recommend food within a restaurant. Danila found an extensive dataset of food recipes that we will use to gather attributes on food.

Drawback:

We can find data on restaurants, but we cannot find data on users. We do not know how we will run experiments without user data​.

Summary of Meeting on August 6

Our domain is still wide. We need a more specific domain to work on. Dr. Farooque will talk to another professor about a narrower domain and will get back to us by Friday night.

We are on the right track, but we need to be more specific right now.

Our TripAdvisor data set is good, but is focused on Europe, not the U.S.

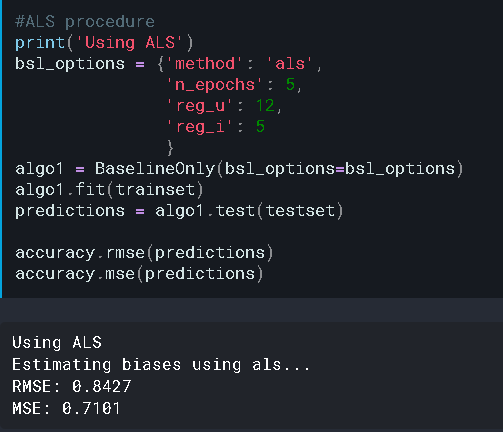
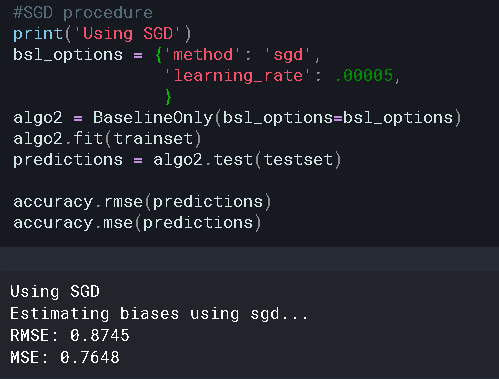
Updates for August 21

Since our last meeting on August 6, we decided that we should start testing algorithms that are used for recommender systems. These algorithms require data, so we had to find a dataset. We decided to use the MovieLens dataset. This contains clean data that is ready to be used in machine learning algorithms. So far we have done exploratory data analysis on the dataset. Our next step is to actually test the algorithms. I decided to test Matrix Factorization algorithms and Danila wants to test K-Nearest Neighbors algorithms.

Updates for August 30

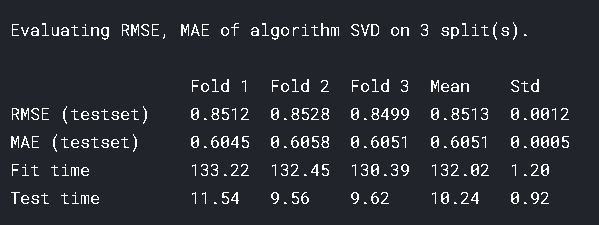
The following week, we were focused on learning how to execute main features of Surprise library (<http://surpriselib.com/>) and use its resources for our tests on the datasets.

Just as we planned to, we first split the data into trainset and tests and tried running two general baselines provided by Surprise (using ALC and SGD) our dataset and compute their accuracies.

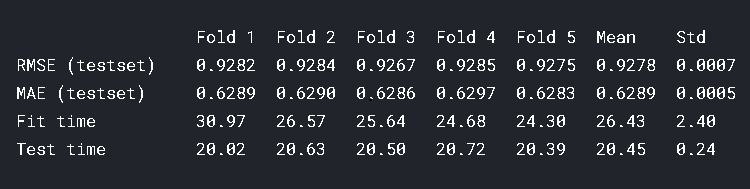
 

Then, we ran cross validation on both KNN and Matrix Factorization algorithms in order to understand the basic principle of Surprise’s prediction algorithms and compare their accuracies on our dataset (RMSE, MAE).

Matrix Factorization: cross validation (3 splits)

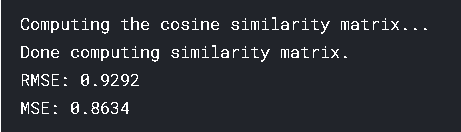


KNNBasic: item-similarity, cross validation (5 splits)

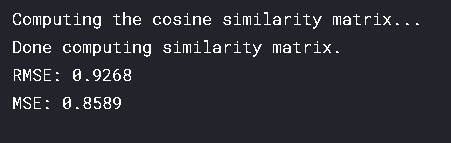


KNNBasic: item-similarity, baselines

SGD:



ASL:

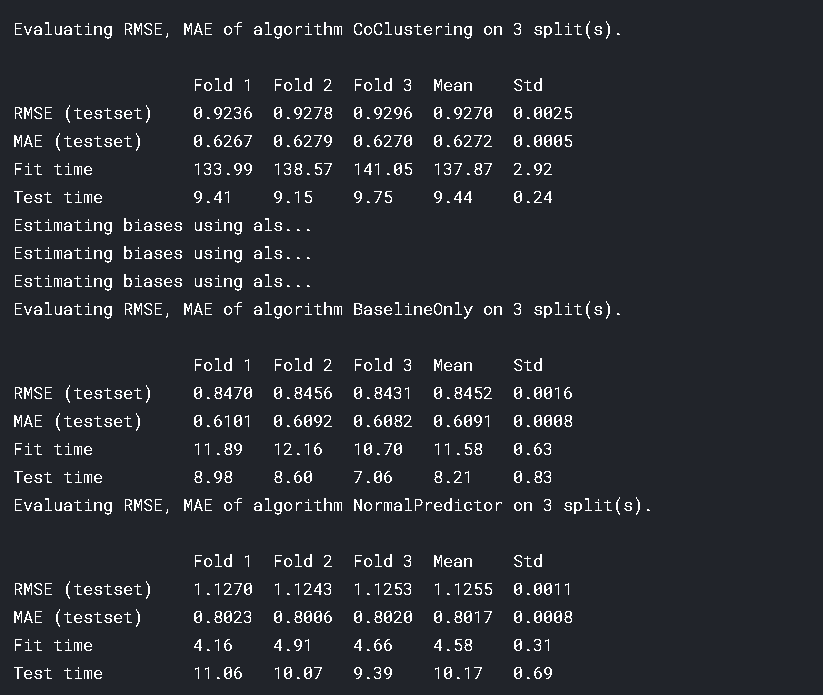


Summary: RMSE and MAE for KNN and SVD (matrix factorization) doesn’t have significant differences and will probably not affect the quality of recommendations.

Updates for September 6:

We ran cross validation on the rest of Surprise algorithms: CoClustering, BaselineOnly, NormalPredictor

Result:



The next is step is trying out GridSearch by Surprise for parameter tuning:

This is the result of GridSearchCV on the SVD algorithm

0.8483719783924384

{'n\_epochs': 10, 'lr\_all': 0.005, 'reg\_all': 0.4}