CSE158 – Assignment 1 Report
Mao Li
Professor: Julian McAuley
Date: Nov 20th, 2018

Assignment One Report

TASK 1 - Purchase Prediction

- **Objective**: given reviewerID, itemID and a training data set from Amazon, predict whether user(reviewerID) would purchase the item(itemID).

- **Idea:**

    - Categories Similarity: Users would purchase items that are similar to their interests/preference. Although there exists no direct data representing their preference, we can approximate what they like based on their purchase history. In particular, we can record all categories a user bought before and check which categories the user is more likely to purchase. If the item is from these categories, we should predict true.

    - Item Popularity: Popular items have higher chance to be bought by any user. Therefore, we can predict true if an item is popular, regardless the properties of users. This case this especially useful if we haven't seen the user.

- **Model Description**:

    - Rule: Combining above two ideas, this model predicts true if either an item and a user has similar categories (same category id, and similar inner list of categories), or an item is very popular based on training data. Otherwise predict false.

    - Implementation:

        - Split the given 200,000 data into **50% of train set** and **50% of validation set**. Then random sample the data to get another **100,000 non-purchased user, item pairs**; append these pairs to the validation set.

        - From training data, for every user/item I record their **most specific and descriptive categories**. These categories are the last categories of inner lists in 'categories' field. Also record all sub-categories users purchased and items belonged to.

- From training data, for every item record into a dictionary the times it has been purchased. Then sort these items by their purchase counts to get a **list of top popular items**.

- On the validation set, for each user-item pair, first check if item's **categoryIDs** (can have multiple) **exactly match** user's purchased categoryIDs. If not, predict false. Second, check if user's list of most specific categories has **intersection** with that of item. Also compute if user-item's categories' **cosine similarity** (please kindly refer to code for specific implementation) is over a certain similarity threshold. If so, predict true. Third, for unseen users check if the i**tem is popular** over a certain popularity threshold. If so, predict true. Otherwise, predict false.

- Based on validation performance, choose popularity threshold **= 0.45**, cosine similarity threshold = **0.75**. I notice the intersection of item and user's most similarity categories list is a better indicator than cosine similarity, so I only use intersection in final submission. (Task 1's Approach 2 in codes)

- **Alternative Models**

  - Other than above model which I submitted as final version, I also considered other approaches, but they didn't outperform current model.

  - SVC alternative: make a feature vector as [item's popularity, user-item's cosine similarity, user-item's categories intersection], and train with LinearSVC to have a binary classifier. Please kindly refer to Task 1's Approach 4 in code section.

  - Cosine Similarity alternative: When predicting, incorporate cosine similarity score with intersection and popularity. Using if statements to check if it's over a threshold. Please kindly refer to Task 1's Approach 3 in code section.

- **Model Performance on Kaggle**
  - Submission model trained on **entire given data**.
  - Rank: top **12%**, 93rd of 816
  - Public leaderboard Accuracy: **0.69842,** Private leaderboard Accuracy: **0.70500**

CSE158 – Assignment 1 Report
Mao Li
Professor: Julian McAuley
Date: Nov 20th, 2018

TASK 2 – Category Prediction

- **Objective**: Train model from given data. Then given test_Categories.json (everything included with category information removed), predict the category of an item from review.

- **Idea:**

    - Review text information: When users are writing reviews for Items from different categories, they may use some category specific words, such as 'bra' for women and 'shirts' for men. Extracting this information will be helpful to determining the item's category.

    - User's purchased categories: For a given review, the user may have purchased different categories of items before. Based on user's purchase history, we can infer which categories the user is likely to purchase/review. The item in given review is likely to be one of these categories.

- **Model Description:**

    - General: Design a length 5005 feature vector for each review to have two parts. First part is length 5000 one hot encoding representing whether review text has training data's **5000 most frequent words**. Second part is length 5 vector representing the user's **purchase counts for each category**. Then train five binary classifiers each for one category, with Linear SVM, and combine them to have final composite classifier (similar method used in HW3, question 8).

    - Implementation:

        - **Random sample 10%** of the given data to construct training and validation sets.

        - From the training data, get **top 5000 most frequent words** by counting the frequency of each words and sort by frequency. Also store into a dictionary a length 5 list of users' purchase counts, each index for a

category. In other words, get **each user's purchase counts for each of five categories**.

- For training data, get each review's feature vectors [length 5005] as described above. Then for each category, train a **binary classifier** to distinguish this category with others. Training is done with Scikit-Learn's LinearSVC.

- **Combine the five classifiers** to together to have a composite classifier. This composite classifier classifies each point to the category where this category's binary classifier gives **highest decision function score**.

- Then evaluate the composite classifier on the validation set to tune the parameter C and choose max iteration for LinearSVC. Based on the performance, I chose **C = 0.1** and **max_iter = 10000**. This gives highest accuracy, ensures LinearSVC converges as much as possible and finishes running in relatively 'short' time.

- **Alternative Models**

  - To come up with above final solution, I've experimented with many other models, but they haven't outperformed this ultimate one. Below are two comparatively good ones.

  - Alternative 1: Length = 10000 Text Feature. Instead of having 500 top words as in hw3 question 8, I design a feature that has 10000 top words. Yet, since this one hot encoding is very sparse and the text feature isn't descriptive, the run time is high, and performance is only around 0.835.

  - Alternative 2: Various features combined. In this feature I append many different features to the top 5000 one-hot encoding. They include the average rating user gave for each category, one-hot encoding for user's purchased categories, and as well as count the top words in review text. Yet, since many features aren't related to categories, this model's performance is still around 0.84. Please kindly refer to old_feature function in Task2's Approach 2 code section

CSE158 – Assignment 1 Report
Mao Li
Professor: Julian McAuley
Date: Nov 20th, 2018

- **Model Performance on Kaggle**
  - Submission model trained on **entire given data**.
  - Rank: top **2%**, 5th of 355
  - Public leaderboard Accuracy: **0.88857,** Private leaderboard Accuracy: **0.88871**

Kaggle Details

Full name: Mao Li
Email: mal131@ucsd.edu (Also used for Gradescope)