Northeastern University, Khoury College of Computer Science

CS 6220 Data Mining — Assignment 1

Due: January 18, 2023(100 points)

YOUR NAME + LDAP

link-to-your-repository

Prerequisite Review

In subsequent lectures, you'll learn about frequent item sets, where relationships between items are learned by observing how often they co-occur in a set of data. This information is useful for making recommendations in a rule based manner. Before looking at frequent item sets, it is worth understanding the space of all possible sets and get a sense for how quickly the number of sets with unique items grows.

Suppose that we've received only a hundred records of items bought by customers at a market. Each line in the file represents the items an individual customer bought, i.e. their basket. For example, consider the following rows.

```
ham, cheese, bread
dates, bananas
celery, chocolate bars
```

Customer 1 has a basket of ham, cheese, and bread. Customer 2 has a basket of dates and bananas. Customer 3 has a basket of celery and chocolate bars. Each of these records is the receipt of a given customer, identifying what they bought.

Please answer the following:

1. The *cardinality* of a set or collection of items is the number of unique items in that collection. Write a function called `cardinality_items` that takes a `.csv` text string file as input, where the format is as the above, and calculates the cardinality of the set of all the grocery items in any given dataset. What is the cardinality in "basket_data.csv"?

2. Taking any `.csv` file as a sample of a larger dataset, we'd occasionally like to understand the space of all possible subsets comprised of unique items. If there are N unique items

(i.e., the cardinality of the entire dataset is N), how many sets with unique items can there possibly be? (Ignore the null set.) NOTE: I only expect the formula, and there is no code associated with this question.

3. Write a module called `all_itemsets` that takes a `.csv` text string file as input, where the format is as the above, and the output is a list of all possible unique sets with non-repeating N items. That is, the output is $L = [S_1, S_2, \cdots S_M]$, a list of all possible sets of N unique items.

4. Let's take the small sample `.csv` provided as reflective of the distribution of the receipts writ large. For example, if the set $S = \{\text{bread}, \text{oatmeal}\}$ occurs twice in a dataset with 100 records, then the probability of item set $\{\text{bread}, \text{oatmeal}\}$ occurring is 0.02. Likewise in another example, if $S =$`["dates", "bananas"]` and the `data_file.csv` looks like :

```
ham, cheese, bread
dates, bananas
celery, chocolate bars
```

then `prob_S("data_file.csv", S)` should yield 0.3333 since this set occurs a third of the time in the data file.

Write a module called `prob_S` that takes a `.csv` text string file as input and an item set (e.g., $S =$"bread, oatmeal") and outputs the probability of seeing S.

Examining Our First Dataset

One of the most famous challenges in data science and machine learning is Netflix's Grand Prize Challenge, where Netflix held an open competition for the best algorithm to predict user ratings for films. The grand prize was $1,000,000 and was won by BellKor's Pragmatic Chaos team. This is the dataset that was used in that competition.

- https://www.kaggle.com/datasets/netflix-inc/netflix-prize-data

In this exercise, we're going to do a bit of exploring in the Netflix Data. Start by downloading the data. Data integrity tends to be a problem in large scale processing, especially if there is little to no support. Therefore, it's important to verify the quality of the file download. If all worked out well, you should have the following files:

- combined_data_1.txt
- combined_data_2.txt
- combined_data_3.txt
- combined_data_4.txt
- movie_titles.csv
- probe.txt
- qualifying.txt
- README

Data Verification and Analysis

A large part of machine learning and data science is about getting data in the right format and ensuring that there is no data corruption. Verify that the schema is the same as the Kaggle Dataset's description, read in the data (hint: some movies have commas in them), and then answer the following questions.

5. Let's look at the aggregate counts of data.
   a) How many total records of movie ratings are there in the entire dataset (over all of `combined_data_*.txt`)?
   b) How many total users are there in the entire dataset (over all of `combined_data_*.txt`)?
   c) How many movies with unique names are there?
   d) What is the movie name that has the most number of movie IDs of the same name?
6. We can bin the movies by year to get a better understanding of the overall trend of the site use and movie availability as well as user sentiments.
   a) What is the range of years that this data is valid over?
   b) Year over year, what is the trend of the number of ratings? Plot the number of movie ratings per year (aggregated over all users). Any guesses as to why this is the case?
   c) Year over year, what is the average movie rating? Plot the average star rating over time (aggregated over all users in that year).
7. It's useful to understand the user population sometimes.
   a) What is the average number of ratings for a user?
   b) Which user rated the most movies? How many movies did they rate?
   c) How many users rated exactly 200 movies? Of these users, take the one with the user ID that is the lowest and print out the names of the movies that they liked the most (5 star ratings).
   d) How many users rated exactly 200 movies? Of these users, take the one with the user ID that is the lowest and print out the names of the movies that they liked the least (1 star ratings).
8. Many massive data processing pipelines typically read data in line by line, where all the information about a record is stored in a single line. Write a function that changes takes data from `combined_data_*.txt` and joins it with `movie_names.csv` to create a file called: `movie_user_ratings_*.txt`, where all movie and user rating information is stored on a single line. The file `movie_user_ratings.txt` should have the format:
   - movie id, user id, star-rating, movie name, date

Grading Criterion

A significant portion of the grading rubric is the presentation of your report. We'll review:

- the answers to questions.
- your code and its legibility

- the clarity of your write-up, including
    1. pipeline and code decisions,
    2. perspectives on the solution,
    3. and algorithmic rationale.