Northeastern University, Khoury College of Computer Science

CS 6220 Data Mining — Assignment 1

Due: September 16, 2024 (100 points)

YOUR NAME + LDAP

Coding 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. Take a look at data: https://course.ccs.neu.edu/cs6220/fall2024/homework-1/basket_data.csv. 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.

```
def cardinality_items( filename ):
    '''
```

```
    Takes a filename "*.csv" and returns an
    integer
    '''
    < YOUR CODE HERE >
    return 0
```

What is the cardinality in "basket_data.csv"?

2. Write a function called `all_itemsets` that takes a list of unique items and an integer k as input, and the output is a list of all possible unique itemsets with non-repeating k items. That is, the output is $L = [S_1, S_2, \cdots S_N]$, a list of all possible sets, where each S_i has k items.

   For example,

   ```
   all_itemsets( ["ham", "cheese", "bread"], 2 )
   ```

   should result in:

   ```
   [ ["ham", "cheese"], ["ham", "bread"], ['cheese", "bread"] ]
   ```

   Please don't use any library functions as you will not need them. Your function signature should look like this:

   ```
   def all_itemsets( unique_items ):
       '''
       Enumerate out all itemsets. Note that for each itemset,
       order does not matter.

       Input:
       - unique_items: a list of items (should be unique)

       Output:
       - all_itemsets: a list of lists: represents all possible
       baskets of items.
       '''
       < YOUR CODE HERE >
       return 0
   ```

   Output your data into `output.txt` (which will be uploaded via Gradescope).

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. The original dataset that was used in that competition was downloaded from Kaggle (netflix-inc/netflix-prize-data), and you can download it here:

- https://course.ccs.neu.edu/cs6220/fall2024/homework-1/netflix-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.

Please answer the following:

3. Let's review `combined_data_*.txt`.
   a) How many total records of movie ratings are there in the entire dataset (over all of `combined_data_*.txt`)?
   b) How many total unique users are there in the entire dataset (over all of `combined_data_*.txt`)?
   c) What is the range of years that this data is valid over?
4. Let's review `movie_titles.csv`.
   a) How many movies with unique names are there? That is to say, count the distinct names of the movies.
   b) How many movie names refer to four different movies?
5. Let's review both.
   a) How many users rated exactly 200 movies?
   b) Of these users, take the lowest user ID and print out the names of the movies that this person liked the most (all 5 star ratings).

Submission Instructions

Add all functions to `homework1.py`. Submit via Gradescope. There, you will upload `output.txt`, PDF, and Python code.