CS-6620 Assignment 1

Status	In progress
Priority	High
Tags	CS 6220 Data Mining
Date	@January 15, 2024

1. Github link: https://github.com/Hui-Hwoo/Data-Mining/tree/main/Assignment/assignment_1

assignment-1.pdf

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. 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) -> int:
    """
    Takes a filename "*.csv" and returns an integer
    """
    # check file extension and existence
    if filename[-4:] != ".csv":
        print("Invalid file extension: ", filename[-4:])
        return 0
    try:
        hashset = set()
        with open(filename) as f:
            data = f.read()

        # read file line by line and split by comma
        for line in data.split("\n"):
            for item in line.split(","):
                # add item to hashset
                hashset.add(item.strip())

        # return the cardinality of the hashset
        return len(hashset)
    except FileNotFoundError:
        print("File not found:", filename)
        return 0
```

   What is the cardinality in “basket data.csv”?

   basket_data.csv

2. Write a function called `all_itemsets` that takes a list of unique items and an integer N as input, and the output is a list of all possible unique item sets with non-repeating N items. That is, the

output is L = [$S_1, S_2, \cdots S_N$], a list of all possible sets of N unique items.

For example,

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

should result in:

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

You should not need any library functions.

```
def all_itemsets(items, N) -> list[str]:
    """
    Takes a list of items and returns a list of all possible itemset
    """
    if N == 0:
        return [[]]
    if len(items) == 0:
        return []
    # get first item
    first = items[0]
    # get all itemsets of size k-1
    itemsets = all_itemsets(items[1:], N - 1)
    # add first to all itemsets
    for itemset in itemsets:
        itemset.insert(0, first)
    # get all itemsets of size k
    itemsets.extend(all_itemsets(items[1:], N))
    return itemsets
```

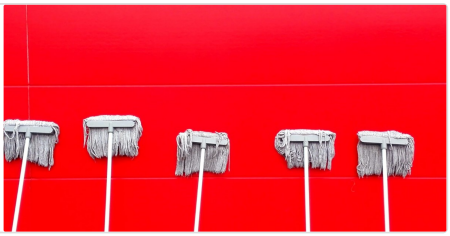
Examining Our First Dataset

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.

Netflix Prize data

Dataset from Netflix's competition to improve their reccommendation algorithm

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.

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)

       100480507

    b. How many total unique users are there in the entire dataset (over all of combined_data_*.txt)?

       480189

c. What is the range of years that this data is valid over?

1999-11-11 to 2005-12-31

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.

   17297

   b. How many movie names refer to four different movies?

   7

5. Let's review both.

   a. How many users rated exactly 200 movies?

   605

   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).

   Lowest user ID: 1001192

   - **Sex and the City: Season 4**
   - **Ghost**
   - **Steel Magnolias**
   - **Pure Country**
   - **Finding Nemo (Full-screen)**

Submission Instructions

Commit all your code and materials to your repository, and submit via Gradescope. There, you will upload your PDF and Python code.