Michael Woo
28 March 2021
Professor Yasser Abduallah
CS 634 104 Data Mining

Midterm Project Report

Implementation & Code Usage

The main concept of the Apriori Algorithm is to create associations. In order to create associations, I had to figure out what items were most frequent when given the list of transactions. Once the items that are most frequent were found, depending on the user's support parameter, the support would have to be calculated for each item. After calculating the support value for each item, we can eliminate the items that do not meet the user-defined support parameter. Next, I can then generate set combinations and calculate the support for those item-sets by cross-referencing them to the list of transactions and examine if these item-sets existing in the transactions list.

We would repeat this process and increase the number of items in the set by one until I no longer meet the user-defined support parameter. Once I have the item-sets that meet the support then I would have to generate the permutations and calculate the confidence with each item set.

When having the support and confidence of each item-set we can then filter the item-sets that meet the both user-defined parameters of support and confidence and acquire our final associations.

As for code usage, it was in my best interest to use the notions of sets, counters, data frames, and lambdas, loops, and well-defined functions to carry out specific tasks to achieve the Apriori Algorithm.

Michael Woo
28 March 2021
Professor Yasser Abduallah
CS 634 104 Data Mining

Screenshots

Here are what the csv files (This program takes in two separate csv files: Item Names & Transactions)

amazon_item_names

item_number	item_name
1	A Beginner's Guide
2	Java: The Complete Reference
3	Java For Dummies
4	Android Programming: The Big Nerd Ranch
5	Head First Java 2nd Edition
6	Beginning Programming with Java
7	Java 8 Pocket Guide
8	C++ Programming in Easy Steps
9	Effective Java (2nd Edition)
10	HTML and CSS: Design and Build Websites

Figure 1: Item Names CSV file

amazon_transactions

transaction_id	transaction
1	A Beginner's Guide, Java: The Complete Reference, Java For Dummies, Android Programming: The Big Nerd Ranch
2	A Beginner's Guide, Java: The Complete Reference, Java For Dummies
3	A Beginner's Guide, Java: The Complete Reference, Java For Dummies, Android Programming: The Big Nerd Ranch, Head First Java 2nd Edition
4	Android Programming: The Big Nerd Ranch, Head First Java 2nd Edition , Beginning Programming with Java,
5	Android Programming: The Big Nerd Ranch, Beginning Programming with Java, Java 8 Pocket Guide
6	A Beginner's Guide, Android Programming: The Big Nerd Ranch, Head First Java 2nd Edition
7	A Beginner's Guide, Head First Java 2nd Edition , Beginning Programming with Java
8	Java: The Complete Reference, Java For Dummies, Android Programming: The Big Nerd Ranch,
9	Java For Dummies, Android Programming: The Big Nerd Ranch, Head First Java 2nd Edition , Beginning Programming with Java,
10	Beginning Programming with Java, Java 8 Pocket Guide, C++ Programming in Easy Steps
11	A Beginner's Guide, Java: The Complete Reference, Java For Dummies, Android Programming: The Big Nerd Ranch
12	A Beginner's Guide, Java: The Complete Reference, Java For Dummies, HTML and CSS: Design and Build Websites
13	A Beginner's Guide, Java: The Complete Reference, Java For Dummies, Java 8 Pocket Guide, HTML and CSS: Design and Build Websites
14	Java For Dummies, Android Programming: The Big Nerd Ranch, Head First Java 2nd Edition
15	Java For Dummies, Android Programming: The Big Nerd Ranch
16	A Beginner's Guide, Java: The Complete Reference, Java For Dummies, Android Programming: The Big Nerd Ranch
17	A Beginner's Guide, Java: The Complete Reference, Java For Dummies, Android Programming: The Big Nerd Ranch
18	Head First Java 2nd Edition , Beginning Programming with Java, Java 8 Pocket Guide
19	Android Programming: The Big Nerd Ranch, Head First Java 2nd Edition
20	A Beginner's Guide, Java: The Complete Reference, Java For Dummies

Figure 2: Transactions CSV file

Michael Woo
28 March 2021
Professor Yasser Abduallah
CS 634 104 Data Mining

Below are screenshots of the code from python file:

```
import numpy as np
import pandas as pd
import os
import time
from itertools import permutations, combinations
from IPython.display import display
```

Figure 3: Imports to run the py file

```
def number_to_store(store_number):
    switcher = {
        1: "data/amazon_transactions.csv",
        2: "data/nike_transaction.csv",
        3: "data/best_buy_transaction.csv",
        4: "data/k_mart_transaction.csv",
        5: "data/walmart_transaction.csv"
    }
    return switcher.get(store_number)


def number_to_item_list_of_store(store_number):
    switcher_dict = {
        1: "data/amazon_item_names.csv",
        2: "data/nike_item_names.csv",
        3: "data/best_buy_item_names.csv",
        4: "data/k_mart_item_names.csv",
        5: "data/walmart_item_names.csv"
    }
    return switcher_dict.get(store_number)


def ns(store_number):
    switcher_store = {
        1: "Amazon",
        2: "Nike",
        3: "Best Buy",
        4: "K-Mart",
        5: "Walmart"
    }
    return switcher_store.get(store_number)
```

Figure 4:These are my dictionaries to choose which store to get based in Key-Value Pairs

Michael Woo
28 March 2021
Professor Yasser Abduallah
CS 634 104 Data Mining

```python
def a_priori_read(item_list, transaction, support_percentage, confidence_percentage):
    # Create two different functions one that is solo for read in the file data and the other that is algorithmic with the data
    if support_percentage > 100 or confidence_percentage > 100 or support_percentage < 0 or confidence_percentage < 0:
        print("Support Percent or Confidence Percent is Invalid. \n Enter a valid number between 0 and 100.\n")
        print("Restarting Apriori 2.0.....\n")
        time.sleep(2)
        os.system("python Aprior_Algo")
    if support_percentage >= 0 and support_percentage <= 100 and confidence_percentage >= 0 and confidence_percentage <= 100:
        df_item_list = pd.read_csv(item_list)
        df_transactions = pd.read_csv(transaction)
        print(df_transactions.head())
        print(df_item_list.head())
        trans = np.array(df_transactions["transaction"])
        items_names = np.array(df_item_list["item_name"])
        k_value = 1
        return items_names, trans, support_percentage, confidence_percentage, k_value
```

Figure 5:We first have to read in the csv files and make sure that the inputs received from the user are valid

```python
def ap_1(items_names, trans, support_percentage, confidence_percentage, k_value):
    counter = np.zeros(len(items_names), dtype=int)
    for i in trans:
        i = list((map(str.strip, i.split(','))))
        s1 = set(i)
        nums = 0
        for x in items_names:
            s2 = set()
            s2.add(x)
            if s2.issubset(s1):
                counter[nums] += 1
            nums += 1
    counter = list(map(lambda x: int((x / len(trans)) * 100), counter))
    df3 = pd.DataFrame({"item_name": items_names, "support": counter, "k_val": np.full(len(items_names), k_value)})
    rslt_df = df3[df3['support'] >= support_percentage]
    print("When K = " + str(k_value))
    print(rslt_df)
    items = np.array(rslt_df["item_name"])
    support_count = np.array(rslt_df["support"])
    k_value += 1
    return items, support_count, k_value, rslt_df
```

Figure 6:The first go around of the Apriori Algorithm we find the items that are most frequent when K=1. This is so that we can find the most frequent items given the transactions

Michael Woo
28 March 2021
Professor Yasser Abduallah
CS 634 104 Data Mining

```
def ap_2(item_comb, k_value, trans, support_percentage):
    boo = True
    comb = combinations(item_comb, k_value)
    comb = list(comb)
    counter = np.zeros(len(comb), dtype=int)
    if k_value > 1:
        for i in trans:
            i = list((map(str.strip, i.split(','))))
            s1 = set(i)
            nums = 0
            for x in comb:
                s2 = set()
                x = np.asarray(x)
                for q in x:
                    s2.add(q)
                if s2.issubset(s1):
                    counter[nums] += 1
                nums += 1
    counter = list(map(lambda x: int((x / len(trans)) * 100), counter))
    df3 = pd.DataFrame({"item_name": comb, "support": counter, "k_val": np.full(len(comb), k_value)})

    # Making sure that user parameters are met for support
    rslt_df = df3[df3['support'] >= support_percentage]
    print("When K = " + str(k_value))
    print(rslt_df)
    items = np.array(rslt_df["item_name"])
    supp = np.array(rslt_df["support"])
    if len(items) == 0:
        boo = False
        return rslt_df, boo
    return rslt_df, boo
```

Figure 7: Then we use this function below to find item sets that are most frequent when K > 1

```
frames = []
items_names, trans, support_percent, confidence_percent, k_value = a_priori_read(
    str(number_to_item_list_of_store(int(store_num))), str(number_to_store(int(store_num))),
    int(support_percent), int(confidence_percent))

items, supp, k_value, df = ap_1(items_names, trans, support_percent, confidence_percent, k_value)
frames.append(df)
boo = True

# Increasing K by 1 until we can longer support the support value
```

Figure 8: Calls of functions and variable saving

```
while boo:
    df_1, boo = ap_2(items, k_value, trans, support_percent)
    frames.append(df_1)
    k_value += 1
```

Figure 9:Increasing K by 1 until we can longer support the support value

Michael Woo
28 March 2021
Professor Yasser Abduallah
CS 634 104 Data Mining

```
print("results of item-sets that meet support are below")
display(pd.concat(frames))
df_supp = pd.concat(frames)
```

Figure 10: Combine the dataframes we have from when we increase K

```
def confidence(val):
    # first
    df_212 = df_supp.loc[df_supp['k_val'] == val]
    stuff_name = np.array(df_212["item_name"])
    support_arr = np.array(df_212['support'])

    # second
    df_temp = df_supp.loc[df_supp['k_val'] == val + 1]
    df_21231 = np.array(df_temp["item_name"])
    sup_arr = np.array(df_temp['support'])

    # variables to save
    sup_ov = list()
    sup_sing = list()
    perm_item = list()

    # When the item set is k =1 and the comparison is k = 2
    if val == 1:
        for w in df_21231:
            temp_list = list(df_21231)
            ov_sup = sup_arr[temp_list.index(w)]
            temp = set()
            for d in w:
                temp.add(d)
            perm = permutations(temp)
            perm_lst = list(perm)
            for y in perm_lst:
                perm_item.append(y)
                sup_ov.append(ov_sup)
                sup_sing.append(int(support_arr[np.where(stuff_name == y[0])]))
```

Figure 11.1: This is the FUNCTION that generates the Associations (Permutations) and calculating the Confidence of the item sets

Michael Woo
28 March 2021
Professor Yasser Abduallah
CS 634 104 Data Mining

```
    if val > 1:
        for w in df_21231:
            temp_list = list(df_21231)
            ov_sup = sup_arr[temp_list.index(w)]
            temp = set()
            for d in w:
                temp.add(d)
            perm = permutations(temp)
            perm_lst = list(perm)
            for y in perm_lst:
                try:
                    temp_set = []
                    for t in range(0, val):
                        temp_set.append(y[t])
                    # lol = tuple(sorted(temp_set))
                    lol = tuple(temp_set)
                    tp_lst = list(stuff_name)
                    ss = support_arr[tp_lst.index(lol)]
                    sup_ov.append(ov_sup)
                    sup_sing.append(ss)
                    perm_item.append(y)
                except:
                    print("itemset below does not exist...")
                    print(y)
                    sup_ov.append(ov_sup)
                    sup_sing.append(0)
                    perm_item.append(y)

    df_main = pd.DataFrame({"association": perm_item, "support_ov": sup_ov, "support_sing": sup_sing})
    df_main = df_main.assign(confidence=lambda x: round(((x.support_ov / x.support_sing) * 100), 0))
    return df_main
```

Figure 10.2: This is still part of the confidence function. When the item set is k > 1 and the comparison is k += k + 1

```
try:
    df_final = df_associations.reset_index().drop(['index', 'support_sing'], axis=1)
    df_final.columns = ["Association", "Support", "Confidence"]
except:
    print("No items or transactions meet the user requirements!")

# Final Associations


try:
    print("Store Name: " + str(ns(int(store_num))))
    print("\nFinal Associations that meet the user standards....")
    print("Support: " + str(support_percent) + "%" + "\t" + "Confidence: " + str(confidence_percent) + '%')
    print(df_final)
except:
    print("\nNo Associations were generated based on the parameters set!")
```

Figure 12: Final output (final associations) of the Apriori Algorithm

Michael Woo
28 March 2021
Professor Yasser Abduallah
CS 634 104 Data Mining

Below are screenshots to show that the program runs in the Terminal.

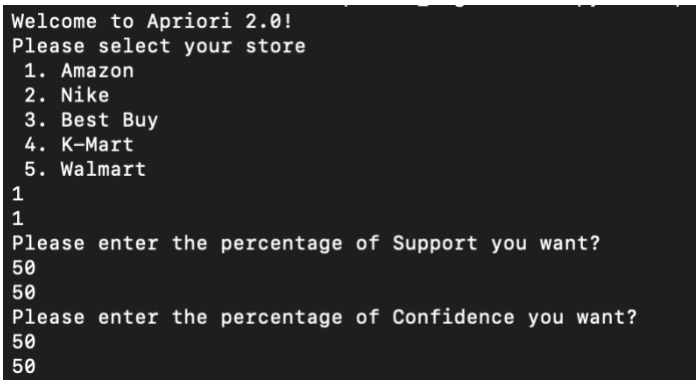

Figure 13: Follow the prompts when asked

The final output should be the following:

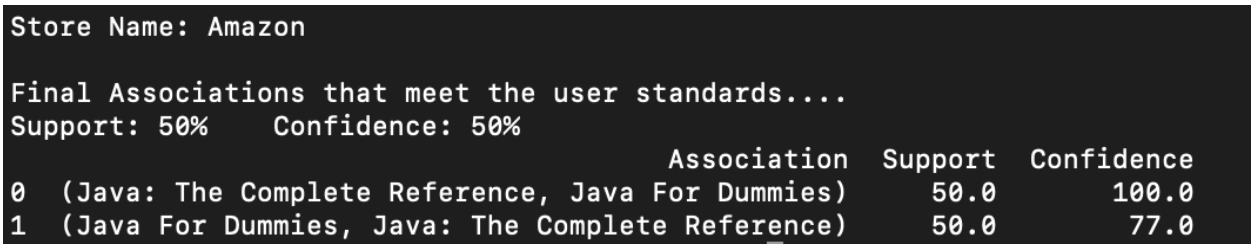

Figure 14: These are the following Associations given the user parameters

Other

The source code (.py file) and data sets (.csv files) will be attached to the zip file.

Link to Git Repository

https://github.com/MichaelWoo-git/Apriori_Algorithm