Mining sequential patterns in web dataset using GSP Algorithm

Lakshmisetti Abhishek Kumar
Elec. Department
Birla Institute of Technology and Science, Pilani, Hyderabad Campus
Hyderabad(Telangana), India
f20160859@hyderabad.bits-pilani.ac.in

***Abstract*—Data Mining is a process to extract information for developing significant relationships with variables stored in large data warehouses. The objective of this assignment is to get the sequential patterns in web dataset using GSP(generalized sequential pattern).Data preprocessing is the most essential part of any data mining tasks.we tried to extract sequences of web data from the given resource.with the help of GSP algorithm we extracted frequent item sets from the given web dataset.**

I. Introduction

The project focuses on application of generalized sequential pattern (GSP) algorithm to mine for sequential patterns in web dataset.A sequential pattern is defined as a series of item-sets; item-sets in which sequences are in specific order.Sequential pattern mining helps to extract the sequences which are most frequent in the sequence database, which in turn can be interpreted as domain knowledge for several purposes.This is used in various areas for different purposes. It can be used for identifying Web data sequences, Customer Shopping Sequences. Various algorithms have been implemented for identifying frequent sequences from sequence databases. One of the approaches used to identify frequent sequences is Apriori approach. “The Apriori approach is based on the apriori property(association rule mining).This property states that if a pattern is not frequent then any pattern that contains cannot be frequent. Two of the most successful algorithms that take this kind of approach are GSP and SPADE. The major difference between the two is that GSP uses a horizontal data format while SPADE uses vertical one. The sequence growth approach does not require candidate generation; it gradually grows the frequent sequences.

II. Data Description

This data set describes the page visits of users who visited msnbc.com on September 28, 1999. Visits are recorded at the level of URL category (see description) and are recorded in time order.

The data comes from Internet Information Server (IIS) logs for msnbc.com and news-related portions of msn.com for the entire day of September, 28, 1999 (Pacific Standard Time). Each sequence in the dataset corresponds to page views of a user during that twenty-four hour period. Each event in the sequence corresponds to a user's request for a page. Requests are not recorded at the finest level of detail---that is, at the level of URL, but rather, they are recorded at the level of page category (as determined by a site administrator). The categories are "frontpage", "news", "tech", "local", "opinion", "on-air", "misc", "weather", "health", "living", "business", "sports", "summary", "bbs" (bulletin board service), "travel", "msn-news", and "msn-sports". Any page requests served via a caching mechanism were not recorded in the server logs and, hence, not present in the data.

Number of users: 989818

Average number of visits per user: 5.7

Number of URLs per category: 10 to 5000

III. Data Analysis

The part of the given dataset

6 7 7 7 6 6 8 8 8 8

6 9 4 4 4 10 3 10 5 10 4 4 4

12 12

1 1

Each category is associated--in order--with an integer starting with "1". For example, "frontpage" is associated with 1, "news" with 2, and "tech" with 3. Each row below "% Sequences:" describes the hits--in order--of a single user.

For example, the first user hits "frontpage" twice, and the second user hits "news" once.

Every line in the Dataset describes the sequence of webpages which the user has entered.Here in the given Dataset we have **989818** number of entries.A sequence is a list or order of items(Ex: 1, 2, 3...)

IV. GSP ALGORITHM IMPLEMENTATION

Since we have the data in Horizontal data format we try to compute the GSP Algorithm on the data as shown below.

```
Algorithm GSP(S)
1   C1 ← init-pass(S);                          // the first pass over S
2   F1 ← {⟨{f}⟩ | f ∈ C1, f.count/n ≥ minsup};  // n is the number of sequences in S
3   for (k = 2; Fk-1 ≠ ∅; k++) do               // subsequent passes over S
4     Ck ← candidate-gen-SPM(Fk-1);
5     for each data sequence s ∈ S do           // scan the data once
6       for each candidate c ∈ Ck do
7         if c is contained in s then
8           c.count++;                          // increment the support count
9       endfor
10    endfor
11    Fk ← {c ∈ Ck | c.count/n ≥ minsup}
12  endfor
13  return F ← ⋃k Fk;
```

Computing C1 & F1:

Given the horizontal sequence database, all frequent 1-sequences can be computed in a single database scan. For each database item we take the count of single items(multiple repetitions of an item in a row will be considered as one count).After this C1 will be generated.

Support Calculation and Pruning(Apriori Method) for calculating F1 :

For implementing the above we implement a for loop where we copy each item set data into a temporary data set and then find the number distinct sequence id associated with the taken item. Then use the following formulae to calculate support:

Support = (No. of Distinct Sequence ID for a given item) / (Total no. of distinct Sequence IDs in dataset)

Now if support of an item is less than min support then the item will be pruned from C1.After pruning the less supportive itemset we will get F1.

After the generation of F1 we generate frequent itemsets until F(k-1) = NULL. Until it satisfies the above condition we run the below steps iteratively.

1. **Join step**. Candidate sequences are generated by joining F(k-1) with F(k-1). A sequence s1 joins with s2 if the subsequence obtained by dropping the first item of s1 is the same as the subsequence obtained by dropping the last item of s2. The candidate sequence generated by joining s1 with s2 is the sequence s1 extended with the last item in s2. There are two cases: the added item forms a separate element if it was a separate element in s2, and is appended at the end of s1 in the merged sequence, and the added item is part of the last element of s1 in the merged sequence otherwise. When joining F1 with F1, we need to add the item in s2 both as part of an itemset and as a separate element. That is, joining {x} with {y} gives us both {x, y} and {x}{y}. Note that x and y in {x, y} are ordered.

2. **Prune step.** A candidate sequence is pruned if any one of its (k-1) subsequences is infrequent (without minimum support).

After completing the above steps frequent item sets will be obtained.

V. ACHIEVEMENTS & LIMITATIONS

Achievements :

1.By implementing the GSP on the web dataset we can extract and observe the frequent pattern of page visits of a user

2.Minimum support can be modified by the user as the minimum support is lowered, more and larger frequent sequences are found. GSP makes a complete dataset scan for each iteration.

3.We didn't use any complicated structure like a Hash tree here.

Limitations:

1. The limitations for GSP are the overhead of generating and searching for subsequences.

2. The generation has a huge set of candidate sequences, which needs multiple scans of the database.

3. The expensive number of short patterns for the mined pattern length and for that reason this algorithm is inefficient for mining long sequential patterns.

Consequently, it is important to review the sequential pattern mining problem to discover more efficient and scalable methods which may reduce the expensive candidate generation.

VI. RESULTS

The minimum support value can be varied but it is inversely proportional to the no. of frequent item sets.If we keep on decreasing the min. support value we will lose the case where we will get the meaning frequent itemsets.So the min. support value has to be saturated somewhere such that we get useful frequent itemsets.with help of these results on web dataset one can arrange the web pages accordingly to have smooth transition between the pages for most of the users.Below are some of the results with different min. support values :

min.support = 0.000000001

```
pattern: ['1'] support value: 136009
pattern: ['1', '1'] support value: 136009
pattern: ['1', '2'] support value: 11627
pattern: ['1', '3'] support value: 7052
pattern: ['1', '4'] support value: 5656
pattern: ['1', '5'] support value: 1659
pattern: ['1', '6'] support value: 8551
pattern: ['1', '7'] support value: 2470
pattern: ['2'] support value: 88065
pattern: ['3'] support value: 85012
pattern: ['4'] support value: 62822
pattern: ['5'] support value: 9554
pattern: ['6'] support value: 147331
pattern: ['7'] support value: 21510
pattern: ['8'] support value: 29002
pattern: ['9'] support value: 48183
pattern: ['1', '2'] support value: 11627
pattern: ['1', '3'] support value: 7052
pattern: ['1', '4'] support value: 5656
pattern: ['1', '5'] support value: 1659
pattern: ['1', '6'] support value: 8551
pattern: ['1', '7'] support value: 2470
pattern: ['1', '8'] support value: 1579
```

```
pattern: ['1', '9'] support value: 2896
pattern: ['1', '1', '2'] support value: 11627
pattern: ['1', '1', '3'] support value: 7052
pattern: ['1', '1', '4'] support value: 5656
pattern: ['1', '1', '5'] support value: 1659
pattern: ['1', '1', '6'] support value: 8551
pattern: ['1', '1', '7'] support value: 2470
pattern: ['2', '3'] support value: 6677
pattern: ['2', '4'] support value: 4770
pattern: ['2', '5'] support value: 1407
pattern: ['2', '6'] support value: 4425
pattern: ['2', '7'] support value: 1852
pattern: ['2', '8'] support value: 1466
pattern: ['2', '9'] support value: 1986
pattern: ['3', '4'] support value: 4629
pattern: ['3', '5'] support value: 1296
pattern: ['3', '6'] support value: 6407
pattern: ['3', '7'] support value: 1148
pattern: ['3', '8'] support value: 1009
pattern: ['3', '9'] support value: 3437
pattern: ['4', '6'] support value: 6414
pattern: ['4', '7'] support value: 3929
pattern: ['4', '8'] support value: 1328
pattern: ['4', '9'] support value: 3643
```

```
pattern: ['5', '6'] support value: 1332
pattern: ['6', '7'] support value: 2305
pattern: ['6', '8'] support value: 2026
pattern: ['6', '9'] support value: 5249
pattern: ['7', '9'] support value: 2047
pattern: ['8', '9'] support value: 1201
```

min.support = 0.00000001

```
pattern: ['1'] support value: 136009
pattern: ['1', '1'] support value: 136009
pattern: ['1', '2'] support value: 11627
pattern: ['2'] support value: 88065
pattern: ['3'] support value: 85012
pattern: ['4'] support value: 62822
pattern: ['6'] support value: 147331
pattern: ['7'] support value: 21510
pattern: ['8'] support value: 29002
pattern: ['9'] support value: 48183
pattern: ['1', '2'] support value: 11627
pattern: ['1', '1', '2'] support value: 11627
```


ACKNOWLEDGEMENT

We thank Dr. Manik Gupta for giving us the opportunity to work on such a project . Dr. Manik Gupta gave special guidance to us during the whole duration of the project, she was very helpful over the period of time. We also thank course teaching assistants Srijanee Mookherji and Deepak Sir for their guidance. Without their guidance this project would not have been possible.

References

- https://github.com/Abhishek859/DM_Assignment/blob/master/GSP_Algo.ipynb