Clustering Attribute Values in Transitional Data Streams

Shankar B. Naik
Department of Computer Science
S.S. Ambiye Govt. College
Pernem, Goa, India
xekhar@rediffmail.com

Jyoti D. Pawar
Department of Computer Science and Technology
Goa University
Goa, India
jyotidpawar@gmail.com

***Abstract*— Millions of users create user profiles on social media. Changes made to an attribute in the user profiles on social media generate a huge volume of data representing a data stream. A framework has been proposed to analyze such data streams and cluster the attribute values related to each other.**

Keywords— ***data stream; social network; clustering; pattern component***

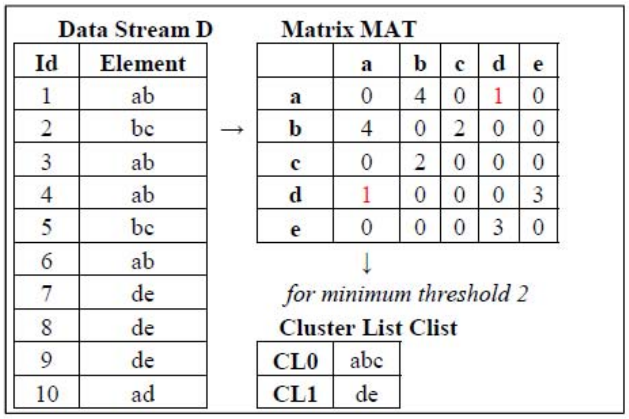
I. INTRODUCTION

Users create their user profiles on social media websites. Each user profiles is a set of attributes pertaining to professional, educational, or useral data. Mostly, these profiles are updated by assigning new values to their attributes. The change in an attribute from an old value to a new one can be studied to mine valuable pieces of information. For example, when a user changes the work company attribute shows that the old and the new companies must be of same /sector/domain and offer similar kind of jobs. This helps both, the employees to find new recruiting companies and employers to find out sources of human resource for their company. A change in address attribute can help learn the migration patterns of humans across places. A change in the attribute mobile number reflects the switching patterns of people between telecom service companies. Analysis of the relation between values of two different attributes like work company and educational institute of a students can help students identify their prospective companies having opportunities in jobs related to their area of studies, and the companies to find out educational institutes passing out students which can be employed in their companies.

Due to the large number of users on social media the number of updates done to their profile is huge. A change made by a user to an attribute on profile can be considered as an event. Due to the large number of users online at a time, the frequency of such events is high. This situation provides a scope to model these events as a data stream of transitions in an attribute. The element of a data stream, for an attribute, is a set (pair) of two values of the attribute, the first one being the old value and the second one being the new one replacing the old value in the user profile.

We have proposed a framework and an approach to model these transitions in attributes from old value to a new one as a data stream and perform analysis to group values of attribute similar to each other.

The approach uses two frameworks. First, the market basket analysis to finding associations between attribute values, and second, the clustering analysis to group these values of attributes using these association patterns. Application of methods like text mining, semantic analysis would make the approach more intelligent.



Id	Element
1	ab
2	bc
3	ab
4	ab
5	bc
6	ab
7	de
8	de
9	de
10	ad

	a	b	c	d	e
a	0	4	0	1	0
b	4	0	2	0	0
c	0	2	0	0	0
d	1	0	0	0	3
e	0	0	0	3	0

CL0	abc
CL1	de

Fig. 1. Framework at a glance

The problem and the approach discussed in this paper is demonstrated in Fig. 1.

This is a new area of research and no similar approach has been proposed earlier to the best of our knowledge. However, network transition analysis has been performed on user profile data in offline manner [2][4]. The remainder of the paper is organized as follows. We have mentioned the work related to this problem in Section II, Sections III defines the problem. The algorithms are proposed in Section IV, followed by the experimental work in Section V. Section VI concludes the paper.

II. RELATED WORK

A. Data Stream

Data steam is a collection of elements where the elements arrive at very high rate [3][4]. Data mining on a data stream is a challenging task because (1) the size of the data stream is unknown; (2) it is not possible to store the elements of the entire data stream for analysis; and (3) the results generated may contain errors. There are three approaches used to mine frequent item sets in data streams: landmark window model, damped window model, and sliding window model. In the landmark window model, transactions between the landmark and the latest transaction are considered. Damped window model considers the recent transactions as more important than the previous ones. In this paper, we have focused on the sliding window model and landmark model.

B. Social network analysis

A similar problem have been worked upon by Cheng et. al.[2] but not from a data stream point of view. In this paper, they analyze the job information only from the social network point of view. Firstly they collect the job-related data from various social media sources and then construct an inter-company job-hopping network. The vertices denote companies and the edges denote the transition of people between companies. Graph mining techniques are used to mine groups of related companies.

Another similar problem have been presented by Xu et. al.[5] from a social networking model point of view. Both the papers have specifically focused on the work company attribute of the user and aim to find associations between companies from employment point of view. The data is analyzed offline. In this paper we propose an approach not limited to the work company attribute. It is applicable to attributes beyond work company. This approach enables the user to analyze the transition data online. Our approach allows the user to specify the value of a minimum threshold to avoid clustering of values together that are weakly associated.

III. Problem Definition

A. Preliminaries

Let A be the attribute of study. For example, A could be name of the work company of a user. Let V={v1,v2,…,vn} be the set of values for the attribute A. For example, V={‘Stickjoy’, ‘Silica’, ’Bluechip’, …}, is the set of names of companies employing people.

A transition element T=(vo,vn), where vo,vnϵV and vn is the value replacing vo for the attribute A. For example, if a user on profile changes the value of his work company from ‘Stickjoy’ to ‘Silica’ then this event generates a transition element T={‘Stickjoy’,’Silica’) having vo=’Stickjoy’ and vn=’Silica’. A change in the user profile on a social media generates such a transition element. The study is limited to unidirectional associations as we are interested in clustering similar values in an attribute, i.e. the elements {‘Stickjoy’,’Silica’} and {‘Silica’,’Stickjoy’} are the same.

More intelligent patterns can be generated from the information about directions in transitions between values of an attribute.

Two values v1 and v2 are similar if v1, v2ϵT. If v1, v2ϵT1 and v2, v3ϵT2 then v1,v2 and v3 are similar. A Cluster C= {vi/ all vi’s are similar} is a cluster of similar values. For example, if a user updates his/her profile to change the work company value from ‘Stickjoy’ to ‘Silica and another user updates his/her profile by changing the work company value from ‘Silica’ to ‘Bluechip’, then ‘Stickjoy’, ’Silica’ and ‘Bluechip’ are similar and they all belong to the same cluster. In this case, a cluster has names of the companies offering similar job.

A data stream D= {T1,T2,…,Tn} is a stream of such transition elements. Two transition elements, T1 and T2 are same if they have same sets of values. For example, If two users change working companies from ’Stickjoy’ to ‘Silica’ at the same or different times, i.e. T1={‘Stickjoy’,’Silica’} and T2={‘Stickjoy’,’Silica’} then T1 and T2 are the same. Two values v1 and v2 are heavily similar to each other if the number of transition elements in the data stream containing v1 and v2, denoted as sim(v1,v2), is greater than or equal to the minimum threshold value s0 i.e. sim(v1,v2)>=s0. The value of s0 is provided by the user of the system. If sim(v1,v2)>= s0 and sim(v2,v3)>=s0 then v1 and v3 are also heavily similar.

The significance of s0 is that it prevents the clustering of values that are not heavily similar, i.e. attribute values that have hardly any transitions happening between them, which is quite often a real world situation. For example, a user while changing his/her profession moves between two companies of different areas. Such companies should not be grouped together as a cluster. Such transitions rarely happen, which are few in number. A minimum support threshold prevents such companies from being clustered together.

B. Problem Statement

The problem is, for an attribute A, a set of values V= {v1, v2,…} for the attribute A, a data stream D= {T1,T2,…} consisting of transitional elements Ti={vo,vn}, vo,vnϵV, generate a set of clusters of values of an attribute such that values in a cluster are similar in terms of the number of transitional elements of the data stream D, they belong to.

IV. The Proposed Framework

A. Intermediate Data Structure

Due to the large size of data streams it is not possible to store all the elements of the data stream on the system for analysis. The elements are processed in batches or one at a time. An intermediate data structure stores the information about the elements of the data stream as they are processed. The data in the intermediate data structure is used to generate the end results. The intermediate data structure in this paper has two parts, a matrix MAT and a list of clusters CList.

1) Adjacency Matrix, MAT: MAT is the adjacency matrix whose rows and columns represent the values of V. The value MAT[i,j] represents the number of transitions happening between vi and vj and vice-versa. Matrix MAT represents an un-directional weighted graph where the vertices represent the attribute values, the vertex a transition between the values represented by the vertices of the edge, and weights representing the number of transitions between the values corresponding to the edges. MAT can be implemented in two ways, as an array and as a list. Implementation as an array requires that the complete list of all the values for the attribute should be already known. Whereas, implementation as a list allows a new value to be dynamically added, no sooner it is encountered the first time in a data stream.In this paper we have implemented the algorithms using a matrix.

2) List of Clusters, CList: CList is a set of clusters of attribute values that are generated at various steps of the algorithm execution. Besides MAT and CList, two bit-vectors. The first one is B= (b1, b2,…bn), where n is the number of values in V, is maintained where bi is set to 1 if vi is already included in some cluster, otherwise it is set to 0.

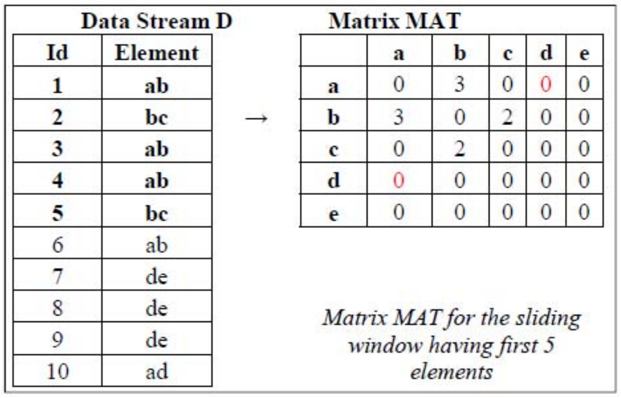
B. The Proposed Algorithms

In this paper we have proposed algorithms for two models, i.e. the sliding window model and the landmark model.

*1) The Sliding Window Model:*This algorithm works in two steps, Update and Generate. The Update step updates the matrix MAT. It is performed when a new element arrives in a data stream and when an old element leave a sliding window. The Generate step generates and/or updates the clusters of attribute values from the matrix MAT. This step is executed at any time by the user.

a) The Update step: This step updates the matrix MAT when a new transitional element either enters (Add step) or leaves (Remove step) the sliding window SW.

b) The Add step: When a new transitional element T={vi,vj} arrives at the sliding window, MAT[i,j] and MAT[j,i] are increased by one each(Fig. 2.).



Data Stream D

Id	Element
1	ab
2	bc
3	ab
4	ab
5	bc
6	ab
7	de
8	de
9	de
10	ad

Matrix MAT

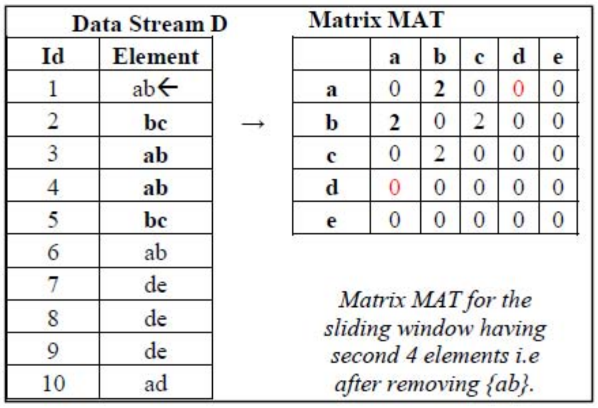
	a	b	c	d	e
a	0	3	0	0	0
b	3	0	2	0	0
c	0	2	0	0	0
d	0	0	0	0	0
e	0	0	0	0	0

Matrix MAT for the sliding window having first 5 elements

Fig. 2. Add step demonstration

c) The Remove step: When a transitional element T={vi,vj} leaves the sliding window, MAT[i,j] and MAT[j,i] are decreased by one each (Fig. 3.).

d) The Generate step: This step generates clusters from the matrix MAT(Fig. 4.). The time complexity of the algorithm is of O(n^2).



Data Stream D

Id	Element
1	ab←
2	bc
3	ab
4	ab
5	bc
6	ab
7	de
8	de
9	de
10	ad

Matrix MAT

	a	b	c	d	e
a	0	2	0	0	0
b	2	0	2	0	0
c	0	2	0	0	0
d	0	0	0	0	0
e	0	0	0	0	0

Matrix MAT for the sliding window having second 4 elements i.e after removing {ab}.

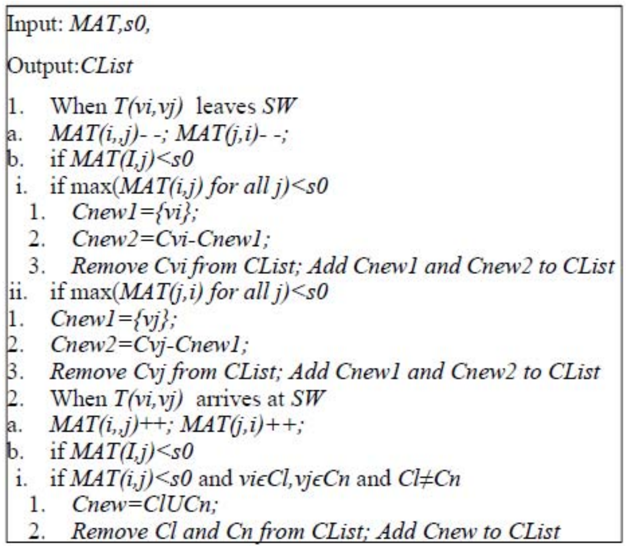
Fig. 3. Remove Step

```
Input: MAT,so

Output: CList={CL1,CL2,...} set of clusters

    1. For every row value vi of matrix MAT, vi∈V, and vi not
       already assigned to any cluster
        a. Co={vi};n=0
        b. S(Cn)={vs/vs is similar to some value in Cn,i.e
           MAT(vs,v)>=so for all v∈Co.}
        c. Cn+1=CnUS(Cn);n++;
        d. Goto step b if Cn-1≠Cn
        e. Add Cn to CList
```

Fig 4. Algorithm Generate

It is noteworthy, that the above algorithm can be executed by the user at any time by specifying the minimum threshold value s0. But, it lacks the ability to incrementally update the clusters in CList. We propose another algorithm (Fig. 5.) to maintain the clusters in CList incrementally.

The motivation behind developing an incremental algorithm to maintain the clusters is as follows. The algorithm in Fig. 4. generates the set of clusters from the matrix MAT containing information for the latest w transition elements of the data stream where w is the size of the sliding window. In-case the user wants to generate clusters immediately after the next Update step, i.e. after the window has slide one element of the data stream, generating the clusters using algorithm in Fig. 4. proves costly. When an old transitional element T={vo,vn} leaves the sliding window, only the cluster containing vo and vn may at the most require a split, if MAT[vo][vn]<s0. When a new transitional element T={vo,vn} enters the sliding window, at the most the clusters containing vo and vn may require a merge, if MAT[vo][vn]>=s0. Hence, the proposed algorithm in Fig. 5. The time complexity of the algorithm is of O(n).



```
Input: MAT,s0,
Output:CList
1.  When T(vi,vj) leaves SW
a.  MAT(i,,j)- -; MAT(j,i)- -;
b.  if MAT(I,j)<s0
 i. if max(MAT(i,j) for all j)<s0
   1.  Cnew1={vi};
   2.  Cnew2=Cvi-Cnew1;
   3.  Remove Cvi from CList; Add Cnew1 and Cnew2 to CList
ii. if max(MAT(j,i) for all j)<s0
1.  Cnew1={vj};
2.  Cnew2=Cvj-Cnew1;
3.  Remove Cvj from CList; Add Cnew1 and Cnew2 to CList
2.  When T(vi,vj) arrives at SW
a.  MAT(i,,j)++; MAT(j,i)++;
b.  if MAT(I,j)<s0
 i. if MAT(i,j)<s0 and vi∈Cl,vj∈Cn and Cl≠Cn
   1.  Cnew=ClUCn;
   2.  Remove Cl and Cn from CList; Add Cnew to CList
```

Fig. 5. Algorithm Update Cluster

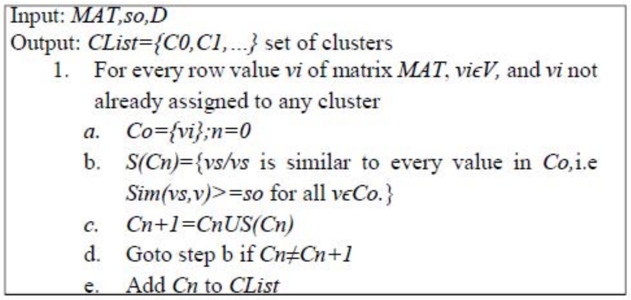
The above algorithm updates the clusters incrementally for a given value of minimum threshold s0. This algorithm can be only executed after having executed the algorithm in Fig. 4. at least once.

2) The Landmark Model

In a landmark model, the part of the data stream considered for analysis are from the element at a time called landmark till the latest element of the data stream. This algorithm works in two steps, Update and Generate. This algorithm is similar to the one used for the sliding model except that there is no Remove step as there is no sliding window to be left by an element of the data stream.

a) The Update step: This step updates the matrix MAT when a new transitional element arrives. When a new transitional element T={vi,vj} arrives at the sliding window, MAT[i][,j] and MAT[j][,i] are increased by one each.

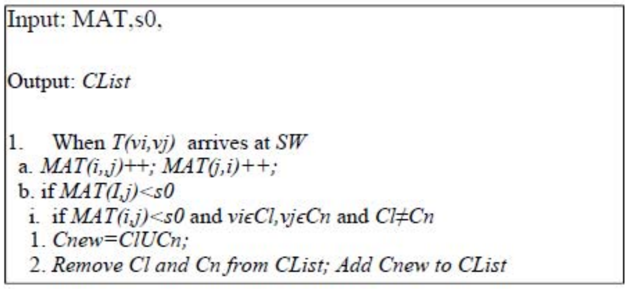
b) The Generate step: This step generates clusters from MAT(Fig. 6.).



```
Input: MAT,so,D
Output: CList={C0,C1,...} set of clusters
  1.  For every row value vi of matrix MAT, vi∈V, and vi not
      already assigned to any cluster
      a.  Co={vi};n=0
      b.  S(Cn)={vs/vs is similar to every value in Co,i.e
          Sim(vs,v)>=so for all v∈Co.}
      c.  Cn+1=CnUS(Cn)
      d.  Goto step b if Cn≠Cn+1
      e.  Add Cn to CList
```

Fig. 6. Algorithm Generate

It noteworthy, that the above algorithm can be executed by the user at the time timestamp by specifying the minimum threshold value s0. But, it lacks the ability to incrementally update the clusters in CList. We propose another algorithm (Fig: 7.) to maintain the clusters in CList incrementally.



```
Input: MAT,s0,

Output: CList

1.  When T(vi,vj) arrives at SW
 a. MAT(i,,j)++; MAT(j,i)++;
 b. if MAT(I,j)<s0
  i. if MAT(i,j)<s0 and vi∈Cl,vj∈Cn and Cl≠Cn
  1. Cnew=ClUCn;
  2. Remove Cl and Cn from CList; Add Cnew to CList
```

Fig. 7. Algorithm

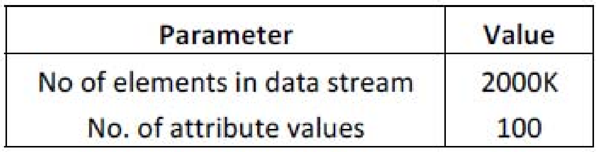
The above algorithm updates the clusters incrementally for a given value of minimum threshold s0. This algorithm can be only executed after having executed the algorithm in Fig. 4.3 at least once.

V. Experiments

The experiments are done on 2.26GHz Intel® Core™ i3 PC with 3 GB memory and running on Windows 7 system. The proposed algorithm is implemented in C++ and compiled using GNU GCC compiler.

The experiments were performed on synthetic data generated using IBM Synthetic Data Generator [1]. The generator was made to generate sets of average 3 items in an item set. The generated data was then pruned to remove item sets with other than two items. The parameters of the data sets is given in the Table I.

TABLE I



Parameter	Value
No of elements in data stream	2000K
No. of attribute values	100

A. Using landmark model

This experiment was performed on the above data set using the landmark model approach to determine the number of clusters by changing the value of minimum threshold (Fig. 8.).

Fig. 8. shows the average time required for the generation of clusters by varying the value of s0. The number transition elements analyzed at a time is 100K.

B. Using sliding window model

This experiment was performed on the above data set using the sliding window model approach to determine the number of clusters by changing the value of minimum threshold. The size of the sliding window was 10K (Fig. 10.).

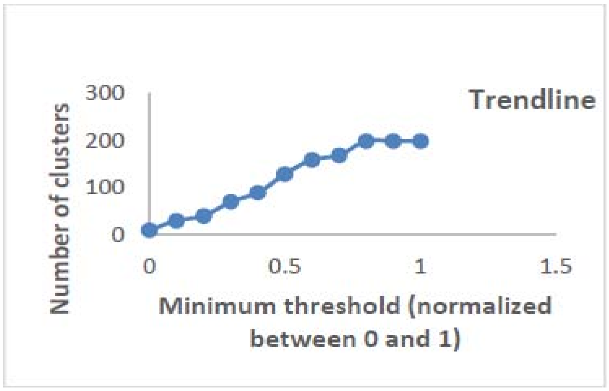


Fig. 8. Experiment using landmark model

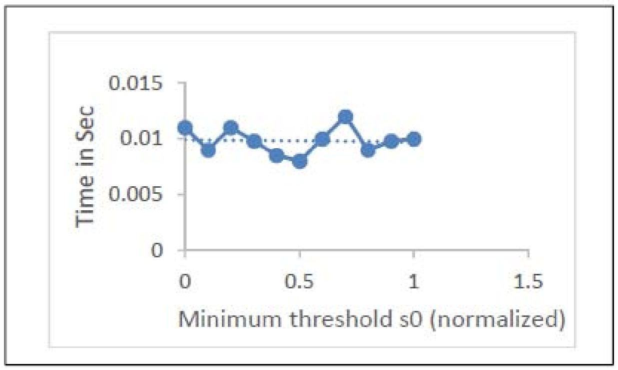


Fig.9. Time requirements

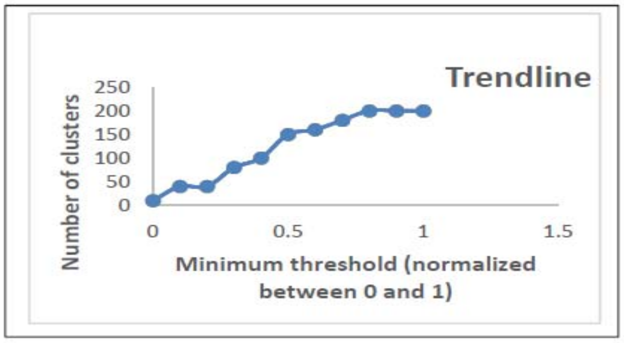


Fig. 10. Experiment using sliding window

VI. Conclusion

In this paper we have proposed a framework and algorithms to analyze data streams having its elements as pair of values of attributes and cluster these attribute to find out groups of related values. These values can be objects like companies, educational institutes, places, etc. This paper considers only the attribute values as specified by the user in his/her profile. There is tremendous scope for more intelligent mining by incorporating data about the values themselves.

In this paper we have limited our study to a data stream pertaining to one attribute only. The same study can be applied to data streams with multiple attributes.

We have limited our experiments to synthetic data. The understanding about the efficiency and limitations of this approach can be better appreciated by performing experiments on real data in a real time environment. Nevertheless, experimentation on synthetic data helped us verify the results as we already knew the properties of synthetically generated data. We are in the process of applying our algorithms to real data from Facebook which we shall publish subsequently by overcoming the short comings of the proposed approach.

Deciding the value of minimum threshold s0 and the size of the sliding window is itself an area of research. These values can be optimized by using approaches which repeatedly learn upon execution from real data in a domain.

Perhaps, our future work shall address these issues.

References

[1] R. Agrawal and R. Shrikant. “Fast algorithms for mining association rules,” Proceedings of the 20th international conference on very large databases, 487-499, 1994.

[2] Y. Cheng, Y. Xie, Z.Chen, A. Agrawal. “Jobminer: A realtime system for mining job-related patterns from social media,” 2013

[3] S. B. Naik, J.D. Pawar. “An efficient incremental algorithm to mine closed frequent itemsets over data streams,” Proceedings of the 19th COMAD 2013

[4] S. B. Naik, J.D. Pawar. “A quick algorithm for incremental mining closed frequent itemsets over data streams,” Proceedings of the Second ACM IKDD Conference on Data Sciences, . CoDS '15

[5] H. Xu, J. Yang, H.Xiong, H.Zhu. “Talent Circle Detection in job transition network,” KDD, 2016

[6] S.B. Naik and J.D.Pawar. "Finding frequent item sets from data streams with supports estimated using trends," Journal of Information and Operations Management 3.1 (2012): 153.