AFRL-AFOSR-UK-TR-2019-0012

Ranking and Clustering in Signed and Weighted Bipartite Graphs

Ismail Hakki Toroslu
ORTA DOGU TEKNIK UNIVERSITESI

02/28/2019
Final Report

Air Force Research Laboratory
AF Office Of Scientific Research (AFOSR)/ IOE
Arlington, Virginia 22203
Air Force Materiel Command

DISTRIBUTION A: Distribution approved for public release.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
05-03-2019	Final	15 Oct 2014 to 14 Oct 2018

4. TITLE AND SUBTITLE
Ranking and Clustering in Signed and Weighted Bipartite Graphs

5a. CONTRACT NUMBER

5b. GRANT NUMBER
FA9550-15-1-0004

5c. PROGRAM ELEMENT NUMBER
61102F

6. AUTHOR(S)
Ismail Hakki Toroslu

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
ORTA DOGU TEKNIK UNIVERSITESI
UNIVERSITELER MAHALLESI, 1
CANKAYA, 06800 TR

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
EOARD
Unit 4515
APO AE 09421-4515

10. SPONSOR/MONITOR'S ACRONYM(S)
AFRL/AFOSR IOE

11. SPONSOR/MONITOR'S REPORT NUMBER(S)
AFRL-AFOSR-UK-TR-2019-0012

12. DISTRIBUTION/AVAILABILITY STATEMENT
A DISTRIBUTION UNLIMITED: PB Public Release

13. SUPPLEMENTARY NOTES

14. ABSTRACT

This project created and analyzed algorithms to cluster weighted graphs - ideas which could be applied to better understand radical subnetworks in social media. The project piggy packed with a larger US DoD effort and connected the PI with researchers working on DoDs Minerva Research Initiative at Arizona State University (ASU). The project produced several conference papers as well as a journal article published jointly with the ASU team. The project successfully created several algorithms based on greedy heuristics to cluster bi-partite and tri-partite graphs. In addition to benchmarks on synthetic data, these algorithms were also tested on real-world Twitter data the goal to cluster UK Tweets from around to time of Brexit discussions to see if politicians, key words, and sentiment could be well identify by the clustering. While the algorithms did show promise, it remains challenging to directly compare these results to other existing clustering methods. Full details are found in the attached report as well as journal articles and conference papers therein. While AFOSR is currently not supporting follow-on efforts, the PI and his team plan to continue to improve their clustering techniques.

15. SUBJECT TERMS
Operations Research, Network Optimization, Bipartite Graphs, Complex Networks, EOARD

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			PETERSON, JESSE
Unclassified	Unclassified	Unclassified	SAR		

Standard Form 298 (Rev. 8/98)
Prescribed by ANSI Std. Z39.18

				19b. TELEPHONE NUMBER *(Include area code)* 314 235 6292

FINAL REPORT

GRANT NUMBER: FA9550-15-1-0004
RESEARCH TITLE: Ranking and Clustering in Signed and Weighted Bipartite Graphs

PI NAME: ISMAIL H. TOROSLU

PERIOD OF PERFORMANCE:14 OCT 2014 -14 Oct 2018

TABLE OF CONTENTS

1. SUMMARY

The project had officially started on October 15, 2014 and it was planned as 2 years. However, due to the problems with the transfer of the funds, an extension is granted, and the project has been extended until Oct 14, 2018. The funds became available at METU project account in May 2016. The total amount was $38.000. Half of the funds were transferred to ASU project account. Using these funds, the following visits are done:

- Prof. Davulcu visited Prof. Toroslu's team at METU, Ankara, Turkey between March 4 and March 17, 2017.
- Prof. Toroslu visited Asst. Prof. Jordan at AFIT, Dayton, OH between May 27 and June 2, 2017.
- Prof. Toroslu visited Prof. Davulcu's team at ASU, Tempe, AZ between January 27 and February 4, 2018.
- Prof. Toroslu visited Prof. Davulcu's team at ASU, Tempe, AZ between July 9 and 20, 2018.

The results of the collaboration has produced 6 paper, one of them is a journal paper [1], and the others are well-known conferences on social networks [2, 3, 4, 5, 6]. One MS thesis of a graduate student, Mr. Sefa Sahin Koc at METU who was supervised by Prof. Toroslu, was directly based on the project subject, and in January 2017 Mr. Koc had completed his MS thesis. In the last part of the project, a PhD student from METU, Riza Aktunc, also joined to the project. Meanwhile, a PhD student at ASU who was supervised by Prof. Davulcu, Mr. Mert Ozer, was also involved in the project. Finally, Asst. Prof. Dr. Jeremy Jordan from Air Force Institute of Technology (AFIT), had also joined to the Prof. Toroslu and Prof. Davulcu research team, and the journal paper has been published together.

In addition to the visits, project funds have also been used by Prof. Toroslu and Prof. Davulcu to participate the most prestigious social network conference (IEEE/ACM ASONAM) in years 2015, 2016, and 2018. In year 2018, two teams published 2 separate papers in that conference. The papers published in AONAM 2015 and 2016, the papers written by the collaboration of two teams. Also, another conference paper has been published in another well-known social network conference, SBP-BRIMS in 2015 as well in collaboration of two teams.

In general, the collaboration was very successful in terms of the scientific output produced, and the funds were extremely helpful in achieving this success. All the papers published acknowledge the support of the grant as well.

2. INTRODUCTION

Project Definition

At the beginning of this project, we started with bipartite graphs, and we modeled a mixed set of blogs/microblog records debating a set of political issues from different camps as signed and/or weighted bipartite graphs. At this point, our model has been evolved and now it is in the form of a tri-partite graph. In this graph, the partitions correspond to people, issues, and keywords people use to express their views related to these issues, respectively. We have developed algorithms for clustering/partitioning the blogs/people, the issues (i.e. topics, leaders, etc.), and the keywords simultaneously comprising the debate into two or more camps, which was published in [1]. In addition, follow up research directions were evolved as well, such as detecting antagonistic and allied communities on social media [3] and event detection by change tracking on community structure of temporal networks [2, 4].

Teams

Prof. Davulcu's team at Arizona State University have been working on this kind of problems for several years, and his team has been supported by US DoD's Minerva Research Initiative Grant (with #: N00014-09-1-0815). In this project, they had applied this model for understanding the structure of counter-radical networks, the ideas on which they are based, social locations of their leaders and followers, and the ways in which radical and counter-radical discourse intersect. One of the main results of that project was extracting multidimensional portrait of counter-radical networks shifting through time and across regions. Since 2011, Prof. Toroslu's team and Prof. Davulcu's team have been working together on some sub-problems in this domain, namely the scaling and clustering of signed and/or weighted bipartite graphs in order to formulate the kind of relationships mentioned above and some results have been published.

Impact of the Project

The idea of this project is based on the problems related to Prof. Davulcu's other similar projects supported by DoD and other government agencies, including MINERVA. The problems studied are related to bipartite/tripartite graphs and different forms of data mining/optimization problems in these graph models. We have identified new problems on this domain and developed effective solutions to them. All these problems are from the new and emerging sub-domain of computer science, called social network analysis. This area has become one of the most active research areas in computer science and more specifically in data mining. The results of this specific project will help us to understand social networks formed by people on the web and other communication networks, and their connections, how they form communities etc.

In terms of collaboration between Turkish and US teams, these contributions not only complement the DoD and similar projects of Prof. Davulcu, but they are also used for developing further research projects. Both parties had benefited from this collaboration, since the parties have skills and experiences complementing each other.

At the end of the project 1 MS student has completed his thesis, 2 PhD students had partially contributed and a professor from AFIT also joined to the project. Finally, 5 conference and 1 journal paper had been published together by Prof. Toroslu and Prof. Davulcu. Both sides plan to continue their collaboration after the completion of the project as well.

3. METHODS, ASSUMPTIONS AND PROCEDURES

Most important part of the methods developed or contributions of the project is based on the journal paper [1], whose early version was also published in a conference [4]. Below details of this work is given, and the other papers are summarized shortly afterwards.

Co-Clustering Signed 3-Partite Graphs

Social network data contains many hidden relationships. The most well-known is the communities formed by users. Moreover, typical social network data, such as Twitter, can also be interpreted in terms of three-dimensional relationships; namely the users, issues discussed by the users, and terminology chosen by the users in these discussions. In this work, we propose a new problem to generate co-clusters in these three dimensions simultaneously. There are three major differences between our problem and the standard co-clustering problem definition: a node can be a member of more than one clusters; not necessarily all the nodes are members of some cluster; edges are signed and clusters are expected to have high density of positive signed edges, and low density of negative signed edges. We apply our method to the tweets of British politicians just before the Brexit referendum. Our motivation is to discover clusters of politicians, issues and the sentimental words politicians use to express their feelings on these issues in their tweets.

In many social network sites users share their views on a variety of subjects. This simple data set can naturally be modelled as a 3-dimensional relationship including users, topics on which users share their ideas and the keywords users use to express their feelings on the subjects as three different dimensions. Users may be clustered according to the similarity of feelings on the same issues. Also, users with different political views or from different social group may prefer to use different keywords with the same negative or positive sentiments in order to express the same kind of feeling on the same issue. Therefore, in clustering the people, it is important to use the feelings of the people on the issues, as well as utilizing keywords they use to express these views.

In this work, we have collected tweets of preselected groups of users on some specific issues, and then, created a 3-dimensional data model by also extracting the sentiment keywords used by these users on the selected issues. More specifically, our selected groups of people are politicians from the UK, and the issues are hot subjects discussed just before the "Brexit" referendum. Using this data, we have created a 3-D data set, whose dimensions correspond to the people, the issues and keywords, and the values of each element of this 3-D data. If the person had written a tweet on the issue including that keyword, the values are the sign of the sentiment keyword (as positive or negative), else the value is empty. This 3-D data can be modelled as a tripartite hypergraph.

The aim of this research is to propose an effective method which generates tri-clusters from hypergraphs with signed tripartite hyperedges. Our focus data set is a signed 3-D relationships obtained from users' feelings represented by the sentimental keywords (either positive or negative) they use their tweets on the issues.

We have developed a new co-clustering algorithm customized to meet the requirements of our problem for 2 main reasons. These reasons are as follows:

1. Typical co-clustering algorithms, such as spectral clustering algorithms, include all the nodes of all the dimensions of the graph in the clusters generated for these dimensions. However, some of these nodes might have very few relationships or they may not have sufficiently common relationships with other nodes. Thus, we may want to exclude such nodes from the clusters generated. In other words, we don't need to include all the nodes in clusters formed from tri-partite graph if they do not contribute the clusters. The purpose of all clustering algorithms is to determine subgraphs with strong connections/relationships among nodes. However, when all the nodes are forced to be included in clusters, their connections weaken. Furthermore, we may want to define the minimum amount of connection in subgraphs as a constraint. This means, we have to construct clusters by excluding some of the nodes that violates these constraints. In this way, we can obtain denser and more meaningful clusters.

2. Again, typical co-clustering algorithms, like spectral clustering method, place every node in exactly one cluster. In our data set, a node may be in more than one cluster. For example, we may detect a person which belongs to two different social groups. So, we have to allow overlaps of nodes (not hyperedges) in clusters.

We call this new form of clustering method as triadic co-clustering. In this work we developed the following:

1. A full-fledged system developed to crawl and collect tweets of selected users on selected issues, and then construct a 3-D relationship among user issue-sentiment keyword triples by parsing these tweets and extracting the issue-sentiment keyword relationships.

2. A flexible tri-partite clustering algorithm (triadic co-clustering) with parameters to control the minimum density and the minimum size of clusters.

3. The clustering method has been applied on real data sets, more specifically on the tri-partite graph obtained from "Brexit" tweets and its effectiveness has been shown both by empirically and by the coverage metric defined.

For the dataset that we have studied in this work clusters cannot overlap. However, one user, or one issue, or one sentiment keyword may be in more than one clusters simultaneously, since, for example, a specific user's feelings toward some issues can be the same as with more than one different set of people (different people clusters), so he can be in both clusters at the same time. Again considering tweets, the same sentiment words may be used by people with different clusters corresponding to different political camps. They may even use the same keywords on the same issues. On people dimension, a person which appears in more than one cluster may be interpreted as a person close to more than one different political camps.

Other Methods Studied

Other 4 papers focus on related but different aspects of the problem.

- Dynamic community detection and event detection by change tracking on community structure of temporal networks [2, 5]
- Predicting hashtag breakouts in Twitter [6]
- Detecting antagonistic and allied communities on social media [3]

Both problems are variations of our original bipartite graph clustering problem. In the first one the graph is not bipartite and the nodes are uniform. However the concept of community in these graphs is similar to clusters in bipartite graphs and similar approaches are applicable. Even though this version of the problem is well-studied in the literature its dynamic version on very large graphs is still a new topic. Another important and popular topic related to blogosphere is detecting hashtag breakouts, which is directly related to community behaviors. A directed graph is formed representing the relationships among users in terms of retweeting, replying or mentioning another user. Thus, this problem also turns into a classification in a directed graph and, we have also studied it and a quite successful solution has been generated.

Dynamic community detection and event detection by change tracking on community structure of temporal networks

Most of the social networks are not static because they evolve in many ways. They may gain or lose users that are represented as nodes in the graphs over time. The users of these social networks may lose contact from each other or there can be new connections among users. In other words, some edges in the graphs may be removed or new edges may be added to the graph over time. All these processes may happen in a very small amount of time in a social network if it has a lot of active users. This kind of a social network may be called as highly dynamic. For example, popular social sites such as Facebook, Twitter, LinkedIn and so on have highly dynamic social networks. The first part of this work focusses on this dynamism [5].

Addition or deletion of an edge or a node from a network which has millions of edges might seem insignificant; but when this additions or deletions of an edge or a node happen very frequently, they begin to change the community structure of the whole network and become very important. This change in the community structure raises the need of re-identification of communities in the network. This need arises frequently and creates a new problem in the community detection research area. This new problem requires somehow fast detection of communities in dynamic networks.

The first solution that comes to mind for community detection in large dynamic networks problem is the execution of static community detection algorithms already defined in the literature all over again to detect the new community structure whenever the network is modified. Nevertheless, this solution takes too much time in every modification of the large networks since it runs the community detection algorithm from scratch each time. A much efficient and less time consuming solution is to run the community detection algorithms not from scratch but from a point in the history of the network by storing and using the historical results of executions of the algorithms whenever network is evolved. In other words, updating previously discovered community structure instead of trying to find communities from scratch each time the network evolves consumes much less time and thus much efficient.

We modified the smart local moving (SLM) algorithm defined by Waltman & Van Eck [5] so that it would detect the communities in rapidly growing large networks dynamically and efficiently. As a result, we propose the dynamic SLM (dSLM) algorithm that dynamically detects communities in large networks by optimizing modularity and using its own historical results. We tested our proposed approach on several different datasets. We demonstrated the effects of our contribution to the SLM algorithm in two ways. One of them is the change in modularity value which determines the quality of the community structure of the network. The other one is the change in running time that determines the pace of the algorithm. The latter is more significant than the former because the community structure of the network must be quickly identified at the given timestamp before the next timestamp is reached. We realized that dSLM improved SLM by decreasing its running time incredibly. Moreover, there are some experiments where modularity value increases while running time decreases.

SLM algorithm changes the reduced network construction step by applying following processes:

1) It iterates over all communities that are formed by the first step. It copies each community and constructs a subnetwork that contains only the specific community's nodes.

2) It then runs the local moving heuristic algorithm on each subnetwork after assigning each node in the subnetwork to its own singleton community.

3) After local moving heuristic constructs a community structure for each subnetwork, the SLM algorithm creates the reduced network whose nodes are the communities detected in subnetworks. The SLM algorithm initially defines a community for each subnetwork. Then, it

assigns each node to the community that is defined for the node's subnetwork. Thus, there is a community defined for each subnetwork and detected communities in subnetworks are placed under these defined communities as nodes in the reduced network.

This is the way that the SLM algorithm constructs the reduced network. After these processes, the SLM algorithm gives the reduced network to the recursive call as input and all the processes starts again for the reduced network. The recursion continues until a network is constructed that cannot be reduced further.

SLM algorithm initially assigns each node to a different community, so each node has its own singleton community. In order convert SLM to dynamic form, we replace that operation with a newly defined procedure. This procedure works as follows:

- Existing communities are read from file as New Communities.
- If exists, the extensions to the network has been determined.
- For each new node, singleton new communities are constructed and added to New Communities.

The effects of other changes in the network, such as adding new edges and deletions of nodes and edges, are handled while executing standard SLM procedure. The new dSLM is available at https://github.com/mertozer/dSLM.

We evaluate our proposed approach dSLM on five real world datasets which are the arXiv citation dataset, the GSM calls dataset, Google Plus, Twitter and YouTube user network datasets.

In the second part of the work [2] we aimed to detect events from online data sources with least possible delay. Most of the previous work focuses on analyzing textual content such as social media postings to detect happenings. In this work, we considered event detection as a change detection problem in network structure, and propose a method that detects change in community structure extracted from communication network. We have studied three versions of the method based on different change models.

We focused especially on detecting the events on Call Detail Record (CDR) data. We firstly extract weekly samples from CDR data and then convert these samples to directed weighted and unweighted networks. The nodes of the networks correspond to phone users in CDR data, while the edges of the networks correspond to the communication between the connected nodes. The

communication can be an SMS or a voice call. For each communication type, we define different networks.

In the literature, event is generally defined as a happening that takes place at a certain time and place, and attracts attentions. In our study, we focus on the time dimension and aim to determine time windows in which an event takes place. The proposed method involves tracking the change in the community structures over temporal networks. A sequence of networks corresponding to time windows along the timeline is analyzed for changes in detected communities in consecutive networks. We model the change in three different ways: change in the number of communities, change in the central nodes of the communities, and change in members of the communities. Within each model, there are variations based on how the defined change is computed.

The proposed method is based on the idea that events can be detected by tracking the amount of change in various attributes of the community structure of the network in consecutive time windows. In addition to the length of the time window, the source of the network structure and the community detection techniques may vary. In this work, we set the time window as one week, and hence construct communities on the basis of the weekly interactions. As the community detection technique, we use dSLM algorithm.

In order to track the change, we compare the community structure parameter values against the previous time window. We studied change in the community structure over three basic parameters, which are Number of Communities, Central Nodes, and Community Members.

A Network-Based Model for Predicting Hashtag Breakouts in Twitter

Online Social Networks (OSNs) such as Twitter have emerged as popular microblogging and interactive platforms for information sharing among people. Twitter provides a suitable platform to investigate properties of information diffusion. Diffusion analysis can harness social media to investigate viral tweets and trending hashtags to create early-warning solutions that can signal if a viral hashtag started emerging in its nascent stages. In this work, first we introduced a simple standard deviation sigma levels based Tweet volume breakout definition, then we proceeded to determine patterns of re-tweet network measures to predict whether a hashtag volume will

breakout or not. We also developed a visualization tool to help trace the evolution of hashtag volumes, their underlying networks and both local and global network measures. We trained a random forest tree classifier to identify effective network measures for predicting hashtag volume breakouts.

Given a set of tweets T = t1, t2, t3, ..., tn where n is number of tweets in our corpus. These tweets comprise textual contents, user interactions and additional meta-data. We explore and analyze both textual contents filtered by a given hashtag from hashtags set H. Then we denote tweet volume as number of tweets per day. We then compute daily means ($\mu(20)$) and standard deviation ($\sigma(20)$) for each hashtag by utilizing its volume distribution during its previous 20 days window. We experimentally determined the best window size by experimenting 10, 15, 20, 25 and 30 days windows. The 20 days window shows the best performance amongst the others.

If the hashtag frequency rises above ($\mu(20) +1\sigma(20)$), then we label that period as an episode, and we mark its previous 20 days as the accumulation period of an episode. We start observing hashtag frequency for two possible outcomes:

- a breakout if hashtag volume rises above($\mu(20) +2\sigma(20)$), without falling below max(0, $\mu(20) - 2\sigma(20)$), or
- non-breakout, if hashtag volume falls below max(0, $\mu(20) - 2\sigma(20)$), without rising above ($\mu(20) +2\sigma(20)$)

In breakout scenario for an episode no further overlapping breakouts are allowed until its volume falls below max(0, $\mu(20) - 2\sigma(20)$). In both scenarios, as episode begins with its accumulation period and continues until the hashtag volume dies out (i.e. it falls below max(0, $\mu(20) - \sigma(20)$)).

The dataset used in this study is a collection of tweets from UK region. These tweets have been crawled based on a set of keywords with the aim to capture political groups, events, and trends in the UK. The dataset consists of more than 3 million tweets, 600K users, with more than 5.2 million interactions (both mentioning and retweeting) between users along with 1,334 hashtags.

In this model we investigate how users get involved in a hashtag h by mentioning, replying or retweeting. Their interactions are depicted as a directed graph. We then incorporated normalized size-independent network features for directed graphs corresponding to accumulation periods of episodes. The network graph is a pair G = (V,E) where V is set of vertices representing users

together with a set of edges E, representing interactions between users. For instance, if a user u1 mentioned, replied, or retweeted one tweet of u2, then a directed edge from u1 to u2 is formed.

We attempted to identify key features that contribute to the network based classification problem for breaking or non-breaking hashtags. We used both local and global measures. Local measures are associated with user interactions during the accumulation period only, whereas global measures draws their information from all interactions beginning from the start date (June 2013) until the end date of any accumulation period under consideration.

Detecting Antagonistic and Allied Communities on Social Media

Community detection on social media has attracted considerable attention for many years. However, existing methods do not reveal the relations between communities. Communities can form alliances or engage in antagonisms due to various factors, e.g., shared or conflicting goals and values. Uncovering such relations can provide better insights to understand communities and the structure of social media. According to social science findings, the attitudes that members from different communities express towards each other are largely shaped by their community membership. Hence, we hypothesize that intercommunity attitudes expressed among users in social media have the potential to reflect their inter-community relations. Therefore, we first validate this hypothesis in the context of social media. Then, inspired by the hypothesis, we develop a framework to detect communities and their relations by jointly modeling users' attitudes and social interactions.

In this work, we propose a framework, namely DAAC, which detects communities and their relations (i.e., antagonism, alliance, or neither) by exploiting users' social interactions (e.g., retweets) and attitudes expressed on social media. Our main contributions are:

- Validating the hypothesis suggesting that intercommunity attitudes that users express towards each other in social media can reflect the relations of their communities;
- Achieving higher performance in detecting communities compared to several standard community detection methods;
- Uncovering inter-community relations, i.e., antagonism, alliance, or no relation.

The problem of detecting communities and their relations on social media can be defined as: Given social interaction matrix R and attitude matrix S, we aim to obtain community membership matrix U and intra/inter-community relation matrix H. Formal definition can be found in [3].

Politics is a domain in which it is common among political parties (i.e., communities) to form alliances or engage in antagonisms. To validate the aforementioned hypothesis and evaluate our proposed framework, we use the following political Twitter datasets:

- US Dataset consists of the tweets posted by 583 politicians from two major US political parties (the Republican Party and the Democratic Party) from August 26 to November 29, 2016. For the period of time that this dataset covers, there were antagonisms between these parties particularly due to the 2016 presidential election campaigning.
- Australia Dataset consists of the tweets posted by 225 user accounts, including politicians and political groups, from five major Australian political parties (the Liberal Party, the National Party, the Liberal National Party, the Greens, and the Labor Party) from January 1 to November 18, 2016. For several decades, there has been a coalition among the Liberal Party, the National Party, and the Liberal National Party. In the 2016 federal election, all relations between the parties were antagonistic except the relations between the members of the coalition,.
- UK Dataset consists of the tweets posted by 389 user accounts, including politicians and political groups, from five major UK political parties (the Conservative Party, the Labour Party, the Scottish National Party, the Liberal Democrats Party, and the UK Independence Party) from January 1 to October 31, 2015. There were antagonism among five major UK political parties in this period of time, especially due to the 2015 general election campaigning.

According to social science findings, the attitudes that members from different communities express towards each other are largely shaped by their community membership. Therefore, we hypothesize that inter-community attitudes expressed among users towards each other in social media have the potential to reflect inter-community relations. However, the findings borrowed from social sciences do not necessarily hold in social media due to many factors, such as the validity and representativeness of available information. Moreover, the attitudes that users

express towards each other in social media might result from users' personal relationships. we aim to verify our hypothesis by answering the following two questions. With this respect, we utilize the Australia dataset since it is the only dataset containing both allied and antagonistic relations.

- Are the communities of two users who express negative attitudes towards each other more likely to be in antagonism?
- Are the communities of two users who express positive attitudes towards each other more likely to be in alliance?

For each pair of users (ui; uj) who are from different communities and have expressed negative attitudes towards each other, we randomly select a user uk where users ui and uk are from different communities and have not expressed negative attitudes towards each other. Then, we check whether there is antagonism between the communities of ui and uj and between the communities of ui and uk. If there is antagonism between the communities of ui and uj , we set tp = 1; otherwise tp = 0. Similarly, if there is antagonism between the communities of ui and uk, we set tr = 1; otherwise tr = 0. Let vector Tp denote the set of all tps for pairs of users from different communities who have expressed negative attitudes towards each other, and vector Tr denote the set of all trs for pairs of users from different communities who have not expressed negative attitudes towards each other.

Therefore, the result of the two sample t-test demonstrates that the communities of two users who express negative attitudes towards each other are highly probable to be in antagonism. we conclude that the communities of two users who express positive attitudes towards each other are highly probable to be in alliance.

We also propose a model which uncovers intra/inter-community relations by exploiting the attitudes users express towards each other. Furthermore, we cluster users into k communities with the most social interactions within each community and the fewest social interactions between communities. Our framework DAAC jointly exploits these two models to uncover communities and their relations. The proposed framework requires solving an optimization problem. Since the optimization problem is not convex there is no guarantee to find the global optimal solution. We introduce an alternative scheme to find a local optimal solution of the optimization problem. The key idea is optimizing the objective function with respect to one of

the variables while fixing the other one. The algorithm keeps updating the variables until convergence.

To evaluate our proposed framework, we design the required experiments to answer the following two questions.

1) How effective is the proposed framework compared to the standard community detection methods?

2) How effective is our framework in discovering intercommunity relations?

We can make the following observations:

- Our proposed framework achieves the highest performance in terms of NMI and ARI for all three datasets. In terms of Purity, it also achieves the best in US and Australia datasets. In the UK dataset, only InfoMap obtains higher Purity compared to our framework since it generates a large number of communities (e.g., 11 communities for the UK dataset) for sparse graphs such as social media networks.
- Our framework achieves its highest performance with large values of regularization parameter (e.g., 107). This implies that social interactions are more effective in detecting communities compared to users' attitudes.

For the inter-community relations we compare the inter-community relations which our framework detects with the real-word inter-community relations. Each community detected by our framework is labeled with the party to which the majority of its members belong. Then, we evaluate inter-community relations detected by our algorithm according to the known ground-truth inter-party relations

The second experiment compares our framework with a two-step approach described as follows. We first utilize social interactions to detect communities. Then, we aggregate the sentiment expressed among the members of different communities in order to figure out their inter-community relations.

We validated the hypothesis that inter-community attitudes that users express towards each other in social media can reflect inter-community relations. As inspired by this hypothesis, our proposed framework DAAC jointly models users' attitudes and social interactions in order to uncover communities and their antagonistic/allied relations. Experimental results on three real-

world social media datasets demonstrated that our framework obtains significant performance in detecting communities compared with several baselines and also detects inter-community relations correctly. Moreover, we showed that a two-step approach, which sequentially detect communities and their relations, can fail to detect correct inter-community relations.

4. RESULTS AND DISCUSSIONS

In this part we only discuss the details of the main part of the study, namely co-clustering three-partite graphs [4].

Experiments on Synthetic Data

In order to evaluate our algorithm, we have generated data sets with varying sizes. We have done several experiments.

In the first set of experiments, we have fixed the positive and negative signed edge density ratios while changing input sizes. In this test, we have generated 6 sample datasets. Each one contains positive hyperedges with 60%, negative hyperedges with 20%, and 20% is empty. The sizes of graphs were (31.25K, 62.5K, 125K, 250K, 500K, 1M) respectively for the 6 experiments.

In the second test, we have generated 5 datasets. In this test, we have fixed the size as 125K and we have varying density ratios for (minimum positive edge, maximum negative edge) pairs as (0.2,0.4), (0.2,0.2), (0.4,0.2), (0.4,0.4), (0.6,0.2).

Naturally a good clustering should contain most (positive) hyperedges in clusters while clusters being non-trivial. The results show that we have achieved very high coverage in that sense, since constructed clusters include almost as many hyperedges as the half of the number of positively signed hyperedges.

The results show that we have achieved very high coverage in that sense, since constructed clusters include almost half of the positive signed hyperedges.

Experiments on Brexit Data

We have generated one data set from Twitter data. We have used Brexit event for this purpose. The Brexit referendum is a very popular political event of 2016. The citizens of the UK voted

whether or not the UK should leave the EU. To affect opinions, many talks have been done by political parties. Social media is a very effective platform where a vast number of people can be readily reached. Therefore, it is particularly focused for sharing ideas about Brexit. Millions of tweets have been posted on the subject. Since there are great numbers of expressive tweets, it is easier to find numerous ones having many common parts among themselves.

We collect tweets from 411 politicians from 5 major political parties in United Kingdom. Twitter Search API is utilized to get the latest 3,200 tweets of each politician. For preprocessing, tweets dated before January 1, 2016 are removed.

To represent each politician's stance towards the issue in binary, we utilize off -the-shelf sentiment analysis tool SentiStrength. We assume that overall sentiment score of the tweet implies the opinion of the tweet towards the issue word the tweet contains. To build input tensor, sentiment scores of the tweets of i'th politician with j'th issue using k'th sentiment-expressing words are summed up and put into the i'th row, j'th column and k'th slice entry.

The input data has 411 users and 48 different issues. The data also contains 6,776 keywords. Keywords are derived from a state-of-the-art sentiment word list. Occurrence of each keyword is counted. Then, most frequent 1000 keywords are selected, by keeping number of users and issues stable.

This results in a maximum of 411 x 48 x 1000 = 19; 728; 000 possible hyperedges in the input data. This is the size of 3D matrix constructed from nodes. The total number of hyperedges in the input data is 26,624. The rest of it is sparse. Thus, the density of hyperedges in the input is roughly 0.001. More than 60% of the hyperedges have negative label. Since the negative hyperedges dominates the positives and the original algorithm mines clusters with high positive density, the labels of hyperedges are switched in order utilize the negative feelings rather than the positive ones in determining the clusters.

A summary of observations are as follows:

- Since most tweets had negative sentiments, we swapped positive and negative signs of the edges in order to obtain correlation between users, issues and negative sentiment words they use on these issues.

- Cluster sizes are much smaller since we only try to construct clusters with high relationships among the nodes of 3 dimensions. Most nodes are not included in these clusters. The largest cluster we obtained with the lowest density requirement, as 25% for minimum negative signed edges, and up to 10% for maximum positive signed edges, was with 24 users on 3 issues, using 22 different negative sentiment words. For this cluster and other clusters, with these density constraints, we have obtained the resulting user clusters containing members from different parties.
- As we have tightened the density constraints we have obtained much smaller and less number of clusters from the experiments. With high density constraint and/or larger minimum size constraint, it was not possible to obtain many clusters. Therefore, we have obtained only a single cluster from most of our tests. As these constraints were slightly relaxed it became possible to generate more clusters. Moreover we have clearly seen that 3 issues stand above almost all other issues, namely “EU", “tax", and “Brexit". Since our method allows node sharing among clusters, we have seen that when more than one cluster is obtained from an experiment, either all, or at least two out of these three issues are in the issue dimension of these clusters. We have also observed that users are distributed to different clusters due to sentiment words they choose to use as they express their feelings against these issues.
- In the experiments with higher negative density requirements the users were mostly from Labour and Conservative parties. Even though these users were from different parties, when we look at their tweets we have seen that they express their negative feelings with similar sentiment words against common issues. Since the number of users using common sentiment words was small, the cluster sizes turned out to be small as well.

5. CONCLUSIONS

In the 4 year term the collaboration between Prof. Toroslu on the Turkish side and Prof. Davulcu on the US side was very successful. Project funds were used to make visits, and participating academic conferences. The results of these collaborations had been published as 5 major conference papers and 1 prestigious journal paper. In the last part of the project the collaboration has been extended by including Prof. Jeremy Jordan from Air Force Institute and Technology, Dayton, OH as well.

Although the project had officially started in September 2014, the first installment of the project fund had been received in May 2016. After the whole fund has reached to METU project account, half of it has been transferred to the project account under the control of co-PI Prof. Davulcu at ASU.

Both teams are continuing the collaborations after the completion of the project. They are also planning to apply for follow-up research projects through the same agency or another one.

REFERENCES

[1] Sefa Sahin Koc, Mert Ozer, Ismail Hakki Toroslu, Hasan Davulcu, Jeremy Jordan, 2018, Triadic co-clustering of users, issues and sentiments in political tweets. *Expert Syst. Appl*. 100: 79-94 (2018).

[2] Riza Aktunc, Ismail Hakki Toroslu, Pinar Karagoz, Event Detection by Change Tracking on Community Structure of Temporal Networks. *ASONAM 2018*: 928-931, Barcelona, Spain.

[3] Amin Salehi, Hasan Davulcu, Detecting Antagonistic and Allied Communities on Social Media. *ASONAM 2018*: 99-106, Barcelona, Spain.

[4] Koc, S. S., Toroslu, I. H., Davulcu, H., 2016, Co-Clustering Signed 3-Partite Graphs, *ASONAM 2016: 945-948*, San Francisco, CA, USA.

[5] Aktunc, R., Ozer, M., Toroslu, I.H., Davulcu, H., 2015, A Dynamic Modularity Based Community Detection Algorithm for Large-scale Networks, *ASONAM 2015: 945-948* Paris, France.

[6] Alzahrani, S., Alashri, S., Koppela, A., Davulcu, H., Toroslu, I., 2015, A Network-Based Model for Predicting Hashtag Breakouts in Twitter, *Proceedings of the International Conference on Social Computing, Behavioral-Cultural Modeling, & Prediction (SBP15)*, pp. 3-12, Washington DC, USA.

Expert Systems With Applications 100 (2018) 79–94

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

Triadic co-clustering of users, issues and sentiments in political tweets

Sefa Şahin Koç[a], Mert Özer[b], İsmail Hakkı Toroslu[a,*], Hasan Davulcu[b], Jeremy Jordan[c]

[a] Computer Engineering Department, Middle East Technical University, Ankara 06530, Turkey
[b] School of Computing, Informatics, Decision Systems Engineering, Arizona State University, Tempe, AZ 85287, Turkey
[c] Department of Mathematics and Statistics, Air Force Institute and Technology, Dayton, OH 45433, Turkey

ARTICLE INFO

Article history:
Received 7 October 2017
Revised 25 January 2018
Accepted 26 January 2018
Available online 2 February 2018

Keywords:
Social network analysis
Co-clustering
Hypergraph
3 partite graph
Sentiment analysis

ABSTRACT

Social network data contains many hidden relationships. The most well known is the communities formed by users. Moreover, typical social network data, such as Twitter, can also be interpreted in terms of three-dimensional relationships; namely the users, issues discussed by the users, and terminology chosen by the users in these discussions. In this paper, we propose a new problem to generate co-clusters in these three dimensions simultaneously. There are three major differences between our problem and the standard co-clustering problem definition: a node can be a member of more than one clusters; all the nodes are not necessarily members of some cluster; and edges are signed and cluster are expected to have high density of positive signed edges, and low density of negative signed edges. We apply our method to the tweets of British politicians just before the Brexit referendum. Our motivation is to discover clusters of politicians, issues and the sentimental words politicians use to express their feelings on these issues in their tweets.

1. Introduction

Although social networks have been around for quite some time, Facebook and Twitter have actually made them readily available to mainstream users. These top social network sites are among the most visited websites in the world. This trend introduces new and interesting problems, the most well-known being the determination of communities or user clusters within these social networks.

The community detection problem or user clustering are among the most popular recent research topics, and many different versions of the problem have been studied. However, co-clustering is still relatively new and less explored than other clustering problems. Most of the research is limited to the 2-dimensional version of the co-clustering problem, also known as 2-mode clustering or biclustering, see (Angiulli & Pizzuti, 2005; Dhillon, 2001; Giannakidou, Koutsonikola, Vakali, & Kompatsiaris, 2008; Zha, He, Ding, Simon, & Gu, 2001). In this work, we focus on the 3-dimensional co-clustering problem, which is motivated by real-life social network data.

On many social network sites, users share their views on a variety of subjects. This simple data set can naturally be modeled as a 3-dimensional relationship which includes users, topics on which users share their ideas, and the keywords they use to express their feelings on the subjects. Users may be clustered according to the similarity of feelings on the same issues. Also, users with different political views or from different social groups, may prefer to use different keywords with the same negative or positive sentiments in order to express the same kind of feeling on the same issue. Therefore, in clustering the people, it is important to use the feelings of the people on the issues, as well as utilizing keywords they use to express these views.

In this work, we have collected tweets of preselected groups of users on specific issues, and subsequently created a 3-dimensional data model by extracting the sentiment keywords used by these users on the selected issues. More specifically, our selected group of people are politicians from the UK, and the issues are hot subjects discussed just before the "Brexit" referendum. Using this data, we have created a 3-D data set, whose dimensions correspond to the people, the issues and keywords, and the values of each element of this 3-D data. If the person had written a tweet on the issue including that keyword, the values are the sign of the sentiment keyword (as positive or negative), else the value is empty. This 3-D data can be modeled as a tri-partite hypergraph.

The aim of this paper is to propose an effective method which generates tri-clusters from hypergraphs with signed tripartite hyperedges. Our focus data set is signed 3-D relationships obtained from users' feelings, represented by the sentimental keywords (ei-

* Corresponding author.
E-mail addresses: sefa.koc@metu.edu.tr (S.Ş. Koç), mozer@asu.edu (M. Özer), toroslu@ceng.metu.edu.tr, toroslu@metu.edu.tr (İ.H. Toroslu), hdavulcu@asu.edu (H. Davulcu), Jeremy.Jordan@afit.edu (J. Jordan).

https://doi.org/10.1016/j.eswa.2018.01.043
0957-4174/

ther positive or negative) they use in their tweets on the issues. The preliminary version of this work is presented in an extended abstract in Dawande, Keskinocak, Swaminathan, and Tayur (2001), where the sketch of the tri-partite clustering method is given and tested on toy data sets.

One of the most well-known and general purpose co-clustering methods is spectral clustering, which could be applied to our signed 3-D data as well (Gao, Liu, Zheng, Cheng, & Ma, 2005; Long, Zhang, Wu, & Yu, 2006). However, we have developed a new co-clustering algorithm customized to meet the requirements of our problem for 2 main reasons. These reasons are as follows:

1. Typical co-clustering algorithms, such as spectral clustering algorithms (Ng, Jordan, Weiss et al., 2001), include all the nodes of all the dimensions of the graph in the clusters generated for these dimensions. However, some of these nodes might have very few relationships or they may not have sufficiently common relationships with other nodes. Thus, we may want to exclude such nodes from the clusters generated. In other words, we don't need to include all the nodes in clusters formed from tri-partite graph if they do not contribute to the clusters. The purpose of all clustering algorithms are to determine subgraphs with strong connections/relationships among nodes. However, when all the nodes are forced to be included in clusters, their connections weaken. Furthermore, we may want to define the minimum amount of connection in subgraphs as a constraint. This means, we have to construct clusters by excluding some of the nodes that violate these constraints. In this way, we can obtain denser and more meaningful clusters.
2. Again, typical co-clustering algorithms, like the spectral clustering method, place every node in exactly one cluster. In our data set, a node may be in more than one cluster. For example, we may detect a person which belongs to two different social groups. Thus, we have to allow overlaps of nodes (not hyperedges) in clusters.

We define this new form of clustering method as *triadic co-clustering*. A full definition of the method will be given in the next section. In this paper we present the following:

1. A full fledged system developed to crawl and collect tweets of selected users on selected issues, and then construct a 3-D relationship among user issue-sentiment keyword triples by parsing these tweets and extracting the issue-sentiment keyword relationships.
2. A flexible tri-partite clustering algorithm (*triadic co-clustering*) with parameters to control the minimum density and the minimum size of clusters.
3. The clustering method has been applied on real data sets, more specifically on the tri-partite graph obtained from "Brexit" tweets, and its effectiveness has been shown both empirically and by the coverage metric defined.

For the dataset we studied in this paper, clusters cannot overlap. This means, for a triple (user u_i, issue i_j, sentiment keyword k_k), all of the following cannot happen at the same time: $u_i \in C_{ux}$, $u_i \in C_{uy}$; $i_j \in C_{ix}$, $i_j \in C_{iy}$, $k_k \in C_{kx}$, and $k_k \in C_{ky}$, where C is a cluster. However, one user, one issue, or one sentiment keyword may be in more than one cluster simultaneously, since, for example, a specific user's feelings toward some issues can be the same as with more than one different set of people (different people clusters), so he can be in both clusters at the same time. Again considering tweets, the same sentiment words may be used by people with different clusters corresponding to different political camps. They may even use the same keywords on the same issues. For the people dimension, a person who appears in more than one cluster may be interpreted as a person close to more than one different political camp.

The rest of the paper is organized as follows. Section 2 discusses current related literature. Section 3 introduces the Triadic Co-Clustering algorithm. Section 4 presents experiments and Section 5 concludes the paper.

2. Related work

In a tri-partite graph, a hyperedge represents the relationships among the three nodes it connects. One of the well-known examples of the tri-partite network relationship from the literature is a social tagging system, which contains three types of nodes (users, tags, resources) (Lu, Chen, & Park, 2009). In this relationship, a hyperedge represents a user annotating a resource with a tag. In order to determine users with common interests on resources, tri-partite clustering may be applied on this hypergraph. As another example, in a biological system, a level of gene in a sample at a particular time can be represented as a tripartite hyperedge. On this hypergraph, tri-partite clusters represent genes showing common characteristics in samples at common time slots (Zhao & Zaki, 2005).

It has already been shown that determining bi-clusters with maximum sizes from a bipartite graph is NP-hard (Dawande et al., 2001). Thus, finding maximum size tri-clusters from tri-partite graphs is also NP-hard, and, works in literature (Lu et al., 2009; Zhao & Zaki, 2005; Zhu, Galstyan, Cheng, & Lerman, 2014) typically propose heuristics to determine clusters.

For example, in Zhu et al. (2014), tri-clusters are constructed by first generating biclusters between each pair of three partitions, and then, by matching each bicluster with the two others in order to construct tri-clusters. However, this approach is very costly. Another work has proposed a faster method (Zhao & Zaki, 2005). In this approach, two partitions are selected, then biclusters of these bipartite graphs are constructed. In order to construct final tri-clusters, each of these biclusters are iterated on the third partition. Since the first two partitions are fixed, this approach has bias against the third partition. In the works of Lin et al. (2009) and Liu and Murata (2010), tri-partite clusters correspond to one-to-one relationships among the nodes. On the other hand, as in social tagging system, a group of users may tag multiple sources with the same set of tags, which represents a many-to-many relationship.

The overlapping clustering concept has been studied as well. There are several works, such as (Blondel, Guillaume, Lambiotte, & Lefebvre, 2008; Gregory, 2010; Lancichinetti, Fortunato, & Kertész, 2009; Li, Xie, Xin, & Mo, 2017; Rhouma & Romdhane, 2014; Rosvall & Bergstrom, 2008; Zhou, Lü, Yang, Wang, & Kong, 2015) dedicated on finding overlapping communities in large graphs. These methods are all heuristic based approaches and all of them approach the problem very similarly. However for higher dimensions, such as 2 or 3, the concept of overlapping is quite different. In the higher dimensional version of the clustering problem, where co-clusters are constructed for each dimension simultaneously by relating them with each other, we can define overlapping of clusters in a dimension as a more constrained concept. In this version of overlapping, co-clusters related with each other may have some overlaps with other clusters in their dimensions, however, all co-clusters cannot have common entities with another set of co-clusters at the same time. We define this version more formally in the following section.

There are also some works focused on partitioning of signed bipartite graphs (Gokalp, Temkit, Davulcu, & Toroslu, 2013; Omeroglu, Toroslu, Gokalp, & Davulcu, 2013). The main difference between these works and our research is related to the density of the bi-partite graphs. Both of these works (Gokalp et al., 2013; Omeroglu et al., 2013) aim to distribute the whole set of nodes of both partitions into clusters, since their main assumption is the bi-partite graph is complete and each node of one partition have

edges to every node of the other partition with a negative or positive sign. If there are a small number of missing edges, then, they are replaced by unsigned edges.

Another paper uses ternary (3-dimensional) relationships also (Missaoui & Kwuida, 2011). However, rather than clustering, the aim of this paper is to find frequent triadic association rules.

To the best of our knowledge, the most similar work to our method is presented in Ignatov, Gnatyshak, Kuznetsov, and Mirkin (2015) paper. This paper introduces triadic clustering also. However, they do not have sign on hyperedges and clusters with overlapping nodes are not considered. They have also performed experiments on data sets such as IMDB and Bibsonomy. Most of these data sets have less than 30 objects at each dimension, or the total number of hyperedges are less than 4000. Example clusters obtained from IMDB dataset has also been presented in the paper.

In our algorithm, unlike most of the above mentioned methods, we address 3 dimensional input data with labeled (either positive or negative) hyperedges. Due to this characteristic of the input, the algorithm considers density of labeled hyperedges and it aims to construct tri-dimensional clusters with high density of positive edges while minimizing the density of negative ones in clusters. Secondly, our algorithm is not biased towards any dimension. This feature increases its applicability in different domains. Thirdly, the nodes can be shared among clusters as long as hyperedges were not repeated in clusters. In some applications, this node sharing helps us to discover more useful clusters.

3. The TRIADIC CO-CLUSTERING algorithm

In a typical co-clustering problem there are N dimensions (usually 2 or 3) and all the items of these N dimensions are clustered simultaneously by using the relationships among the items of different dimensions. For example, in the case of N=2, the dimensions correspond to the partitions of a bipartite graph, and the relationship used for clustering is defined by the edges between two partitions. This means, for the 2-cluster case, all the nodes of both partitions will be placed into some clusters which are formed simultaneously. However, this approach does not work well when the relationship is too sparse. Consider a bipartite graph with a very few edges connecting two partitions. In this case most nodes from both partitions may have no edges incident to them. If our aim is to capture the clusters defined by the edges, then we should exclude the nodes with no or very little connections while forming the clusters. For example, consider a bipartite graph, where, one of the partitions with size 1 million represents users, the other partition with size 10 thousand represents movies, and the edges between two partitions define the relationship of people watching the movies. This bi-partite graph will most likely be very sparse. If we are trying to determine a group of people with similar taste in movies, they should have watched similar movies. When such a group of people have been determined, the common movies they have watched would also have been determined simultaneously, as two clusters from two different partitions. Notice that, we are talking about forming two clusters in two partitions simultaneously, such that the relationship between these two dimensions is expected to be dense. Also, in this formulation, a cluster of one partition will have a one-to-one relationship with a cluster from the other partition. We can assume that only a small portion of movies and people will have a chance to be in such clusters, thus, leaving most movies and most people without being in any cluster at all. Furthermore, some people may have common taste with more than one group of people, or, some movies might be in the watch list of more than one group of people, implying there can be overlap between clusters of a partition. However, we cannot have the same person and the same movie repeated in two clusters simultaneously.

Table 1
Cluster control parameters and constraints.

Symbol	Meaning
Input parameters	
ϵ_p	Minimum h_+ edges ratio parameter in a cluster
ϵ_n	Maximum h_- edges ratio parameter in a cluster
λ_i	Minimum size of dimension i parameter in a cluster
Cluster properties	
L_i	Number of nodes for type i in a cluster (size of type i)
C_p	Number of h_+ edges in a cluster
C_n	Number of h_- edges in a cluster
Constraints	
$\epsilon_p \leq \frac{C_p}{L_1 \times L_2 \times L_3}$	Ratio of h_+ edges constraint in a cluster
$\epsilon_n \geq \frac{C_n}{L_1 \times L_2 \times L_3}$	Ratio of h_- edges constraint in a cluster
$L_i \geq \lambda_i$	Size for dimension i constraint in a cluster

In this work, we propose a feasible solution to this problem. Even though this problem can be defined for any dimension, we focus on the three dimensional case. Our modeling can easily be reduced to 2 dimensions, or can be extended to higher dimensions with a little additional effort. Below we define the 3-dimensional form formally.

Definition (Triadic Co-Clustering): Given 3 sets U_1, U_2, U_3 and 3D-relationships $U_1 \times U_2 \times U_3$ among the items of these 3 sets (i.e., hyperedges *hyperedge* (x, y, z) such that $x \in U_1$, $y \in U_2$, $z \in U_3$), density and size thresholds as ϵ and λ_i for each dimension, *triadic co-clustering* determines k 3-clusters as $\{(C_{11}, C_{21}, C_{31}), (C_{12}, C_{22}, C_{32}), \ldots, (C_{1k}, C_{2k}, C_{3k})\}$ satisfying the following:

- Clusters satisfy size constraint. That is for each cluster C_{ij}, $size(C_{ij}) \geq \lambda_i$.
- Clusters satisfy density constraints. That is for each 3-cluster (C_{1j}, C_{2j}, C_{3j}), $number\_of\_hyperedges\ (C_{1k} \times C_{2k} \times C_{3k}) \geq \epsilon$.
- No common hyperedges. That is, if $x \in U_1$, $y \in U_2$, $z \in U_3$, then, it is not possible to generate two different 3-clusters as (C_{1i}, C_{2i}, C_{3i}) and (C_{1j}, C_{2j}, C_{3j}), such that, $x \in C_{i1}$, $y \in C_{i2}$, $z \in C_{i3}$ and at the same time $x \in C_{j1}$, $y \in C_{j2}$, $z \in C_{j3}$.

The first item in the above definition enforces forming dense clusters with strong relationships among the sets from three different dimensions. The second item prevents forming trivial and very small clusters satisfying the density constraints. Finally, the third item allows node-overlapping among the clusters of the same dimension while disallowing hyperedge-overlapping among 3-clusters.

In this work, we extend this problem with one more feature by defining signs (positive or negative) for hyperedges. Therefore, in this signed version, there will be two density parameters to maximize the number of positive edges while minimizing the number of negative edges (or vice versa) in clusters.

The input parameters, together with related cluster features and the definitions of the size and the ratio constraints, are given in Table 1. Since, the positive sign of the hyperedge means three nodes forming this edge agree, and the negative sign means they disagree, the idea is to form co-clusters on three dimensions using this agreement relationship and maximizing the agreement while minimizing the disagreement among the nodes of co-clusters. Furthermore, we want to generate the largest possible clusters. So, we have two conflicting objectives. For a given tri-partite graph, we would like to construct the largest possible tri-clusters, as well as generate clusters with maximum density of positive signed hyperedges and minimum density of negative signed hyperedges. There is a trade-off between these two objectives. By trying various values of the parameters in Table 1, clusters with different properties/qualities may be generated. As expected, only very small size or trivial perfect clusters can be generated with full positive la-

beled edges. In order to generate more meaningful and useful clusters with larger sizes, we may tolerate some negative labeled edges in clusters and reduce the minimum positive edge ratio requirement.

To simplify the process, we have generated a fully connected tri-partite graph by adding hyperedges with no sign between all 3 nodes from 3 different dimensions, if they are not already connected in the original graph with negative or positive signed edges. We have developed a simple and efficient greedy heuristic in order to generate clusters satisfying the above mentioned constraints. The general idea is to start with the whole graph as a cluster, then trim less effective nodes until ratio constraints are satisfied, or it cannot satisfy the size constraint. Effectiveness of the nodes is also defined using simple formulas which will be discussed below. Our method works as follows:

1. Start with a single co-cluster of the whole graph and trim the least effective node from it in order to increase its positive density and decrease its negative density as much as possible. Notice that this trimming operation reduces the size of the cluster.
2. Repeat this trimming operation until a cluster is obtained satisfying both the density constraints and the cluster size constraints, or until one of the minimum cluster size constraints are violated.
   - If the obtained cluster satisfies all the constraints, it is added to the cluster list. Then, in order to remove the hyperedges used in this cluster, least effective nodes incident to each hyperedge are removed from the graph. The process then repeats itself from the beginning using the remaining graph.
   - If the obtained cluster violates one of the size constraints, all node removals from this iteration were backtracked, and only one hyperedge with negative sign is selected from the graph, and one of the nodes incident to that edge is removed from the graph. Following this, the process repeats itself from the beginning using the remaining graph.

In this paper, we use the notations given in Table 1. Triadic Co-Clustering Algorithm takes a set of hyperedges, Γ as an input, such that each hyperedge h connects three nodes from three different types $U = (U_1, U_2, U_3)$. Fig. 1 illustrates hyperedges given as 3D matrix. These hyperedges have either positive or negative labels, represented by positive and negative signs respectively in Fig. 1. Remaining entries (white cells) correspond to node triples without connecting hyperedges. In the example, nodes are $\{\{a_1, a_2\}, \{b_1, b_2, b_3, b_4, b_5\}, \{c_1, c_2, c_3, c_4, c_5\}\}$ from types U_1, U_2, U_3 respectively. An example cluster obtained from the input graph given in Fig. 1 is depicted in Fig. 2.

The aim of Triadic Co-Clustering is to find tripartite clusters of hyperedges with highly positive labels. To be a valid tripartite cluster, it has to satisfy threshold values for both density and size. The density threshold values are ϵ_p and ϵ_n, such that $0 \leq \epsilon_p, \epsilon_n \leq 1$, $(\epsilon_p + \epsilon_n) \leq 1$. The former one represents the minimum ratio density of positive hyperedges (h_+) among all possible hyperedges (i.e., there may be $L_1 \times L_2 \times L_3$ number of possible hyperedges for a cluster with size (L_1, L_2, L_3), where L_i is number of nodes with U_i type in the cluster). If C_p is the number of h_+, then:

$$\epsilon_p \leq \frac{C_p}{L_1 \times L_2 \times L_3}, \tag{1}$$

If $\epsilon_p = 1$, generated tripartite clusters become tripartite cliques as well. ϵ_n is the value to control the density of negatively signed hyperedges (h_-). If C_n represents the number of h_-, then:

$$\epsilon_n \geq \frac{C_n}{L_1 \times L_2 \times L_3}, \tag{2}$$

shows the maximum allowed tolerance of h_- in a cluster if $\epsilon_n \neq 0$.

In order to prevent constructing very small clusters, λ_i is defined, such that:

$$L_i \geq \lambda_i, \tag{3}$$

for $1 \leq i \leq 3$, and this constraint should also be satisfied by every cluster.

The input parameters presented in Table 1 have different effects on the structures and the qualities of the clusters constructed by the algorithm. By adjusting them, different clusters may be constructed. Therefore, how these parameters are going to be adjusted can be very important. To do this adjustment well, both the structure of the input data and what is expected from the clusters must be known in advance. These metrics are: density of positively and negatively labeled edges, percentage of sparseness, and sizes of each dimension. For densely connected input data, density parameters can be chosen as high, or depending on the requirements, they can also be set to a lower value as well. If it is known that positive connections are dominant, then ϵ_p can be set to a higher value. On the other hand, for a sparse data set, the algorithm may behave more severely on eliminating potential clusters. If ϵ_p is not low enough or λ_i is high, then no clusters may be found. In order to be able to find the best settings of these input parameters, some exploration of potential values may be needed.

Since in our problem, we have both positive and negative labelled edges, without some apriori knowledge about their densities, arbitrary settings of the relevant density parameters may produce uninteresting clusters. We assume that interesting clusters will have high density of positive and low density of negative edges. For example, if the input graph is very sparse, we may obtain trivial clusters with a very few nodes only, if the (positive) density parameter is set to a high value. On the other hand, for dense graphs, a small (positive) density parameter may generate a giant single cluster including almost all the nodes. The difference between the densities of positive and negative edges in the graph is also very important. For example, if the negative edge density is close to the positive edge density in the graph, relatively high negative density ratio may produce clusters with so many negative edges, maybe very close to the number of positive edges. Typically, such clusters are not considered as interesting.

In addition to the density, the dimension sizes of the input graphs are also very important. Since each dimension size can be different, their corresponding size constraint parameters may also be set to different values. While we can choose a small size constraint for clusters for a dimension with small size, we can set the size constraint of larger dimension to a higher value.

Since dimension size parameters and density parameters conflict with each other, some kind of exploration of these parameters are needed in order to be able to find more interesting clusters. As we increase the size parameters it becomes more difficult to find high (positive) density clusters. In general, we are not interested in very small clusters even if they have high densities. Therefore, a simple exploration process may start with high density and large size parameters and reduce them to values that produce clusters with acceptable densities and non-trivial sizes.

Triadic Co-Clustering algorithm (Algorithm 1) starts by generating a potential cluster α which contains all hyperedges in Γ. The main loop (lines from 3 to 23 in Algorithm 1) iteratively constructs clusters satisfying both size and density constraints.

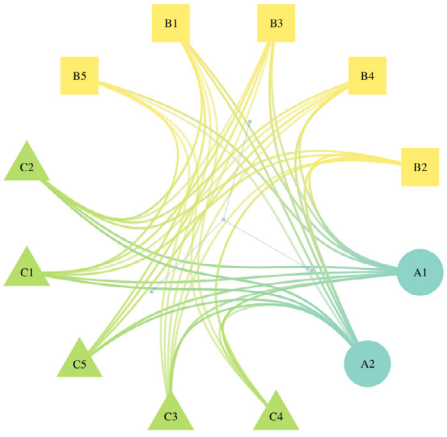
It begins with the remaining nodes of the graph and removes least effective nodes (through while loop in lines 5–7), until either all constraints are satisfied (conditions of the if statement at lines 9 and 10), or until the graph becomes too small to satisfy minimum size constraints due to these node removals.

If both the density and size constraints are satisfied, the remaining nodes of the graph form a cluster which is added to the cluster list (at line 11). In addition, its edges (all signed edges) are

a_1	b_1	b_2	b_3	b_4	b_5
c_1		−		+	−
c_2	−	+	+	+	
c_3	+				+
c_4		−			+
c_5	+		−	+	

a_2	b_1	b_2	b_3	b_4	b_5
c_1		+	−		+
c_2	+	+		+	
c_3	+	+	−		−
c_4	−	+	+		+
c_5	+				

(a) Matrix Representation



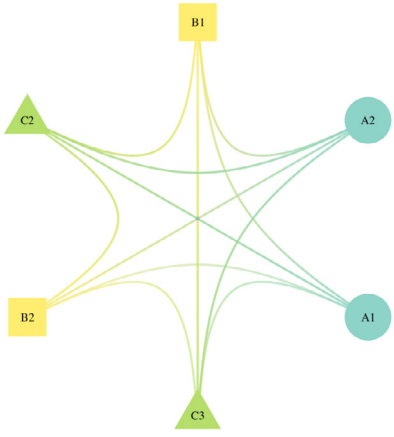
(b) Graph Representation

Fig. 1. Input data.

a_1	b_1	b_2
c_2	−	+
c_3	+	

a_2	b_1	b_2
c_2	+	+
c_3	+	+

(a) Matrix Representation



(b) Graph Representation

Fig. 2. Output cluster, when ϵ_p=0.75, ϵ_n=0.15.

removed from the graph (by adding them to invalid hyperedge list at lines 12 and 13), so they cannot be used in the construction of a new cluster. This way its guaranteed clusters will not share edges, but can still share nodes.

As mentioned above, due to trimming of the graph by removing its least effective nodes to satisfy density constraints, the graph may become too small and unable to satisfy the size constraints any more, and thus, the else branch of the if statement between the lines 15 and 18 of the algorithm is executed. A very simple heuristic is done here. A negative edge is randomly selected from the graph and removed (adding it to an invalid edge set). This way, the next round of iteration of the main loop starts with a graph with one less negative edge, and the chance for generating higher positive density clusters increases.

After this main if statement (from lines 9 to 18 in the algorithm) either one cluster is generated and its edges are added to the invalid edge list, or just one negative edge is added to the invalid edge list. Then, these invalid edges must be removed from the graph. This is done by removing one of the nodes incident to these edges in order to be able to reduce the graph size as well. The procedure at line 19 removes one node for each of the invalid edges from the original graph, then the remaining graph (Γ') is used in the next iteration to discover another cluster from it. However, the remaining graph may be too small, and if it does not satisfy the size constraints (checked at line 20), the process ends and clusters obtained so far are returned (at line 21).

Algorithm 1 Triadic Co-Clustering Algorithm.

Input: Γ : all hyperedges
Input: ϵ_p : lowest ratio value of h_+ in a cluster
Input: ϵ_n : highest ratio value of h_- in a cluster
Input: λ_i : minimum size value for type i in a cluster
Output: $\Re$: list of clusters

```
procedure TriadicCluster(Γ, ε_p, ε_n, λ_1, λ_2, λ_3)
    Γ' ← Γ
    ℜ ← ∅
    loop
        α ← Γ'
        while (SizeCheck(α, λ_1, λ_2, λ_3)) and
            not DensityCheck(α, ε_p, ε_n) do
            ~~ RemLeastEffNode(α)
        end while
        if DensityCheck(α, ε_p, ε_n) and
            SizeCheck(α, λ_1, λ_2, λ_3) then
            ℜ ← ℜ ⋃ {α}
            for each h_{-/+} ∈ α do
                Γ_inval ← Γ_inval ⋃ {h_{-/+}}
            end for
        else
            h_- = RandomNegativeEdge(Γ)
            Γ_inval ← h_-
        end if
        CleanInvalids(Γ, Γ_inval, Γ')
        if not SizeCheck(Γ', λ_1, λ_2, λ_3) then
            return ℜ
        end if
    end loop
end procedure
```

3.1. Density Check

DensityCheck is a sub-procedure with 3 input parameters. They are a potential cluster α and two constraints values ϵ_p and ϵ_n. The purpose of this sub-procedure is controlling the quality of a given cluster due to constraints 1 and 2. If both constraints are satisfied, then this sub-procedure returns **true**. Otherwise, it returns **false** to point out the fact that removing a node is necessary.

3.2. Size Check

SizeCheck is a sub-procedure with 4 input parameters. The first one is a potential cluster α, the others are size constraints for each of 3 types. This sub-procedure checks the size of a given cluster. If the size of each dimension is not below its corresponding limit (constraint 3), then this sub-procedure returns **true**. Otherwise, it returns **false**.

3.3. Remove Least Effective Node

RemLeastEffNode sub-procedure takes one input parameter which is a potential cluster α. It removes a node from α.

Heuristic calculations are used to determine which node to remove. There is a simple approach behind this. If a node is connected by high number of h_+, it should be less likely to be removed. If a node is highly linked by negatively signed hyperedges or it is loosely connected, its probability of being removed is high. Due to the statement of the problem, positively labeled hyperedges are valuable. Therefore, it is not desired to extract them from a cluster. In order to keep them inside, they are valued by a positive number. In contrast, a negative number is given the ones carrying negative labels (Eq. (4)).

$$val(h) = \begin{cases} 1 & \text{if } h \text{ has positive label} \\ -1 & \text{if } h \text{ has negative label.} \end{cases} \tag{4}$$

With the help of it, effectiveness of each node is determined. For this purpose, formula 5 is used.

$$E_{ir} = \frac{1}{S_i} \times \sum_{h \in \alpha} \begin{cases} val(h) & \text{if } r \in h \\ 0 & \text{otherwise} \end{cases}. \tag{5}$$

E_{ir} is the effectiveness value of node r from type i. It is a density calculation of hyperedges which connect a node to the cluster. S_i refers to the maximum number of hyperedges which can be linked to that node. For each type, S_i value differs (Table 2). The lowest effectiveness value indicates the node to remove. An example is given in Fig. 3, where the least effective node is b_3.

Table 2
Maximum possible number of hyperedges for each type.

Type	Value
S_1	$L_2 \times L_3$
S_2	$L_1 \times L_3$
S_3	$L_1 \times L_2$

3.4. Random Negative Edge

RandomNegativeEdge is a sub-procedure taking 1 input parameter, which is a potential cluster α. This sub-procedure helps to invalidate a hyperedge. It finds and returns one which is not significant whether it is covered or not. Predictably, it is the negatively signed one. The sub-procedure traverses all hyperedges and returns the first (h_-) it finds.

3.5. Clean invalids

Invalid hyperedges are not desired to be part of future clusters. The hyperedges of all tripartite clusters previously found are invalid. Moreover, hyperedges returned by RandomNegativeEdge sub-procedure are added to the list of invalids. CleanInvalids sub-procedure aims to remove all invalid hyperedges (Γ_{inval}) from (Γ'). Γ is copied as Γ' at the beginning of this sub-procedure. As a result, (Γ') does not contain any hyperedge in Γ_{inval}, and this sub-procedure terminates. Then, the algorithm searches a valid cluster inside Γ'.

In order to remove a hyperedge, one of the nodes linked by this hyperedge should be removed. In this manner, to remove an invalid hyperedge, CleanInvalids sub-procedure looks for a node, then removes it. It repeats the same removing action until no invalid hyperedge is left in Γ'. While selecting a node, it uses a heuristic that reduces side effects of removing nodes on the cluster size as much as possible.

On the heuristic calculation, first, the number of invalid hyperedges connected to each node is counted. This value is then divided by S_i where i refers to the type of that node (Eq. (6)). The final value (θ_{ir}) is the density of invalids for that node. If (θ_{ir}) is high, the node r is more likely to be removed.

$$Q_{ir} = \frac{1}{S_i} \times \sum_{h \in \alpha} \begin{cases} 1 & \text{if } r \in h \wedge h \text{ is invalid} \\ 0 & \text{otherwise} \end{cases}. \tag{6}$$

Next, the effectiveness of each node in Γ' for all the valid hyperedges is calculated. This calculation is a modified version of Eq. (5). Additionally it checks if hyperedges are valid. If not, they are not

a_1	b_1	b_2	b_3	b_4	b_5
c_1		−		+	−
c_2	−	+	+	+	
c_3	+				+
c_4		−			+
c_5	+		−	+	

a_2	b_1	b_2	b_3	b_4	b_5
c_1		+	−		+
c_2	+	+		+	
c_3	+	+	−		−
c_4	−	+	+		+
c_5	+				

Fig. 3. Removing node b_3.

counted in the calculation (Eq. (7)). The final value (E^v_{ir}) is the effectiveness value for node r. A node with high (E^v_{ir}) will likely not be extracted.

$$E^v_{ir} = \frac{1}{S_i} \times \sum_{h \in \alpha} \begin{cases} 1 + val(h) & \text{if } r \in h \wedge h \text{ is valid} \\ 0 & \text{otherwise} \end{cases}. \tag{7}$$

(θ_{ir}) is directly proportional with selecting a node to remove, as (E^v_{ir}) value is inversely proportional. Therefore, the final calculation is performed as in Eq. (8). c is a constant value which is generally set to 1.

$$\gamma_{ir} = \theta_{ir} \times \frac{1}{c + E^v_{ir}}. \tag{8}$$

Among all nodes, the one with the highest γ value is removed. The removing node is performed iteratively until there is no invalid hyperedge inside Γ'.

For the input data in Fig. 1, Triadic Co-Clustering Algorithm finds the cluster in Fig. 2 in the first iteration. Then, hyperedges of this newly generated cluster are labeled as invalid. In the next iteration, a new potential cluster Γ' (Fig. 4a) is generated from Γ. But Γ' contains some invalid hyperedges (colored with purple in Fig. 4a). Therefore, Γ' is passed to CleanInvalids sub-procedure which cleans Γ' from invalid hyperedges. First, node c_3 is removed since γ_{2c_3} is $3 \div 0.3 = 30$, is the maximum among γ values. Then, nodes b_2 and b_1 are selected and removed respectively (Fig. 4b, and c). This will result a clean Γ' (Fig. 4d) and the sub-procedure terminates.

3.6. Complexity analysis

Let us assume that the size of each dimension is equal to L. Thus, $S_i = L^2$ and $L^3 = n$, where n represents the maximum potential number of hyperedges. In this algorithm, the main calculations are (E_{ir}) and (γ_{ir}). (E_{ir}) is calculated for each node from each dimension. Therefore, the total number of calculations are $(L \times S + L \times S + L \times S) = 3 \times n$. ($\gamma_{ir}$) value includes ($\theta_{ir}$) and ($E^v_{ir}$). This makes total calculation number for (γ_{ir}) $6 \times n$. Thus, the asymptotic complexity of (E_{ir}) and (γ_{ir}) is $\mathcal{O}(n)$.

Worst case scenario for Triadic Co-clustering algorithm happens when no tripartite cluster is generated. In this case, RemLeastEffNode is called until SizeCheck returns **false** in line 6. This loop is iterated at most $(3 \times L)$ times. Time complexity of the loop between lines 6 and 9 is $\mathcal{O}(L \times n)$. CleanInvalids sub-procedure has same time complexity as well. The outer loop (lines between 4–24) eliminates a single node in each iteration in worst case. Therefore, until it ends, it iterates $(3 \times L)$ times. As a result, the worst case asymptotic complexity of Triadic Co-clustering algorithm is $\mathcal{O}(L^2 \times n) = \mathcal{O}(n^{\frac{5}{3}})$.

In the best case, the whole input is a complete triadic co-cluster. If the input satisfies density and size constraints (line 6–7), it will be added into $\Re$ (line 12). RemLeastEffNode sub-procedure will not be called. Then, all nodes inside the input will be invalid. Therefore, they will be removed in CleanInvalids (line 20), and Γ' will be empty. Thus, the algorithm will conclude (line 22). As a result, asymptotic running time complexity of Triadic Co-clustering algorithm in best case will be $\mathcal{O}(n)$.

The algorithm concludes when Γ' does not satisfy the size constraint. Therefore, values for λ_i, which are determined by the user may effect the number of iterations, and thus, the running time complexity. In these calculations, we assume (λ_1, λ_2, λ_3) are all set to 1. Furthermore, the running time of the algorithm is highly dependent on the quality of the input. If positivity of the input data is high, large clusters are found in early iterations, and the algorithm quickly converges. If the negativity or the sparseness is high, then RemLeastEffNode operation is called more often. Also, negativity will decrease the chance of finding a cluster in line 10. This will increase the number of iterations of the outer loop (lines between 4–24). In this case, the algorithm will behave closely to the worst case.

4. Experiments

4.1. Experiment metrics

We have performed two different kinds of experiments. We have generated synthetic data sets in order to analyze the behavior of our method, and we have also tested our algorithm with a real data set in order to illustrate the applicability of our method. Synthetic data is denser. Therefore, in these experiments, ϵ_p and ϵ_n parameters are set to relatively high values. Size constraints are kept at minimum in order to not eliminate small clusters. In the end, many clusters are found. Real world data set is much sparser. When we have tried to find larger clusters, the number of clusters obtained became very small. All experiments are performed on MacBook Pro-Mid 2015 (Intel i7 2,5 GHz, 16GB memory).

In our experiments, we have used internal evaluation measures. Namely, to evaluate the baseline and proposed methods, we have used well known clustering quality metrics such as coverage, density, and purity. The definitions of the first two are as follows:

$$Coverage(\Re) = \sum_{\alpha \in \Re} L_1 \times L_2 \times L_3, \tag{9}$$

where $\Re$ is a cluster list. α is a cluster inside $\Re$ and (L_1, L_2, L_3) are sizes of dimensions of α.

$$DensityP(\alpha) = \frac{C_p}{L_1 \times L_2 \times L_3}, \tag{10}$$

$$DensityN(\alpha) = \frac{C_n}{L_1 \times L_2 \times L_3}, \tag{11}$$

where α is a cluster. C_p is number of positive hyperedges and C_n is number of negative hyperedges inside α. (L_1, L_2, L_3) are sizes of dimensions of the given cluster.

a_1	b_1	b_2	b_3	b_4	b_5
c_1		−		+	−
c_2	−	+	+	+	
c_3	+				+
c_4		−			+
c_5	+		−	+	

a_2	b_1	b_2	b_3	b_4	b_5
c_1		+	−		+
c_2	+	+		+	
c_3	+	+	−		−
c_4	−	+	+		+
c_5	+				

(a) Removing First Node

a_1	b_1	b_2	b_3	b_4	b_5
c_1		−		+	−
c_2	−	+	+	+	
c_4		−			+
c_5	+		−	+	

a_2	b_1	b_2	b_3	b_4	b_5
c_1		+	−		+
c_2	+	+		+	
c_4	−	+	+		+
c_5	+				

(b) Removing Second Node

a_1	b_1	b_3	b_4	b_5
c_1			+	−
c_2	−	+	+	
c_4				+
c_5	+	−	+	

a_2	b_1	b_3	b_4	b_5
c_1		−		+
c_2	+		+	
c_4	−	+		+
c_5	+			

(c) Removing Third Node

a_1	b_3	b_4	b_5
c_1		+	−
c_2	+	+	
c_4			+
c_5	−	+	

a_2	b_3	b_4	b_5
c_1	−		+
c_2		+	
c_4	+		+
c_5			

(d) Clean Γ'

Fig. 4. Cleaning invalids.

Purity (Manning, Raghavan, & Schütze, 2008) is formally defined as;

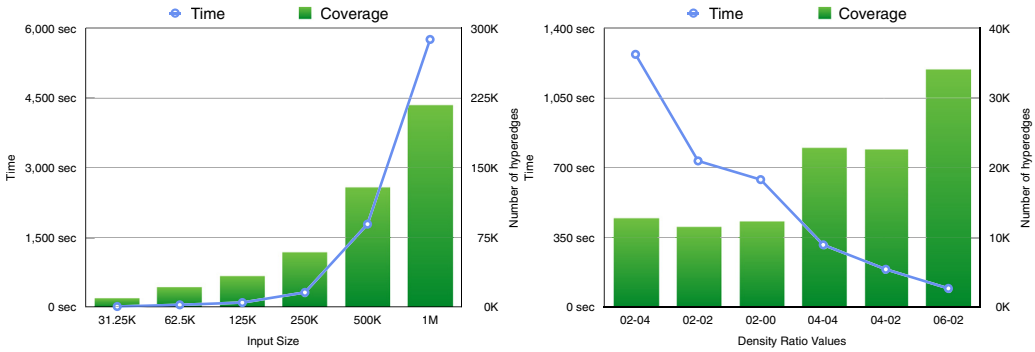
$$Purity(\Re) = \frac{1}{n}\sum_{i=1}^{k} max_j |C_i \cap l_j| \tag{12}$$

where k is the number of clusters that model produces, n is the number of instances, l_j is the set of instances which belong to the ground-truth cluster j, and C_i is the set of instances that are members of cluster i of the model's output.

4.2. Experiments on synthetic data

This experiment is conducted to determine the effectiveness and the scalability of our proposed method. In order to evaluate TRIADIC CO-CLUSTERING algorithm, we generate datasets with varying sizes.

In the first set of experiments, we fix h_+ and h_- density ratios while changing input sizes. We generate 6 sample datasets. Each set contains positive hyperedges with 60%, negative hyperedges with 20%, and 20% empty. $(L_1 \times L_2 \times L_3)$ values for the samples are (31.25K, 62.5K, 125K, 250K, 500K, 1M), respectively. Fig. 5a presents the results. Input parameters are fixed as $\epsilon_p = 0.75$, $\epsilon_n = 0.10$, $\lambda_i = (2,2,2)$. ϵ_p value has been chosen larger than the positivity ratio of the input data. If it was smaller, the whole input would have been accepted as one large cluster. Similarly, ϵ_n value is chosen smaller than the negativity ratio of the input data. Therefore, elimination of negative hyperedges was possible. Additionally, we set the size constraints (λ_i) small in order to find as many clus-



(a) Varying Input Size (b) Varying Density Ratios in Input Data

Fig. 5. Performance graphs of experiments on synthetic data.

ters as possible. The way, we have managed to cover as many hyperedges as possible by these clusters. Test results are plotted in Fig. 5a.

In the second test, we have generated 6 datasets. In this test, we fix the total size as $(L_1 \times L_2 \times L_3) = 125K$ and we have performed tests with varying density ratios for (h_+, h_-) pairs as {(0.2,0.4), (0.2,0.2), (0.2,0.0), (0.4,0.2), (0.4,0.4), (0.6,0.2)}. We did not change any of input parameters for test 2 as well. Chosen ϵ_p value is larger than positivity ratios of all inputs while ϵ_n value is relatively small (which is 0.1). λ_i values are chosen to be small in order to increase the number of clusters found even with small sizes. The results are shown in Fig. 5b. Negativity of an input data effects the execution time of the algorithm poorly. However, if positivity is high, the algorithm finds large clusters and terminates more quickly.

The figures show both execution times and the number of hyperedges included in the constructed clusters. We prefer most (positive) hyperedges to be included in clusters while clusters being non-trivial. The results show that we achieve very high coverage in that sense, since constructed clusters include almost half of the positive signed hyperedges.

4.3. Experiments on brexit data

Since the problem that we define is new, previous datasets made publicly available by other scholars do not meet our constraints. Given the scarcity of publicly available datasets containing hypergraphs with signed tri-partite hyperedges, we decide to exploit well-known social media data source Twitter, thanks to its generous APIs providing researchers social data freely over the last decade. We choose to focus on recent Brexit referendum in the United Kingdom to be able to generate signed tri-partite edges from a social media data.

4.3.1. Data acquisition

We collect tweets from 411 politicians[1] from 5 major political parties in the United Kingdom. Twitter Search API is utilized to get the latest 3200 tweets of each politician. For preprocessing, tweets dated before January 1, 2016 are removed. Then, tweets that do not contain the words shown in Table 3 are eliminated. Number of tweets that contain relevant keywords after preprocessing can be seen in Table 3.

To represent each politician's stance towards the issue in binary format, we utilize an off-the-shelf sentiment analysis tool called SentiStrength[2]. We assume that overall sentiment score of the tweet implies the opinion of the tweet towards the issue word the tweet contains in this work. Whether the assumption holds or not is not the focus of this study, yet some insights based on our experiments are discussed in Section 4.3.2. To build input tensor Γ, sentiment scores of the tweets of the i'th politician with j'th issue using k'th sentiment-expressing words are summed up and put into the i'th row, j'th column and k'th slice entry.

Table 3
Keywords and number of tweets they occur in the dataset.

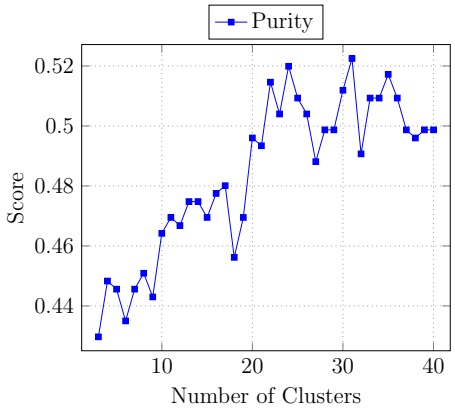
Keyword	Count	Keyword	Count
eu	12,206	#leave	157
Tax	3278	Hospitals	150
#brexit	2362	Gas	147
nhs	2312	daesh	126
#strongerin	2249	Muslims	125
#voteleave	1802	Worker	123
Workers	1250	healthcare	114
Immigration	1066	Tuition	86
#remain	807	Immigrants	72
#takecontrol	460	Visas	50
Debt	456	Abortion	49
Hospital	411	Deport	45
Brussels	396	Immigrant	34
Wage	364	Islam	33
Borders	343	Citizenship	32
Border	271	Deportation	22
isis	262	Vaccination	11
Wages	259	Same-sex	9
Nuclear	247	lgbtq	7
lgbt	232	Monarchy	6
Muslim	223	#remainin	3
Medical	204	samesex	2
oil	200	al-qaeda	1
Taxes	166	lbtq+	1
Total number of tweets:		26,624	

The major 5 political party viewpoints can be summarized as below (Hobolt, 2016);

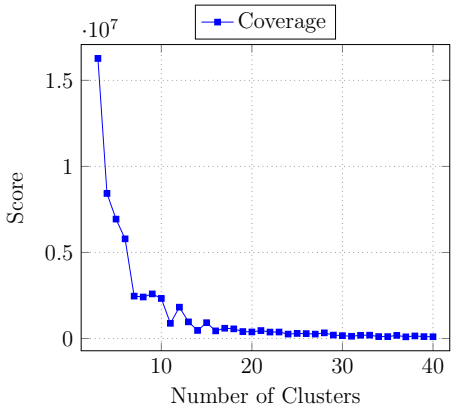
- Labour: Overwhelming majority of Labour Party members campaign for staying in European Union although there were raising concerns about the structure and function of the European Union.
- Conservatives: The leader of the Conservative Party, David Cameron offered the referendum and started the campaign for remaining in EU. There was a clear leaning towards leaving the EU despite the Cameron's efforts.
- Libdem: Liberal Democrats campaigned for staying in the EU.
- UKIP: UK Independence Party was a prominent figure in the referendum campaign. They passionately advocated to leave the

[1] Users' Twitter id lists can be obtained from http://mlg.ucd.ie/aggregation/index.html.

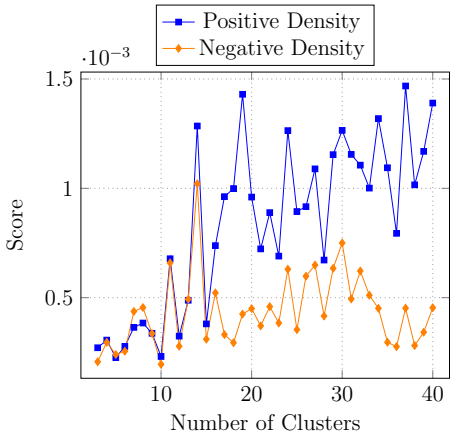
[2] http://sentistrength.wlv.ac.uk/.



(a) Effect of number of clusters ($c = c_1 = c_2 = c_3$) on purity of user clusters



(b) Effect of number of clusters ($c = c_1 = c_2 = c_3$) on coverage of hyper-clusters



(c) Effect of number of clusters ($c = c_1 = c_2 = c_3$) on density of hyper-clusters

Fig. 6. Experimental results for tucker decomposition.

EU. Blocking the refugees from entering the country, opposing international and EU-wide trade agreements, defending UK-born workers' rights over immigrants' rights were standing out as motivating factors in their campaign.
- SNP: Scottish National Party campaigns to stay in EU.

The input data has 411 users and 48 different issues. The data also contains 6776 keywords. Keywords are derived from a state-of-the-art sentiment word list.[3] Occurrence of each keyword is counted. Then, the most frequent 1000 keywords are selected, by keeping number of users and issues stable.

This results in a maximum of $411 \times 48 \times 1000 = 19,728,000$ possible hyperedges in the input data. This is the size of 3D matrix constructed from nodes. The total number of hyperedges in the input data is 26,624. The rest of it is sparse. Thus, the density of hyperedges in the input is $\sim$0.001. More than 60% of the hyperedges have a negative label. Since the negative hyperedges dominate the positives and the original algorithm mines clusters with high positive density, the labels of hyperedges are negated in order to utilize negative feelings rather than the positive feelings in determining the clusters.

[3] http://www.cs.uic.edu/~liub/FBS/opinion-lexicon-English.rar.

4.3.2. Discussions of experiment results

Tucker decomposition results. As a baseline method, we apply Tucker decomposition to Γ to find user, issue and sentiment word clusters in the Brexit data set. To that end, we utilize the Tucker decomposition component (Kolda & Sun, 2008) of MATLAB Tensor Toolbox Version 2.6 of Bader and Kolda (2007) which has the following objective function;

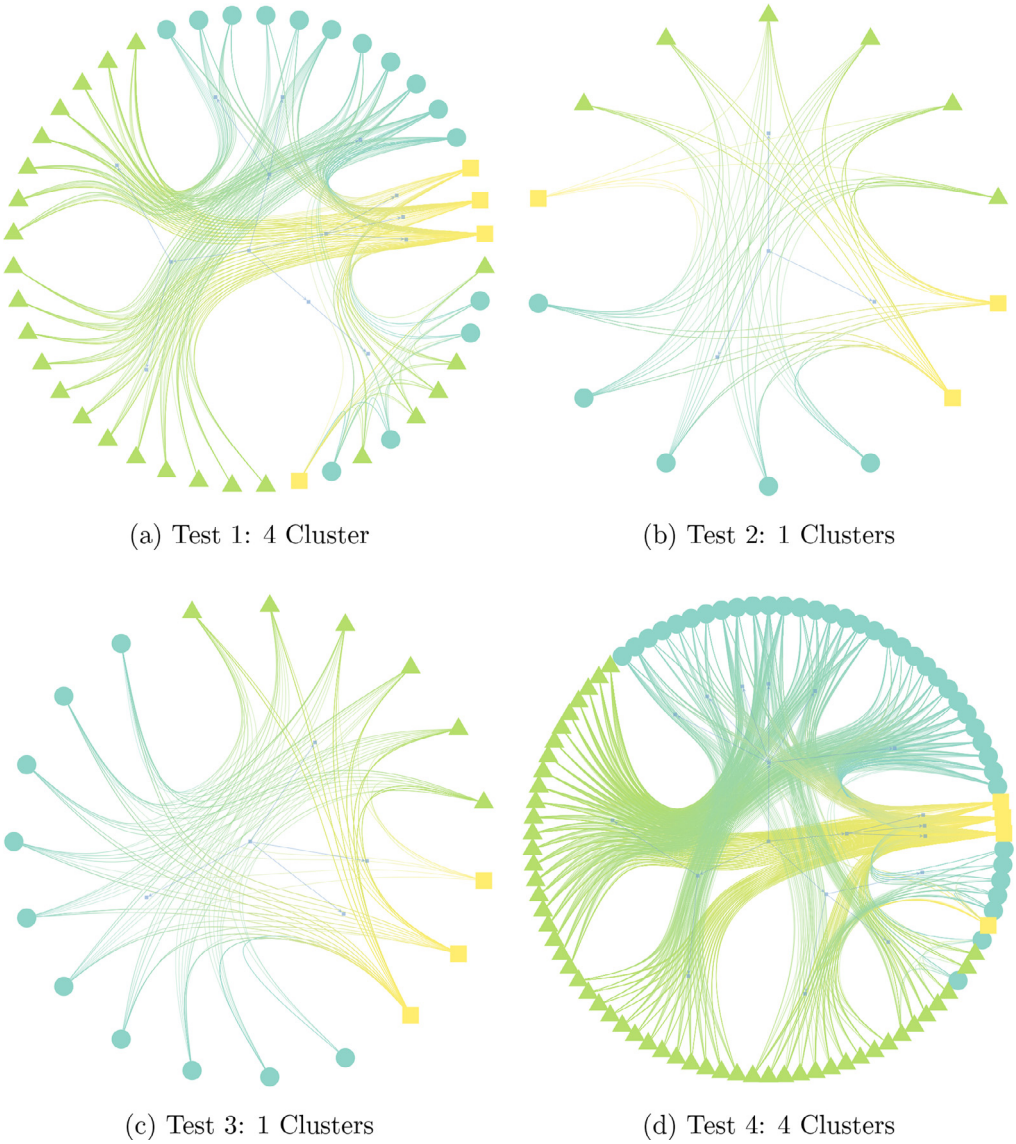
$$\begin{aligned} &\min_{\mathbf{C},\mathbf{U},\mathbf{I},\mathbf{K}} ||\mathbf{\Gamma} - \mathbf{C} \times_1 \mathbf{U} \times_2 \mathbf{I} \times_3 \mathbf{K}||_F^2 \\ &\text{s.t.} \quad \mathbf{\Gamma} \in \mathbb{R}^{u \times i \times k}, \mathbf{C} \in \mathbb{R}^{c_1 \times c_2 \times c_3}, \\ &\quad \mathbf{U} \in \mathbb{R}^{u \times c_1}, \mathbf{I} \in \mathbb{R}^{i \times c_2}, \mathbf{K} \in \mathbb{R}^{k \times c_3} \end{aligned} \tag{13}$$

where u is the number of users, i is the number of issues, and k is the number of keywords.

Our experiment setup for measuring the performance of Tucker decomposition spans different values of c_1, c_2 and c_3 between 3 and 40. For brevity, we set all parameters of c_1, c_2 and c_3 equal to each other.

To evaluate the purity of clusters, we simply check cluster assignments in matrix $\mathbf{U}$. Purity measure of user clusters are reported for different cs (where $c = c_1 = c_2 = c_3$) between 3 and 40.

To be able to evaluate the tucker decomposition against density and coverage metrics, we form hyper-clusters of users, issues and keywords according to the core tensor $\mathbf{C}$. A simple heuristic is followed when choosing which user, issue and keyword clusters to



(a) Test 1: 4 Cluster (b) Test 2: 1 Clusters

(c) Test 3: 1 Clusters (d) Test 4: 4 Clusters

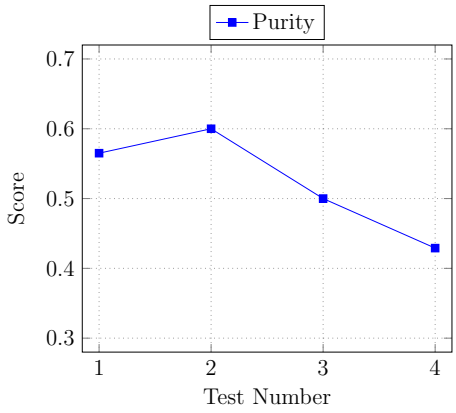
Fig. 7. Test results on Brexit data.

form a hyper-cluster. Top c (where $c = c_1 = c_2 = c_3$) entries of core tensor **C** with highest absolute values are picked in a way where none of the picks share any row, column or slice index. Therefore, each cluster of user, issue and keyword represented in **U, I, K** is assigned to a hyper-cluster. Density and coverage measures of hyper-clusters are reported for different c's between 3 and 40.

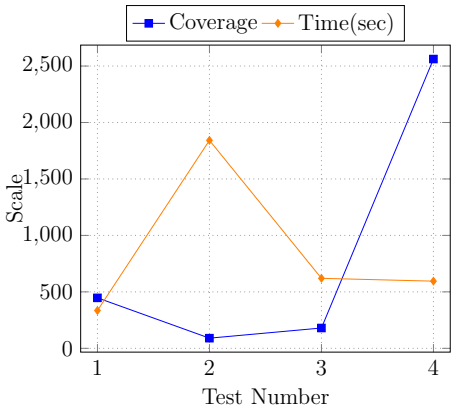
Purity of the clusters found tend to increase with increasing numbers of clusters. It oscillates between 0.43 and 0.52 for different c's. Coverage of hyper-clusters tend to decrease dramatically with increasing number of clusters. This signals the fact that splitting hyper-clusters to further smaller hyper-clusters does not help to find dense sub-clusters. Density of the clusters tend to increase with increasing number of clusters, yet not with any significance. It supports the claim we make about tucker decomposition's inability to find dense sub-clusters with increasing number of clusters.

For qualitative analysis of the Tucker decomposition's performance, with the expectation to determine *remain* and *leave* camps, we focus on the case of 2 clusters for the user dimension ($c_1 = 2$), and 3 clusters for the issues and sentiment words dimensions ($c_2 = c_3 = 3$), respectively. When two opposing camps exist, it is also likely that the other dimensions will have reflections of these two camps as two clusters, and there may be a third cluster for the remaining issues and the sentiment words. Since spectral clustering includes all the nodes of all these dimensions, we obtain very large clusters with very low edge densities as seen in Fig. 6. Notable results that we observed from this experiment are as follows:

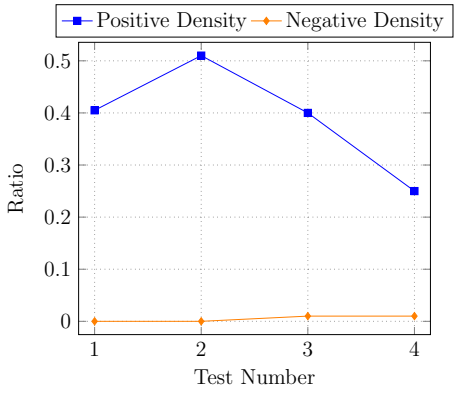
- Issues are unevenly distributed to 3 clusters. The first one contains three of the most popular issues, namely "citizenship", "brussels" and "worker". There is a strong negative reaction towards these issues from the first user cluster. The next cluster contains 11 issues, and towards those issues, there is less negative reaction from the second user cluster. This issue cluster contains issues such as "humanrights", "tuition", "eu". The reaction is not very clear on remaining clusters. Thus, it is not possible to obtain any useful information from issue dimension.
- Sentiment keywords are also unevenly distributed across three clusters. Even the smallest cluster contains 105 sentiment words. Positive and negative sentiment words are also distributed through the clusters. There is no useful result that can be obtained from these clusters either.



(a) Purity score of clusters



(b) Quality of clusters



(c) Plot of Average Density Ratios of Clusters in Test Cases

Fig. 8. Result plots of test cases.

Table 4
Variables for tests with real data.

Test No	λ_i	ϵ_p	ϵ_n
1	(5,3,5)	0.4	0.08
2	(5,3,5)	0.5	0.05
3	(10,3,5)	0.4	0.08
4	(10,3,5)	0.25	0.1

Table 5
Sizes of clusters found in tests.

Num	Test No 1	Test No 2	Test No 3	Test No 4
1	$8 \times 3 \times 8$	$5 \times 3 \times 6$	$10 \times 3 \times 6$	$24 \times 3 \times 22$
2	$5 \times 3 \times 6$			$12 \times 3 \times 13$
3	$5 \times 3 \times 6$			$10 \times 3 \times 10$
4	$5 \times 3 \times 5$			$10 \times 3 \times 7$

- The user clusters obtained from this method are also not very informative. The first cluster contains 133 politicians from a variety of parties. The largest group in this cluster is Labours with 62 members, which is followed by 44 members of the Conservatives. It also contains 8 SNP, 8 Liberal, and 5 UKIP members, as well as 6 members from other parties. The second cluster contains 239 politicians. Labours increase to 95, Conservatives almost double to 90, and other parties also increase, Liberals to 25, UKIP to 11, and SNP to 15. So, these clusters do not give any information about the party membership vs issue relationship either.

Triadic co-clustering results. When we applied our method on the Brexit dataset, we tried different density and cluster size constraints in order to be able to explore different clustering structures. Four different runs are performed on various values for λ_i, ϵ_p and ϵ_n. The values are listed on Table 4 with respect to test numbers. λ_i values are for users, issues, and keywords, respectively. The sizes of clusters obtained from these tests are given in Table 5. Cluster sizes are written in the form of (Users $\times$ Issues $\times$ Keywords).

Since most tweets have negative sentiments, we swap positive and negative signs of the edges in order to obtain correlation between users, issues and negative sentiment words they use on these issues.

On Fig. 7, circles refers to users, while triangles for keywords and squares for issues. Bundles of edges are expressing the diversity. Nodes with similar connections are adjacent. Coverage of clusters can be seen on Fig. 8b. Average density of clusters in each test are plotted in Fig. 8c.

A summary of observations are as follows:

- Purity score shows the homogeneity of clusters. High purity score together with high coverage for results are highly desired. For 4 test cases, purity scores are plotted in Fig. 8a. It can be seen that homogeneity of clusters is around 0.5. We observed

that generally half of the tweets are from same division. Others are mixed.
- Cluster sizes are much smaller since we only constructed clusters with high relationships among the nodes of 3 dimensions. Most nodes are not included in these clusters. The largest cluster we obtain with the lowest density requirement, as 25% for minimum negative signed edges, and up to 10% for maximum positive signed edges, is with 24 users on 3 issues, using 22 different negative sentiment words (test number 4). For this cluster and other clusters, with these density constraints, we obtain the resulting user clusters containing members from different parties. The visual representation of clusters obtained from these experiments are given in Fig. 7.
- As we tighten the density constraints we obtain much smaller and a lower number of clusters from the experiments. With high density constraint and/or larger minimum size constraint, it is not possible to obtain many clusters. Therefore, we obtain only a single cluster from tests number 2 and 3. As these constraints are slightly relaxed, it became possible to generate more clusters, as it can be seen from test number 1. Moreover, we clearly see that 3 issues stand above almost all other issues, namely “EU”, “tax”, and “Brexit”. Since our method allows node sharing among clusters, we see that when more than one cluster is obtained from an experiment, either all, or at least two out of these three issues are in the issue dimension of these clusters. We also observe that users are distributed to different clusters due to sentiment words they choose to use as they express their feelings against these issues.
- In the experiments with higher negative density requirements, the users are mostly from Labour and Conservative parties. Even though these users are from different parties, when we look at their tweets we see they express their negative feelings with similar sentiment words against common issues. Since the number of users using common sentiment words are small, the cluster sizes turn out to be small as well.

The details of all the clusters generated by all of the above discussed experiments are given in Appendix A.

In addition to these general observations, we also investigated the details of the clusters formed by our approach. Consider the first clusters generated from the first experiment, which contains 8 politicians, 3 issues, and 8 keywords. In order to understand this cluster, we need to look into the tweets written by these politicians on these issues including these keywords, which are given in Appendix B.

When we look these tweets we see the following:

- The 8 politicians are distributed to 3 parties as 4 UKIP members, 3 Conservatives, and 1 Labour. 7 out of these 8 have strong feelings against the EU and are for the Leave campaign.
- The three issues, namely, “EU”, “tax”, and “Brexit”, are also very common almost for all clusters formed in our experiments. This is not surprising since these tweets are collected just before Brexit referendum. As it can also be seen from Table 3, these are the top three issues of the collected tweets. The issues with smaller counts did not have enough tweets to form clusters with required densities.
- As it can also be seen from Appendix B these tweets include the same set of negative sentiment words, such as “crisis”, “risk”, “bad”, “worse”, etc. Since these 8 politicians have tweets on the same set of issues and they use the same sentiment words, our method clustered the politicians, issues, and the sentiment words as three clusters of these three dimensions, respectively.
- When the tweets of a Labour member has been investigated, it can easily be seen that he does not have the same political view as the other 7 politicians. Unfortunately, since we have not studied natural language semantic implications, we cannot capture such outlier behaviors. This politician has used the same negative sentiment words in his tweets on the same issues, just like the other seven, but, in a completely reverse way. Since currently we only syntactically analyze tweets, we cannot capture these kind of anomalies.

As it can be seen from the above observations, our approach had been quite successful in finding strong small clusters from large, very sparse data sets. Considering the nature of politicians, each one trying to be different from others, even if they want to say exactly the same thing, they would probably try to be different by using different terminology, focusing on different aspects etc. That is why forming large clusters, even among the same party members, are unlikely. We have demonstrated that there are at least some small sets of politicians with very similar views.

In terms of running times, as expected the spectral clustering method is much faster than our method and determines the clusters in almost real-time. However, spectral clustering does not solve exactly the same problem. In our spectral clustering experiments we have ignored negative signed edges and all there dimensions of clusters are distinct from each other, but, as we have mentioned finding overlapping clusters with comon objects in some of their dimensions is the main novality of our problem. Moreover, since this clustering problem does not require a real-time response, and the quality of the clusters obtained is more important, we aimed to improve the quailty while sacrificing execution time. For the above mentioned experiments, our method took less than an hour to complete.

5. Conclusions

In this paper, a new version of co-clustering problem is defined. In this problem, from a given signed 3-partite graph, clusters with high positive labeled edge, and low negative labeled edge density are determined which potentially can have common nodes. We have proposed a greedy heuristic solution for this problem. The effectiveness of our method has been shown using both synthetic and real data sets. Similar to many data mining applications, the success of our proposed method also depends on the adjustments of several parameters. Thus, apriori knowledge about the structure of the data set, such as its density, dimension sizes etc., are needed in adjusting these parameters properly.

In order to evaluate the quality of the clusters generated from our real-world experiment, we mainly used internal evaluation metrics. As a future work, we plan to use an external evaluation method by obtaining comments from the domain experts, such as political scientists, about the clusters generated.

Furthermore, we are also planning to expand our Brexit experiments using more tweets in order to increase the density of input graphs. Even with relatively small experiments that we have presented in this paper, we were able to obtain very encouraging results by obtaining highly dense clusters. As the input graph gets denser, we are expecting to produce much larger and more meaningful clusters. Moreover, we also plan to use different kinds of social network data sets with 3 dimensions and signed relationships, such as tagged (as like and unlike) images, comments on movies etc. in order to apply our method.

Acknowledgments

This research was supported partially by USAF Grant FA9550-15-1-0004.

Appendix A. Tweets of one cluster

Table A.1
Tweets of 1st cluster in Test 1.

User	Issue	Keyword	Tweet
15157283 : conservative	EU	bad	RT @Vote_LeaveMedia: Professor David Blake gives 10 reasons why staying in the EU ranges from "pretty bad to very, very bad" for pensions h
		crisis	RT @Telegraph: Leaving the EU will halve net migration, boost pay and help to solve housing crisis, according to a new study https://t.co/a
		risk	RT @TelegraphNews: Theresa May warns Tory Brexit rebels that they risk 'incentivising' EU to offer UK a bad deal https://t.co/Y3ekbmMrawht
		risks	RT @bbckamal: Carney says there are greater risks for the continent than for the UK during the #Brexit process. EU wd face financial capaci
		threat	RT @suttonnick: Friday's Telegraph front page: Britain can fight terror threat better outside EU #tomorrowspaperstoday #euref https://t.co/
		threaten	RT @MustBeRead: If leaving the EU would really be so ruinous, asks @PaulGoodmanCH, why did @David_Cameron threaten to do so? https://t.co/Y
		worse	RT @Change_Britain: Jacob Rees Mogg MP reminds people how wrong economists were about the EU Referendum https://t.co/qnXwqi7ULa
	tax		
	#brexit	threat	RT @AndrewRosindell: The biggest threat to #Brexit now is complacency. We need to fight this campaign with every spare minute of the day. W
		worse	RT @BrexitTheMovie: EU protectionism raises the price of goods from the rest of the world and makes us all worse off #brexit https://t.co/6
107254637 : labor	EU	bad	bad: 746810051512393729: RT @TradingJeremy: EU warning that clearing houses should not be in London needs more public airing and explanation. Catastrophically bad f
		crisis	Greek financial crisis hasn't gone away: deal needs to be done by end of May or post-UK referendum on EU membership https://t.co/3sWHXK62LK
		risk	RT @pdacosta: #Brexit talks must be quick, City of London at risk of losing 'EU passport': @ECB's Villeroy https://t.co/XELL6JG7Jo
		risks	Couldn't be clearer from @bankofengland MPC today: leaving EU risks lower UK growth & investment, higher unemployment & prices #StrongerIn
	tax	worse	RT @resfoundation: PT worker on UC cld be 28 a wk worse off compared to tax credits. They'd need to work 10 hrs more to offset losses http
	#brexit	bad	IMF's Lagarde: consequences of #Brexit range from "pretty bad to very, very bad" #EURef https://t.co/PlARumMOaU
		risk	Lots emerged from @IMFNews report on UK economy: 1: echoes dire warnings on #Brexit risk to growth, prices, funding current account deficit
		risks	July @ECB monetary policy minutes out: spots stronger growth risks including #Brexit but no policy loosening for now https://t.co/LFcrMgBlTh
		worse	RT @EuropeElects: UK: Especially young voters are concerned that a #Brexit would leave them financially worse off (TNS poll). #EURef https:
478679663 :ukip	EU	bad	@FredLitten I understand this article bland claims UK entered eu at bad time and did well. The lag in UK simply caught up whilst EU declined
		crisis	Just posted: When will EU and the world recognise that #mi-gration crisis is a @UN issue too https://t.co/KqIRT2WXIu https://t.co/0IpY2iVNXR
		risks	Breaking: Europol reporting that 800,000 migrants in Libya alone waiting to come to EU. Huge security risks & we have no control of borders. (+)
		threat	Read: My article for @heatstreet. The threat to European Peace is the EU itself NOT #Brexit. Hope will beat fear. https://t.co/EsDtqYVNVy
		threaten	Don't be scared and threatened of the EU powerbrokers. The power and size of our market will do the talking.
		worse	Mr Cameron, EU's expansionist foreign policy agenda, provoking Russia has made Europe less safe. Will only get worse as EU grows.
		wrong	Britain will do well outside of the EU: Project fear is wrong says top economist Roger Bottle https://t.co/6HXokYgLj1
	tax	risks	For the City of London, the risks of a regulatory body based in Frankfurt & EU financial transaction tax are far greater than Brexit.
	#brexit	threat	Must Read: EU's own Frontex force recognises terrorist porous borders threat https://t.co/aJOSqD8M3w #Brexit https://t.co/MbNk26VESp
		threaten	Great morning #Brexit bill passed but the Lords threaten to demolish it. Well here's brilliant @SuellaFernandes dem https://t.co/5HGoTgKzCO
		wrong	Major wrong on what #Brexit is about. Brexit is not about isolationism, it's about globalism and mutual cooperation.
1668992125 : ukip	EU	bad	RT @Nigel_Farage: I don't just think the EU has been bad for Britain, I think it's been disastrous for the whole of Europe. https://t.co/yi...
		crisis	RT @V_of_Europe: Juncker: No Matter How Bad Migrant Crisis, Terrorism Gets, We'll Never Give Up Open Borders https://t.co/EdxqwxV67i https:.
		risks	RT @hermannkelly: Farage: Staying In EU Risks More Cologne -like Sex Attacks .@UKLabour @MumsnetTowers #EURef https://t.co/DRqY80eGoi http...
		threat	RT @BBCRealityCheck: Criminal record not enough to reject EU citizens, must pose current threat https://t.co/B4tnV6nK7N #BBCDebate https://...
		threaten	RT @MikeHookemMEP: Blundering Corbyn vows to veto TTIP trade deal threatening NHS - but EU rules say he CAN'T https://t.co/6hztGhlGFh
	tax	wrong	RT @richardcalhoun: Petition: Abolish Inheritance Tax. It is wrong to tax assets that have already been taxed please RT https://t.co/7wJprH...
	#brexit	bad	Remember the 'good day to bury bad news'? Well every I'll in the world now due to #Brexit ... Apparently! https://t.co/8678egiedB

(*continued on next page*)

Table A.1 (*continued*)

User	Issue	Keyword	Tweet
		crisis	RT @LeaveEUOcial: Another Greek crisis is coming. The Eurocrats are trying to hush it up before #Brexit vote: https://t.co/rHZG5t6c0A ht...
121171051 : ukip	EU	crisis	RT @KulganofCrydee: EU migrant crisis 'is colossal': British borders face threat from terrorists and smugglers https://t.co/3PaoM2zKJK
		risk	RT @oflynnmep: What does Mr Carney think about risk of UK being in EU at next € zone crisis? Predecessor Lord King thinks € will lurch from...
		risks	RT @oflynnmep: Janet Daley right to wish Leave would emphasise more the economic risks of remaining in the EU. #VoteLeave
		threat	RT @KulganofCrydee: EU migrant crisis 'is colossal': British borders face threat from terrorists and smugglers https://t.co/3PaoM2zKJK
		threaten	RT @ConHome: If leaving the EU would really be so ruinous, why did @David_Cameron threaten to do so only recently? https://t.co/MCtydB0w1N
		worse	RT @andrealeadsom: EU is making the migrant crisis so much worse, says Defence Minister @PennyMordauntMP https://t.co/pVWlQ1nZRX @vote leav...
		wrong	RT @montie: Tim Farron wrong. This isn't about Little Britain but about Little Europe. The EU is a 28 nation lo-growth zone that is obsesse...
	tax	threat	RT @terencehooson: Chancellor plotting 'punishment' Budget with threat to hike income tax https://t.co/iMldSIkexmvia@MailOnline
	#brexit	bad	RT @Charlton_UKIP: #BREXIT not so bad: UK employ-ment hits record high: fundamentals of the British economy are strong https://t.co/r81UEb4E...
		risk	RT @DavidJo52951945: With open borders to 500m & ISIS terrorists flooding the EU,the EU is a risk to UK security. We're wide open #Brexit ht...
		risks	RT @minefornothing: The Spanish PM is warning Britain about the risks of leaving the EU. Youth unemployment in Spain is 50%!!!! #Brexit
		threat	Fury at PMs EU pensions threat: Vindictive PM tries Brexit blackmail https://t.co/hgp7etDuIC Shameful behaviour have your say #Brexit
		wrong	@Renegade_Inc Brit PM wrong Brits who support #Brexit put their people and country first!
14758838 : conservative	EU	bad	Leaving the EU & copying Canada's trade deal would be a bad deal for Britain. Here's why: https://t.co/51QRQ15FcZ https://t.co/APqkRlctRo
		risk	Manufacturing jobs are at risk if we leave the EU. Here's why: https://t.co/BwNDgkE4io #StrongerIN https://t.co/5f69NeJgUu
		risks	RT @David_Cameron: The IMF is right - leaving the EU would pose major risks for the UK economy. We are stronger, safer and better off in th...
		threat	Suggestion UK's vital intelligence 'five-eyes' relationship is under any sort of threat from the EU is wrong. Membership makes us safer.
		worse	RT @George_Osborne: Britain will be worse off by over 6% of GDP, to the tune of £4300 per household if we vote to leave the EU on June 23,
		wrong	Suggestion UK's vital intelligence 'five-eyes' relationship is under any sort of threat from the EU is wrong. Membership makes us safer.
	#brexit	risk	Years of "gruelling negotiations" would follow a #Brexit vote - hurting jobs and investment. Its not worth the risk. https://t.co/VbDhj63NK9
		risks	
		threat	New figures show that the threat of a #Brexit is hitting business investment https://t.co/GWRGDR36l6@ConservativesIN
		worse	Comprehensive @hmtreasury report shows #Brexit would make British families £4,300 worse o https://t.co/zNUQBfCzEk https://t.co/3wFfkx2ALr
118984824 : conservative	EU	bad	RT @montie: if things are so bad in Norway, outside of the EU, why do 18% want to join but 70% want to stay out? #marr https://t.co/ILoXjRS...
		crisis	@grahamstuart You mean two richest countries in Europe outside the EU? Remain have to explain risk of Greek default & migrant/Euro crisis
		risk	So just today: Heseltine says we'll be forced to join Euro, Turkey opening EU negotiations, Juncker says no new deal for UK.Why risk Remain?
		risks	RT @NadineDorriesMP: Time for Cameron and Osborne to debate live with Gove and Johnson the risks of staying in the EU People need to know!...
		threat	RT @suttonnick: Monday's Daily Express: Fury at PM's EU pension threat #tomorrowspaperstoday #bbcpapers https://t.co/wCOserkLPa
		threaten	RT @Vote_LeaveMedia: If leaving the EU would really be so ruinous, why did Cameron threaten to do so only recently? https://t.co/omOhNWZkwP...
		worse	RT @vote_leave: .@wdjstraw just admitted live on #r4today that 'we will be worse off if we stay in the EU'
		wrong	RT @David_Cameron: The Leave campaign is wrong to say there'll be a 2nd referendum if we vote to remain in the EU. This is a referendum and...
	tax	risk	RT @MrRBourne: So following Cameron logic, he'd have been willing to risk war to secure minor changes to tax credits.
	#brexit	risk	RT @BeeAHoney_: Hey @David_Cameron please explain 'in detail': what changed from No. 2015, to NOW? YOU implied #BREXIT not a RISK?! https:/...
		threat	So John Major's scaremongering over "threat" to peace in Northern Ireland of #Brexit is rejected by a whopping 61%-21% in IpsosMori poll
		wrong	RT @ajcdeane: Look at this rogues' gallery. Wrong then, wrong now. #Brexit https://t.co/WWFzERiUrc

(*continued on next page*)

Table A.1 (*continued*)

User	Issue	Keyword	Tweet
2673995912 : ukip	EU	crisis	Germany's largest bank's profits drop 98% as EU banking crisis spreads https://t.co/TSo2biDhAc
		risk	Biggest canard of Remainians is staying in EU is risk free. In fact, it's a massive gamble we escape federalist tide https://t.co/CwMBc8MKHn
		threat	RT @UKIP: "Despite clear evidence the EU is under threat from the migration crisis, our PM will not secure our borders" @DianeJamesMEP #UKI...
		threaten	Here's the deal. EU gives Turkey € 3B to house refugees and Erdogan threatens to send millions more migrants https://t.co/HnjuI1U2q5#Cushty!
	#brexit	bad	If things are going to be as bad after #BREXIT as the #scaremongering #Remainians suggest, surely we will need to be even more creative!
		crisis	If you would like to know what is imperiling the global economy its not #Brexit it is the European banking crisis https://t.co/g5oXXtrFsX
		risks	Another example of risks we face if #Remain in a federalist EU. Only way to make lawmakers accountable #Brexit https://t.co/b791dDCC6u
		wrong	Cameron's #ProjectFear speech Straw Man 3: #Leave equals uncertainty. Wrong. Staying in EU SuperState means Decades of uncertainty #Brexit

Table A.2
Clusters of Test 1.

Cluster No	Users	Issues	Keywords
1	Conservative	eu	Bad
	Labour	tax	Crisis
	ukip	#brexit	Risk
	ukip		Risks
	ukip		Threat
	Conservative		Threaten
	Conservative		Worse
	ukip		Wrong
2	ukip	eu	Desperate
	Conservative	tax	Failing
	ukip	#brexit	Fails
	ukip		Fear
	Conservative		Sorry
			Threatening
3	Labour	eu	Fears
	ukip	Tax	Risky
	Labour	#brexit	Rival
	ukip		Shock
	Conservative		Threats
			Worried
4	Labour	nhs	Benefit
	Labour	eu	top
	Labour	Tax	Attacks
	Labour		Lose
	ukip		Worst

References

Angiulli, F., & Pizzuti, C. (2005). Gene expression biclustering using random walk strategies. In *Proceedings of the international conference on data warehousing and knowledge discovery* (pp. 509–519). Springer.

Bader, B. W., & Kolda, T. G. (2007). Efficient matlab computations with sparse and factored tensors. *SIAM Journal on Scientific Computing, 30*(1), 205–231.

Blondel, V. D., Guillaume, J.-L., Lambiotte, R., & Lefebvre, E. (2008). Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment, 2008*(10), P10008.

Dawande, M., Keskinocak, P., Swaminathan, J. M., & Tayur, S. (2001). On bipartite and multipartite clique problems. *Journal of Algorithms, 41*(2), 388–403.

Dhillon, I. S. (2001). Co-clustering documents and words using bipartite spectral graph partitioning. In *Proceedings of the seventh ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 269–274). ACM.

Gao, B., Liu, T.-Y., Zheng, X., Cheng, Q.-S., & Ma, W.-Y. (2005). Consistent bipartite graph co-partitioning for star-structured high-order heterogeneous data co-clustering. In *Proceedings of the eleventh ACM SIGKDD international conference on knowledge discovery in data mining* (pp. 41–50). ACM.

Giannakidou, E., Koutsonikola, V., Vakali, A., & Kompatsiaris, Y. (2008). Co-clustering tags and social data sources. In *Proceedings of the ninth international conference on Web-age information management, 2008. WAIM'08* (pp. 317–324). IEEE.

Gokalp, S., Temkit, M., Davulcu, H., & Toroslu, I. H. (2013). Partitioning and scaling signed bipartite graphs for polarized political blogosphere. In *Proceedings of the 2013 international conference on social computing (socialcom)* (pp. 168–173). IEEE.

Gregory, S. (2010). Finding overlapping communities in networks by label propagation. *New Journal of Physics, 12*(10), 103018.

Hobolt, S. B. (2016). The brexit vote: A divided nation, a divided continent. *Journal of European Public Policy, 23*(9), 1259–1277.

Ignatov, D. I., Gnatyshak, D. V., Kuznetsov, S. O., & Mirkin, B. G. (2015). Triadic formal concept analysis and triclustering: searching for optimal patterns. *Machine Learning, 101*(1), 271–302. doi:10.1007/s10994-015-5487-y.

Kolda, T. G., & Sun, J. (2008). Scalable tensor decompositions for multi-aspect data mining. In *Proceedings of the eighth IEEE international conference on Data mining, 2008* (pp. 363–372). IEEE.

Lancichinetti, A., Fortunato, S., & Kertész, J. (2009). Detecting the overlapping and hierarchical community structure in complex networks. *New Journal of Physics, 11*(3), 033015.

Li, W., Xie, J., Xin, M., & Mo, J. (2017). *An overlapping network community partition algorithm based on semi-supervised matrix factorization and random walk*.

Lin, Y.-R., Sun, J., Castro, P., Konuru, R., Sundaram, H., & Kelliher, A. (2009). Metafac: community discovery via relational hypergraph factorization. In *Proceedings of the 15th ACM SIGKOD international conference on knowledge discovery and data mining* (pp. 527–536). ACM.

Liu, X., & Murata, T. (2011). Detecting communities in K-Partite K-Uniform (Hyper) networks. *J. Comput. Sci. Technol., 26*(5), 778–791.

Long, B., Zhang, Z. M., Wu, X., & Yu, P. S. (2006). Spectral clustering for multi-type relational data. In *Proceedings of the 23rd international conference on machine learning* (pp. 585–592). ACM.

Lu, C., Chen, X., & Park, E. (2009). Exploit the tripartite network of social tagging for web clustering. In *Proceedings of the 18th ACM conference on information and knowledge management* (pp. 1545–1548). ACM.

Manning, C. D., Raghavan, P., & Schütze, H. (2008). *Introduction to information retrieval* (pp. 356–360)). New York, NY, USA: Cambridge University Press.

Missaoui, R., & Kwuida, L. (2011). *Mining triadic association rules from ternary relations*. Berlin, Heidelberg: Springer Berlin Heidelberg. doi:10.1007/978-3-642-20514-9_16.

Ng, A. Y., Jordan, M. I., Weiss, Y., et al. (2001). On spectral clustering: Analysis and an algorithm. In *Nips: 14* (pp. 849–856).

Omeroglu, N. B., Toroslu, I. H., Gokalp, S., & Davulcu, H. (2013). K-partitioning of signed or weighted bipartite graphs. In *Proceedings od the 2013 international conference on social computing (socialcom)* (pp. 815–820). IEEE.

Rhouma, D., & Romdhane, L. B. (2014). An efficient algorithm for community mining with overlap in social networks. *Expert Systems with Applications, 41*(9), 4309–4321.

Rosvall, M., & Bergstrom, C. T. (2008). Maps of random walks on complex networks reveal community structure. *Proceedings of the National Academy of Sciences, 105*(4), 1118–1123.

Zha, H., He, X., Ding, C., Simon, H., & Gu, M. (2001). Bipartite graph partitioning and data clustering. In *Proceedings of the tenth international conference on information and knowledge management* (pp. 25–32). ACM.

Zhao, L., & Zaki, M. J. (2005). Tricluster: an effective algorithm for mining coherent clusters in 3d microarray data. In *Proceedings of the 2005 ACM sigmod international conference on management of data* (pp. 694–705). ACM.

Zhou, L., Lü, K., Yang, P., Wang, L., & Kong, B. (2015). An approach for overlapping and hierarchical community detection in social networks based on coalition formation game theory. *Expert Systems with Applications, 42*(24), 9634–9646.

Zhu, L., Galstyan, A., Cheng, J., & Lerman, K. (2014). Tripartite graph clustering for dynamic sentiment analysis on social media. In *Proceedings of the 2014 acm sigmod international conference on management of data* (pp. 1531–1542). ACM.

Event Detection by Change Tracking on Community Structure of Temporal Networks

Riza Aktunc
METU Computer Eng. Dept.
Ankara, Turkey
riza.aktunc@ceng.metu.edu.tr

I. Hakki Toroslu
METU Computer Eng. Dept.
Ankara, Turkey
toroslu@ceng.metu.edu.tr

Pinar Karagoz
METU Computer Eng. Dept.
Ankara, Turkey
karagoz@ceng.metu.edu.tr

***Abstract*—Event detection is a popular research problem, aiming to detect events from online data sources with least possible delay. Most of the previous work focus on analyzing textual content such as social media postings to detect happenings. In this work, we consider event detection as a change detection problem in network structure, and propose a method that detects change in community structure extracted from communication network. We study three versions of the method based on different change models. Experimental analysis on benchmark data set reveals that change in the community can be used as an indication of an event.**

***Index Terms*—Event detection, temporal network, community detection, network features, change**

I. Introduction

Event detection has attracted attention as a research problem due to its applicability in several applications from getting news in a timely manner [1] to taking emergency steps for city management or disasters [2]. There is a rich literature on event detection, however, most of the previous studies focus on analyzing textual content from social media and web data [3], [4], [5], [6].

In this work, we follow a comparatively newer track in the event detection literature, and study the problem from network point of view [7], [8]. In addition to posting textual messages, events trigger certain behaviors on individuals, such as gathering or communicating, affecting the social context and network among individuals. In this work, we hypothesize that examining and tracking changes in community structure can reveal that an event is happening. We focused especially on detecting the events on Call Detail Record (CDR) data. We firstly extract weekly samples from CDR data and then convert these samples to directed weighted and unweighted networks. The nodes of the networks correspond to phone users in CDR data, while the edges of the networks correspond to the communication between the connected nodes. The communication can be an SMS or a voice call. For each communication type, we define different networks. There are previous efforts on detecting events on CDR data published in the literature [9], [10], [11]. These works focus on tracking graph attributes such as degree of nodes, and several probabilistic models to detect events on the graph extracted from CDR data. Thus, the main difference and contribution of our proposed method is that it is based on tracking the change on community structure of the consecutive networks.

In the literature, event is generally defined as a happening that takes place at a certain time and place, and attracts attentions. In our study, we focus on the time dimension and aim to determine time windows in which an event takes place.

The proposed method involves tracking the change in the community structures over temporal networks. A sequence of networks corresponding to time windows along the timeline is analyzed for changes in detected communities in consecutive networks. We model the change in three different ways: change in the number of communities, change in the central nodes of the communities, and change in members of the communities. Within each model, there are variations based on how the defined change is computed.

Event detection performance of the methods are evaluated on a benchmark data set [12] in terms of precision, recall and f1-measure under varying change thresholds. The results reveal the potential of the approach, especially for the use of change model involving the number of communities, for event detection problem.

II. Proposed Method: Community Structure Change based Event Detection

The proposed method is based on the idea that events can be detected by tracking the amount of change in various attributes of the community structure of the network in consecutive time windows. In addition to the length of the time window, the source of the network structure and the community detection techniques may vary. In this work, we set the time window as one week, and hence construct communities on the basis of the weekly interactions. As the community detection technique, we use dSLM algorithm given in [13].

In order to track the change, we compare the community structure parameter values against the previous time window. We studied change in the community structure over three basic parameters, which are *Number of Communities*, *Central Nodes*, and *Community Members*. In the rest of this section, the details of the methods through each of these parameters are presented.

This research was supported partially by USAF Grant FA9550-15-1-0004.
This work is partially supported by TUBITAK under project number 117E566.

IEEE/ACM ASONAM 2018, August 28-31, 2018, Barcelona, Spain
978-1-5386-6051-5/18/$31.00 © 2018 IEEE

Algorithm 1 Event Detection via Change on # of Comm

Input: setOfComm, minCommSize, maxCommSize, changeThreshold
Output: events
  prev = NumberOfComm(setOfComm(t_1), minCommSize, maxCommSize)
  for i=2 **to** timeWindowCount **do**
    cur = NumberOfComm(setOfComm(setOfComm(t_i), minCommSize, maxCommSize)
    change = abs(cur − prev)/prev
    if change ≥ changeThreshold **and** abs(cur - prev) > 1 **then**
      add i to events
    end if
    prev = cur
  end for
  return events

Algorithm 2 Event Detection via Change On Central Nodes

Input: setOfComm, changeComptMethod, changeThreshold
Output: events
  prev = setOfComm(t_1).centralNodes
  for i=2 **to** timeWindowCount **do**
    cur =setOfComm(t_i).centralNodes
    change = ComputeChangeOnCentralNodes (changeCompMethod, prev, cur)
    changeInSize = abs(cur.size - prev.size)
    if change ≥ changeThreshold **and** changeInSize > 1 **then**
      add i to events
    end if
    prev = cur
  end for
  return events

A. Change on the Number of Communities

Our first hypothesis is that the change on the total number of communities is an indication of an event. In other words, if the amount of change in the number of communities of two consecutive time windows exceeds a given threshold, we may mark the latter window as an event. However, size of the communities is an important factor to be considered. Considering all kinds of communities may lead to incorrect predictions due to the fact that events may arise from communities of certain size. In order to analyze the effect of the community size, in experiments (as given in Section III), we partitioned the set of communities into size groups. For instance, we find the number of communities that contain 3 to 5 people in each time window, and compute the change only for these communities. The exception is that if the amount of change is only one (despite the change threshold is fulfilled), we consider that this change is not a strong indication, and hence do not mark that window as an event. This algorithm is given in Algorithm 1.

B. Change on the Central Nodes

As another important indicator of an event, we consider the change on the *central nodes* of the communities. Given a ranked list of nodes with respect to centrality score, we define central nodes of a community as the top 20% of the nodes in the list. We compute the change of central nodes in four different ways. The simplest one is that we compute the change on the size of the central nodes in consecutive time windows (denoted as SIZE). The second one is computing the change as the number of central nodes of previous window that are not central nodes any more in the current window (denoted as NOT_ANY_MORE). The third way is that we compute change as the number of central nodes of current window that were not central nodes in the previous window (denoted as NEW). The last and the most complex change computation method involves the number of central nodes of previous window that are not central nodes any more in current window, added with the number of central nodes of current window that were not central nodes in previous window (denoted as NOT_ANY_MORE_NEW). The algorithm is presented in Algorithm 2 and Algorithm 3.

Algorithm 3 Compute Change on Central Nodes

Input: changeCompMethod, prev, cur
Output: change
  if changeCompMethod is SIZE **then**
    return abs(cur. size - prev.size) / prev.size
  else if changeCompMethod is NOT_ANY_MORE **then**
    return notAnyMoreCentralNodeCount / prev.size
  else if changeCompMethod is NEW **then**
    return newCentralNodeCount / cur.size
  else if changeCompMethod is NOT_ANY_MORE_NEW **then**
    return (notAnyMoreCentralNodeCount + newCentralNodeCount) / cur.size
  end if

C. Change on the Community Members

As another indicator for event, we consider the change on the community members. However, community detection techniques do not assign global identifiers to communities to be tracked over time windows. To be able to track the change within the same community, we assume that two communities in consecutive windows are the same community if they have common central nodes. Therefore, we firstly find the communities of consecutive windows around similar central nodes. Then, we compute the change in the number of members within the community. We compute the change in three different way, such that we consider the *minimum (MIN)* of change value, the *average (AVG)* of these change values, and the *maximum (MAX)* change value. The algorithm is given in Algorithm 4 and Algorithm 5.

Algorithm 4 Event Detection via Change on Comm Members

Input: setOfComm, changeCompMethod, changeThreshold
Output: events

```
prev = setOfComm(t_1).members
for i=2 to timeWindowCount do
    cur = setOfComm(t_i).members
    change = ComputeChangeOnMembers(
    changeCompMethod, prev, cur)
    if change ≥ changeThreshold then
        add i to events
    end if
    prev = cur
end for
return events
```

Algorithm 5 Compute Change on Community Members

Input: changeCompMethod, prev, cur
Output: change

```
if changeCompMethod is MIN then
    return min of the change values on the comm members
else if changeCompMethod is AVERAGE then
    return avg of the change values on the comm members
else if changeCompMethod is MAX then
    return max of the change values on the comm members
end if
```

III. Experiments and Results

Data Set. The experiments are conducted on *Reality Mining* data set that involves mobile phone call logs of 97 faculty, student, and staff at MIT over 50 weeks [12]. The ground truth captures semester breaks, exam and sponsor weeks, and holidays. In data set, among voice call, SMS and bluetooth activities, we used voice call logs. We built a sequence of temporal networks corresponding to 1-week time windows. We applied a modularity based community detection algorithm [13] on each week's network.

Experiments. We conducted experiments on two versions of the network (weighted and unweighted) under varying change thresholds from 5% to 85%. The results are presented in Tables I to V. The experiments of the method that detects events via tracking the change on community members have the same results for both weighted and unweighted versions of the network. Therefore, we present the results of the experiments of this method as a single table in Table III. In all tables, the second column shows the applied change threshold. The first column includes the change modeling parameter, which depends on the change model applied. In the first method, it includes the community size, in the second one, the type of change in central node, and in the third one, the type of change in the community members. For each method, we present the best three results per parameter setting in f1-measurement.

Discussion. We can summarize the results of the study as follows:

- Among three community change models, event detection according to the change in number of communities has the highest scores in terms of precision and f1-measure. Although it is the simplest approach, it provides a stronger indicator. Since it is computationally simpler as well, it has an advantage to be applied online.
- Among different community size groupings, change tracking on smallest communities (communities with 3-5 members) provides the highest precision and recall results. This indicates that an event triggers communication among small groups that emerges new, possibly short-lived communities.
- As a general observation for the first two methods under all settings, lower threshold values provides the highest

TABLE I
Directed Weighted Voice Network - Number of Communities

Type	Chng	Precision	Recall	F1-meas.
3-5 nodes	0.05	0.62	0.81	0.71
3-5 nodes	0.15	0.62	0.81	0.71
3-5 nodes	0.25	0.6	0.75	0.67
6-10 nodes	0.05	0.38	0.56	0.46
6-10 nodes	0.15	0.38	0.56	0.46
6-10 nodes	0.25	0.36	0.5	0.42
11-20 nodes	0.05	0.48	0.88	0.63
11-20 nodes	0.15	0.48	0.88	0.63
11-20 nodes	0.25	0.5	0.75	0.6
21-30 nodes	0.05	0.32	0.44	0.38
21-30 nodes	0.15	0.32	0.44	0.38
21-30 nodes	0.25	0.32	0.44	0.38
21-30 nodes	0.35	0.17	0.19	0.18
31-40 nodes	0.05	0.55	0.38	0.45
31-40 nodes	0.15	0.55	0.38	0.45
31-40 nodes	0.25	0.55	0.38	0.45

TABLE II
Directed Weighted Voice Network - Central Nodes

Type	Chng	Precision	Recall	F1-meas.
NEW	0.05	0.33	0.94	0.49
NEW	0.15	0.33	0.94	0.49
NEW	0.25	0.33	0.94	0.49
NOT_ANY_MORE	0.05	0.33	0.94	0.49
NOT_ANY_MORE	0.15	0.33	0.94	0.49
NOT_ANY_MORE	0.65	0.39	0.75	0.52
NOT_ANY_MORE_NEW	0.05	0.33	0.94	0.49
NOT_ANY_MORE_NEW	0.15	0.33	0.94	0.49
NOT_ANY_MORE_NEW	0.25	0.33	0.94	0.49
SIZE	0.05	0.32	0.75	0.45
SIZE	0.15	0.36	0.5	0.42
SIZE	0.25	0.27	0.19	0.23

TABLE III
Directed Voice Network - Community Members

Type	Chng	Precision	Recall	F1-meas.
AVG	0.05	0.33	1	0.5
AVG	0.15	0.33	1	0.5
AVG	0.25	0.33	1	0.5
MAX	0.05	0.33	1	0.5
MAX	0.15	0.33	1	0.5
MAX	0.25	0.33	1	0.5
MIN	0.05	0	0	0
MIN	0.15	0	0	0
MIN	0.25	0	0	0

TABLE IV
DIRECTED UNWEIGHTED VOICE NETWORK - NUMBER OF COMMUNITIES

Type	Chng	Precision	Recall	F1-meas.
3-5 nodes	0.05	0.62	0.81	0.71
3-5 nodes	0.15	0.62	0.81	0.71
3-5 nodes	0.25	0.62	0.81	0.71
6-10 nodes	0.05	0.43	0.62	0.51
6-10 nodes	0.15	0.43	0.62	0.51
6-10 nodes	0.25	0.43	0.62	0.51
11-20 nodes	0.05	0.54	0.94	0.69
11-20 nodes	0.15	0.54	0.94	0.69
11-20 nodes	0.25	0.58	0.88	0.7
21-30 nodes	0.05	0.48	0.69	0.57
21-30 nodes	0.15	0.48	0.69	0.57
21-30 nodes	0.25	0.48	0.69	0.57
31-40 nodes	0.05	0.53	0.56	0.55
31-40 nodes	0.15	0.53	0.56	0.55
31-40 nodes	0.25	0.5	0.5	0.5

TABLE V
DIRECTED UNWEIGHTED VOICE NETWORK - CENTRAL NODES

Type	Chng	Precision	Recall	F1-meas.
NEW	0.35	0.34	0.94	0.5
NEW	0.45	0.34	0.94	0.5
NEW	0.55	0.34	0.94	0.5
NOT_ANY_MORE	0.05	0.33	0.94	0.49
NOT_ANY_MORE	0.15	0.33	0.94	0.49
NOT_ANY_MORE	0.65	0.39	0.81	0.53
NOT_ANY_MORE_NEW	0.05	0.33	0.94	0.49
NOT_ANY_MORE_NEW	0.15	0.33	0.94	0.49
NOT_ANY_MORE_NEW	0.25	0.33	0.94	0.49
SIZE	0.05	0.31	0.75	0.44
SIZE	0.15	0.36	0.5	0.42
SIZE	0.25	0.25	0.19	0.22

accuracy scores. Except for a few cases, under increasing change threshold, both precision and recall values decrease.

- Event detection according to the change in the community members appears to be insensitive to change threshold. The accuracy values remain the same for each of AVG, MAX and MIN settings. This method provides the highest recall score (recall value 1.00) under AVG and MAX, however, low precision value shows that it has tendency towards labeling time windows as event.
- The similar observation holds for the method tracking the change of central nodes as well. Although high recall values (recall 0.94) are obtained, precision values remain low.
- The accuracy values under weighted and unweighted network structures do not indicate a strong difference. The results are the same for the method with change of community members for both of the graphs. For the other two methods, the accuracy values slightly higher for the unweighted network structure. Thus, unweighted network structure may be preferable due to its lightweight structure.

IV. CONCLUSION AND FUTURE WORK

In this work, we study event detection problem from social network point of view and model it as a change detection problem in community structure of temporal networks. Hence, we detect communities in a network corresponding to a time window, an then we track the change in the community structures along the timeline. We propose event detection under three different change models: the change in the number of communities, the change in the central nodes of the communities and the change in the size of each community. Experiments conducted on a benchmark data set under various settings show that change in the number of communities is a stronger indication for an event.

This work can be extended in several dimensions. As the first one, a hybrid model combining these three change models can be constructed. As another research direction, using the change values under various change models as features, a learning based method can be studied on. In our method, we considered a particular centrality metric to determine the central nodes. Alternative link based ranking methods can be employed in the method.

REFERENCES

[1] J. Sankaranarayanan, H. Samet, B. E. Teitler, M. D. Lieberman, and J. Sperling, "TwitterStand: News in Tweets," in *ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (GIS)*, 2009, pp. 42–51.

[2] T. Sakaki, M. Okazaki, and Y. Matsuo, "Earthquake Shakes Twitter Users: Real-time Event Detection by Social Sensors," in *International Conference on World Wide Web (WWW)*, 2010, pp. 851–860.

[3] X. Zhou and L. Chen, "Event Detection over Twitter Social Media Streams," *The VLDB Journal*, vol. 23, no. 3, pp. 381–400, 2014.

[4] O. Ozdikis, P. Karagoz, and H. Oğuztüzün, "Incremental Clustering with Vector Expansion for Online Event Detection in Microblogs," *Social Network Analysis and Mining*, 2017.

[5] O. Ozdikis, P. Senkul, and H. Oguztuzun, "Semantic Expansion of Tweet Contents for Enhanced Event Detection in Twitter," in *International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, 2012, pp. 20–24.

[6] F. Atefeh and W. Khreich, "A Survey of Techniques for Event Detection in Twitter," *Computational Intelligence*, vol. 31, no. 1, pp. 132–164, 2015.

[7] S. Rayana and L. Akoglu, "Less is more: Building selective anomaly ensembles with application to event detection in temporal graphs," in *SDM*, 2015.

[8] S. Rayana and L. Akogli, "Less is more: Building selective anomaly ensembles," *ACM Trans. Knowl. Discov. Data*, vol. 10, no. 4, pp. 42:1–42:33, May 2016.

[9] Y. Dong, F. Pinelli, Y. Gkoufas, Z. Nabi, F. Calabrese, and N. V. Chawla, "Inferring unusual crowd events from mobile phone call detail records," *CoRR*, vol. abs/1504.03643, 2015.

[10] I. A. Karatepe and E. Zeydan, "Anomaly detection in cellular network data using big data analytics," in *European Wireless 2014; 20th European Wireless Conference*, May 2014, pp. 1–5.

[11] V. A. Traag, A. Browet, F. Calabrese, and F. Morlot, "Social event detection in massive mobile phone data using probabilistic location inference," in *2011 IEEE Third International Conference on Privacy, Security, Risk and Trust and 2011 IEEE Third International Conference on Social Computing*, Oct 2011, pp. 625–628.

[12] N. Eagle, A. Pentland, and D. Lazer, "Inferring Friendship Network Structure by Using Mobile Phone Data," in *Proceedings of the National Academy of Sciences*, vol. 106, no. 36, 2009, pp. 15 274–15 278.

[13] R. Aktunc, I. H. Toroslu, M. Ozer, and H. Davulcu, "A dynamic modularity based community detection algorithm for large-scale networks: Dslm," in *Proceedings of the 2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2015*, ser. ASONAM '15, 2015, pp. 1177–1183.

Detecting Antagonistic and Allied Communities on Social Media

Amin Salehi, Hasan Davulcu
Computer Science and Engineering
Arizona State University, Tempe, USA
{asalehi1, hdavulcu}@asu.edu

***Abstract*—Community detection on social media has attracted considerable attention for many years. However, existing methods do not reveal the relations between communities. Communities can form alliances or engage in antagonisms due to various factors, e.g., shared or conflicting goals and values. Uncovering such relations can provide better insights to understand communities and the structure of social media. According to social science findings, the attitudes that members from different communities express towards each other are largely shaped by their community membership. Hence, we hypothesize that inter-community attitudes expressed among users in social media have the potential to reflect their inter-community relations. Therefore, we first validate this hypothesis in the context of social media. Then, inspired by the hypothesis, we develop a framework to detect communities and their relations by jointly modeling users' attitudes and social interactions. We present experimental results using three real-world social media datasets to demonstrate the efficacy of our framework.**

I. Introduction

Although community detection plays an important role in providing insights into the structure and function of social media [1], existing community detection methods do not reveal inter-community relations, which are indispensable to deepen our insights. Moreover, to better understand communities, there is a need to uncover their relations. Indeed, social scientists suggest that "the understanding of policies and practices prevailing within groups will be inadequate unless relations among them are brought into the picture" [2]. A community, or group in social sciences, is defined as a set of users with many intra-group social interactions and few inter-group ones [3], who tend to have mainly positive attitudes towards each other [4], [5].

Several methods [6], [7], [8], [9], [10], [11] have been proposed to detect antagonistic communities. There are generally two categories of such methods: (1) those which detect antagonistic communities from signed networks [6], [7], [8], [9], and (2) those which mine antagonistic communities by finding frequent patterns in users' ratings [10], [11]. However, these methods suffer from two main limitations. First, they cannot be applied to a majority of popular social network platforms (e.g., Facebook and Twitter) since these platforms do not provide signed links or users' ratings explicitly. Second, inter-community relations are not restricted to antagonisms. Indeed, communities can also form alliances.

IEEE/ACM ASONAM 2018, August 28-31, 2018, Barcelona, Spain
978-1-5386-6051-5/18/$31.00

According to social science findings, inter-community attitudes that individuals express towards each other are largely shaped by their community membership rather than their characteristics or personal relationships [12], [13]. Moreover, Tajfel [14] observed a pair of characteristics in inter-community behavior. First, the members of a community display uniformity in their behavior and attitude towards any other community. Second, they tend to perceive the characteristics and behavior of the members of any other community as undifferentiated. Moreover, social scientists suggest that "the social psychology of intergroup relations is concerned with intergroup behaviour and attitudes" [14]. According to these observations, inter-community attitudes that users express towards each other in social media have the potential to reflect inter-community relations.

In this paper, we propose a framework, namely DAAC, which detects communities and their relations (i.e., antagonism, alliance, or neither) by exploiting users' social interactions (e.g., retweets) and attitudes expressed on social media. Our main contributions are:

- Validating the hypothesis suggesting that inter-community attitudes that users express towards each other in social media can reflect the relations of their communities;
- Achieving higher performance in detecting communities compared to several standard community detection methods;
- Uncovering inter-community relations, i.e., antagonism, alliance, or no relation.

The rest of the paper is organized as follows. In Section II, we review related work. In Section III, we formally define the problem of detecting communities and their relations on social media. Section IV describes three real-world social media datasets used in our experiments. In Section V, we first validate the aforementioned hypothesis and then present our framework. In Section VI, we demonstrate the effectiveness of the proposed framework. Section VII concludes the paper and discusses future work.

II. Related Work

There has been a lot of efforts to detect communities efficiently and accurately. To this end, a wide variety of approaches have been utilized. Modularity-based methods are among the most well-know techniques to detect communities.

The modularity measure proposed in [15] evaluates whether a division is good enough to form communities. Many variants of modularity-based community detection [16], [17] have been developed. Another well-known category includes spectral algorithms [18], [19], [20], [21] which aims to divide the network into several communities in which most of the interactions are within communities while the number of interactions across communities are minimized. Probabilist approaches [22], in which users are assigned to clusters in a probabilistic way, are also applied to the problem of community discovery. There are a variety of approaches such as information theory based methods [23], random walk techniques [24], [25], and model-based methods [26], [27] to tackle this problem.

Although there has been a great deal of efforts to detect communities, to the best of our knowledge, no previous work has been proposed to uncover the existence of antagonism and alliance between communities. However, some efforts have been made [6], [7], [8], [9], [10], [11] to detect only antagonistic communities. These methods can be roughly divided into two main categories. First category includes the methods [10], [11] utilizing frequent patterns in users' ratings to mine antagonistic communities. Second category includes the methods [6], [7], [8], [9] utilizing signed networks, having trust and distrust links, to detect antagonistic communities. A majority of these methods [7], [8], [9] detect a pair of subgraphs with most trust links preserved between the members of each subgraph and most distrust links remained between the members of different subgraphs. These methods are limited to detecting only a pair of antagonistic communities. To address this limitation, another method [6] has been proposed to detect multiple antagonistic communities by finding several dense subgraphs with the mentioned property. However, as experiments in [6] show such methods usually end up with large number of small subgraphs due to high sparsity of users' interactions in social media.

III. Problem Statement

We first begin with the introduction of the notations used in the paper as summarized in Table I. Let $\mathcal{U} = \{u_1, u_2, ..., u_n\}$ be the set of n users and $\mathcal{C} = \{c_1, c_2, ..., c_k\}$ indicate the set of k communities. $\mathbf{R} \in \mathbb{R}_+^{n \times n}$ denotes the social interaction matrix, where $\mathbf{R}_{i,j}$ corresponds to the number of social interactions between user u_i and user u_j. $\mathbf{S} \in \mathbb{R}^{n \times n}$ indicates the attitude matrix, where the positive/negative value of $\mathbf{S}_{i,j}$ corresponds to the positive/negative attitude strength of user u_i towards user u_j. $\mathbf{U} \in \mathbb{R}_+^{n \times k}$ indicates the community membership matrix, in which $\mathbf{U}_{i,l}$ corresponds to the membership strength of user u_i to community c_l. $\mathbf{H} \in \mathbb{R}^{k \times k}$ denotes intra/inter-community relation matrix, where $\mathbf{H}_{i,j}$, if $i \neq j$, corresponds to the strength and type of inter-community relation between community c_i and community c_j; the negative, positive, and zero value of $\mathbf{H}_{i,j}$ indicates antagonism, alliance, or no relation between community c_i and community c_j, respectively. Moreover, $\mathbf{H}_{i,i}$ corresponds to the intra-community attitudes that the members of community c_i have expressed towards each other. We define the symmetric normalization of $\mathbf{R}$ as $\widetilde{\mathbf{R}} = \mathbf{D}^{-1/2}\mathbf{R}\mathbf{D}^{-1/2}$, where $\mathbf{D} = diag(d_1, d_2, ..., d_n)$ is the degree matrix of $\mathbf{R}$ and the degree of user u_i is $d_i = \sum_{j=1}^{n} \mathbf{R}_{i,j}$. We separate positive and negative parts of matrix $\mathbf{A}$ as $\mathbf{A}^+_{i,j} = (|\mathbf{A}_{i,j}| + \mathbf{A}_{i,j})/2$ and $\mathbf{A}^-_{i,j} = (|\mathbf{A}_{i,j}| - \mathbf{A}_{i,j})/2$.

TABLE I: Notations used in the paper

Notation	Explanation
$\mathcal{U}$	The set of users
$\mathcal{C}$	The set of communities
n	The number of users
k	The number of communities
$\mathbf{R}$	The social interaction matrix
$\mathbf{S}$	The attitude matrix
$\mathbf{U}$	The community membership matrix
$\mathbf{H}$	The community intra/inter-relation matrix
$\widetilde{\mathbf{R}}$	Symmetrically normalized matrix $\mathbf{R}$
$\mathbf{D}$	Degree matrix of $\mathbf{R}$
$\mathbf{A}^+$	The positive part of matrix $\mathbf{A}$ (i.e., $(\|\mathbf{A}\| + \mathbf{A})/2$)
$\mathbf{A}^-$	The negative part of matrix $\mathbf{A}$ (i.e., $(\|\mathbf{A}\| - \mathbf{A})/2$)

By using the aforementioned notations, the problem of detecting communities and their relations on social media can be defined as: *Given social interaction matrix* $\mathbf{R}$ *and attitude matrix* $\mathbf{S}$*, we aim to obtain community membership matrix* $\mathbf{U}$ *and intra/inter-community relation matrix* $\mathbf{H}$.

IV. Data Description

Politics is a domain in which it is common among political parties (i.e., communities) to form alliances or engage in antagonisms. To validate the aforementioned hypothesis and evaluate our proposed framework, we use the following political Twitter datasets:

- **US Dataset** consists of the tweets posted by 583 politicians from two major US political parties (the Republican Party and the Democratic Party) from August 26 to November 29, 2016. For the period of time that this dataset covers, there were antagonisms between these parties particularly due to the 2016 presidential election campaigning [28].
- **Australia Dataset** consists of the tweets posted by 225 user accounts, including politicians and political groups, from five major Australian political parties (the Liberal Party, the National Party, the Liberal National Party, the Greens, and the Labor Party) from January 1 to November 18, 2016. For several decades, there has been a coalition among the Liberal Party, the National Party, and the Liberal National Party [29]. In the 2016 federal election, all relations between the parties were antagonistic except the relations between the members of the coalition,.
- **UK Dataset** consists of the tweets posted by 389 user accounts, including politicians and political groups, from five major UK political parties (the Conservative Party, the Labour Party, the Scottish National Party, the Liberal Democrats Party, and the UK Independence Party) from January 1 to October 31, 2015. There were antagonism among five major UK political parties in this period

TABLE II: The statistics of the cleaned datasets.

	US	Australia	UK
# of tweets	111,743	159,499	267,085
# of retweets	17,724	21,111	14,892
# of mentions	8,470	14,996	33,462
# of user accounts	583	225	389
# of true communities	2	5	5
# of allied relations	0	3	0
# of antagonistic relations	1	7	10

of time, especially due to the 2015 general election campaigning [30].

Preprocessing: For all datasets, we remove the users who do not have any retweet (i.e., social interaction). Table II shows the statistics of the preprocessed datasets. All users in the datasets have been labeled with their corresponding parties, and these labels are used to evaluate our proposed method.

Although aspect-based sentiment classification techniques [31] have been proposed to capture users' attitudes towards entities, publicly available training datasets are either too small or domain-oriented, making such techniques incapable to tackle real-world problems. Therefore, we use the following technique to extract the attitudes that users express towards each other in social media. Given each message in which author u_i has mentioned user u_j, we add the strength of the message's sentiment to the corresponding elements of matrix $\mathbf{S}$ (i.e., $\mathbf{S}_{i,j}$). Even though some messages may carry a negative sentiment, the author may not necessarily have an antagonistic attitude towards a mentioned user. To alleviate this problem, we ignore such messages if there is a social interaction (i.e., retweet) between the author and the mentioned user since a social interaction indicates the presence of a good relationship [32]. We utilize SentiStrength [33] to detect the sentiment polarity and strength of messages. We have made the code and datasets used in this paper available[1].

V. The Proposed Framework

In this section, we first demonstrate the existence of a significant level of correlation between the type of inter-community relation (i.e., alliance or antagonism) between two communities and the type of sentiment (i.e., positive or negative) that members from these communities expressed towards each other. Next, we propose our framework.

A. Validating the Hypothesis

According to social science findings [12], [13], the attitudes that members from different communities express towards each other are largely shaped by their community membership. Therefore, we hypothesize that inter-community attitudes expressed among users towards each other in social media have the potential to reflect inter-community relations. However, the findings borrowed from social sciences do not necessarily hold in social media due to many factors, such as the validity and representativeness of available information [34], [35]. Moreover, the attitudes that users express towards each other in social media might result from users' personal relationships. Therefore, in this section, we aim to verify our hypothesis by answering the following two questions. With this respect, we utilize the Australia dataset since it is the only dataset containing both allied and antagonistic relations.

- Are the communities of two users who express negative attitudes towards each other more likely to be in antagonism?
- Are the communities of two users who express positive attitudes towards each other more likely to be in alliance?

We first answer the former by using the following procedure inspired by [36]. For each pair of users (u_i, u_j) who are from different communities and have expressed negative attitudes towards each other (i.e., $\mathbf{S}_{i,j} < 0$), we randomly select a user u_k where users u_i and u_k are from different communities and have not expressed negative attitudes towards each other (i.e., $\mathbf{S}_{i,k} \geq 0$). Then, we check whether there is antagonism between the communities of u_i and u_j and between the communities of u_i and u_k. If there is antagonism between the communities of u_i and u_j, we set $t_p = 1$; otherwise $t_p = 0$. Similarly, if there is antagonism between the communities of u_i and u_k, we set $t_r = 1$; otherwise $t_r = 0$. Let vector T_p denote the set of all t_ps for pairs of users from different communities who have expressed negative attitudes towards each other, and vector T_r denote the set of all t_rs for pairs of users from different communities who have not expressed negative attitudes towards each other.

We conduct a two-sample t-test on T_p and T_r. The null hypothesis H_0 and alternative hypothesis H_1 are defined as follows:

$$H_0 : T_p \leq T_r, \quad H_1 : T_p > T_r \tag{1}$$

The null hypothesis is rejected at significance level $a = 0.01$ with p-value of $3.56\text{e}{-105}$. Therefore, the result of the two-sample t-test demonstrates that *the communities of two users who express negative attitudes towards each other are highly probable to be in antagonism*. We apply a similar procedure to answer the second question. For brevity, we only report the result of the two-sample t-test. The null hypothesis is rejected at significance level $a = 0.01$ with p-value of $1.57\text{e}{-26}$. As a result, we conclude that *the communities of two users who express positive attitudes towards each other are highly probable to be in alliance*.

B. Modeling Users' Attitudes

In the previous section, we demonstrated that inter-community attitudes expressed by users can reflect the relation of their communities in the context of social media. Inspired by this observation, we propose a model which uncovers intra/inter-community relations by exploiting the attitudes users express towards each other as,

$$\begin{aligned} \min_{\mathbf{U},\mathbf{H}} \quad & ||\mathbf{W} \odot (\mathbf{S} - \mathbf{U}\mathbf{H}\mathbf{U}^T)||_F^2 \\ \text{s.t.} \quad & \mathbf{U} \geq 0. \end{aligned} \tag{2}$$

[1]https://github.com/amin-salehi/DAAC

where $\odot$ is Hadamard product, $\mathbf{W}_{i,j}$ controls the contribution of $S_{i,j}$ in the model, and a typical choice of $\mathbf{W} \in \mathbb{R}_{+}^{n \times n}$ is,

$$\mathbf{W} = \begin{cases} 0, \text{ if } \mathbf{S} = 0 \\ 1, \text{ otherwise} \end{cases} \tag{3}$$

Given communities c_i and c_j, Eq. (2) aims to uncover their inter-community relation $\mathbf{H}_{i,j}$ by using their attitudes. To this end, $\mathbf{U}_{:,i}\mathbf{H}_{i,j}\mathbf{U}_{:,j}^T$ estimates the inter-community attitudes among the members of these two communities as presented in matrix $\mathbf{S}$. Since the non-negativity constraint only holds on $\mathbf{U}$, $\mathbf{H}_{i,j}$ will be negative, positive, or zero if the members of two communities have generally expressed negative, positive, or no attitudes towards each other, respectively. The lower the negative value of $\mathbf{H}_{i,j}$ is, the more antagonistic communities c_i and c_j are. On the other hand, the larger the positive value of $\mathbf{H}_{i,j}$ is, the more allied communities c_i and c_j are. Moreover, $\mathbf{H}_{i,i}$ indicates the intra-community attitudes that the members of community c_i have expressed towards each other.

C. Modeling Social Interactions

Social interactions are one of the most effective sources of information to detect communities [1]. In this section, we aim to cluster users into k communities with the most social interactions within each community and the fewest social interactions between communities. To this end, we use the following model,

$$\begin{aligned} \max_{\mathbf{U}} \quad & Tr(\mathbf{U}^T\widetilde{\mathbf{R}}\mathbf{U}) \\ \text{s.t.} \quad & \mathbf{U} \geq 0, \mathbf{U}^T\mathbf{U} = \mathbf{I}. \end{aligned} \tag{4}$$

where $\mathbf{I}$ is the identity matrix with the proper size. In fact, Eq. (4) is equivalent to the nonnegative relaxed normalized cut as put forth in [19].

D. The Proposed Framework DAAC

We separately introduced our models to utilize users' attitudes and social interactions. In this section, we propose our framework DAAC, which jointly exploits these two models to uncover communities and their relations. The proposed framework requires solving the following optimization problem,

$$\begin{aligned} \min_{\mathbf{U},\mathbf{H}} \quad & \boldsymbol{\mathcal{F}} = ||\mathbf{W} \odot (\mathbf{S} - \mathbf{U}\mathbf{H}\mathbf{U}^T)||_F^2 - \lambda Tr(\mathbf{U}^T\widetilde{\mathbf{R}}\mathbf{U}) \\ \text{s.t.} \quad & \mathbf{U} \geq 0, \mathbf{U}^T\mathbf{U} = \mathbf{I}. \end{aligned} \tag{5}$$

where λ is a non-negative regularization parameter controlling the contribution of social interactions in the final solution.

Since the optimization problem in Eq. (5) is not convex with respect to variables $\mathbf{U}$ and $\mathbf{H}$ together, there is no guarantee to find the global optimal solution. As suggested by [37], we introduce an alternative scheme to find a local optimal solution of the optimization problem. The key idea is optimizing the objective function with respect to one of the variables $\mathbf{U}$ or $\mathbf{H}$, while fixing the other one. The algorithm keeps updating the variables until convergence.

Optimizing the objective function $\mathcal{F}$ with respect to $\mathbf{U}$ leads to the following update rule,

Algorithm 1 The Proposed Algorithm for DAAC

Input: attitude matrix $\mathbf{S}$ and social interaction matrix $\mathbf{R}$
Output: community membership matrix $\mathbf{U}$ and intra/inter-community relation matrix $\mathbf{H}$

1: Initialize $\mathbf{U}$ and $\mathbf{H}$ randomly where $\mathbf{U} \geq 0$
2: **while** not convergent **do**
3: Update $\mathbf{U}$ according to Eq. (6)
4: Update $\mathbf{H}$ according to Eq. (14)
5: **end while**

$$\mathbf{U} = \mathbf{U} \odot \sqrt{\frac{\mathbf{E}_1^+ + \mathbf{E}_2^+ + \mathbf{E}_3^- + \mathbf{E}_4^- + \lambda\widetilde{\mathbf{R}}\mathbf{U} + \mathbf{U}\boldsymbol{\Gamma}^-}{\mathbf{E}_1^- + \mathbf{E}_2^- + \mathbf{E}_3^+ + \mathbf{E}_4^+ + \mathbf{U}\boldsymbol{\Gamma}^+}} \tag{6}$$

where,

$$\mathbf{E}_1 = -(\mathbf{W} \odot \mathbf{W} \odot \mathbf{S})\mathbf{U}\mathbf{H}^T \tag{7}$$

$$\mathbf{E}_2 = -(\mathbf{W} \odot \mathbf{W} \odot \mathbf{S})^T\mathbf{U}\mathbf{H} \tag{8}$$

$$\mathbf{E}_3 = (\mathbf{W} \odot \mathbf{W} \odot \mathbf{U}\mathbf{H}\mathbf{U}^T)\mathbf{U}\mathbf{H}^T \tag{9}$$

$$\mathbf{E}_4 = (\mathbf{W} \odot \mathbf{W} \odot \mathbf{U}\mathbf{H}\mathbf{U}^T)^T\mathbf{U}\mathbf{H} \tag{10}$$

$$\boldsymbol{\Gamma} = -\mathbf{U}^T\mathbf{E}_1 - \mathbf{U}^T\mathbf{E}_2 - \mathbf{U}^T\mathbf{E}_3 - \mathbf{U}^T\mathbf{E}_4 \tag{11}$$

$$+ \lambda\mathbf{U}^T\widetilde{\mathbf{R}}\mathbf{U} \tag{12}$$

The details are given in the Appendix.
The derivative of $\boldsymbol{\mathcal{F}}$ with respect to $\mathbf{H}$ is as follows:

$$\begin{aligned} \frac{\partial \boldsymbol{\mathcal{F}}}{\partial \mathbf{H}} = & -2\mathbf{U}^T(\mathbf{W} \odot \mathbf{W} \odot \mathbf{S})\mathbf{U} \\ & - 2\mathbf{U}^T(\mathbf{W} \odot \mathbf{W} \odot \mathbf{U}\mathbf{H}\mathbf{U}^T)\mathbf{U} \end{aligned} \tag{13}$$

Thus, the update rule of $\mathbf{H}$ is as follows:

$$\mathbf{H} = \mathbf{H} - \alpha\frac{\partial \boldsymbol{\mathcal{F}}}{\partial \mathbf{H}} \tag{14}$$

where α is the learning rate for updating $\mathbf{H}$.

The detailed algorithm for DAAC is shown in Algorithm 1. We briefly review Algorithm 1. In line 1, it randomly initializes $\mathbf{U}$ and $\mathbf{H}$. From line 2 to 5, it updates $\mathbf{U}$ and $\mathbf{H}$ until convergence is achieved.

E. Time Complexity

In Algorithm 1, the most costly operations are the matrix multiplications in update rules Eq. (6) and Eq. (14) on which we focus in this section. $\mathbf{W}$ and $\mathbf{R}$ are usually very sparse matrices, so let N_w and N_r denote the number of non-zero elements of $\mathbf{W}$ and $\mathbf{R}$, respectively. The time complexities of Eq. (6) and Eq. (14) are described as follows:

- We first focus on the time complexity of Eq. (6). Note that $\mathbf{W} \odot \mathbf{W} \odot \mathbf{S}$ needs to be calculated once. Therefore, the time complexities of both $\mathbf{E_1}$ and $\mathbf{E_2}$ are $\mathcal{O}(N_w k + nk^2)$ thanks to the sparsity of matrices $\mathbf{W}$ and $\mathbf{S}$. The time complexity of $\mathbf{W} \odot \mathbf{W} \odot \mathbf{U}\mathbf{H}\mathbf{U}^T$ is $\mathcal{O}(N_w n + nk^2 + n^2 k)$. The number of non-zero values of $\mathbf{W} \odot \mathbf{W} \odot \mathbf{U}\mathbf{H}\mathbf{U}^T$ is the same as $\mathbf{W}$ owing to the sparsity of $\mathbf{W}$. Thus, the time complexities of both $\mathbf{E_3}$ and $\mathbf{E_4}$ are $\mathcal{O}(N_w n + nk^2 + n^2 k)$. Using a similar procedure, the time complexities of

TABLE III: Comparison of community detection methods.

Method	US dataset			Australia dataset			UK dataset		
	NMI	ARI	Purity	NMI	ARI	Purity	NMI	ARI	Purity
Louvain	0.431095	0.386347	0.943396	0.825234	0.833025	0.942222	0.858118	0.841718	0.987147
InfoMap	0.431437	0.351938	0.946826	0.831903	0.831737	0.942222	0.909684	0.928716	**0.992288**
Leading eigenvectors	0.580117	0.678051	0.938250	0.779931	0.573488	0.693334	0.913700	0.9533703	0.982005
CNM	0.502925	0.487620	0.945111	0.842446	0.848269	0.937778	0.939093	0.971631	0.984576
Label propagation	0.600777	0.655619	0.958833	0.822237	0.826724	0.937778	0.958379	0.979031	0.989717
Soft clustering	0.735760	0.829228	0.955403	0.841264	0.812856	0.844444	0.951260	0.974292	0.987147
DAAC	**0.768307**	**0.854484**	**0.962264**	**0.903691**	**0.908264**	**0.951111**	**0.958806**	**0.978770**	0.989717

$\widetilde{\mathbf{R}}\mathbf{U}$ and $\mathbf{\Gamma}$ are $\mathcal{O}(N_r k)$ and $\mathcal{O}(N_w n + nk^2 + n^2 k + N_r k)$, respectively. As a result, the time complexity of Eq. (6) is $\mathcal{O}(N_w(n+k) + N_r k + nk^2 + n^2 k)$.

- Now we provide the time complexity of Eq. (14). The cost of $\mathbf{U}^T(\mathbf{W} \odot \mathbf{W} \odot \mathbf{S})$ is $\mathcal{O}(N_w k)$ thanks to the sparsity of $\mathbf{W}$. Thus, the time complexity of $\mathbf{U}^T(\mathbf{W} \odot \mathbf{W} \odot \mathbf{S})\mathbf{U}$ is $\mathcal{O}(N_w k + nk^2)$. Similarly, the cost of $\mathbf{U}^T(\mathbf{W} \odot \mathbf{W} \odot \mathbf{U}\mathbf{H}\mathbf{U}^T)\mathbf{U}$ is $\mathcal{O}(N_w n + nk^2 + n^2 k)$. Therefore, the time complexity of Eq. (14) is $\mathcal{O}(N_w(n+k) + nk^2 + n^2 k)$.

Hence, the time complexity of Algorithm 1 is $\mathcal{O}(i(N_w(n+k) + N_r k + nk^2 + n^2 k))$ where i is the number of iterations required for the convergence. Our framework can be applied to large scale social network platforms by exploiting distributed approaches outlined in [38], [39], [40].

VI. Experiments

To evaluate our proposed framework, we design the required experiments to answer the following two questions.

1) How effective is the proposed framework compared to the standard community detection methods?
2) How effective is our framework in discovering inter-community relations?

In the next section, we first compare the performance of several well-known community detection methods with DAAC. Then, we evaluate the effectiveness of our framework in uncovering inter-community relations. Finally, we study the sensitivity of our framework with respect to regularization parameter λ. For the experiments, we set the number of communities for any method, if it is required, as the true number of communities (i.e., parties) in each dataset.

A. Evaluation of Community Detection

1) Baselines: In order to demonstrate the efficacy of DAAC, we compare it with six well-known community detection methods presented as follows:

- **Louvain:** This method [17] greedily maximizes the benefit function known as modularity to detect communities.
- **InfoMap:** This baseline [23] is based on information theory and compresses the description of random walks in order to find communities.
- **Leading eigenvectors:** Newton [20] presents a formulation of modularity in a matrix form, namely modularity matrix. Then, he proposes to use the eigenvectors of modularity matrix to detect communities.

TABLE IV: The uncovered relations between detected communities (i.e., parties) by using DAAC in the US dataset.

	Republicans	Democrats
Republicans	259	-138
Democrats	-138	112

Note: all values in the table are rounded.

- **CNM:** This method [16] uses a greedy approach to find the divisions of the network which maximizes the modularity.
- **Label propagation:** [26] This method initially assigns unique labels to users. Then, in each iteration, users adopt the label that most of their neighbors posses. Finally, users with the same label fall into the same community.
- **Soft clustering:** This baseline [22] assigns users to communities in a probabilistic way.

2) Performance Measures: To evaluate the performance of the methods, we utilize three following measures which are frequently used for community detection evaluation: Normalized Mutual Information (NMI), Adjusted Rand Index (ARI), and Purity.

3) Experimental Results: We run all methods with their hyperparameters initialized from $\{10^x | x \in [0, 9]\}$. Table III shows the best result for each method. According to the table, we can make the following observations:

- Our proposed framework achieves the highest performance in terms of NMI and ARI for all three datasets. In terms of Purity, it also achieves the best in US and Australia datasets. In the UK dataset, only InfoMap obtains higher Purity compared to our framework since it generates a large number of communities (e.g., 11 communities for the UK dataset) for sparse graphs such as social media networks.
- Our framework achieves its highest performance with large values of regularization parameter λ (e.g., 10^7). This implies that social interactions are more effective in detecting communities compared to users' attitudes. We will study more on the impact of the regularization parameter in Section VI-C.

B. Evaluation of Inter-community Relations

In this section, we evaluate the effectiveness of our proposed framework in uncovering inter-community relations by conducting two experiments. To the best of our knowledge, there

TABLE V: The uncovered relations between detected communities (i.e., parties) by using DAAC in the Australia dataset.

	Liberals	Nationalists	Liberal Nationalists	Labors	Greens
Liberals	87	61	34	-21	-32
Nationalists	61	52	46	-4	-22
Liberal Nationalists	34	46	39	-4	-61
Labors	-21	-4	-4	121	-31
Greens	-32	-22	-61	-31	64

Note: all values in the table are rounded.

is no previous work to discover inter-community antagonistic and allied relations. Therefore, as the first experiment, we compare the inter-community relations which our framework detects with the real-word inter-community relations. Each community detected by our framework is labeled with the party to which the majority of its members belong. Then, we evaluate inter-community relations (i.e., matrix $\mathbf{H}$) detected by our algorithm according to the known ground-truth inter-party relations as previously presented in Section IV.

Table IV shows intra/inter-community relation matrix $\mathbf{H}$ for the US dataset as well as the parties corresponding to the detected communities. In 2016, the Republican Party and the Democratic Party were strongly antagonistic towards each other, especially due to the 2016 presidential election campaigning[2] [28]. As Table IV shows, our framework uncovers the existence of strong antagonism between these two parties. It also discovers that intra-community attitudes among the members of each community are highly positive as expected owing to the election campaign dynamics.

Table V shows intra/inter-community relation matrix $\mathbf{H}$ for the Australia dataset as well as the parties corresponding to the detected communities. The Liberal Party, the National Party, and the Liberal National party forged a coalition in the 2016 federal election. Except the relations between the members of the coalition, other relations among all parties were antagonistic[3]. As shown in Table V, our framework uncovers the coalition in which the three involved parties are in alliance with each other. It also discovers antagonism between the members of the coalition and other parties as well as the antagonism between the Greens and the Labor Party. Moreover, it detects high positive intra-community attitudes among the members of communities as expected.

Table VI shows intra/inter-community relation matrix $\mathbf{H}$ for the UK dataset as well as the parties corresponding to the detected communities. In 2015, there were antagonisms between all five major UK political parties, especially due to the 2015 general election campaigning[4] [30]. As shown in Table VI, our framework correctly detects all antagonistic relations between these parties. It also discovers that intra-community attitudes among the members of each community are highly positive as expected.

The second experiment compares our framework with a two-step approach described as follows. We first utilize social interactions to detect communities. Then, we aggregate the sentiment expressed among the members of different communities in order to figure out their inter-community relations. To have a fair comparison, we use Eq. (4) to detect communities for the two-step approach; which is the main component in DAAC for utilizing social interactions. As Table VII shows, the two-step approach is able to detect correct relations in US and Australia datasets. However, it fails to detect two out of ten inter-community relations in the UK dataset. This result shows that our proposed framework can detect inter-community relations more accurately by jointly using and social interactions and attitudes among users compared to a approach which sequentially detects communities and their relations.

[2]https://en.wikipedia.org/wiki/United_States_presidential_election,_2016
[3]https://en.wikipedia.org/wiki/Australian_federal_election,_2016
[4]https://en.wikipedia.org/wiki/United_Kingdom_general_election,_2015

TABLE VI: The uncovered relations between detected communities (i.e., parties) by using DAAC in the UK dataset.

	Conservatives	Labours	Lib dems	SNPs	UKIPs
Conservatives	154	-37	-7	-21	-9
Labours	-37	242	-8	-11	-26
Lib dems	-7	-8	63	-3	-14
SNPs	-21	-11	-3	55	-5
UKIPs	-9	-26	-14	-5	30

Note: all values in the table are rounded.

TABLE VII: Inter-community detection performance between DAAC and the two-step approach.

	US	**Australia**	**UK**
Two-step approach	1.0	1.0	0.8
DAAC	1.0	1.0	1.0

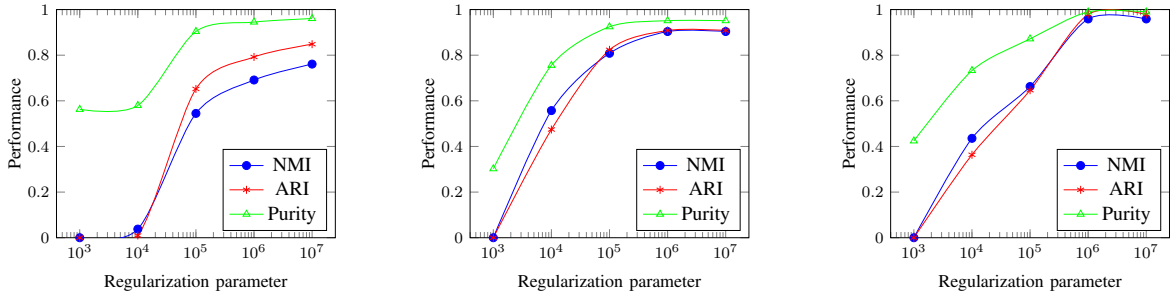
C. Study on the Regularization Parameter

In this section, we investigate the sensitivity of our framework with respect to regularization parameter λ. We vary the value of λ, and plot NMI, ARI and Purity measures in Figure 1 for all three datasets used in the study. Similarly, we plot the correct number of inter-community relations discovered by DAAC in Figure 2 for all three datasets with respect to different values of λ.

As we observe from Figure 1, very large values of λ (e.g., 10^6 and 10^7) for all datasets result in the highest performance of DAAC in detecting communities. Similarly, Figure 2 shows that very large values of λ also result in the highest number of correct inter-community relations discovered by DAAC. The rationale behind this is that inter-community relations cannot be correctly identified unless communities are accurately detected.

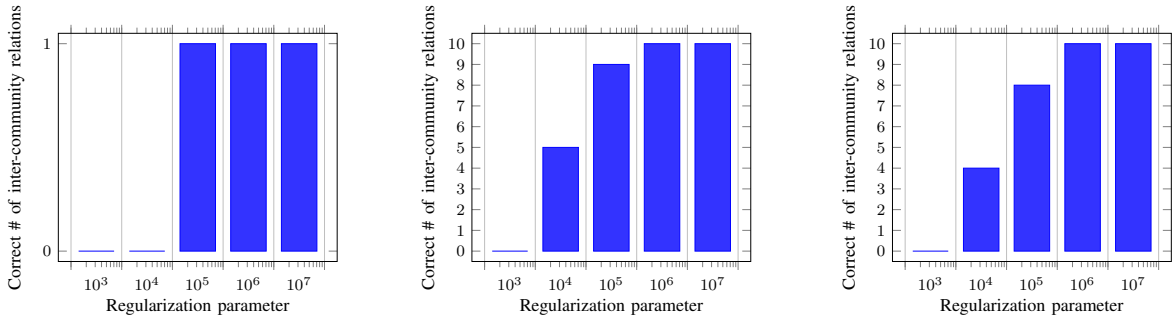
VII. Conclusion and Future Work

In this paper, we proposed a framework to discover communities and their relations by exploiting social interactions and user-generated content. We validated the hypothesis that inter-community attitudes that users express towards each other in social media can reflect inter-community relations. As inspired by this hypothesis, our proposed framework DAAC jointly models users' attitudes and social interactions in order



(a) US dataset (b) Australia dataset (c) UK dataset

Fig. 1: Community detection performance with regard to λ.



(a) US dataset (b) Australia dataset (c) UK dataset

Fig. 2: The correct number of inter-community relations with regard to λ.

to uncover communities and their antagonistic/allied relations. Experimental results on three real-world social media datasets demonstrated that our framework obtains significant performance in detecting communities compared with several baselines and also detects inter-community relations correctly. Moreover, we showed that a two-step approach, which sequentially detect communities and their relations, can fail to detect correct inter-community relations.

Since communities and their relations evolve over time, studying such dynamics provides deeper insights into understanding communities. In our future work, we aim to study uncovering the dynamics of communities and their relations and the motives behind these dynamics.

APPENDIX

Optimizing the objective function $\mathcal{F}$ in Eq. (5) with respect to U is equivalent to solving

$$\begin{aligned} \min_{\mathbf{U}} \quad & \mathcal{F}_{\mathbf{U}} = ||\mathbf{S} - \mathbf{U}\mathbf{H}\mathbf{U}^T||_F^2 - \lambda Tr(\mathbf{U}^T\widetilde{\mathbf{R}}\mathbf{U}) \\ \text{s.t.} \quad & \mathbf{U} \geq 0, \mathbf{U}^T\mathbf{U} = \mathbf{I}. \end{aligned} \tag{15}$$

Let $\mathbf{\Gamma}$ and $\mathbf{\Lambda}$ be the Lagrange multiplier for constraints $\mathbf{U}^T\mathbf{U} = \mathbf{I}$ and $\mathbf{U} \geq 0$, respectively, and the Lagrange function is defined as follows:

$$\begin{aligned} \min_{\mathbf{U}} \quad & \mathcal{L}_{\mathbf{U}} = ||\mathbf{S} - \mathbf{U}\mathbf{H}\mathbf{U}^T||_F^2 - \lambda Tr(\mathbf{U}^T\widetilde{\mathbf{R}}\mathbf{U}) \\ & - Tr(\mathbf{\Lambda}\mathbf{U}^T) + Tr(\mathbf{\Gamma}(\mathbf{U}^T\mathbf{U} - \mathbf{I})) \end{aligned} \tag{16}$$

The derivative of $\mathcal{L}_{\mathbf{U}}$ with respect to $\mathbf{U}$ is

$$\begin{aligned} \frac{\partial \mathcal{L}_{\mathbf{U}}}{\partial \mathbf{U}} = & -2(\mathbf{W} \odot \mathbf{W} \odot \mathbf{S})\mathbf{U}\mathbf{H}^T - 2(\mathbf{W} \odot \mathbf{W} \odot \mathbf{S})^T\mathbf{U}\mathbf{H} \\ & + 2(\mathbf{W} \odot \mathbf{W} \odot \mathbf{U}\mathbf{H}\mathbf{U}^T)\mathbf{U}\mathbf{H}^T \\ & + 2(\mathbf{W} \odot \mathbf{W} \odot \mathbf{U}\mathbf{H}\mathbf{U}^T)^T\mathbf{U}\mathbf{H} \\ & - 2\lambda\widetilde{\mathbf{R}}\mathbf{U} - \mathbf{\Lambda} + 2\mathbf{U}\mathbf{\Gamma} \end{aligned} \tag{17}$$

For the sake of simplicity, let us assume that,

$$\mathbf{E}_1 = -(\mathbf{W} \odot \mathbf{W} \odot \mathbf{S})\mathbf{U}\mathbf{H}^T \tag{18}$$

$$\mathbf{E}_2 = -(\mathbf{W} \odot \mathbf{W} \odot \mathbf{S})^T\mathbf{U}\mathbf{H} \tag{19}$$

$$\mathbf{E}_3 = (\mathbf{W} \odot \mathbf{W} \odot \mathbf{U}\mathbf{H}\mathbf{U}^T)\mathbf{U}\mathbf{H}^T \tag{20}$$

$$\mathbf{E}_4 = (\mathbf{W} \odot \mathbf{W} \odot \mathbf{U}\mathbf{H}\mathbf{U}^T)^T\mathbf{U}\mathbf{H} \tag{21}$$

By setting $\frac{\partial \mathcal{L}_{\mathbf{U}}}{\partial \mathbf{U}} = 0$, we get

$$\Lambda = -2\mathbf{E}_1 - 2\mathbf{E}_2 + 2\mathbf{E}_3 + 2\mathbf{E}_4 - 2\lambda\widetilde{\mathbf{R}}\mathbf{U} + 2\mathbf{U}\mathbf{\Gamma} \tag{22}$$

With the KKT complementary condition for the nonnegativity of $\mathbf{U}$, we have

$$\mathbf{\Lambda}_{ij}\mathbf{U}_{ij} = 0 \tag{23}$$

Therefore, we have

$$(\mathbf{E}_1 + \mathbf{E}_2 + \mathbf{E}_3 + \mathbf{E}_4 - \lambda\widetilde{\mathbf{R}}\mathbf{U} + 2\mathbf{U}\mathbf{\Gamma})_{ij}\mathbf{U}_{ij} = 0 \tag{24}$$

where

$$\mathbf{\Gamma} = -\mathbf{U}^T\mathbf{E}_1 - \mathbf{U}^T\mathbf{E}_2 - \mathbf{U}^T\mathbf{E}_3 - \mathbf{U}^T\mathbf{E}_4 + \lambda\mathbf{U}^T\widetilde{\mathbf{R}}\mathbf{U} \tag{25}$$

Since $\mathbf{E}_1$, $\mathbf{E}_2$, $\mathbf{E}_3$, $\mathbf{E}_4$, and $\mathbf{\Gamma}$ can take mixed signs. Suggested by [41], we separate positive and negative parts of any matrix A as

$$\begin{aligned} \mathbf{A}_{ij}^{+} &= (|\mathbf{A}_{ij}| + \mathbf{A}_{ij})/2 \\ \mathbf{A}_{ij}^{-} &= (|\mathbf{A}_{ij}| - \mathbf{A}_{ij})/2 \end{aligned} \tag{26}$$

Then, we get the following update rule of $\mathbf{U}$,

$$\mathbf{U} = \mathbf{U} \odot \sqrt{\frac{\mathbf{E}_1^{+} + \mathbf{E}_2^{+} + \mathbf{E}_3^{-} + \mathbf{E}_4^{-} + \lambda\widetilde{\mathbf{R}}\mathbf{U} + \mathbf{U}\mathbf{\Gamma}^{-}}{\mathbf{E}_1^{-} + \mathbf{E}_2^{-} + \mathbf{E}_3^{+} + \mathbf{E}_4^{+} + \mathbf{U}\mathbf{\Gamma}^{+}}} \tag{27}$$

ACKNOWLEDGMENTS

This work was partially supported by ONR Grant N00014-16-1-2015 and USAF Grant FA9550-15-1-0004.

REFERENCES

[1] S. Papadopoulos, Y. Kompatsiaris, A. Vakali, and P. Spyridonos, "Community detection in social media," *Data Mining and Knowledge Discovery*, vol. 24, no. 3, pp. 515–554, 2012.

[2] M. Sherif and C. W. Sherif, "Groups in harmony and tension; an integration of studies of intergroup relations." 1953.

[3] M. Girvan and M. Newman, "Community structure in social and biological networks," *Proc. Natl. Acad. Sci. USA*, vol. 99, no. cond-mat/0112110, pp. 8271–8276, 2001.

[4] L. Festinger, K. W. Back, and S. Schachter, "The spatial ecology of group formation," in *Social pressures in informal groups: A study of human factors in housing*. Stanford University Press, 1950, vol. 3, ch. 4.

[5] A. J. Lott and B. E. Lott, "Group cohesiveness as interpersonal attraction: a review of relationships with antecedent and consequent variables." *Psychological bulletin*, vol. 64, no. 4, p. 259, 1965.

[6] L. Chu, Z. Wang, J. Pei, J. Wang, Z. Zhao, and E. Chen, "Finding gangs in war from signed networks," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2016, pp. 1505–1514.

[7] M. Gao, E.-P. Lim, D. Lo, and P. K. Prasetyo, "On detecting maximal quasi antagonistic communities in signed graphs," *Data Mining and Knowledge Discovery*, vol. 30, no. 1, pp. 99–146, 2016.

[8] D. Lo, D. Surian, P. K. Prasetyo, K. Zhang, and E.-P. Lim, "Mining direct antagonistic communities in signed social networks," *Information Processing & Management*, vol. 49, no. 4, pp. 773–791, 2013.

[9] D. Lo, D. Surian, K. Zhang, and E.-P. Lim, "Mining direct antagonistic communities in explicit trust networks," in *Proceedings of the 20th ACM international conference on Information and knowledge management*. ACM, 2011, pp. 1013–1018.

[10] K. Zhang, D. Lo, and E.-P. Lim, "Mining antagonistic communities from social networks," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 2010, pp. 68–80.

[11] K. Zhang, D. Lo, E.-P. Lim, and P. K. Prasetyo, "Mining indirect antagonistic communities from social interactions," *Knowledge and information systems*, vol. 35, no. 3, pp. 553–583, 2013.

[12] H. Tajfel, "Human intergroup conflict: Useful and less useful forms of analysis," *Human ethology: Claims and limits of a new discipline*, pp. 369–422, 1979.

[13] M. Billig and H. Tajfel, "Social categorization and similarity in intergroup behaviour," *European Journal of Social Psychology*, vol. 3, no. 1, pp. 27–52, 1973.

[14] H. Tajfel, *Social identity and intergroup relations*. Cambridge University Press, 2010.

[15] M. E. Newman and M. Girvan, "Finding and evaluating community structure in networks," *Physical review E*, vol. 69, no. 2, p. 026113, 2004.

[16] A. Clauset, M. E. Newman, and C. Moore, "Finding community structure in very large networks," *Physical review E*, vol. 70, no. 6, p. 066111, 2004.

[17] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," *Journal of statistical mechanics: theory and experiment*, vol. 2008, no. 10, p. P10008, 2008.

[18] I. S. Dhillon, Y. Guan, and B. Kulis, "Kernel k-means: spectral clustering and normalized cuts," in *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2004, pp. 551–556.

[19] C. H. Ding, X. He, and H. D. Simon, "On the equivalence of nonnegative matrix factorization and spectral clustering." in *SDM*, vol. 5. SIAM, 2005, pp. 606–610.

[20] M. E. Newman, "Finding community structure in networks using the eigenvectors of matrices," *Physical review E*, vol. 74, no. 3, p. 036104, 2006.

[21] A. Salehi, M. Ozer, and H. Davulcu, "Sentiment-driven community profiling and detection on social media," in *Proceedings of the 29th ACM Conference on Hypertext and Social Media*. ACM, 2018.

[22] K. Yu, S. Yu, and V. Tresp, "Soft clustering on graphs," in *Advances in neural information processing systems*, 2005, pp. 1553–1560.

[23] M. Rosvall and C. T. Bergstrom, "Maps of random walks on complex networks reveal community structure," *Proceedings of the National Academy of Sciences*, vol. 105, no. 4, pp. 1118–1123, 2008.

[24] D. Harel and Y. Koren, "On clustering using random walks," in *FSTTCS*. Springer, 2001, pp. 18–41.

[25] P. Pons and M. Latapy, "Computing communities in large networks using random walks." *J. Graph Algorithms Appl.*, vol. 10, no. 2, pp. 191–218, 2006.

[26] U. N. Raghavan, R. Albert, and S. Kumara, "Near linear time algorithm to detect community structures in large-scale networks," *Physical review E*, vol. 76, no. 3, p. 036106, 2007.

[27] S. Gregory, "Finding overlapping communities in networks by label propagation," *New Journal of Physics*, vol. 12, no. 10, p. 103018, 2010.

[28] D. Lilleker, D. Jackson, E. Thorsen, and A. Veneti, "Us election analysis 2016: Media, voters and the campaign." 2016.

[29] D. Clune, "Contemporary australian political party organisations," 2016.

[30] M. Moran, *Politics and Governance in the UK*. Palgrave Macmillan, 2015.

[31] M. Pontiki, D. Galanis, H. Papageorgiou, I. Androutsopoulos, S. Manandhar, A.-S. Mohammad, M. Al-Ayyoub, Y. Zhao, B. Qin, O. De Clercq *et al.*, "Semeval-2016 task 5: Aspect based sentiment analysis," in *Proceedings of the 10th international workshop on semantic evaluation (SemEval-2016)*, 2016, pp. 19–30.

[32] M. Conover, J. Ratkiewicz, M. R. Francisco, B. Gonçalves, F. Menczer, and A. Flammini, "Political polarization on twitter." *ICWSM*, vol. 133, pp. 89–96, 2011.

[33] M. Thelwall, K. Buckley, G. Paltoglou, D. Cai, and A. Kappas, "Sentiment strength detection in short informal text," *Journal of the American Society for Information Science and Technology*, vol. 61, no. 12, pp. 2544–2558, 2010.

[34] Z. Tufekci, "Big questions for social media big data: Representativeness, validity and other methodological pitfalls," *arXiv preprint arXiv:1403.7400*, 2014.

[35] D. Ruths and J. Pfeffer, "Social media for large studies of behavior," *Science*, vol. 346, no. 6213, pp. 1063–1064, 2014.

[36] G. Beigi, J. Tang, and H. Liu, "Signed link analysis in social media networks." in *ICWSM*, 2016, pp. 539–542.

[37] D. D. Lee and H. S. Seung, "Algorithms for non-negative matrix factorization," in *Advances in neural information processing systems*, 2001, pp. 556–562.

[38] C. Liu, H.-c. Yang, J. Fan, L.-W. He, and Y.-M. Wang, "Distributed nonnegative matrix factorization for web-scale dyadic data analysis on mapreduce," in *Proceedings of the 19th international conference on World wide web*. ACM, 2010, pp. 681–690.

[39] R. Gemulla, E. Nijkamp, P. J. Haas, and Y. Sismanis, "Large-scale matrix factorization with distributed stochastic gradient descent," in *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2011, pp. 69–77.

[40] F. Li, B. Wu, L. Xu, C. Shi, and J. Shi, "A fast distributed stochastic gradient descent algorithm for matrix factorization," in *Proceedings of the 3rd International Conference on Big Data, Streams and Heterogeneous Source Mining: Algorithms, Systems, Programming Models and Applications-Volume 36*. JMLR. org, 2014, pp. 77–87.

[41] C. H. Ding, T. Li, and M. I. Jordan, "Convex and semi-nonnegative matrix factorizations," *IEEE transactions on pattern analysis and machine intelligence*, vol. 32, no. 1, pp. 45–55, 2010.

Co-Clustering Signed 3-Partite Graphs

Sefa Şahin Koç, İsmail Hakkı Toroslu
Computer Engineering Department
Middle East Technical University, Ankara, Turkey
Email: {sefa.koc, toroslu}@ceng.metu.edu.tr

Hasan Davulcu
Computer Science and Engineering Department
Arizona State University, Tempe, AZ
Email: hdavulcu@asu.edu

***Abstract*—In this paper, we propose a new algorithm, called STRICLUSTER, to find tri-clusters from signed 3-partite graphs. The dataset contains three different types of nodes. Hyperedges connecting three nodes from three different partitions represent either positive or negative relations among those nodes. The aim of our algorithm is to find clusters with strong positive relations among its nodes. Moreover, negative relations up to a certain threshold is also allowed. Also, the clusters can have no overlapping hyperedges. We show the effectiveness of our algorithm via several experiments.**

I. INTRODUCTION

A hyperedge in a tri-partite graph represents the relationship among the three nodes it connects. For example, in a social tagging system, which contains three types of nodes (users, tags, resources), a hyperedge means that a user annotates a resource with a tag [1]. A tripartite cluster of these hyperedges may give many information such as users' attitudes to multiple resources or users with common interests. As another example, in a biological analysis system, a level of gene in a sample at a particular time can be represented as a tripartite hyperedge. By mining tripartite clusters, genes showing common characteristics in samples at common time slots could be extracted [2].

Finding biclusters with maximum size from a bipartite graph is proven to be NP-hard, as well as discovering tripartite clusters with maximum size [3]. Therefore, works in literature [2], [4], [1] apply heuristics to determine clusters. As a common strategy, tri-clusters are generated by first constructing biclusters between each pair of three partitions [4]. Then, each bicluster is matched with two others in order to construct tri-clusters. Since this approach is very costly, as an alternative, first, two partitions are selected, and, then, biclusters of these bipartite graphs are constructed. After that, by iterating each one of these biclusters on the third partition, tripartite clusters can be constructed [2]. However, since the first two partitions are fixed, this approach has bias against the third partition. As another approach, tripartite clusters focus on one-to-one correspondence among the nodes [5], [6]. However, the real-world data is usually more complex. For example, in a social tagging system, a group of users may tag multiple sources with the same set of tags, which corresponds to many-to-many relationship.

In this paper, we present an effective algorithm which generates tri-clusters from tripartite hyperedges with positive signs. Our method has the following properties: 1) A minimum threshold for positive signed hyperedge density ratio over all possible hyperedges among tri-partitions of the cluster is defined, and, it must be satisfied by clusters. 2) A simple greedy approach is used in order to trim the hyperedges from tri-clusters with negative signs to increase the positive density ratio of the cluster. 3) In order to prevent constructing very small clusters, both negative signed hyperedges and triples with no connections are also allowed as long as they satisfy user defined density threshold constraints. 4) Clusters are not allowed to have overlaps in terms of hyperedges. A simple heuristic is used to mark hyperedges in order to prevent hyperedge overlaps among clusters, and fast termination of the algorithm while searching potentially maximal clusters. 5) The effectiveness of our approach is shown using a coverage-based metric.

To the best of our knowledge this is the first work that attempts to find co-clusters with potentially overlapping nodes from signed tri-partite graphs. In our work, the clusters are constructed considering the hyperedges, and thus, it is possible to generate clusters with common nodes from the same dimension. A typical problems that motivates this work is finding co-clusters from sentiments of tweets on issues. The three dimensions of this problems are people who write tweets, the selected set of issues (or named entities) and the chosen sentiment words by the users on these issues. The sign of the sentiment words also sepresent the sign of the hyperedge between the three nodes of these dimenstions. It is very likely that same sentiment words are used by people with different clusters corresponding to different camps. Similarly more than one camp may have similar sentiments towards the same issue as well. Therefore, clusters generated on sentiment words and on issues which corresponds to positive feelings of different camps may have many common items. Even on people dimension, it is likely to generate clusters with several common people, which may be interpreted as these people being close to more than one different political camps.

The rest of the paper is organized as follows. Section II introduces STRICLUSTER algorithm. Section III presents experiments and section IV concludes the paper.

II. THE STRICLUSTER ALGORITHM

In this paper, we use the notations given in Table 1. STRICLUSTER algorithm takes a set of hyperedges, Γ as an input, such that each hyperedge h connects three nodes from three different types $U = (U_1, U_2, U_3)$. Figure 1 illustrates

TABLE I
SYMBOL TABLE

Symbol	Meaning
Γ	set of hyperedges
$\Gamma_{v/iv}$	set of valid/invalid hyperedges
α	a tripartite cluster
ϵ_p	minimum ratio of h_+ in a cluster
ϵ_n	maximum ratio of h_- in a cluster
λ_i	minimum size for type i in a cluster
L_i	number of nodes for type i in a cluster (size of type i)
$\Re$	set of tripartite clusters
$h_{+/-}$	a hyperedge with positive/negative label
U_i	set of nodes for type i
E_{ir}	affectiveness value for node r of type i
U_{ir}	minimum E_{ir} in U_i, which belongs to node r
S_i	maximum number of h in which a node from type i can be

hyperedges given as 3D matrix. These hyperedges have either positive or negative labels which are also represented by green and red colors respectively in Figure 1. Remaining entries (white cells) corresponds to node triples without connecting hyperedges. In the example, nodes are {{A,B}, {a,b,c,d,e}, {1,2,3,4,5}} from types U_1, U_2, U_3 respectively.

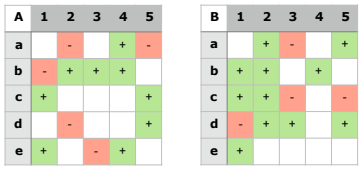


Fig. 1. Input Data

The aim of STRICLUSTER is to find tripartite clusters of hyperedges with highly positive labels. To be a valid tripartite cluster, it has to satisfy threshold values for both density and size. The density threshold values are ϵ_p and ϵ_n, such that $0 \leq \epsilon_p, \epsilon_n \leq 1$, $(\epsilon_p + \epsilon_n) \leq 1$. The former one represents the minimum ratio density of positive hyperedges (h_+) among all possible hyperedges (i.e., there may be $L_1 \times L_2 \times L_3$ number of possible hyperedges for a cluster with size (L_1, L_2, L_3), where L_i is number of nodes with U_i type in the cluster). If C_p is the number of h_+, then:

$$\epsilon_p \leq \frac{C_p}{L_1 \times L_2 \times L_3}, \tag{1}$$

If $\epsilon_p = 1$, generated tripartite clusters become tripartite cliques as well. ϵ_n is the value to control the density of negatively signed hyperedges (h_-). If C_n represnts the number of h_-, then:

$$\epsilon_n \geq \frac{C_n}{L_1 \times L_2 \times L_3}, \tag{2}$$

shows maximum allowed tolerance of h_- in a cluster if $\epsilon_n \neq 0$.

In order to prevent constructing very small clusters λ_i is defined, such that:

$$L_i \geq \lambda_i, \tag{3}$$

for $1 \leq i \leq 3$, and this constraint should also be satisfied by every cluster.

Algorithm 1 STriCluster Algorithm

```
1:  procedure STRICLUSTER(Γ, ε_p, ε_n, λ_1, λ_2, λ_3)
2:      loop
3:          generate α from Γ
4:          β = CLEANINVALIDS(Γ, Γ_iv, α, λ_1, λ_2, λ_3)
5:          if not β then
6:              return ℜ
7:          end if
8:          DENSITYCHECKING(α, ε_p, ε_n, λ_1, λ_2, λ_3)
9:          if α ⇀ formula (3) then          ▷ if α satisfies
10:             ℜ ← ℜ ⊕ α                    ▷ ⊕ means appending
11:         end if
12:         for each h in α do               ▷ h is a hyperedge in α
13:             Γ_iv ← Γ_iv ⊕ h
14:         end for
15:     end loop
16: end procedure
```

STRICLUSTER algorithm (Algorithm 1) starts by generating a potential cluster α which contains all hyperedges in Γ. For the example input data in Figure 1, α initially is equal to the whole graph. If there are invalid hyperedges (used to prevent hyperedge overlaps), they will be removed from α (Section 2.B). After invalid hyperedges are removed, if α does not satisfy the condition (3), (i.e., β is *FALSE*), the algorithm terminates.

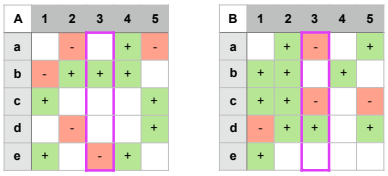
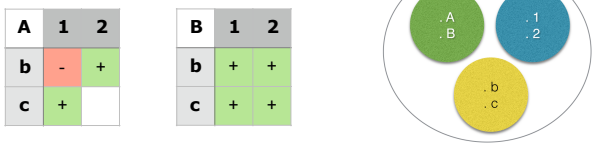


Fig. 2. Removing Node (3) From a Potential Cluster

After a potential cluster α is generated, density check operation is applied on α (Section 2.A). This operation aims to get α to satisfy conditions (1), (2), and (3). There are three possible cases that can happen: In the first case (case I), if conditions (1) or (2) are not satisfied, the least useful node is removed from α (Figure 2) iteratively, until both constraints are satisfied. For the example in Figure 1, this step will remove nodes {{a,d,e}, {3,4,5}} from types U_2, U_3 respectively from α. As the second case (case II), if the removal of a node from α violates the constraint (3), the process stops. In this case, one h_- in α is labeled as invalid. This prevents the construction of exactly the same potential tripartite cluster again, because of CLEANINVALIDS operation (Line 4) of the algorithm. As the third case (case III), α satisfies conditions (1), (2), and (3). Then, DENSITYCHECKING operation returns

α (as a reference parameter). In the example, returned cluster α contains nodes {{A,B}, {b,c}, {1,2}} shown in Figure 3. In this case, α is added to the cluster list $\Re$ (Line 7). After that, all hyperedges in α are labeled as invalid and added into Γ_{iv} which is the list of invalid hyperedges (Line 13). The following sub-section describes the details of density checking procedure which handles these operations.



(a) Matrix Representation (b) Cluster Representation

Fig. 3. A Tripartite Cluster Mined in Given Input

A. Density Checking

If given cluster α does not satisfy conditions (1) and (2), the density checking algorithm searches for nodes to exclude until α satisfies these constraints. If a node is connected by high number of h_+, it should be less likely to be removed. We define hyperedge's usefullness as follows:

$$val(h) = \begin{cases} 2 & \text{if } h \text{ has positive label} \\ -1 & \text{if } h \text{ has negative label.} \end{cases} \quad (4)$$

Then, the effectiveness of a node (r) is determined with the following formula where S_i represents the maximum number of hyperedges which contains that node in U_i:

$$E_{ir} = \frac{\sum_{h \in \alpha} \begin{cases} val(h) & \text{if } r \in h \\ 0 & \text{otherwise} \end{cases}}{S_i}. \quad (5)$$

Then, for each partition, all nodes are checked to find a node with minimum effectiveness value:

$$U_{ir} = min(\sum_{r \in U_i} E_{ir}). \quad (6)$$

At the end of this stage, there will be three U_{ir} values which are U_{1a}, U_{2b}, U_{3c} corresponding to partitions U_1, U_2, U_3 respectively. Minimum of them will be the effectiveness value E_{ix} of node x. This node will be the one to be removed from type i in α in this iteration.

TABLE II
NUMBER OF POSSIBLE HYPEREDGES FOR EACH NODE TYPE

Type	Value
S_1	$L_2 \times L_3$
S_2	$L_1 \times L_3$
S_3	$L_1 \times L_2$

For the input in Figure 1, when DENSITYCHECKING operation applied on α, node 3 will be removed in the first iteration (Figure 2). Node 3 has the lowest effectiveness value, E_{33}, compared to others. E_{33} is obtained as $\frac{2+2-1-1-1}{2\times}5 = 0.1$ using formula (5). In following iterations, nodes $\{a, d, e, 3, 4, 5\}$ will be removed from α. Then, DENSITYCHECKING will reach to case III satisfying all three conditions (1), (2), and (3) and, then returns α which contains nodes $\{A, B, b, c, 1, 2\}$.

If α does not satisfy conditions (1) and (2) and if node removal results the violation of condition (3), it means that DENSITYCHECKING is in case II. In this case, the procedure marks the first h_- as invalid.

B. Clean Invalids

Invalid hyperedges include the hyperedges of all tripartite clusters previously generated as well as all edges marked as invalid by DENSITYCHECKING. In order to remove a hyperedge one of the nodes from this hyperedge should be removed. To do this, CLEANINVALIDS procedure picks a node to remove and it repeats the same action until no invalid hyperedge is left in α. While selecting a node, it uses a heuristic that reduces the cluster size as minimum as possible.

In order to do this, we first determine the number of invalid hyperedges connected to each node. If the ratio of this number to S_i is high, that node is more likely to be removed. This ratio, called as θ_{ir} for node r from type i. Then, we calculate the effectiveness for all the valid nodes of α, which is called as E^v_{ir}. These two values values are combined with the following formula:

$$\gamma_{ir} = \frac{\theta_{ir}}{0.9 + E^v_{ir}}. \quad (7)$$

Among all nodes, the one which has highest γ value is the one to be removed.

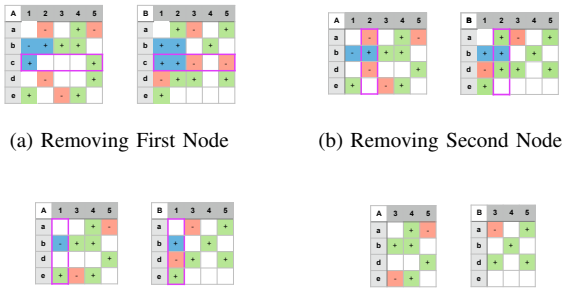
As a constraint, if removing node x will result violation of condition (3), CLEANINVALIDS procedure returns **FALSE**. If there is no invalid hyperedge left in α, CLEANINVALIDS returns **TRUE**.

For the input data in Figure 1, STRICLUSTER algorithm finds the cluster in Figure 3 in the first iteration. Then, hyperedges of this newly generated cluster are labeled as invalid. In the next iteration, new potential cluster α (Figure 4-a) is generated from Γ. But α contains some invalid hyperedges (colored with blue in Figure 4-a). Therefore, α is passed to CLEANINVALIDS procedure to be cleaned from invalid hyperedges. First, node c is removed since γ_{2c} is $3 \div 0.9 = 3.33$, is the maximum among γ values. Then, nodes 2 and 1 are selected and removed respectively (Figure 4-b, 4-c). This will result a clean α (Figure 4-d) and the procedure terminates.

III. EXPERIMENTS

In order to evaluate our algorithm, we have generated data sets with varying sizes. We have done all the experiments on MacBook Pro Mid 2015 (Intel i7 2,5 GHz, 16GB memory).

In the first set of experiments, we have fixed h_+ and h_- density ratios while changing input sizes. Other parameters are also fixed as ϵ_p = 0.75, ϵ_n = 0.10, λ_i = (2,2,2). In this test, we have generated 6 sample datasets. Each one contains



(a) Removing First Node (b) Removing Second Node

(c) Removing Third Node (d) A Clean Tripartite Cluster

Fig. 4. A Scenario of CLEANINVALIDS Procedure

positive hyperedges with 60%, negative hyperedges with 20%, and 20% is empty. $(L_1 \times L_2 \times L_3)$ values for these samples are (31.25K, 62.5K, 125K, 250K, 500K, 1M) respectively. Figure 5 presents the results.

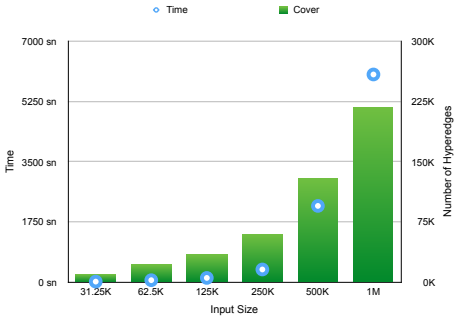


Fig. 5. Test Scenario Depending on Input Size

In the second test, we have generated 5 datasets. In this test, we have fixed the size as $(L_1 \times L_2 \times L_3) = 125K$ and we have varying density ratios for (h_+, h_-) pairs as {(0.2,0.4), (0.2,0.2), (0.4,0.2), (0.4,0.4), (0.6,0.2)}. The results are shown in Figure 6.

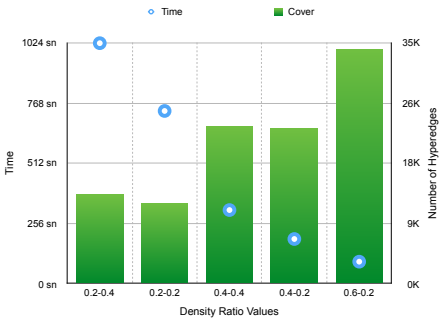


Fig. 6. Test Scenario Depending on Density Ratios in Input Data

The figures show both execution times and the number of hyperedges included in the constructed clusters. We prefer most (positive) hyperedges to be included in clusters while clusters being non-trivial. The results show that we have achieved very high coverage in that sense, since constructed clusters include almost as many hyperedges as the half of the number of positively signed hyperedges.

We have also tested our approach using real data set, which corresponds to the tweets of users on selected issues. These tweets are processed in order to determine the sentiments of users towards these issues. From these tweets, a three dimensional data set is generated. These dimensions are users, issues, and sentiment words chosen by the users on these issues, which also represent the sign of the hyperedge connecting these three items. We have large datasets with 10K users, 45K sentiment words and 20 different issues. This 3-dimensional data is very sparse with only 280K non-empty entries. So far, we have applied our algorithm to a fraction of this dataset which corresponds to randomly select few percentages of it. We have obtained fairly large and overlapping clusters in all three dimensions. Some largest clusters have as many items as the 10% of the nodes of its corresponding dimensions, even for user or sentiment words dimensions.

IV. CONCLUSION

In this paper, we have proposed a new method, called STRICLUSTER, to mine tripartite clusters of positively labeled hyperedges. The input data is composed of three dimensions. Each hyperedge connects three nodes from each dimension. Clusters are generated depending on density ratio of positively (minimum) and negatively (maximum) labeled hyperedges.

We have showed the effectiveness of our approach using both syntetic and real data sets.

ACKNOWLEDGMENT

This research was supported partially by USAF Grant FA9550-15-1-0004.

REFERENCES

[1] C. Lu, X. Chen, and E. Park, "Exploit the tripartite network of social tagging for web clustering," in *Proceedings of the 18th ACM conference on Information and knowledge management*. ACM, 2009, pp. 1545–1548.

[2] L. Zhao and M. J. Zaki, "Tricluster: an effective algorithm for mining coherent clusters in 3d microarray data," in *Proceedings of the 2005 ACM SIGMOD international conference on Management of data*. ACM, 2005, pp. 694–705.

[3] M. Dawande, P. Keskinocak, J. M. Swaminathan, and S. Tayur, "On bipartite and multipartite clique problems," *Journal of Algorithms*, vol. 41, no. 2, pp. 388–403, 2001.

[4] L. Zhu, A. Galstyan, J. Cheng, and K. Lerman, "Tripartite graph clustering for dynamic sentiment analysis on social media," in *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*. ACM, 2014, pp. 1531–1542.

[5] X. Liu and T. Murata, "Detecting communities in tripartite hypergraphs," *arXiv preprint arXiv:1011.1043*, 2010.

[6] Y.-R. Lin, J. Sun, P. Castro, R. Konuru, H. Sundaram, and A. Kelliher, "Metafac: community discovery via relational hypergraph factorization," in *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2009, pp. 527–536.

A Dynamic Modularity Based Community Detection Algorithm for Large-scale Networks: DSLM

Riza Aktunc and Ismail Hakki Toroslu
Computer Engineering Department
Middle East Technical University
Ankara, Turkey 06530
Email: {riza.aktunc, toroslu}@ceng.metu.edu.tr

Mert Ozer and Hasan Davulcu
School of Computing, Informatics,
Decision Systems Engineering
Arizona State University
Tempe, USA 85287
Email: {mozer, hdavulcu}@asu.edu

***Abstract*—In this work, a new fast dynamic community detection algorithm for large scale networks is presented. Most of the previous community detection algorithms are designed for static networks. However, large scale social networks are dynamic and evolve frequently over time. To quickly detect communities in dynamic large scale networks, we proposed dynamic modularity optimizer framework (DMO) that is constructed by modifying well-known static modularity based community detection algorithm. The proposed framework is tested using several different datasets. According to our results, community detection algorithms in the proposed framework perform better than static algorithms when large scale dynamic networks are considered.**

I. Introduction

In the last decade, the notion of social networking is emerged and produced very large graphs that consist of the information of their users. These graphs generally consist of the nodes that represent the users; and edges that represent the relations among users. The nodes in these graphs generally tend to get together and construct communities of their own. Thus, it can be stated that social networks commonly have a community structure. These networks can be divided into groups of nodes that have denser connections inside the group; but fewer connections to the outside of the group. For example, in a GSM network, a group of users who call each other more densely than they call other users may construct their own community. In this case, the nodes represent the users and the edges represent the calls that users made. The detection of communities in these large networks is a problem in this area; therefore a lot of community detection algorithms such as [1], [2], [3], [4], [5], [6], [7], [8], [9] proposed in the literature. Almost all these community detection algorithms are static and designed for static networks.

However, most of the social networks are not static because they evolve in many ways. They may gain or lose users that are represented as nodes in the graphs over time. The users of these social networks may lose contact from each other or there can be new connections among users. In other words, some edges in the graphs may be removed or new edges may be added to the graph over time. All these processes may happen in a very small amount of time in a social network if it has a lot of active users. This kind of a social network may be called as highly dynamic. For example, popular social sites such as Facebook, Twitter, LinkedIn and so on have highly dynamic social networks. Moreover, most GSM networks have millions of users and hundreds of calls made in seconds; therefore, they can also be labeled as highly dynamic networks. Addition or deletion of an edge or a node from a network which has millions of edges might seem insignificant; but when this additions or deletions of an edge or a node happen very frequently, they begin to change the community structure of the whole network and become very important. This change in the community structure raises the need of re-identification of communities in the network. This need arises frequently and creates a new problem in the community detection research area. This new problem requires somehow fast detection of communities in dynamic networks.

The first solution that comes to mind for community detection in large dynamic networks problem is the execution of static community detection algorithms already defined in the literature all over again to detect the new community structure whenever the network is modified. Nevertheless, this solution takes too much time in every modification of the large networks since it runs the community detection algorithm from scratch each time. A much efficient and less time consuming solution is to run the community detection algorithms not from scratch but from a point in the history of the network by storing and using the historical results of executions of the algorithms whenever network is evolved. In other words, updating previously discovered community structure instead of trying to find communities from scratch each time the network evolves consumes much less time and thus much efficient. This solution method for the problem of detecting communities in large dynamic networks is the main focus of our study in this paper.

In this paper, we modified the smart local moving (SLM) algorithm defined by Waltman & Van Eck [9] so that it would detect the communities in rapidly growing large networks dynamically and efficiently. As a result, we propose the dynamic SLM (dSLM) algorithm that dynamically detects communities in large networks by optimizing modularity and using its own historical results. We tested our proposed approach on several different datasets. We demonstrated the effects of our contribution to the SLM algorithm in two ways. One of them is the change in modularity value which determines the quality of the community structure of the network. The other one is the change in running time that determines the pace of the algorithm. The latter is more significant than the former because the community structure of the network must be quickly identified at the given timestamp before the next

timestamp is reached. We realized that dSLM improved SLM by decreasing its running time incredibly. Moreover, there are some experiments where modularity value increases while running time decreases.

The rest of the paper is organized as follows. Section II introduces previous researches done in the area. Section III explains the modularity. The proposed solution for dynamic community detection in large networks called as dSLM and its static version SLM are described in Section IV. In Section V, the results of the experiments of SLM and dSLM are demonstrated. Finally, the paper is concluded in Section VI.

II. Related Work

The idea of modularity-based community detection is to try to assign each vertex of the given network to a community such that it maximizes the modularity value of the network. Optimizing modularity is an NP-hard problem. [10] Exact algorithms that maximize modularity such as [11], [10], [12] can be used only for small networks.

For large-scale modularity optimization, heuristic algorithms are proposed. We basically focus on three well known algorithms, namely; CNM, Louvain and SLM. The first one is Clauset et al.'s [13] CNM algorithm. It is a greedy modularity maximization algorithm that searches for best community assignment for each node. The second one is referred as Louvain algorithm and proposed by Blondel et al. [7] in 2008. By considering each community as a single node, it further searches for new community merges after the local optimum satisfied using CNM. The last one is called as Smart Local Moving (SLM) algorithm that is proposed by Waltman and Jan van Eck in 2013. [9] SLM algorithm is explained in detail in chapter III.

Due to the dynamic features of many social networks [14], the need for detecting communities dynamically in the large networks is emerged in the latest years. There have been many community detection algorithms proposed in the literature to fulfill this need. Xu et al. divides the current research on community evolution into the following categories. Parameter estimation methods and probabilistic models have been proposed in the literature. [15], [16] A methodology that tries to find an optimal cluster sequence by detecting a cluster structure at each timestamp that optimizes the incremental quality can be classified as evolutionary clustering. [17], [18] Furthermore, tracking algorithms based on similarity comparison have also been studied in order to be able to describe the change of communities on the time axis. [19], [20] Apart from these algorithms that are focused on the evolution procedures of communities, community detection in dynamic social networks aims to detect the optimal community structure at each timestamp. For this purpose, incremental versions of both CNM and Louvain algorithm are proposed by Dinh et al.[21] and Aynaud et al. [22]. To the best of our knowledge, this is the first work considering the incremental version of Smart Local Moving algorithm in literature. Our algorithm can be classified as the last mentioned category which aims to detect optimal community structure at each timestamp with minimum running time.

III. Modularity

Modularity is a function that is used for measuring the quality of the results of community detection algorithms. If the modularity value of a partitioned network is high, it means that the network is partitioned well. Apart from quality measurement, modularity is used as the basis of some community detection algorithms. These algorithms try to detect communities (partitions) in a network by trying to maximize the modularity value of the network. Thus, modularity is a function that is used for both quality measurement and community detection.

Modularity is based on the idea that a randomly created graph is not expected to have community structure, so comparing the graph at hand with a randomly created graph would reveal the possible community structures in the graph at hand. This comparison is done through comparing the actual density of edges in a subgraph and the expected edge density in the subgraph if the edges in the subgraph were created randomly. This expected edge density depends on how random the edges created. This dependency is tied to a rule that defines how to create the randomness and called as null model. A null model is a copy of an original graph and it keeps some of this original graphs structural properties but not reflects its community structure. There can be multiple null models for a graph such that each of them keeps different structural properties of the original graph. Using different null models for the calculation of the modularity leads to different modularity calculation methods and values. The most common null model that is used for modularity calculation is the one that preserves the degree of each vertex of the original graph. With this null model, modularity is calculated as the fraction of edges that fall in the given communities minus such fraction in the null model. [23], [24] The formula of modularity can be written as in Equation 1

$$Q = \frac{1}{2m} \sum_{ij} (A_{ij} - P_{ij}) \delta(C_i, C_j) \tag{1}$$

m represents the total number of edges of the graph. Sum iterates over all vertices denoted as i and j. A_{ij} is the number of edges between vertex i and vertex j in the original graph. P_{ij} is the expected number of edges between vertex i and vertex j in the null model. The δ function results as 1 if the vertex i and vertex j are in the same community ($C_i = C_j$), 0 otherwise. The null model can be created by cutting the edges between vertices; thus, creating stubs (half edges) and rewiring them to random vertices. Thus, it obeys the rule of keeping degrees of vertices unchanged. Cutting edges into half, creates $m * 2 = 2m$ stubs. In the null model, a vertex could be attached to any other vertex of the graph and the probability that vertices i and j, with degrees k_i and k_j , are connected, can be calculated. The probability p_i to pick a ramdom stub connection for vertex i is $\frac{k_i}{2m}$, as there are k_i stubs of i out of a total of $2m$ stubs. The probability of vertex i and vertex j being connected is $p_i p_j$, since stubs are connected independently of each other. Since there are $2m$ stubs, there are $2m p_i p_j$ expected number of edges between vertex i and vertex j. [24] This yields to equation 2

$$P_{ij} = 2mp_ip_j = 2m\frac{k_ik_j}{4m^2} = \frac{k_ik_j}{2m} \quad (2)$$

By placing equation 2 into equation 1, modularity function is presented as in equation 3.

$$Q = \frac{1}{2m}\sum_{ij}(A_{ij} - \frac{k_ik_j}{2m})\delta(C_i, C_j) \quad (3)$$

The resulting values of this modularity function lie in the range $\left[\frac{-1}{2}, 1\right)$. It would be positive if the number of edges within subgraphs is more than the number of expected edges in the subgraphs of null model. Higher values of the modularity function mean better community structures. [9]

This modularity function also applies to weighted networks. [9], [25] The modularity function for the weighted graphs can be calculated as in equation 4.

$$Q_w = \frac{1}{2W}\sum_{ij}(W_{ij} - \frac{s_is_j}{2W})\delta(C_i, C_j) \quad (4)$$

There are three differences. The first difference is that in the case of a weighted network W_{ij}, instead of A_{ij}, may take not just 0 or 1 but any non-negative value that represents the weight of the edge. The second one is that instead of m, which is total number of edges, W, which is the sum of the weights of all edges is used in the equation. The last one is that s_i and s_j which represents the sum of the weights of edges adjacent to vertex i and vertex j respectively is used in the equation instead of k_i and k_j which means the degree of vertex i and vertex j respectively. [24]

Apart from weighted networks, the modularity function defined in 3 has been extended in order to be also applicable to directed networks. [26], [27] When the edges are directed, stubs will also be directed and it changes the possibility of rewiring stubs and connecting edges. The calculation of this possibility in the directed case depends on the in- and out-degrees of the end vertices. For instance, there are two vertices A and B. A has a high in-degree and low out-degree. B has a low in-degree and high out-degree. Thus, in the null model of modularity, an edge will be much more likely to point from B to A than from A to B. [24] Therefore, the expression of modularity for directed graphs can be written as in equation 5

$$Q_d = \frac{1}{m}\sum_{ij}(A_{ij} - \frac{k_i^{out}k_j^{in}}{m})\delta(C_i, C_j) \quad (5)$$

The sum of the in-degrees (out-degrees) equals m not $2m$ as in the case of undirected graph. Therefore, the factor 2 in the denominator of the first and second summand has been dropped. In order to get the modularity function to be applicable to directed weighted networks, the equations 4 and 5 can be merged; thus, equation 6 can be constructed as the most general expression of modularity. [24]

$$Q_{dw} = \frac{1}{W}\sum_{ij}(W_{ij} - \frac{s_i^{out}s_j^{in}}{W})\delta(C_i, C_j) \quad (6)$$

There have been a few proposals of modified version of the modularity functions defined above as alternative modularity functions. These modified, extended versions for instance offer a resolution parameter that makes it possible to customize the granularity level at which communities are detected and to mitigate the resolution limit problem defined by Fortunato and Barthlemy [28]. [29] Moreover, there are modularity functions with a somewhat modified mathematical structure in the literature such as Reichardt & Bornholdt, 2006; Traag, Van Dooren, & Nesterov, 2011; Waltman, Van Eck, & Noyons, 2010. [9], [28], [30], [31]

IV. SLM and dSLM Algorithms

A. SLM Algorithm

SLM is a community detection algorithm that is evolved from Louvain algorithm. Louvain algorithm is a large scale modularity based community detection algorithm that is proposed by Blondel et al in 2008. [7] The quality of detected communities by Louvain algorithm is measured by the method called modularity. The modularity of a network is a value that is between -1 and 1. This value presents the density of links inside communities over the density of links between communities. [23] When this value is close to 1, then the measured network can be called as modular network. In the case of weighted networks, modularity function can take weights into consideration and measure the quality of detected communities. Louvain algorithm uses modularity function as not only a measurement function but also an objective function to optimize.

Louvain algorithm is a recursive algorithm which has two steps running in each recursive call. Before the recursion starts, the algorithm assigns a different community to each node of the network whose communities are going to be detected. Therefore, in the initial case each node has its own community. In each recursive call the following steps are run:

1) It runs a local moving heuristic in order to obtain an improved community structure. This heuristic basically moves each node from its own community to its neighbors' community and run the modularity function. If the result of the modularity function, which means quality, increased, the node would be kept in the new community; else, the node would be moved back to its previous community. This process is applied to each node for its each neighbor in random order and thereby heuristically the quality is tried to be increased.
2) The algorithm constructs a reduced network whose nodes are the communities that are evolved in the first step. Moreover, the weights of the edges in this reduced network are given by the sum of weights of the links between the nodes which reside in the corresponding two communities. Links between nodes of the same community in the old network are presented as self-links for the node that represents that community in the new reduced network. When this reduced network is fully constructed, then algorithm calls itself recursively and first step is applied to this reduced network.

The algorithm keeps recursing until no further improvement in modularity is measured and thereby there are no changes in the community structure. [7]

Louvain algorithm detects community structures whose modularity values are locally optimal with respect to community merging, but not necessarily locally optimal with respect to individual node movements. Since the Louvain algorithm applies local moving heuristic in the beginning of its recursive block and merges communities by reducing network in the end of its recursive block, calling it iteratively ensures that the resulting community structure cannot be improved further either by merging communities or by moving individual nodes from one community to another. Like the iterative variant of these algorithms SLM algorithm constructs community structures that are locally optimal with respect to both individual node movements and community merging. Besides these capabilities, SLM also tries to optimize modularity by splitting up communities and moving sets of nodes between communities. This is done by changing the way that local moving heuristic and network reduction runs.[9]

Louvain algorithm runs local moving heuristic algorithm on the present network as the first step, and then construct the reduced network as the second step. However, the SLM algorithm changes the reduced network construction step by applying following processes:

1) It iterates over all communities that are formed by the first step. It copies each community and constructs a subnetwork that contains only the specific community's nodes.
2) It then runs the local moving heuristic algorithm on each subnetwork after assigning each node in the subnetwork to its own singleton community.
3) After local moving heuristic constructs a community structure for each subnetwork, the SLM algorithm creates the reduced network whose nodes are the communities detected in subnetworks. The SLM algorithm initially defines a community for each subnetwork. Then, it assigns each node to the community that is defined for the node's subnetwork. Thus, there is a community defined for each subnetwork and detected communities in subnetworks are placed under these defined communities as nodes in the reduced network.

This is the way that the SLM algorithm constructs the reduced network. After these processes, the SLM algorithm gives the reduced network to the recursive call as input and all the processes starts again for the reduced network. The recursion continues until a network is constructed that cannot be reduced further. To sum up, the SLM algorithm has more freedom in trying to optimize the modularity by having the ability to move sets of nodes between communities which cannot be done by Louvain algorithm. [9]

B. Dynamic Smart Local Moving Algorithm

SLM algorithm initially assigns each node to a different community, so each node has its own singleton community. In order convert SLM to dynamic form, we replace that operation with a newly defined procedure, called initialize communities which is given in Figure 1. This procedure works as follows:

Procedure: Initialize Communities
Input: Old_Communities, Old_Network, New_Network
Output: New_Communities

1: $j = 0$
2: **for** $i = 0 \rightarrow Old\_Communities.size$ **do**
3: $\quad New\_Communities = ReadCommunitiesFile()$
4: **end for**
5: $Delta\_Network = New\_Network - Old\_Network$
6: **for** $j = i \rightarrow Delta\_Network.size$ **do**
7: $\quad new\_communities[j] = Delta\_Network[j - i]$
8: **end for**

Fig. 1. Initialize Communities Procedure

- Existing communities are read from file as New Communities.
- If exists, the extensions to the network has been determined.
- For each new node, singleton new communities are constructed and added to New Communities.

The effects of other changes in the network, such as adding new edges and deletions of nodes and edges, are handled while executing standard SLM procedure. The new dSLM is available at https://github.com/mertozer/dSLM.

Since after some iterations, the increase of modularity drops to very small values, it might make sense to stop the iterations using either the amount of changes or by setting a target modularity value. We have implemented the second option, which is called dSLMEVS in the experiments. We have made the tests by setting the modularity values as the one obtained for SLM in order to be able to observe the differences in the execution times for exactly the same modularity values.

C. Running Example

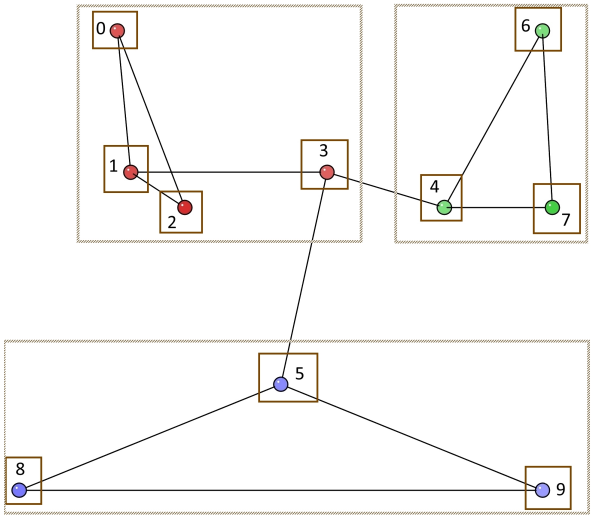


Fig. 2. Network in time t (analyzed by SLM)

Let the sample network depicted in Figure 2 to be a network that changes in time and needs to be analyzed continuously in each time frame. So, the network is analyzed and communities are detected in time t. Figure 2 presents the

beginning and end states of the community structure of the network analyzed by SLM algorithm in time t. Solid rectangles present the initial community structure; whereas the colors of the nodes (and dashed rectangles) present the resulting community structure. From time t to time t+1, a node which is numbered as 10 and an edge between this new node and the node which is numbered as 9 are added to the network. This evolved network in time t+1 can be seen in Figure 3 and Figure 4. Figure 3 presents the community detection process

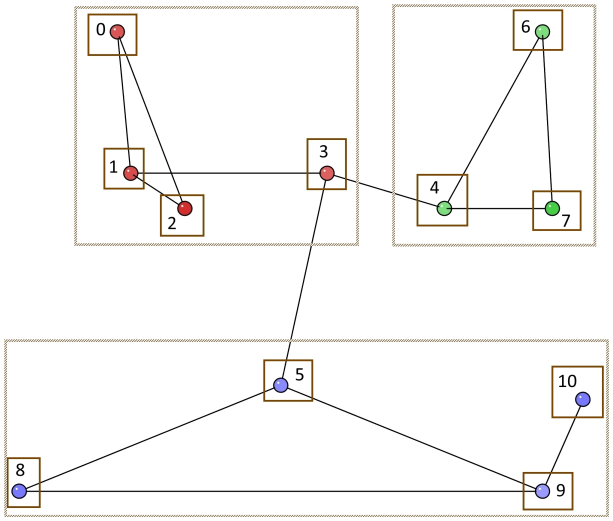


Fig. 3. Network in time t+1 (analyzed by SLM)

of the network in time t+1 performed by SLM algorithm in the same way as Figure 2. As the difference of Figure 2 and Figure 3, a new node and a new edge are only seen in Figure 3. Since they both demonstrate the SLM process, the initial communities are singleton. Figure 4 demonstrates the

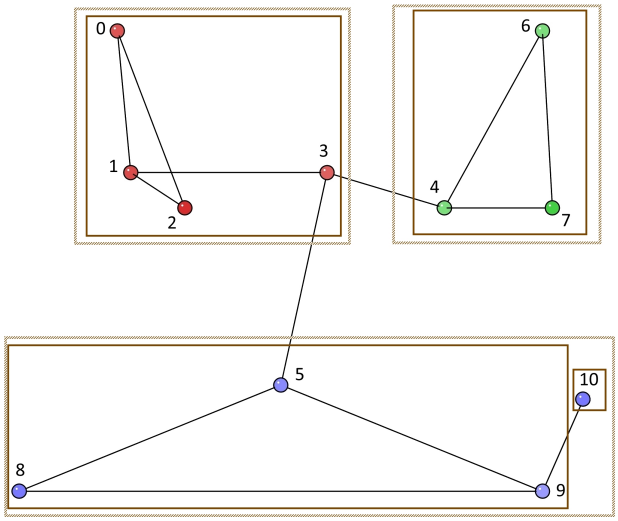


Fig. 4. Network in time t+1 (analyzed by dSLM)

dSLM process of the network in time t+1. In this process, the community structure of the network in time t is used as the initial states of communities which can be seen as rectangles in Figure 4. Both SLM and dSLM algorithms place the newly added node in blue community. SLM constructs the community structure from scratch by trying and finding node movements that maximize the modularity of the network. However, dSLM needs to try only one node movement which is to move newly added node from its singleton community to its only neighbor (blue) community. Since it appears to increase the modularity of the network, dSLM places the new node to blue community and that is it. Because the initial community structure is known to be the one that maximizes the modularity of the network, there is no other node movement trying that can increase modularity. By this way, the dSLM runs faster than SLM . This run time difference between SLM and dSLM gets much greater while the network size increases.

V. Experiments & Results

We evaluate our proposed approach dSLM on five real-world datasets which are the arXiv citation dataset, the GSM calls dataset, Google Plus, Twitter and Youtube user network datasets..

The arXiv[1] citation dataset is published in the KDD Cup 2003. It contains approximately 29,000 papers and their citation graph. In this graph, each vertex represents a paper and each edge represents the citation between its connected vertexes. There are around 350,000 edges which represent citations in this graph.

We have used call detail record (CDR) dataset, obtained from one of the largest GSM operators in Turkey. This produced GSM calls dataset contains 12,521,352 nodes and 44,768,912 edges. These two datasets are used for edge deletion and addition experiments purposes.

The Google Plus and Twitter user network data is collected by Stanford Network Analysis Project[2]. The Google Plus data consists of 107,614 nodes and 13,673,453 edges. The Twitter data consists of 81,306 nodes and 1,768,149 edges. The Youtube user network data is provided by Mislove et al. [32]. It consists of 1,134,890 nodes and 2,987,624 edges. The users of the Youtube are the nodes, and the friendships are represented by edges. We used these 3 datasets for node deletion and addition experiments purposes.

In the first set of experiments we have assumed the dynamic feature is in the form of edge insertions and deletions. Table I and II gives the results for these experiements. In the second set, on the other hand, node insertions and deletions represent the dynamic feature. Table III and IV presents the results for these experiments.

TABLE I. The effect of dSLM for edge insertions

Algorithm	Dataset	Base (# of Edges)	# of Edges Added	Change in Modularity Value	Decrease in Running Time
dSLM	arxiv	300,000	1,000	0.02% increased	26%
dSLM	arxiv	300,000	10,000	0.15% increased	26%
dSLM	GSM	10,000,000	1,000	no change	27%
dSLM	GSM	10,000,000	10,000	no change	29%
dSLM	GSM	10,000,000	100,000	no change	20%
dSLM	GSM	10,000,000	1,000,000	0.02% increased	17%
dSLMEVS	GSM	10,000,000	1,000	no change	91%
dSLMEVS	GSM	10,000,000	10,000	no change	63%
dSLMEVS	GSM	10,000,000	100,000	no change	64%
dSLMEVS	GSM	10,000,000	1,000,000	no change	80%

[1]http://www.cs.cornell.edu/projects/kddcup/datasets.html
[2]http://snap.stanford.edu/data

TABLE II. THE EFFECT OF DSLM FOR EDGE DELETIONS

Algorithm	Dataset	Base (# of Edges)	# of Edges Deleted	Change in Modularity Value	Decrease in Running Time
dSLM	arxiv	300,000	1,000	0.08% increased	32%
dSLM	arxiv	300,000	10,000	0.04% increased	7%
dSLM	GSM	10,000,000	1,000	no change	38%
dSLM	GSM	10,000,000	10,000	no change	27%
dSLM	GSM	10,000,000	100,000	0.01% increased	24%
dSLM	GSM	10,000,000	1,000,000	0.01% increased	16%
dSLMEVS	GSM	10,000,000	1,000	no change	92%
dSLMEVS	GSM	10,000,000	10,000	no change	61%
dSLMEVS	GSM	10,000,000	100,000	no change	91%
dSLMEVS	GSM	10,000,000	1,000,000	no change	80%

TABLE III. THE EFFECT OF DSLM FOR NODE INSERTIONS

Algorithm	Dataset	Base (# of Nodes)	# of Nodes Added	Change in Modularity Value	Decrease in Running Time
dSLM	Twitter	81,296	10	no change	67%
dSLM	Twitter	81,206	100	no change	90%
dSLM	Twitter	80,306	1,000	no change	89%
dSLM	Twitter	71,306	10,000	0.04% increased	70%
dSLM	GPlus	107,604	10	0.02% decreased	72%
dSLM	GPlus	107,514	100	no change	53%
dSLM	GPlus	106,614	1,000	no change	54%
dSLM	GPlus	97,614	10,000	0.97% decreased	11%
dSLM	Youtube	1157728	100	1.77% increased	73%
dSLM	Youtube	1156828	1000	1.36% increased	65%
dSLM	Youtube	1147828	10000	0.03% increased	77%
dSLM	Youtube	1057828	100000	0.12% increased	41%
dSLMEVS	Twitter	81,296	10	no change	82%
dSLMEVS	Twitter	81,206	100	no change	90%
dSLMEVS	Twitter	80,306	1,000	no change	79%
dSLMEVS	Twitter	71,306	10,000	no change	75%
dSLMEVS	GPlus	107,604	10	0.02% decreased	70%
dSLMEVS	GPlus	107,514	100	no change	83%
dSLMEVS	GPlus	106,614	1,000	no change	83%
dSLMEVS	GPlus	97,614	10,000	0.04% increased	30%
dSLMEVS	Youtube	1,157,728	100	0.04% increased	92%
dSLMEVS	Youtube	1,156,828	1000	0.13% increased	99%
dSLMEVS	Youtube	1,147,828	10000	0.08% increased	98%
dSLMEVS	Youtube	1,057,828	100000	0.15% increased	98%

TABLE IV. THE EFFECT OF DSLM FOR NODE DELETIONS

Algorithm	Dataset	Base (# of Nodes)	# of Nodes Added	Change in Modularity Value	Decrease in Running Time
dSLM	Twitter	81,306	10	0.06% increased	85%
dSLM	Twitter	81,306	100	no change	79%
dSLM	Twitter	81,306	1,000	0.02% increased	87%
dSLM	Twitter	81,306	10,000	0.03% decreased	28%
dSLM	GPlus	107,614	10	0.35% decreased	90%
dSLM	GPlus	107,614	100	0.37% decreased	78%
dSLM	GPlus	107,614	1,000	0.35% decreased	75%
dSLM	GPlus	107,614	10,000	0.35% decreased	73%
dSLM	Youtube	1,157,828	100	0.28% decreased	88%
dSLM	Youtube	1,157,828	1000	0.03% decreased	87%
dSLM	Youtube	1,157,828	10000	0.21% decreased	73%
dSLM	Youtube	1,157,828	100000	0.09% decreased	73%
dSLMEVS	Twitter	81,306	10	0.05% increased	92%
dSLMEVS	Twitter	81,306	100	0.02% decreased	81%
dSLMEVS	Twitter	81,306	1,000	0.02% increased	87%
dSLMEVS	Twitter	81,306	10,000	0.05% decreased	28%
dSLMEVS	GPlus	107,614	10	0.36% decreased	90%
dSLMEVS	GPlus	107,614	100	0.35% decreased	78%
dSLMEVS	GPlus	107,614	1,000	0.36% decreased	84%
dSLMEVS	GPlus	107,614	10,000	0.34% decreased	69%
dSLMEVS	Youtube	1,157,828	100	0.29% decreased	99%
dSLMEVS	Youtube	1,157,828	1000	0.01% decreased	99%
dSLMEVS	Youtube	1,157,828	10000	0.19% decreased	99%
dSLMEVS	Youtube	1,157,828	100000	0.07% decreased	98%

In general, dSLM does not decrease the number iterations of convergence of SLM, however, it decreases the number of node movements needed in each iteration of SLM. This indicates that each iteration of dSLM runs faster than each iteration of SLM. Therefore, overall running time of dSLM is less than SLM's overall execution time. The overall results can be seen in Table I, II, III and IV.

In order to be able to decrease the overall running time of dSLM algorithm even more, we added another parameter called expected modularity value that enables the algorithm stop when it is reached. We named this kind of new algorithm as dSLMEVS and made same experiments on it with this new parameter set to the modularity value resulted from SLM algorithm. By this new algorithm and parameter, we aimed to decrease running time as much as possible while keeping the modularity value unchanged or increased. We reached our aim and decreased running time drastically and keep modularity value unchanged or increased as seen in all of the tables.

VI. CONCLUSION

Waltman & Van Eck proposed and implemented the SLM algorithm in order to detect communities in large networks. We extended their implementation to define the community structure in a dynamic rather than static way. We made use of the past calculation results of the SLM algorithm in order to calculate the current networks community structure. This usage is the main extension and contribution to the SLM algorithm. In the basics, it is what extends the SLM to be dSLM.

To sum up, we extended SLM to be incremental and dynamic by using the historical results of community detection algorithms for the initial community assignments of the nodes. Thus, the number of node movement actions tried to maximize the modularity value is decreased. This led to decrease in running time of the algorithms. Moreover, it can lead to decrease in number of iterations to converge. Thus, if the algorithms run with a constant number of iterations parameter, the modularity value may result as increased.

ACKNOWLEDGMENT

This research was supported partially by USAF Grant FA9550-15-1-0004.

REFERENCES

[1] A. Clauset, M. Newman, and C. Moore, "Finding community structure in very large networks," *Physical Review E*, vol. 70, p. 066111, 2004. [Online]. Available: http://www.citebase.org/cgi-bin/citations?id=oai:arXiv.org:cond-mat/0408187

[2] R. Guimera, M. Sales-Pardo, and L. Amaral, "Modularity from fluctuations in random graphs and complex networks," *Physical Review E*, vol. 70, no. 2, p. 025101, 2004.

[3] J. Duch and A. Arenas, "Community detection in complex networks using extremal optimization," *Physical Review E*, vol. 72, p. 027104, 2005. [Online]. Available: http://www.citebase.org/abstract?id=oai:arXiv.org:cond-mat/0501368

[4] M. Newman, "Finding community structure in networks using the eigenvectors of matrices," *Physical Review E*, vol. 74, no. 3, p. 36104, 2006.

[5] S. Lehmann and L. K. Hansen, "Deterministic modularity optimization," *The European Physical Journal B*, vol. 60, no. 1, pp. 83–88, 2007. [Online]. Available: http://dx.doi.org/10.1140/epjb/e2007-00313-2

[6] J. Lee, S. P. Gross, and J. Lee, "Mod-csa: Modularity optimization by conformational space annealing," *CoRR*, vol. abs/1202.5398, 2012.

[7] V. Blondel, J. Guillaume, R. Lambiotte, and E. Mech, "Fast unfolding of communities in large networks," *J. Stat. Mech*, p. P10008, 2008.

[8] R. Rotta and A. Noack, "Multilevel local search algorithms for modularity clustering." *ACM Journal of Experimental Algorithmics*, vol. 16, 2011.

[9] L. Waltman and N. J. van Eck, "A smart local moving algorithm for large-scale modularity-based community detection." *CoRR*, vol. abs/1308.6604, 2013.

[10] U. Brandes, D. Delling, M. Gaertler, R. Goerke, M. Hoefer, Z. Nikoloski, and D. Wagner, "On modularity clustering," *IEEE Transactions on Knowledge and Data Engineering*, vol. 20, no. 2, pp. 172–188, 2008.

[11] D. Aloise, S. Cafieri, G. Caporossi, P. Hansen, L. Liberti, and S. Perron, "Column generation algorithms for exact modularity maximization in networks," *Physical Review E*, vol. 82, no. 4, article, pp. –, jan 2010.

[12] G. Xu, S. Tsoka, and L. G. Papageorgiou, "Finding community structures in complex networks using mixed integer optimisation," *The European Physical Journal B*, vol. 60, no. 2, pp. 231–239, 2007. [Online]. Available: http://dx.doi.org/10.1140/epjb/e2007-00331-0

[13] A. Clauset, M. E. J. Newman, and C. Moore, "Finding community structure in very large networks," *Phys. Rev. E*, vol. 70, p. 066111, Dec 2004. [Online]. Available: http://link.aps.org/doi/10.1103/PhysRevE.70.066111

[14] P. Holme and J. Saramäki, "Temporal networks," *Physics Reports*, vol. 519, no. 3, pp. 97–125, 2012.

[15] T. Yang, Y. Chi, S. Zhu, Y. Gong, and R. Jin, "Detecting communities and their evolutions in dynamic social networks - a bayesian approach." *Machine Learning*, vol. 82, no. 2, pp. 157–189, 2011.

[16] X. Tang and C. C. Yang, "Dynamic community detection with temporal dirichlet process." in *SocialCom/PASSAT*. IEEE, 2011, pp. 603–608.

[17] D. Chakrabarti, R. Kumar, and A. Tomkins, "Evolutionary clustering," in *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, ser. KDD '06. New York, NY, USA: ACM, 2006, pp. 554–560. [Online]. Available: http://doi.acm.org/10.1145/1150402.1150467

[18] M.-S. Kim and J. Han, "A particle-and-density based evolutionary clustering method for dynamic networks." *PVLDB*, vol. 2, no. 1, pp. 622–633, 2009.

[19] D. Greene, D. Doyle, and P. Cunningham, "Tracking the evolution of communities in dynamic social networks." in *ASONAM*, N. Memon and R. Alhajj, Eds. IEEE Computer Society, 2010, pp. 176–183.

[20] P. Brodka, S. Saganowski, and P. Kazienko, "Group evolution discovery in social networks," in *Advances in Social Networks Analysis and Mining (ASONAM), 2011 International Conference on*, 2011, pp. 247–253.

[21] T. Dinh, Y. Xuan, and M. Thai, "Towards social-aware routing in dynamic communication networks," in *Performance Computing and Communications Conference (IPCCC), 2009 IEEE 28th International*, Dec 2009, pp. 161–168.

[22] T. Aynaud and J.-L. Guillaume, "Static community detection algorithms for evolving networks," in *Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks (WiOpt), 2010 Proceedings of the 8th International Symposium on*, May 2010, pp. 513–519.

[23] M. Newman, "Modularity and community structure in networks," *Proceedings of the National Academy of Sciences*, vol. 103, no. 23, pp. 8577–8582, 2006.

[24] S. Fortunato, "Community detection in graphs," *Physics Reports*, vol. 486, pp. 75–174, 2010.

[25] M. E. J. Newman, "Analysis of weighted networks," *Phys. Rev. E*, vol. 70, no. 5, p. 056131, Nov. 2004. [Online]. Available: http://pre.aps.org/abstract/PRE/v70/i5/e056131

[26] A. Arenas, J. Duch, A. Fernandez, and S. Gmez, "Size reduction of complex networks preserving modularity," *CoRR*, vol. abs/physics/0702015, 2007.

[27] E. A. Leicht and M. E. J. Newman, "Community structure in directed networks," *Phys. Rev. Lett.*, vol. 100, no. 11, p. 118703, Mar. 2008. [Online]. Available: http://prl.aps.org/abstract/PRL/v100/i11/e118703

[28] S. Fortunato and M. Barthlemy, "Resolution limit in community detection," *Proceedings of the National Acadamy of Sciences of the United States of America (PNAS)*, vol. 104, no. 1, pp. 36–41, 2007.

[29] J. Reichardt and S. Bornholdt, "Statistical mechanics of community detection," *Arxiv preprint cond-mat/0603718*, 2006.

[30] V. A. Traag, P. Van Dooren, and Y. Nesterov, "Narrow scope for resolution-limit-free community detection," *Physical Review E*, vol. 84, no. 1, p. 016114, 2011.

[31] L. Waltman, N. J. van Eck, and E. C. Noyons, "A unified approach to mapping and clustering of bibliometric networks," *Journal of Informetrics*, vol. 4, no. 4, pp. 629–635, 2010. [Online]. Available: http://www.sciencedirect.com/science/article/B83WV-50RFN28-1/2/809f89176e8076cac3862b6589bc6fd5

[32] A. Mislove, M. Marcon, K. P. Gummadi, P. Druschel, and B. Bhattacharjee, "Measurement and Analysis of Online Social Networks," in *Proceedings of the 5th ACM/Usenix Internet Measurement Conference (IMC'07)*, San Diego, CA, October 2007.

A Network-Based Model for Predicting Hashtag Breakouts in Twitter

Sultan Alzahrani[1], Saud Alashri[1], Anvesh Reddy Koppela[1], Hasan Davulcu[1(✉)], and Ismail Toroslu[2]

[1] School of Computing, Informatics and Decision Systems Engineering, Arizona State University, Tempe, AZ 85287, USA
{ssalzahr,salashri,akoppela,hdavulcu}@asu.edu
[2] Department of Computer Engineering, Middle East Technical University, Ankara, Turkey
toroslu@ceng.metu.edu.tr

Abstract. Online information propagates differently on the web, some of which can be viral. In this paper, first we introduce a simple standard deviation sigma levels based Tweet volume breakout definition, then we proceed to determine patterns of re-tweet network measures to predict whether a hashtag volume will breakout or not. We also developed a visualization tool to help trace the evolution of hashtag volumes, their underlying networks and both local and global network measures. We trained a random forest tree classifier to identify effective network measures for predicting hashtag volume breakouts. Our experiments showed that "local" network features, based on a fixed-sized sliding window, have an overall predictive accuracy of 76 %, where as, when we incorporate "global" features that utilize all interactions up to the current period, then the overall predictive accuracy of a sliding window based breakout predictor jumps to 83 %.

Keywords: Information diffusion · Hashtag volumes · Prediction · Social networks · Diffusion networks

1 Introduction

Online Social Networks (OSNs) such as Twitter have emerged as popular microblogging and interactive platforms for information sharing among people. Twitter provides a suitable platform to investigate properties of information diffusion. Diffusion analysis can harness social media to investigate viral tweets and trending hashtags to create early-warning solutions that can signal if a viral hashtag started emerging in its nascent stages. In this paper, we utilize the 68-95-99.7 rule to define a simple method of hashtag volume breakouts. In statistics, the 68-95-99.7 rule, also known as the three-sigma rule or empirical rule, states that nearly all values lie within three standard deviations (σ) of the mean (μ) in a normal distribution. We utilize a fixed sized sliding window (of length 20 daily

N. Agarwal et al. (Eds.): SBP 2015, LNCS 9021, pp. 3–12, 2015.
DOI: 10.1007/978-3-319-16268-3_1

intervals), to compute a running average and standard deviation for each hashtag's volume distribution. Then, we identify non-overlapping *episodes* within a time-series of daily volumes for each hashtag whenever its daily volume exceeds ($\mu + 1\sigma$) of the previous 20 day periods. We label the 20 day periods preceeding an episode as the *accumulation period* of an episode. We categorize an episode as *breaking* if the hashtag volume goes on to exceed ($\mu + 2\sigma$) without falling below max(0, μ - 2σ), or else as a *non-breaking* episode otherwise. Next, we examine multiple network metrics associated with the accumulation period of each episode and proceed to build a classifier that aims to predict whether an episode will lead to a breakout volume or not. We employ a network based classification model and to discover latent patterns for the breakout phenomena, particularly we examine which factors contribute to make hashtag volumes breakout. We also build a visualization tool called Trending Hashtag Forecaster (THF). Our THF tool helps reveal the underlying network structures, patterns and properties that lead to breakout volumes. Our experiments showed that "local" network features during an accumulation period have an overall predictive accuracy of 76%, where as, when we incorporate "global" features that utilize measures extracted from all of the network up to the current accumulation period, then the overall predictive accuracy of the Trending Hashtag Forecaster jumps to 83%.

2 Problem Formulation

Given a set of tweets $T = t_1, t_2, t_3, ..., t_n$ where n is number of tweets in our corpus. These tweets comprise textual contents, user interactions and additional meta data. We explore and analyze both textual contents filtered by a given hashtag from hashtags set H. Then we denote tweet volume as number of tweets per day. We then compute daily means ($\mu(20)$) and standard deviation ($\sigma(20)$) for each hashtag by utilizing its volume distribution during its previous 20 days window. We experimentally determined the best window size by experimenting 10, 15, 20, 25 and 30 days windows. The 20 days window shows the best performance amongst the others.

If the hashtag frequency rises above ($\mu(20)$ $+1\sigma(20)$), then we label that period as an episode, and we mark its previous 20 days as the accumulation period of an episode. We start observing hashtag frequency for two possible outcomes:

- a breakout if hashtag volume rises above($\mu(20)$ $+2\sigma(20)$), without falling below max(0, $\mu(20)$ - $2\sigma(20)$), or
- non-breakout, if hashtag volume falls below max(0, $\mu(20)$ - $2\sigma(20)$), without rising above ($\mu(20)$ $+2\sigma(20)$)

In breakout scenario for an episode no further overlapping breakouts are allowed until its volume falls below max(0, $\mu(20)$ - $2\sigma(20)$). In both scenarios, as episode begins with its accumulation period and continues until the hashtag volume dies out (i.e. it falls below max(0, $\mu(20)$ - $2\sigma(20)$)). Figure 1, shows the histograms of all daily hashtag volumes in our corpus.

Next, in Section 3 we present related work. In Section 4 we describe out Tweet corpus. In Section 5, we describe our Trending Hashtags Forecaster visualization tool. In Section 6, we introduce our network based model, local and global network features to predict hashtag episode breakouts following accumulation periods. In Section 7, we present experimental results and findings. Section 8 concludes the paper and presents the future work.

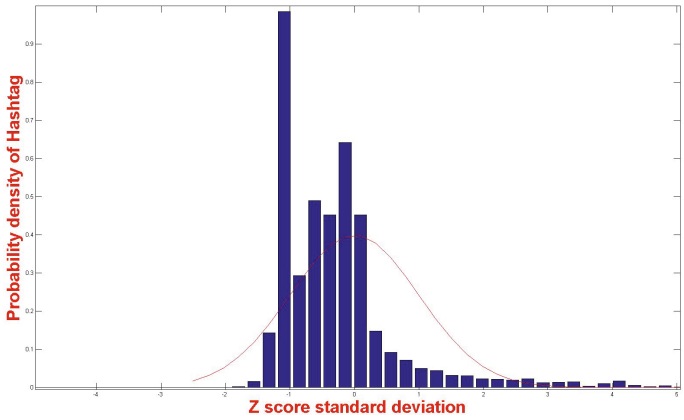


Fig. 1. Probability distribution function of all Hashtags

3 Related Work

Twitter network has more than 271 million monthly active members and 500 million tweets are generated daily [1]. The vast size and reach of Twitter enables examination of potential factors that might be correlated with breakout events and viral diffusion. We found that diffusion related studies fall into two categories. In the first category, many studies start by analyzing social networks as a graph of connected interacting nodes i.e. between users, friends or followers, and these studies investigate different factors that drive propagation and diffusion of information Arruda et al. [7] proposes that network metrics play an important role in identifying influential spreaders. They examined the role of nine centrality measures on a pair of epidemics models (i.e. disease spread on SIR model and spreading rumors on a social network). According to the authors, epidemic networks are different from social networks such that infected individuals in SIR become recovered by a probability μ while in social networks a spreader of a rumor becomes a carrier by contacts. They found centrality measures such as closeness and average neighborhood degree are strongly correlated with the outcome of spreading rumors model.

[1] https://about.twitter.com/company

The second category looked into the diffusion problem through content analysis by incorporating different natural language processing techniques. For instance, one study hypothesized that a specific group of words is more likely to be contained in viral tweets. Li et al. analyzed tweets in terms of emotional divergence aspects (or sentiment analysis) and they noted that highly interactive tweets tend to contain more negative emotions than other tweets [1], [8].

Weng et al. [5] investigated the prediction of viral hashtags by first defining a threshold for a hashtag to be viral, and then by examining metrics and patterns related to the community structure. They achieved a precision of 72% when threshold is set statically to 70. Romero et al studied the diffusion of information on Twitter and presented some sociological patterns that make some types of political hashtags spread more than others. Asur [11] presented factors that hinder and boost trends of topics on Twitter. They found content related to mainstream media sources tends to be main driver for trends. Trending topics are further spread by propagators who re-tweet central and influential individuals.

We propose a model that predicts hashtag breakouts thru adaptive dynamic thresholds, and by utilizing generic content-independent network measures that draws their information from (i) local networks corresponding to accumulation periods, as well as (2) from the global networks corresponding to the entire network history preceeding an accumulation period.Our experiments showed that local network features yield an overall predictive accuracy of 76%, and, global network features yield an overall predictive accuracy of 83%.

4 Data Source

The dataset we are using in this study is a collection of tweets from UK region. These tweets have been crawled based on a set of keywords with the aim to capture political groups, events, and trends in the UK. The dataset consists of more than 3 million tweets, 600K users, with more than 5.2 million interactions (both mentioning and retweeting) between users along with 1,334 hashtags.

5 Visualization Tool: Trending Hashtags Forecaster

In order to visualize and understand breaking hashtag phenomena, we built a visualization tool, depicted in Figure 2, that facilitate exploring temporal dynamics of hashtags and their underlying networks during accumulation period of each episode. Local and global network measures are also computed and displayed as network and node features. These network measures are utilized to train and test a predictive classifier, presented in the next section.

6 Methodology

In this study, we crawled tweets containing hashtags (case insensitive) which related to political groups in UK from June, 2013 to July, 2014. After crawling,

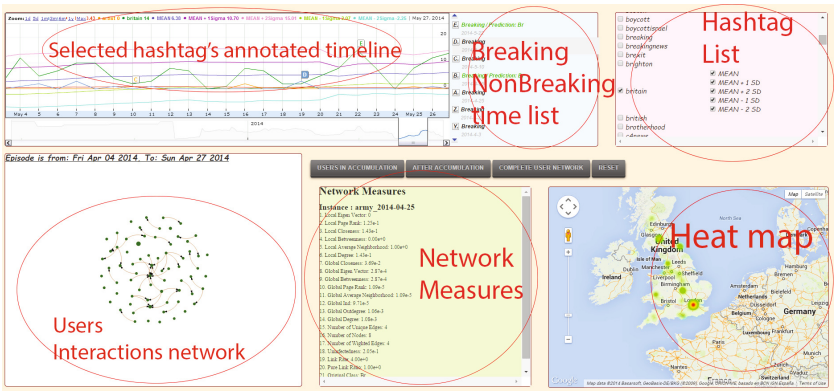


Fig. 2. THF visualization tool

we detected hashtag episodes using techniques described in Section 2. We identified the accumulation period and accumulation network of each episode, and extracted network measures corresponding to its accumulation network. Each eposide was also labeled as breaking or non-breaking based on its spread.

THF visualization tool reveals some of the discriminative patterns between breaking and non-breaking hashtags. Figure 3 shows the user interaction network for a non-breaking hashtag. User interaction network denoted by number 1 was captured during its accumulation period. Later on, this Hashtag did not breakout (i.e. did not cross its $\mu(20) + 2\sigma(20)$, but it fall back to zero volume, hence considered as a non-breaking episode. Figure 4, illustrates a breakout hashtag. Following a 20 period accumulation period, its volume exceeds $\mu(20) + 1\sigma(20)$ (denoted by network number 1), and it's volume exceeds breakout levels (by exceeding it's $\mu(20)+2\sigma(20)$) threshold (denoted by network number 2). Network 3 shows the entire reach this episode before it's demise (i.e. by falling below max(0, $\mu(20)$ - $2\sigma(20)$). An interesting observation related in the network 1 is a highly central green node, which attracts many new re-tweeters in network 2 and network 3. This observation indicates that existence of a large number of highly central nodes during the accumulation phase of an episode could be a good predictor for a following breakout. Other instances' patterns could not be cached by naked eye, yet they carry latent centrality measures correlate with our definition.

6.1 Network Based Model

In this model we investigate how users get involved in a hashtag h by mentioning, replying or retweeting. Their interactions are depicted as a directed graph G_{h_i}. We then incorporated normalized size-independent network features for directed graphs corresponding to accumulation periods of episodes. The network graph is a pair $G = (V, E)$ where V is set of vertices representing users together with a set of edges E, representing interactions between users. For instance, if a user

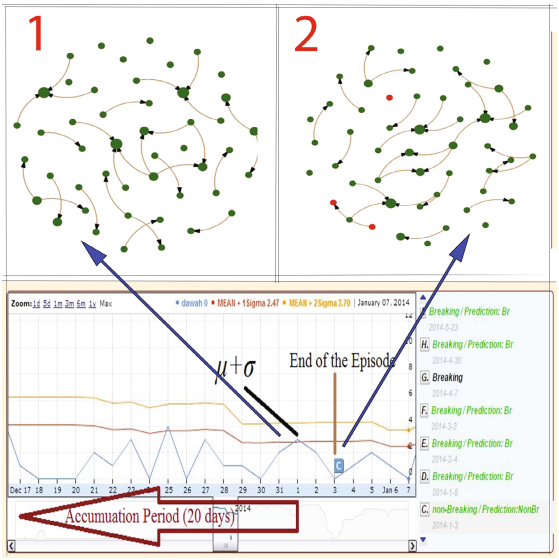


Fig. 3. Non breaking #Dawah Hashtag episode

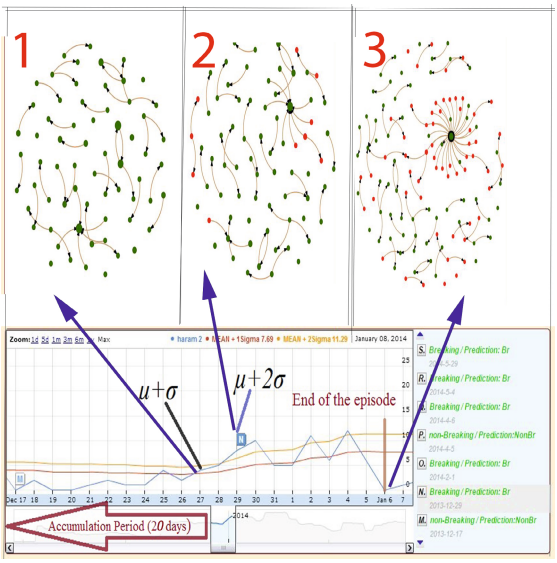


Fig. 4. Breaking #haram Hashtag episode

$u1$ mentioned, replied, or retweeted one tweet of $u2$, then a directed edge from $u1$ to $u2$ is formed.

We attempted to identify key features that contribute to the network based classification problem for breaking or non-breaking hashtags. Table 1 list all features that we used for local and global measures. Local measures are associated with user interactions during the accumulation period only, where as global measures draws their information from all interactions beginning from the start date (June 2013) until the end date of any accumulation period under consideration.

Table 1. Feature description

Feature	Description
Eigen Vector Centrality	Node's centrality depends on its neighbors centralities. If your neighbor are important you most likely are important too.
Page Rank	IVariant of Eigenvector where a node don't pass its entire centrality to its neighbors. Instead, its centrality divided into the neighbors. [3]
Closeness Centrality	A node is considered important if it is relatively close to all other nodes in the network [2].
Betweeness centrality	Measuring the importance of a node in connecting other parts of the graph [6]. This measure possesses the highest space and time complexity.
Degree centrality	It measures the number of ties a node has in undirected graph.
Indegree Centrality	It measures number of edges pointing into a node in a directed graph.
Outdegree Centrality	It is similar to the two above measure but it concerns on the number of outgoing links from a user, and it is normalized for each node.
Link Rate	Number of URLs in the tweets during the accumulation period divided by number of tweets.
Distinct Link Rate	Similar to link rate but without considering similar URLs.
Number of uninfected neighbors of early adopters	It is total number of retweets or mentioned (edges) a user has ever received globally, normalized by max-min retweets within local network in a current period being measured. [5]
Neighborhood average degree	it measures the average degree of the neighborhood of each node. [4]

7 Experiment Results and Findings

As a preprocessing step, We had 2790 for the non break out instances, while 1331 were for the break out. We sampled (without replacement) instances from both classes with oversampling for the lower represented class. We next examined the correlation between features and breaking hashtags using Principle Component Analysis (PCA). PCA is a dimensionality reduction approach that analyzes dataset to find which features give highest variance among instances and it maps the given features into lesser number of factors called components [9]. After that, in order to predict whether a given hashtag will breakout or not, we run a supervised network based learning model.

7.1 Features Correlated with Breaking Hashtags

PCA identified nine factors shown in Table 1. According to Kaiser Criterion [10], the factors to consider are the ones with eigenvalue above 1. In this study, we will focus on the first two components since they reveal interesting insights. Table 2

shows the correlation between our features and the first two components shown in Table 1. The first component is strongly correlated (negatively) with global measures, where as the second component is strongly correlated (negatively) with local measures. These two components give us a hint that global features should be grouped together and they contribute heavily (36%) to the variation in our dataset. Also, some of the local measures are also grouped together in a single factor and they somewhat contribute (21%) to the variation in our dataset.

Table 2. PCA components

Component	Eigenvalue	Variance	Cumulative Variance
1	5.79	36.16	36.17
2	3.30	20.62	56.78
3	1.669	10.43	67.21
4	1.24	7.73	74.94
5	1.01	6.31	81.24
6	0.88	5.54	86.78
7	0.663	4.14	90.92
8	0.48	3.01	93.93
9	0.42	2.65	96.57

Table 3. Correlation Between Table and Components

Feature	Component 1	Component 2	Feature	Component 1	Component 2
PageRank Local	0.14	-0.45	PageRank Global	-0.36	-0.10
Closeness Local	0.05	-0.51	Closeness Global	-0.24	0.09
Betweeness Local	0.05	-0.44	Betweeness Global	-0.35	-0.07
Avg Neighbor Degree Local	-0.11	-0.04	Avg Neighbor Degree Global	-0.3552	-0.10
Degree Cent. Local	0.11	-0.49	Degree Global	-0.3897	-0.0554
Uninfected Neighbor	0.19	0.02	In Degree Global	-0.39	-0.09
Link Rate	-0.03	0.14	Distinct Link Rate	0.0282	0.1889
Outdegree Global	-0.21	0.02	-	-	-

7.2 Network Based Model

For this model, we measured two sets of features: local and global. For local features: we have eigenvector, pagerank, closeness, betweeness, average neighborhood degree, uninfected neighbors before break out, and degree centrality. For global features we have the previous features measured globally plus in degree, out degree, and link rate. Next, we train and test a random forest classifier

with 10 fold cross-validation using three approaches: prediction using all features shown in Table 3, prediction using global features that are correlated with the first factor identified by PCA shown in Table 4, and prediction using local features that are correlated with the second factor returned by PCA shown in Table 5. We achieve the highest precision of 84%, recall of 81% and F-measure of 82% for breakout prediction with the global features. We also achieve the highest precision of 82%, recall of 85% and F-measure of 84% for non-breakout prediction with the global features. On the other hand, local features archive overall lower precision and recall of roughly 76%. These findings suggest that global measures outperform local measures in predictive accuracy.

Table 4. Break out results

Network	TP	FP	Precision	Recall	F-measure
Local	0.73	0.2	0.77	0.73	0.75
Global	**0.81**	**0.15**	**0.84**	**0.81**	**0.82**
All Features	0.8	0.16	0.83	0.8	0.81

Table 5. Non break out results

Network	TP	FP	Precision	Recall	F-measure
Local	0.79	0.27	0.75	0.79	0.77
Global	**0.85**	**0.19**	**0.82**	**0.85**	**0.84**
All Features	0.84	0.20	0.81	0.84	0.83

8 Conclusion and Future Work

In this paper, we develop a model for predicting breaking hashtags using a content independent network model comprising both local and global network features drawn from an indicative accumulation period of hashtag volumes. For the network model, we measured and experimented with the predictive accuracies of global and local features. We also examined their importance and rankings using PCA. Global features drawn for the accumulation period network showed higher predictive accuracy compared to the local features. Network based model with global centralities for the accumulation period network can be used as a general framework to predict breaking hashtags with an overall accuracy of 82%.As future work, we propose to study the utility of content based features such as sentiment analysis, and different types of sources.

Acknowledgments. This research was supported by US DoD ONR grant N00014-14-1-0477 and USAF AFOSR grant FA9550-15-1-0004.

References

1. Li, C., Sun, A., Datta, A.: Twevent: segment-based event detection from tweets. In: Proceedings of the 21st ACM International Conference on Information and Knowledge Management, pp. 155–164. ACM (2012)
2. Newman, M.E.J. A measure of betweenness centrality based on random walks. Social networks 27.1, 39–54 (2005)
3. Brin, S., Page, L.: The anatomy of a large-scale hypertextual Web search engine. Computer Networks and ISDN Systems **30**, 107–117 (1998). doi:10.1.16/S0169-7552(98)00110-X
4. Barrat, A., Barthelemy, M., Pastor-Satorras, R., Vespignani, A.: The architecture of complex weighted networks. In: Proceedings of the National Academy of Sciences of the United States of America 101.11, PP. 3747–3752 (2004)
5. Weng, L., Menczer, F., Ahn, Y.-Y.: Virality prediction and community structure in social networks. Scientific reports 3 (2013)
6. Freeman, L.C.: A set of measures of centrality based on betweenness.Sociometry, 35–41 (1977)
7. Arruda, G., Barbieri, A., Rodrigues, F., Moreno, Y., Costa, L.: The role of centrality for the identification of influential spreaders in complex networks. Physical Review E **90**, 032812 (2014)
8. Cheng, J., Adamic, L., Dow, P.A., Kleinberg, J.M., Leskovec, J.: Can cascades be predicted?. In: Proceedings of the 23rd International Conference on World Wide Web, pp. 925–936. International World Wide Web Conferences Steering Committee (2014)
9. Pearson, K.: LIII. On lines and planes of closest fit to systems of points in space. The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science **2**(11), 559–572 (1901)
10. Bandalos, D.L., Boehm-Kaufman, M.R.: Four common misconceptions in exploratory factor analysis. Statistical and methodological myths and urban legends: Doctrine, verity and fable in the organizational and social sciences, 61–87 (2009)
11. Asur, S., et al.: Trends in social media: persistence and decay. ICWSM (2011)