Πανεπιστημιο Θεσσαλιας

Παρακολούθηση κοινοτήτων σε χρονικά μεταβαλλόμενα κοινωνικά δίκτυα

Συγγραφέας:
Μιχαήλ Αποστολου

Επιβλέποντες:
Δημητριος Κατσαρος
Αναπληρωτής Καθηγητής

University of Thessaly

Community Tracking in Temporal Social Networks

Author:
Michail Apostolou

Supervisor:
Dimitrios Katsaros
Assistant Professor

Περιγραφή

Με την συνεχόμενη εξάπλωση των Κοινωνικών Δικτύων στην καθημερινή ζωή μας,έχει αυξηθεί και η επιστημοκή ερεύνα για την ανάλυση της εξέλιξης τους. Χρήστες δημιουργούν πολλών ειδών διασυνδέσεις με όμοιους τους που μοιράζονται κοινά γούστα και ενδιαφέροντα. Εμείς επικεντρωνόμαστε στις χρονικά δυναμικές μεταβολές που συμβαίνουν στα Κοινωνικά Δίκτυα,προσπαθώντας να εξάγουμε χρήσιμη πληροφορία βασιζόμενοι στη δομή των δικτύων και του περιεξομένου που παράγουν τα μέλη τους.
Σε αυτή τη διπλωματική χρησιμοποιήσαμε ένα κατασκευασθέν σύνολο δεδομένων από το MathOverflow dump[1].Επεκτείναμε τον προτεινόμενο αλγόριθμο από τους P.Brodka et al.[2],ο οποίος εκμεταλευόταν μόνο τη δομή των Κοινωνικών Δικτύων, για να αναγνωρίσει γεγονότα που συμβαίνανε στις κοινότητες σε κάθε χρονική στιγμή. Η επέκτασή μας εκμεταλεύεται και το κείεμνο που παράγεται από τις ερωτήσεις, απαντήσεις και σχόλια των χρηστών.
Τα γεγονότα κατηγοριοποιούνται, για δική μας διευκόλυνση,σε 3 κλάσεις. Τα ένα προς ένα γεγονότα,τα οποία είναι η συνέχεια,η αύξηση και η μείωση των κοινοτήτων σε δύο συνεξόμενες ξρονικές στιγμές.Η μια προς πολλές και πολλές προς μια,που είναι η διάσπαση μιας κοινότητας και η συγχώνευση πολλών σε μια,αντίστοιχα.Και τέλος, τα μονομερή γεγονότα που είναι η εξάληφη μιας υπάρχουσας κοινότητας και η γέννηση μιας καινούργιας.

Abstract

With the ever expanding of Social Networks in our daily lives, research interest for analyzing their evolution has also been increased. Users form all kinds of connection with other peers sharing same taste and interests.We focus on the temporal and dynamic changes on these Social Networks to extract potential useful information based upon the structure of the networks and the content being produced by their members.

On this diploma thesis we used a custom dataset from the MathOverflow dump[1]. We extended an algorithm proposed by P.Brodka et al.[2] which was exploiting only the structure of the Social Network to classify events happening to the communities per time instance.Our extension took advantage of the text generated by the questions,answers and comments of the users.

The events can be categorized ,for our convenience,into 3 classes.The one-to-one match which are the events of the continuation,growth or shrinking of a community in two successive timeframes.The one-to-many and many-to-one,which are the split of one community to many, and the inverse the merging of many communities to a single.Finally the unilateral events which are the dissolution and birth of a community.

Acknowledgements

First of all I would like to thank my supervisors Dimitrios Vogiantzis and George Paliouras for their guidance, insight and providing useful feedback throughout the course of the project. Also I would like to thank the rest of the research team of SKEL lab of the National Center for Scientific Research ”Demokritos” for the opportunity I had to work besides them.Finally I would like to thank all of my friends for their constructive criticism and wonderful ideas.

Contents

1 Introduction **1**
- 1.1 Motivation . . . 1
  - 1.1.1 Thesis Structure . . . 1
- 1.2 Related Work . . . 2

2 Background **4**
- 2.1 Graph Theory . . . 4
- 2.2 Social Network Analysis . . . 7

3 Dataset **8**
- 3.1 Inapplicable Datasets . . . 8
- 3.2 Chosen Data . . . 9
  - 3.2.1 MathOverflow description . . . 10
  - 3.2.2 Data cleanup . . . 10
  - 3.2.3 Creating Row Data . . . 10

4 The Algorithm **12**
- 4.1 Tools and libraries . . . 12
- 4.2 Custom Libraries and Scripts . . . 14
- 4.3 Representation abstraction . . . 14
- 4.4 Metrics . . . 15
  - 4.4.1 Extended Inclusion . . . 15
  - 4.4.2 Jensen-Shannon Divergence . . . 15
- 4.5 Event Recognition . . . 16
  - 4.5.1 General Example . . . 18
- 4.6 Pseudocode . . . 19
- 4.7 Complexity . . . 21

5 Experiments **22**
- 5.1 Ground Truth . . . 22
- 5.2 Number of LDA topics . . . 22
- 5.3 Thresholds α and β . . . 25
- 5.4 Overlapping data . . . 26
- 5.5 Centrality Comparison . . . 28
- 5.6 Weight Comparison . . . 30

6 Final Words **31**
- 6.1 Conclusion . . . 31
- 6.2 Future Work . . . 31

List of Tables

1 Confusion Matrix of 50 LDA topics 23
2 Evaluation of 50 LDA topics 23
3 Confusion Matrix of 10 LDA topics 24
4 Evaluation of 10 LDA topics 24
5 Confusion Matrix of 0 LDA topics 24
6 Evaluation of 0 LDA topics 24
7 Accuracy comparison between number of lda topics 24
8 Confusion matrix of overlap = 0 with degree centrality 27
9 Confusion Matrix of overlap = 0 with PageRank 28
10 Evaluation of Degree Centrality 29
11 Evaluation of PageRank Centrality 29
12 Evaluation of Closeness 29
13 Accuracy comparison between centralities 29

1 Introduction

Nowadays internet is an essential part of our everyday life.Used to get us informed for various topics ranging from the standard news (politics, sports, weather etc.) to more personalized topics (entertainment, skills improval, self hygiene etc.). However the internet holds a vast quantity of information which an individual is unable to properly filter out, in order to get the truthfully useful information. Moreover,over the years social media and social networks in general affect our intake of information, by either influencing us in a way [3] or us being overwhelmed by the multiple input streams of information. This thesis aims to alleviate this problem by exploiting and providing information of peers similar to each other.

1.1 Motivation

The ongoing research on the social network analysis, focuses on more on the structure of the network,rather than the content that is produced by the peers. Thus missing out another stream of information input. On the other hand, research focusing in the topic either exploits the information produced by the user for his own benefit [4] or trying to find the most influential peers of the network[5][6].

Community tracking aims to help various other fields of scientific study. Sociologists [7] for one,have interest in understanding communities on networking level and correlating these results with their real life counterparts. In the field of criminology [8] , a study of the role that social networks play in facilitating and contributing to the creation,growth and continuation of cliques of delinquent individuals,can be beneficial as the results can help deal with this behavior.

Community tracking can also contribute to the imporevement of the smart advertising [3] on a social networking scale as well as the information diffusion [9] . It may also contribute on the field of community evolution prediction [10].

Also we try to enhance the performance of the G.E.D. classifier [2] , and tackle some of its incompatibilities such as 2 events happening at the same time involving the same communities.

1.1.1 Thesis Structure

This thesis is divided into 5 Chapters.

- ***Chapter 2***, presents the theoretical background needed by the reader to understand the techniques used in this thesis.

- ***Chapter 3***, presents the requirements and features of our data. Also presents the datasets that first were tried and later got rejected,as well as, which dataset was ultimately used.

- ***Chapter 4***, presents the tools that were used for the algorithm, as well as, the pseudocode of the algorithm and its complexity.

- ***Chapter 5***, first presents the creation of the ground truth for the data and later experiments done altering various parameters of the algorithm. Also we discuss the results of each experiment.

- Finally, ***Chapter 6*** , provides an overall conclusion of the experiments and thoughts of future work.

1.2 Related Work

In recent years, there has been growing interest in the analysis of community structures. In this section, we present the work already done on topics of community tracking and community evolution based on structural criteria.The field of community tracking is the one that tries to find dense structures in a network whilst community evolution is related to the identification of events such as grow, split,merge that may happen to a community at consecutive time frames. We will present below some of the most important methods for community evolution tracking, which are based on structural features of the communities, such as node identity and importance of interactions between nodes.
Greene et al. (2011) proposed a scalable community tracking strategy which tries to match a community C_i of time-frame F_t to the communities of the next time-frame F_{t+1} with an evolution event [11]. In order to achieve this matching, they used the Jaccard coefficient between the two communities and checked if the similarity exceeded a certain threshold.Then, the pair of communities were matched and $C_{i\alpha}$ (incoming step community) was added to the timeline for the dynamic community D_j.If there was no match for an incoming community, a new dynamic community was created containing $C_{i\alpha}$.
A similar method named Group Evolution Discovery (GED) is based on inclusion, which metric capitalizes the group quantity (the common nodes between groups) and the group quality (the importance of the nodes comprising the network).The group quality can be represented by any major graph centrality (degree, betweenness, page rank etc.).The next step is the identification of the type of the events (continuation, growth, shrinking, dissolution, merging, splitting, formation) that occurred between time frames based on various thresholds (Piotr Brodka et al. 2012)[2].
A slightly different approach on community tracking is followed in Takaffoli's et al. paper [12] (2011) in which they used both structural and topic-based measures to improve the accuracy on their method. Another approach on community and topic co-evolution is proposed by Bi et al. [13] (2014). The authors

created a model which discovers well-connected, topical meaningful communities and the co-evolution of topics over time in dynamic social networks by automatically capturing the dynamic features of communities. Specifically, they observed how the community structure, which is based on a graph partitioning algorithm, changes over time with the evolution of topics, which is basically the change of interests of each user. On a different domain,there is reacher for the problems of community detection. A research article by Sekara et al. (2015) [14], focuses on the impact of the different time scales, spanning from several minutes to days to years. On this article they monitored the evolution of social gatherings and how can they be identified, as well as discriminated the social core of said gatherings. They also investigated the predictability of human social activity. Another work done by Delvenne et al. (2010) [15] , measures which clusters are relevant at different time scales. To achieve this they used Markov chains represented by graphs with edges weighted by probabilities. This provides an interpretation of community detection: natural communities correspond to persistent dynamical basins , that is, to sets of states from which escape is unlikely within the given time scale. Additionally, Mucha et al. [16] (2010) study the detection of communities in multiscale and multiplex networks. On the bursty evolution of Blogspace paper [17] the evolution of directed graphs is done by tracking time-dense communities. Based on their data,which are mainly coming from blogs, they create time graphs by an automatic analysis of their internal time stamps and then study the evolution of community structure in that time graph.

2 Background

In this chapter we will discuss the bare essentials of graph theory and social network analysis and what should the reader know.

2.1 Graph Theory

In mathematics graph theory is the study of graphs, which are mathematical structures used to model pairwise relations between objects. A graph in this context is made up of vertices, nodes, or points which are connected by edges, arcs, or lines. A graph may be undirected, meaning that there is no distinction between the two vertices associated with each edge, or its edges may be directed from one vertex to another.

Graph

A graph is an ordered pair $G = (V, E)$ comprising a set V of *vertices* or *nodes* together with a set E of *edges*, which are 2-element subsets of V.

Degree

Degree, is the number of incoming and outgoing edges of a node.It can be further categorized depending whether we have a undirected or directed graph. In the case of undirected the degree is the sum of the edges,while on the directed graphs we have two separate degrees, the in-degree (the node is the finishing end) and the out-degree (the node is the starting end).

Centrality

In graph theory and network analysis, indicators of centrality identify the most important vertices within a graph. Applications include identifying the most influential person(s) in a social network, key infrastructure nodes in the Internet or urban networks, and super-spreaders of disease. Centrality concepts were first developed in social network analysis, and many of the terms used to measure centrality reflect their sociological origin. They should not be confused with node influence metrics, which seek to quantify the influence of every node in the network.

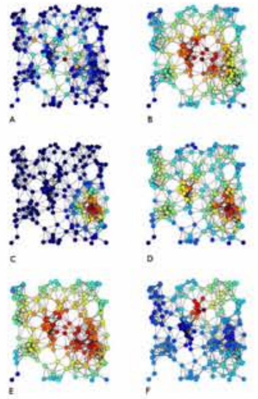

Figure 1: Examples of A) Betweenness centrality, B) Closeness centrality, C) Eigenvector centrality, D) Degree centrality, E) Harmonic Centrality and F) Katz centrality of the same graph.

Closeness Centrality

Closeness centrality of a node u is the reciprocal of the sum of the shortest path distances from u to all $n-1$ other nodes. Since the sum of distances depends on the number of nodes in the graph, closeness is normalized by the sum of minimum possible distances $n-1$.

$$C(u) = \frac{n-1}{\sum_{u=1}^{n-1} d(v,u)}$$

where $d(v,u)$ is shortest-path distance between v and u, and n is the number of nodes in the graph.

Special Graphs

There are several types of graphs.It is imperative that the reader is capable to identify the type of graphs are formed during the analysis of the network.

Bipartite Graph

Bipartite graph (or bigraph) is a graph whose vertices can be divided into two disjoint sets U and V(that is,U and V are each independent sets) such that every edge connects a vertex in U to one in V. Vertex sets U and V are usually called the parts of the graph.An example of a bipartite graph is shown in Figure 2

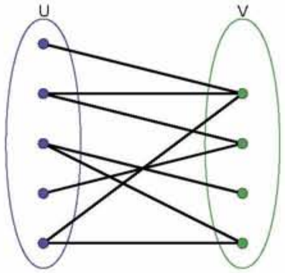

Figure 2: Example of a bipartite graph without cycles

Regular Graph

A regular graph is one in which all nodes have the same degree. A regular graph where all nodes have degree 2 is called a 2-regular graph. More generally, a graph where all nodes have degree k is called a k-regular graph.Regular graphs can be connected or disconnected. Complete graphs are examples of regular graphs, where all n nodes have degree n - 1 (i.e.,k = n - 1). Cycles are also regular graphs, where k = 2. Another example for k = 3 is shown in Figure 3

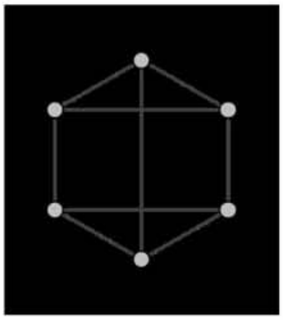

Figure 3: Example of a regular graph with k = 3.

Bridge

Consider a graph with several connected components. Edges in this graph whose removal will increase the number of connected components are called bridges. As the name suggests, these edges act as bridges between connected components. The removal of these edges results in the disconnection of formerly connected components and hence an increase in the number of components. An example graph and all its bridges are depicted in Figure 4.

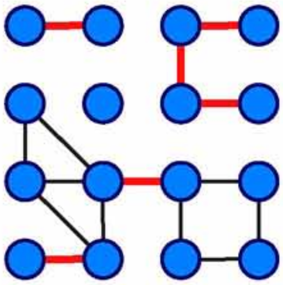

Figure 4: A graph with 16 vertices and 6 bridges (highlighted in red)

2.2 Social Network Analysis

Social network analysis (SNA) is the process of investigating social structures through the use of network and graph theories.[18] It characterizes networked structures in terms of nodes (individual actors, people, or things within the network) and the ties, edges, or links (relationships or interactions) that connect them. Examples of social structures commonly visualized through social network analysis include social media networks, friendship and acquaintance networks, collaboration graphs, kinship, disease transmission, and sexual relationships.[19]

3 Dataset

The dataset should be consisted of 3 features:

1. The data should be temporal and highly dynamic
2. The data should have ground truth communities with labels
3. The data should have text in order to create topics for the topic similarity measure.

I would like to note that in this thesis we don't try to solve the problem of community detection,since we expect the data to provide this information for us.

3.1 Inapplicable Datasets

On the task to find an appropriate dataset, we've came across these datasets.We will list their potential for other projects, and we will clarify why they didn't fit our needs.

Yelp Academic challenge

For those who don't know, Yelp is an online service helping people finding local services (i.e.,dentists,hair stylists etc.) from reviews written by other users/customers.As a challenge Yelp provides a dataset for academic research [20].

Pros:

- The dataset was vast in size,with 2.7M reviews, 687K users with 4.2M social edges and 86K businesses.
- Rich in textual content from the reviews.
- The categorization of the businesses helps create user defined labels.

Cons:

- There was no explicit time stamp of the social edges' formation,thus making it impossible to find the chronological order. Subsequently we couldn't divide the dataset into timeframes.
- There were no ground truth labels for communities.

Stanford Network Analysis Project

Stanford Network Analysis Project or SNAP is a project by Stanford's Associate Professor Jure Leskovec.The SNAP library is being actively developed since 2004 and is growing as a result of research pursuits in analysis of large social and information networks [21]. The datasets range from simple web graphs to complex social networks.
Unfortunately none of these dataset fulfill the above requirements. For example there were "community" labels but no user friendship information (reddit dataset). Others missing context (Web graphs).

Last FM

Another dataset we investigated was a dataset containing social networking,tagging,and music artist listening information from a set of 2K users from Last.fm online music system.The dataset is released in the framework of the 2nd International Workshop on Information Heterogeneity and Fusion in Recommender Systems (HetRec 2011)[22].

Pros:

- The dataset was large enough for our purposes.
- The provided tags could help form "music" communities.
- Timestamps on the formation of the edges between users.

Cons:

- No text content for the topic similarity

Synthetic Data

In early work,where we were testing the algorithm with only the structural measurement,we used a synthetic dataset provided from a page[23],which contained supplementary material for the paper[11].
This dataset was inadequate since it provided only a pair of events per dataset(birth-death,grow-shrink, etc.) of a community.

3.2 Chosen Data

After days of browsing the internet trying to find a dataset to fulfill our requirements,me and my supervisor came to a consensus that we will take an existing dataset and customize it to our needs.Eventually, we decided that we would use the data provided by math overflow.

3.2.1 MathOverflow description

Mathoverflow is a forum-like site where individuals post questions concerning maths.Other users answer these questions and comment said questions and answers.Every question has a single or multiple tags describing the context of the question.

3.2.2 Data cleanup

The data provided from StackExchange[1] , came split into multiple .xml files.

1. **badges.xml:** a user receives badges for being especially helpful,something like an achievement.
2. **comments.xml:** contains the comments on both questions and answers.
3. **posthistory.xml:** contains changes that may happened to a post (i.e. body edit,post migration etc.).
4. **posts.xml:** contains all the information about a post.
5. **users.xml:** contains the personal information of a user.
6. **votes.xml:** a vote type characterizes the nature of a post (i.e. Favorite,Spam,Closed etc.)

After studying the files, we came to the conclusion that all the useful information is contained in just two files, comments and posts.
The process of the data cleanup we checked the consistency of the data. There were some faults that either were fixed or ignored. For example there were answer with timestamp earlier than the question.This was making a conflict in the chronological order.Another example was that some question had their tags or body empty.Since both features were important we had to discard these types of posts.

3.2.3 Creating Row Data

Following the cleanup process, was the process to create to row data for our algorithm input.We went with creating two json files containing all the useful information we needed.

Posts.json

This json file held in an array all the post IDs that would be used.Each post ID held 4 features: Tags,Text,User ID,Post Type.
Tags: Was an array of tags to which the post belonged.We have to note that the answers and comments didn't have tags, so we had to associate them with their parent questions.
Text: the body of the post/comment.
User ID: the ID of the user who posted.
Post_Type: to identify if it was question,answer or comment. It may be used for future work for multirelational graphs.

Edges.json

Since posts and comments where different files,our first objective was to merge them and sort the transaction between two users in an ascending chronological order.Hence, when we split the Edges array into timeframes,it would make chronological sense. The edges had six features.
From: The user ID who was the starting end of the transaction.
To: The user ID who was the finishing end of the transaction.
Post ID: The ID of the post.
Parent ID: The ID of the initial question.
Creation Time: A timestamp in unix time marking the time of the edge creation.
Post Type: Signifies if the edge is a comment or an answer.

4 The Algorithm

The basic idea behind the algorithm is that we compare every community in timeframe TF_t to every community in timeframe TF_{t+1} in order to get three measurement results.Two of them originate from the Inclusion measurement which measures the structural similarity between the networks of the community.It is worth noting that the Inclusion metric is not symmetric,thus the two results,the direct inclusion and the inverse inclusion.The third result is the topic similarity of these two communities.
Afterwards we take these numbers and use them to classify what kind of event has occurred between the two or more communities.Eventually we store the events happened in every timeframe pair and produce a confusion-like matrix to test the algorithms performance.

4.1 Tools and libraries

Before continuing in full detail how the algorithm runs,I would to address the external tools and libraries I used to deliver this project.I will present briefly the objective of the external libraries also the custom libraries and script files that i used.
The whole project was written in python v.2.7.

External Libraries

json.py

As I clarified above, the input of the algorithm was contained in .json files. The library json.py was used during the process of compile the input data from the .xml files.Also it was used for read reading the .json files.

Networkx.py

Networkx is the most valuable library in this project.Is a light and high speed representation of network structures.It was used in almost all of the custom auxiliary files.

xml.etree.py

During the data cleanup,this library was used to access the .xml files.

gensim.py

Gensim was used in the process of creating the corpus for the topics,which were fed to the LDA classifier implemented by the same library.

time.py

The standard time library.Two purposes, the first convert the timestamps from ISO 8601 to epoch for sorting chronologically, and second to time the performance of the algorithm.

nltk.py

Natural Language Toolkit or nltk , was used to help me develop my own tokenizer for the text which was used for the topic extraction as mentioned above.

stop_words.py

A predefined list of stop words in english language.

numpy.py

NumPy is the fundamental package for scientific computing with Python.It was used to develop the Jensenb•“Shannon divergence, which is used to quantify the similarity of the topic probability distributions.Later will examine in detail the JSD method.

Beautifulsoup.py

During the data cleanup process, the body of some posts appeared with html tags,so it was difficult to extract meaningful content.With the use of beautifulsoup,we were able to bypass these html tags and acquire the actual post body.

langid.py

The topic similarity works properly with the assumption that all the context is written in English.Some post were written in other languages such russian,german,ancient greek.With the use of langid we were able to identify the language thus we either accepted it or rejected it.

4.2 Custom Libraries and Scripts

config.py

A configuration file,holding different parameters about the algorithm. The thresholds α and β , the percentage of the window overlap for the row data, the size of the window in rows.

preprocessing.py

A library containing functions for processing the row data to convert them to algorithm input.

event.py

A library containing all the rules for the recognition of the occurring events.

inclusion.py

A library containing functions and methods for calculating the inclusion between the communities.Also there are options for choosing what centrality the user wishes to use.

hypergraph.py

A auxiliary library which holds the class hypergraph.I implemented this class to add another level of abstraction in the algorithm.With this class we're in position of considering the timeframe pair as a bipartite graph whose nodes are the communities.Thus, a simple data structure withholds lots of information and it's lighter in a computational manner to classify the events.

MyTokenizer.py

A library for extracting the text for the posts and create a corpus to feed the LDA topic model.

4.3 Representation abstraction

As we mentioned above, we add another level of abstraction to our algorithm.Instead of having community graphs with users as nodes,we scale up to have hyper graphs with the communities as nodes.In figure 5 we can see that communities are directed graphs,whose starting node is the answer or comment and the finishing node is the question or answer.In this example users "142" and "55" answer or comment "23",same as "23" and "55" to "71".

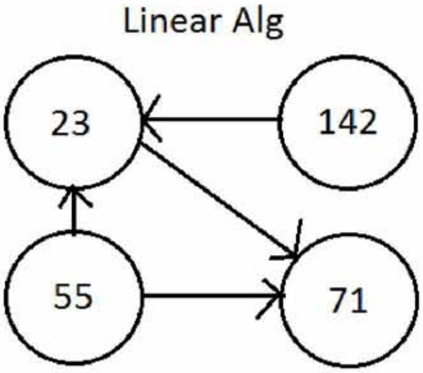

Figure 5: Example community structure.

4.4 Metrics

4.4.1 Extended Inclusion

The extended inclusion in contrast to the regular inclusion metric, has a weighted factor of the topic probability distribution similarity calculated by the Jensen-Shannon Divergence. The first half of the equation 1 can be further analyzed to a product of two components. The first one $\frac{|G_1 \cap G_2|}{|G_1|}$ describes the so called "group quantity",whilst the second component $\frac{\sum_{x \in (G_1 \cap G_2)} C_{G1}(x)}{\sum_{x \in (G_1)} C_{G1}(x)}$ the "group quality", meaning how essential are the common nodes to the graph.It is very essential for the event recognition to denote that this metric is not symmetric $I(G_1, G_2) \neq I(G_2, G_1)$, and that is the reason we used both the direct inclusion and the inverse. Furthermore, the notation C_{Gi} , means that user can you whichever centrality the user finds more suitable. In during the experiments , we used both degree centrality and PageRank centrality.

$$I(G_1, G_2) = \frac{|G_1 \cap G_2|}{|G_1|} * \frac{\sum_{x \in (G_1 \cap G_2)} C_{G1}(x)}{\sum_{x \in (G_1)} C_{G1}(x)} * w_1 + (1 - JSD(p, q)) * w_2 \quad (1)$$

4.4.2 Jensen-Shannon Divergence

The Jensen-Shannon divergence is a popular method of measuring the similarity between two probability distributions.It is based on the Kullback-Leibler divergence, with some notable (and useful) differences, including that it is symmetric and it is always a finite value. Also it is bounded by 1 for two probability distributions, given that one uses the base 2 logarithm in the equation 2.The lower bound 0 means that the distribution are identical,while the upper bound 1 means they are divergent.That is the reason at the second sum factor of eq. 1 is $1 - JSD(p, q)$. The symmetry comes of the use of the auxiliary distribution M , which is the mean of the distributions p and q.

$$JSD(P,Q) = \frac{1}{2}\sum_i P(i)\log\frac{P(i)}{M(i)} + \frac{1}{2}\sum_i Q(i)\log\frac{Q(i)}{M(i)} \tag{2}$$

$$M = \frac{1}{2}(P+Q)$$

4.5 Event Recognition

The event recognition is based on two thresholds α, β, corresponding to direct and inverse inclusion respectively, and in the relative size between the two groups. Let *inc* be the direct inclusion, and *inv* the inversed,calculated by the equation 1, explained in section 4.4.1.

Matching events

Matching events, are called the events of one-to-one relation,meaning continue, grow, shrink and mismatch of two communities between two successive timeframes. The three first are self-explanatory, while the forth means that there was only one relation (independent to which one) between two communities, but those communities share different topic/tag feature. Below, we present the condition which must be true to recognize an event.

- **Continue:** $inc \geq \alpha$ && $inv \geq \beta$ && $|G_1| = |G_2|$
- **Grow:** $inc \geq \alpha$ && $inv \geq \beta$ && $|G_1| < |G_2|$
- **Shrink:** $inc \geq \alpha$ && $inv \geq \beta$ && $|G_1| > |G_2|$

In figure 6,the nodes represent the communities,while the edges are one of the above events.

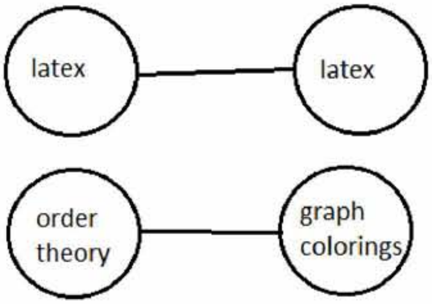

Figure 6: Example of matching and mismatching communities.

Many to one

There are cases where more than one pair ,with one common end, fulfill the rules described above. In this scenario we have a **split** (fig.7), a **merge** (fig.8) or both events. Later we will present an example of such case.

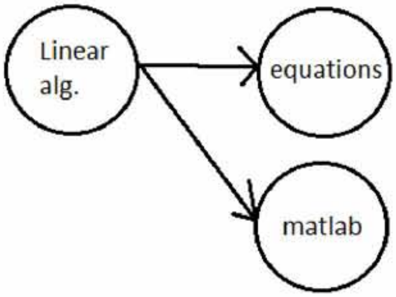

Figure 7: Example of a splitting community.

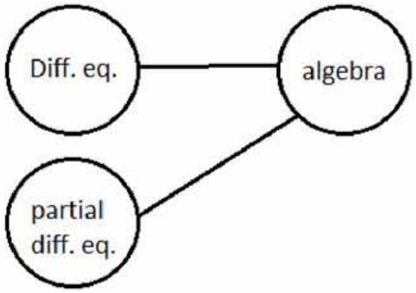

Figure 8: Example of merging communities.

Unilateral events

Unilateral events are the events of dissolution or formation of a community. In order to recognize a dissolution or a formation, Brodka et al. set the threshold values to 10% of the inclusion.

- **Dissolution:** If a community C_i in TF_t has $inc < 0.1$ && $inv < 0.1$ with all the communities in TF_{t+1}
- **Formation:** If a community C_i in TF_{t+1} has $inc < 0.1$ && $inv < 0.1$ with all the communities in TF_t

No event

We can see through all the rules, that there is a region between $[0.1, \alpha)\&\&[0.1, \beta)$ where we have no rules.In this case we claim that there is too large similarity for dissolution but too little for a match, thus we name it **"No event"** event.

4.5.1 General Example

In fig.9 below we can detect all the events described above. Note that the algorithm fails to recognize two matching events, instead recognizes a two splits and two merges between the two communities. This phenomenon depends on how strict are the thresholds. The higher the thresholds the more you expect the communities to be similar. Also let us not forget,that the communities form from the tags of questions. Meaning that if there are multiple questions with the same tags, the communities will close to identical (i.e. question with tags "type-theory,lambda-calculus".

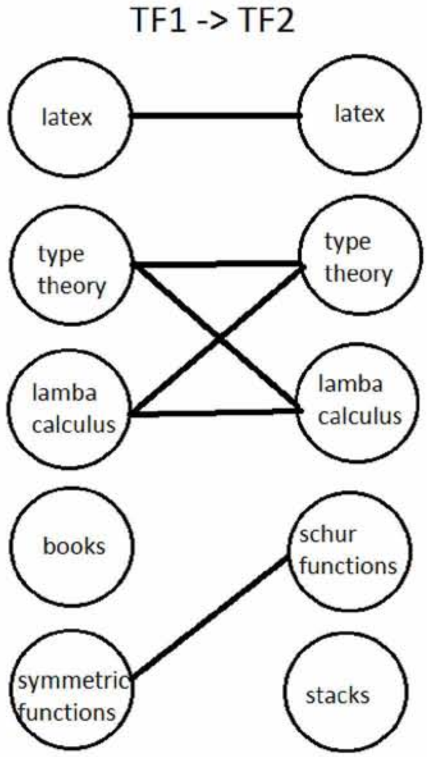

Figure 9: Example of event recognition.

4.6 Pseudocode

In this subsection we will investigate in some detail the steps of the algorithm. Here in Algorithm1 we see the conversion for row data files to the actual input for the main function.The procedure *GetGraphs* requires as input an edge file,a posts file,a float number for the overlap and an integer number for the size of the timeframe.As output this procedure exports an array of size N (where N is the total number of timeframes) of arrays containing graph items (which are our communities per timeframe).

Procedure 1 Convert row data to input data

```
Input: Edges row data,posts row data,overlap,window_length
Output: Array of arrays of community graph items
procedure GetGraphs(edges_file, posts_file, overlap, window_length)
    data ← getEdges(edges_file)
    intervals ← split_to_intervals(data,overlap,window_length)
    posts_dict ← create_post_dictionary(posts_file)
    graphs_array ← create_communities(intervals,posts_dict)
```

The variable *posts_dict* is a hashmap containing all the information we want for the posts.In the procedure create_communities(*intervals*,*posts_dict*) we create the communities,which are networks whose nodes are the users and their edge come from their transactions. Each community node has three features:

1. **Cid:** It is the community id in the form $TF_{t}\_c_{i}$ meaning that in Timeframe t we ID the community i.
2. **Topic:** To characterize the topic of the community,we assign a tag coming from a question. (i.e."Linear-Alg")
3. **Text:** The feature text contains the aggregated text from all questions, answers, comments coming from all threads mark with the above tag in the time instance t.

Below we can examine the pseudocode for the main function,where we compare the communities in each timeframe pair, and classify the events occurring. In every step of the loop in line 3 we have to initialize the inclusion in every community pair combination as shown by the function in line 4. During that procedure we aggregate the texts of each community, so as in line 5 we can train the LDA classifier. Later with a nested loop we compare all the communities, in lines 13 and 14 we use the metric to calculate the topic and structural similarity,respectively as described in sections 4.4.2 and 4.4.1. Then we classify the event by the rules of section 4.5. Finally in line 17 , we check the in-degree and out-degree of the nodes to determine if a split or a merge occurred.

Procedure 2 Main function

Input: Array of arrays of graph items
Output: Array of hypergraph items

```
 1: procedure MAIN(graphs_array)
 2:     hypergraphs ← ∅
 3:     for each timeframe_pair in graphs_array do
 4:         texts ← initialize_inclusions()
 5:         lda_model ← train_ldaModel(texts)
 6:         hypergraph ← new Hypergraph()
 7:         for each past_community in timeframe_pair[0] do
 8:             hypergraph.append(past_community)        ▷ Done only in first iteration
 9:             p ← topic probability distribution
10:             for each future_community in timeframe_pair[1] do
11:                 hypergraph.append(future_community)
12:                 q ← topic probability distribution
13:                 topic_similarity ← 1 - JSD(p,q)
14:                 direct_inclusion ← $w_1 * calc\_inclusion() + w_2 * topic\_similarity$

15:                 inverse_inclusion ← $w_1 * calc\_inclusion() + w_2 * topic\_similarity$
16:                 classify_events()
17:         hypergraph.classify()
18:         hypergraphs.append(hypergraph)
```

4.7 Complexity

Assuming we have N communities on average per time window and M time windows. In the main function , pseudocode 2, we see two nested loops in lines 7 and 10. Each loop iterates through all the communities of the respective time window. Since we compare all the communities in both time windows , this means we have $O(N^2)$ time complexity per time window pair.
In line 4 where we initialize the inclusion pairs , direct and inverse , it is the same condition as above, thus $O(N^2)$ time complexity. In total the algorithm has $O(N^2M)$ time complexity and $O(MN)$ space complexity.

5 Experiments

As we can see, the algorithm depends on a number of different parameters.This gave us motivation to run a sequence of experiments tinkering these parameter in order to observe the behavior of algorithms.These parameter are :

- The thresholds for the event recognition, α and β.
- The percentage of overlap between the successive timeframes.
- The centrality that is used in the "group quality" component of the inclusion metric.
- The number of LDA topics that are used to train the model.
- The weights on the components of structural and topic similarity.

For all the experiments, the size of the dataset and window length remained constant at 15000 and 1000 rows respectively. The graphs ,which will presented in later subsections, are comprised of an x-axis naming the type of event recognized by the algorithm. More specifically the "matched, "dissolution", "form" and "other" events. The class "other" consists of the events merge,split and in some case, we note when,the "no event" event.The y-axis contains the number of the aforementioned events recognized.

5.1 Ground Truth

Due to the fact that there wasn't given any ground truth labeling, we were motivated to create one so as to evaluate the performance of the algorithm. After the process of splitting the row data into time intervals, we compared the topic feature of the communities between the two time instances.

- **Match:** If a topic exists in both TF_t and TF_{t+1} timeframes, it means there is a match. Note that we can't discriminate if that community grows, shrinks or remains the same in size.
- **Dissolve:** If a topic exists in TF_t and not in TF_{t+1}.
- **Form:** If a topic exists in TF_{t+1} and not in TF_t.

5.2 Number of LDA topics

In this subsection, we will compare the effectiveness of the number of LDA topics parameter. We compared an algorithm run with no topics, only structural similarity, to a run with 10 LDA topics and another with 50 LDA topics. The rest of the parameters we set to:
$\alpha = 0.75$ $\beta = 0.75$ overlap $= 0.5$ Centrality $=$ Degree $w_1 = 0.5$ $w_2 = 0.5$
As we can see for the fig.10 below, the runs with topic similarity perform better

than that without topic similarity. Between the two runs,the run with 10 LDA topics seem to recognize more events in all cases.

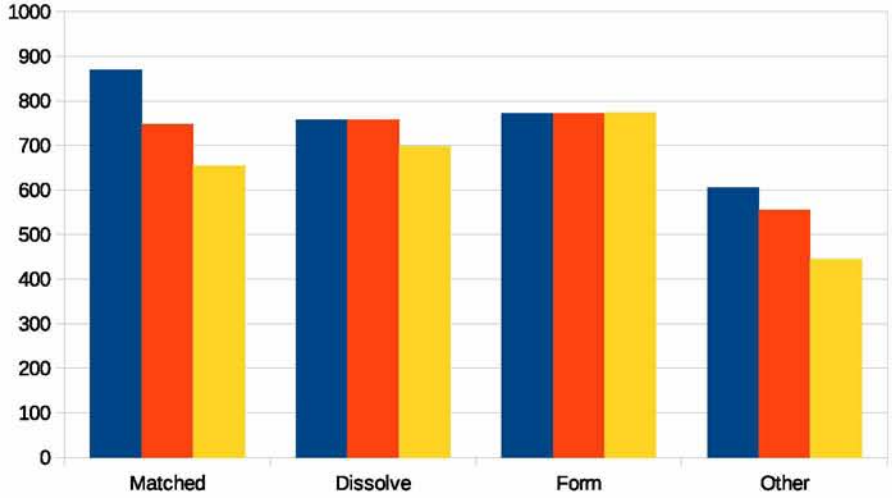

Figure 10: Blue: 10 LDA topics , Red: 50 LDA topics, Yellow: 0 LDA topics

Table 1: Confusion Matrix of 50 LDA topics

	Matched	Dissolve	Form	Other	Total
Matched	748	604	626	554	2532
Dissolve	0	154	0	0	154
Form	0	0	146	1	147
Other	0	0	0	0	2833
Total	748	758	772	555	2833

Table 2: Evaluation of 50 LDA topics

	Recall	Precision	F-measure
Matched	0.2954186414	1	0.456097561
Dissolve	1	0.2031662269	0.3377192982
Form	0.9931972789	0.189119171	0.3177366703

Table 3: Confusion Matrix of 10 LDA topics

	Matched	Dissolve	Form	Other	Total
Matched	868	604	626	604	2702
Dissolve	1	154	0	0	155
Form	0	0	146	1	147
Other	0	0	0	0	0
Total	869	758	772	605	3004

Table 4: Evaluation of 10 LDA topics

	Recall	Precision	F-measure
Matched	0.3212435233	0.998849252	0.4861383366
Dissolve	0.9935483871	0.2031662269	0.3373493976
Form	0.9931972789	0.189119171	0.3177366703

Table 5: Confusion Matrix of 0 LDA topics

	Matched	Dissolve	Form	Other	Total
Matched	654	228	314	444	1640
Dissolve	0	470	0	0	470
Form	0	0	460	1	461
Other	0	0	0	0	0
Total	654	698	774	445	2571

Table 6: Evaluation of 0 LDA topics

	Recall	Precision	F-measure
Matched	0.3987804878	1	0.5701830863
Dissolve	1	0.6733524355	0.8047945205
Form	0.9978308026	0.5943152455	0.7449392713

Table 7: Accuracy comparison between number of lda topics

	50 topics	10 topics	0 topics
Accuracy	0.3699258736	0.3888149134	0.6161026838

Above in the tables 1,3 and 5 , we present in detail the numbers of the recognized events in each run. From the table 7 we can see that 0 LDA topics achieves better accuracy. This occurs because without in our measures we don't count the ïio eventëvents. We see in the LDA experiments, true matched events are misclassified as dissolve or form. This occurs because of the inconsistency of the ground truth. During the process of creating the ground truth, we assume that if the name of a community is present in both time windows , it means that there is a matched event. Although the event recognition operates based on the population of the communities. Meaning that if in a time window there are lots of nodes in a community and in the next there are only two (the minimum),the ground will record a match but the algorithm a dissolve. That is the case.

5.3 Thresholds α and β

In this section we run experiments observing the sensitivity of the algorithm to the event thresholds.
LDA topics=10 overlap = 0.5 Centrality = Degree $w_1 = 0.5$ $w_2 = 0.5$

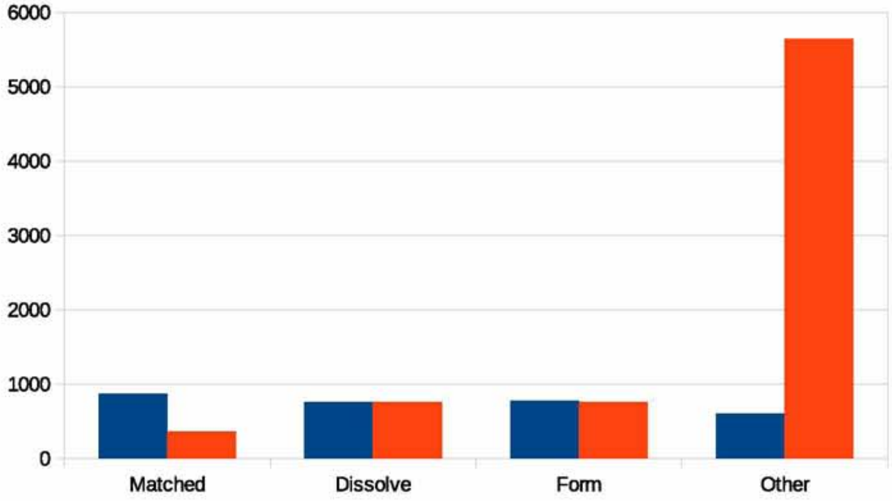

Figure 11: Blue: $\alpha = \beta = 0.75$, Red: $\alpha = \beta = 0.5$ 10 LDA topics

LDA topics=50 overlap = 0.5 Centrality = Degree $w_1 = 0.5$ $w_2 = 0.5$

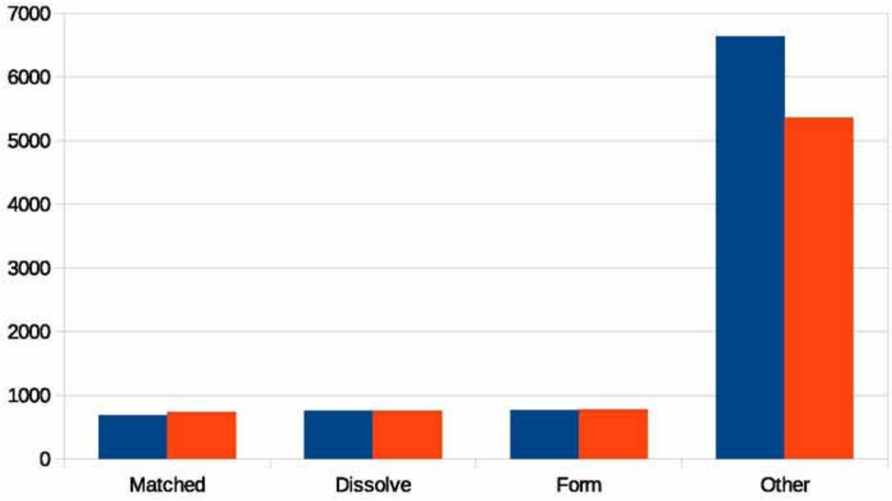

Figure 12: Blue: $\alpha = \beta = 0.5$, Red: $\alpha = \beta = 0.75$ 50 LDA topics

As we can see from figures 11 and 12 , with lower thresholds the "other" events,meaning splitting and merging, dominate over the matching events, independently the number of LDA topics. Also we see that dissolve and form are unaffected which is a rational result.Now comparing only the matched events , we notice that the higher the thresholds, the more matching events. This is also a rational conclusion, because we let less many-to-one relations to be formed.

5.4 Overlapping data

The use of overlapping data aims to smooth the transition through the data. We test how dissimilar are the result between overlapping and non-overlapping data. In this subsection we ran tests using two centralities, degree and Pagerank LDA topics=50 $\alpha = \beta = 0.75$ Centrality = Degree $w_1 = 0.5$ $w_2 = 0.5$

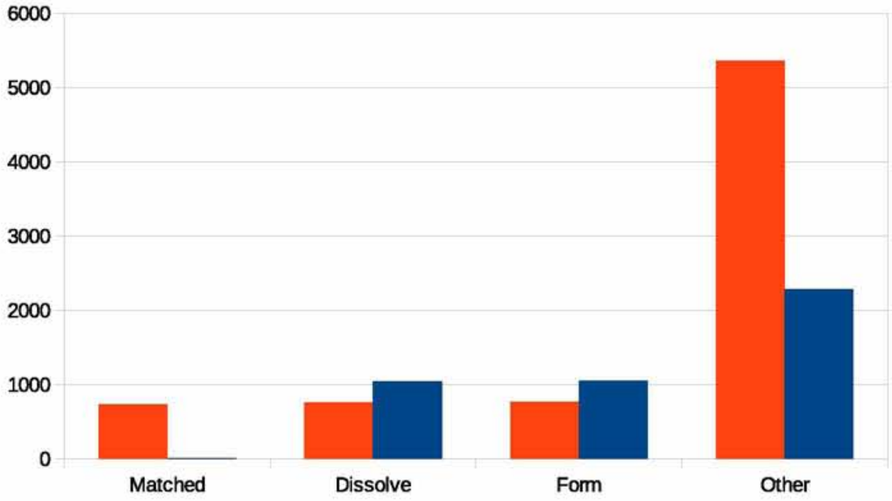

Figure 13: Red: overlap = 0.5 , Blue: overlap = 0 , Degree Centrality

Table 8: Confusion matrix of overlap = 0 with degree centrality

	Matched	Dissolve	Form	Other	Total
Matched	9	689	702	1342	2742
Dissolve	0	352	0	487	839
Form	0	0	353	454	807
Other	0	0	0	0	0
Total	9	1041	1055	2283	4388

Comparing the above confusion matrix 8 to matrix 5 , we clearly see that "matched events" are really scarce. From the elevated number of dissolve and form events, we can deduce that the communities in the consecutive timeframes are highly dissimilar.The same pattern can be found if we change the centrality ,as shown in the figure 14 below.

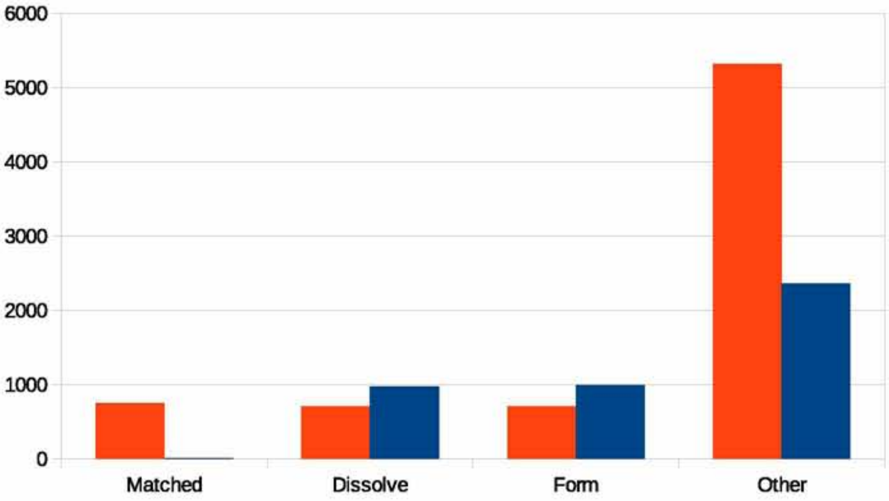

Figure 14: Red: overlap = 0.5 , Blue: overlap = 0 , PageRank Centrality

Table 9: Confusion Matrix of overlap = 0 with PageRank

	Matched	Dissolve	Form	Other	Total
Matched	7	647	667	1377	2698
Dissolve	0	332	0	509	841
Form	0	0	330	477	807
Other	0	0	0	0	0
Total	7	979	997	2363	4346

5.5 Centrality Comparison

In this subsection we tested how important is the choice of the centrality that is used in the group quality component. LDA topics=50 $\alpha = \beta = 0.75$ overlap $= 0.5$ $w_1 = 0.5$ $w_2 = 0.5$
As we can see in the figure 15 the result of both centralities are almost identical.

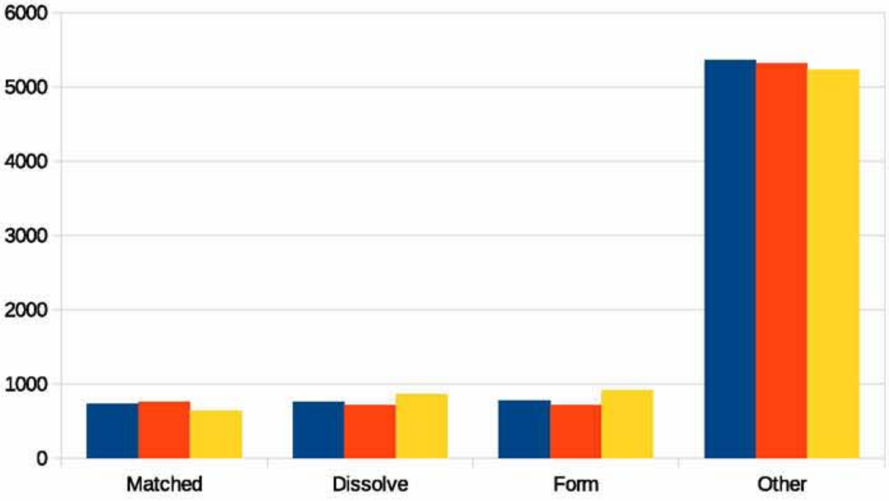

Figure 15: Red: Degree centrality , Blue: PageRank , Yellow: closeness

Table 10: Evaluation of Degree Centrality

	Recall	Precision	F-measure
Matched	0.1270555652	1	0.2254645984
Dissolve	0.1599169263	0.2031662269	0.1789657176
Form	0.1645997745	0.189119171	0.1760096444

Table 11: Evaluation of PageRank Centrality

	Recall	Precision	F-measure
Matched	0.1335579993	1	0.2356438742
Dissolve	0.1453790239	0.1971830986	0.1673640167
Form	0.1454340474	0.1814345992	0.1614518148

Table 12: Evaluation of Closeness

	Recall	Precision	F-measure
Matched	0.1100362757	1	0.1982570806
Dissolve	0.1796469367	0.2013969732	0.1899012075
Form	0.1848928974	0.1798245614	0.1823235131

Table 13: Accuracy comparison between centralities

	Degree	PageRank	Closeness
Accuracy	0.1355709978	0.1364850427	0.1275035999

From the overall numbers ,we see that Degree centrality and PageRank perform similarly , as expected, while closeness performs a little worse. Apparently in the longer the thread the lower the closeness between the questioner and the answer.

5.6 Weight Comparison

In this subsection we assigned different values to the weights corresponding to the structural similarity and topic similarity component, W_1 and W_2, respectively.

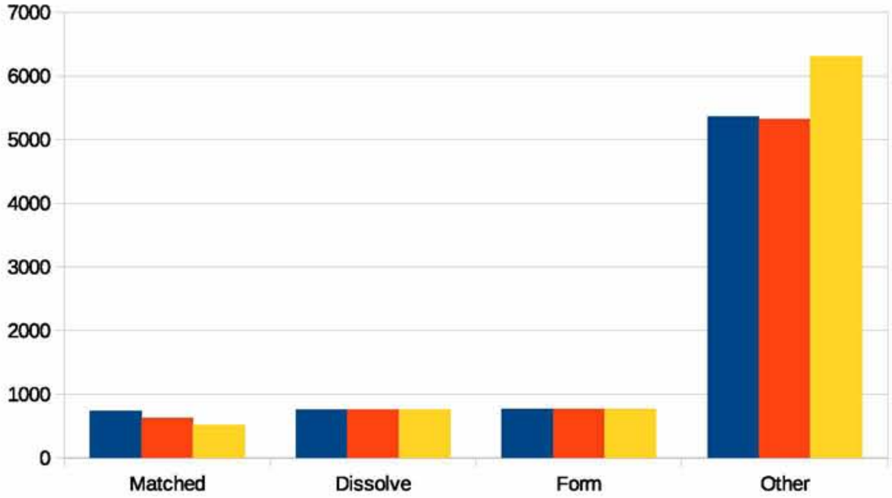

Figure 16: Blue: $W_1 = W_2 = 0.5$, Red: $W_1 = 0.75$ $W_2 = 0.25$, Yellow: $W_1 = 0.25$ $W_2 = 0.75$

Due to the particularity of the data set, forming communities form the same question/thread, the topics are highly related. As a result when the weight is larger for the topic similarity community similarity scores are higher, thus forming many-to-one relations meaning merges/splits and less matches.

6 Final Words

6.1 Conclusion

We have investigated the topic of Community Tracking in Temporal Social Networks with content information. This thesis present one of many way that we can export useful information from the content , using an unsupervised classifier.

Through the experiments, we can conclude that topic similarity helps the algorithm recognize more events, comparing to its simpler version using only structural information. Also we saw that the use of overlapping data is very essential in these types of experiments, as well as that the tuning of the LDA model is very important for the performance of the algorithm.

6.2 Future Work

For more reliable results, this algorithm should be tested with other datasets where there are ground truth events. Moreover , the speed of the algorithm can be improved by processing the timeframe pairs in parallel since they are independent. This can be achieved by using distributed computing frame works such as hadoop or spark.

References

[1] J. Doe. `http://dumps.mathoverflow.net/`, 2010.

[2] P. Brodka, S. Saganowski, and P. Kazienko, "Ged: the method for group evolution discovery in social networks," 2012.

[3] D. Hampe, J. Kleinberg, and E. Tardos, "Maximizing the spread of influence through a social network," in *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 137–146, 2003.

[4] T. H. Haveliwala, "Topic-sensitive pagerank," in *Proceedings of the 11th international conference on World Wide Web*, pp. 517–526, ACM, 2002.

[5] J. Weng, E.-P. Lim, J. Jiang, and Q. He, "Twitterrank: finding topic-sensitive influential twitterers," in *Proceedings of the third ACM international conference on Web search and data mining*, pp. 261–270, ACM, 2010.

[6] S. Saganowski, P. Brodka, and P. Kazienko, "Influence of the dynamic social network timeframe type and size on the group evolution discovery," 2012.

[7] S. Wasserman and K. Faust, *Social network analysis: Methods and applications*, vol. 8. Cambridge university press, 1994.

[8] A. Calvó-Armengol and Y. Zenou, "Social networks and crime decisions: The role of social structure in facilitating delinquent behavior," *International Economic Review*, vol. 45, no. 3, pp. 939–958, 2004.

[9] E. Bakshy, I. Rosenn, C. Marlow, and L. Adamic, "The role of social networks in information diffusion," in *Proceedings of the 21st international conference on World Wide Web*, pp. 519–528, ACM, 2012.

[10] `http://revealproject.eu/about-reveal`.

[11] D. Greene, D. Doyle, and P. Cunningham, "Tracking the evolution of communities in dynamic social networks," in *Advances in Social Networks Analysis and Mining (ASONAM), 2010 International Conference on*, pp. 176–183, ieee, 2010.

[12] M. Takaffoli, F. Sangi, J. Fagnan, and O. R. Zäıane, "Community evolution mining in dynamic social networks," *Procedia-Social and Behavioral Sciences*, vol. 22, pp. 49–58, 2011.

[13] J. Bi, Z. Qin, and J. Huang, "Detecting community and topic co-evolution in social networks," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 12, no. 5, pp. 4063–4070, 2014.

[14] V. Sekara, A. Stopczynski, and S. Lehmann, "The fundamental structures of dynamic social networks," *arXiv preprint arXiv:1506.04704*, 2015.

[15] J.-C. Delvenne, S. N. Yaliraki, and M. Barahona, "Stability of graph communities across time scales," *Proceedings of the National Academy of Sciences*, vol. 107, no. 29, pp. 12755–12760, 2010.

[16] P. J. Mucha, T. Richardson, K. Macon, M. A. Porter, and J.-P. Onnela, "Community structure in time-dependent, multiscale, and multiplex networks," *science*, vol. 328, no. 5980, pp. 876–878, 2010.

[17] R. Kumar, J. Novak, P. Raghavan, and A. Tomkins, *On the Bursty Evolution of Blogspace*, vol. 8. Springer, 2005.

[18] E. Otte and R. Rousseau, "Social network analysis: a powerful strategy, also for the information sciences," 2002.

[19] C. A. R. Pinheiro, "Social network analysis in telecommunications," 2011.

[20] "Yelp Academic Dataset." `http://www.yelp.com/academic_dataset`.

[21] J. Leskovec and A. Krevl, "SNAP Datasets: Stanford large network dataset collection." `http://snap.stanford.edu/data`, June 2014.

[22] I. Cantador, P. Brusilovsky, and T. Kuflik, "2nd workshop on information heterogeneity and fusion in recommender systems (hetrec 2011)," in *Proceedings of the 5th ACM conference on Recommender systems*, RecSys 2011, (New York, NY, USA), ACM, 2011.

[23] D. Green, "Dynamic community finding benchmark." `http://mlg.ucd.ie/snam/`.