Predicting the Evolution of Communities with Online Inductive Logic Programming

George Athanasopoulos[1]
geotha1995@hotmail.com

George Paliouras[2]
paliourg@iit.demokritos.gr

Dimitrios Vogiatzis[3,2]
dimitrv@iit.demokritos.gr

Grigorios Tzortzis[2]
gtzortzi@iit.demokritos.gr

Nikos Katzouris[2]
nkatz@iit.demokritos.gr

[1]Department of Informatics and Telecommunications, National and Kapodistrian University of Athens, Athens, Greece
[2]Institute of Informatics and Telecommunications, NCSR "Demokritos", Athens, Greece
[3]The American College of Greece, Deree, Athens, Greece

Abstract

In the recent years research on dynamic social network has increased, which is also due to the availability of data sets from streaming media. Modeling a network's dynamic behaviour can be performed at the level of communities, which represent their mesoscale structure. Communities arise as a result of user to user interaction. In the current work we aim to predict the evolution of communities, i.e. to predict their future form. While this problem has been studied in the past as a supervised learning problem with a variety of classifiers, the problem is that the "knowledge" of a classifier is opaque and consequently incomprehensible to a human. Thus we have employed first order logic, and in particular the event calculus to represent the communities and their evolution. We addressed the problem of predicting the evolution as an online Inductive Logic Programming problem (ILP), where the issue is to learn first order logical clauses that associate evolutionary events, and particular Growth, Shrinkage, Continuation and Dissolution to lower level events. The lower level events are features that represent the structural and temporal characteristics of communities. Experiments have been performed on a real life data set form the Mathematics StackExchange forum, with the OLED framework for ILP. In doing so we have produced clauses that model both short term and long term correlations.

2012 ACM Subject Classification Human-centered computing → Social network analysis, Computing methodologies → Machine learning, Computing methodologies → Online learning settings, Computing methodologies → Inductive logic learning

Keywords and phrases Social Network Analysis, Community Evolution Prediction, Machine Learning, Inductive Logic Programming, Event Calculus, Online Learning

Digital Object Identifier 10.4230/LIPIcs.CVIT.2016.23

1 Introduction

A social network is a structure which contains individuals, who are linked to other individuals. The link among them states an interaction which has one or more types of interdependency

such as friendship, kinship, common interest, financial exchange. Social networks are often represented as graphs, with nodes representing users and edge representing interactions. Usually a social network changes over time because new individuals join the network, new interactions are developed, or some individuals cease to be active for a short or a long period. This is actually the predominant behaviour especially in streaming social media, such as forums.

The social networks are often studied at the level of communities, which represent their meso-scale structure. A group of nodes forms a community if it densely connected, and sparsely connected to other communities. The said communities are not explicitly formed but rather implicitly as a result of the actions of individual users, that are not random but tend to follow a certain pattern that is related to their similarity to other users. There are many algorithms that have been developed for the detection of communities in networks that are static [6].

In dynamic networks, the communities are influenced over the time by its users' interaction. This influence causes changes in the structure of the communities. Many researchers, consider that the structure of a community contains important information for network evolution as a whole. Thus, it is highly imperative to model the dynamic behavior in social networks and try to predict their evolution.

In this paper we study the problem of community evolution prediction in dynamic social networks. We address this problem as a supervised learning task where we predict four types of community evolutionary events, *growth*, *shrinkage*, *continuation* and *dissolution*. Various features were investigated in order to understand how they influence the results. Among them, are the structural and temporal characteristics of communities. What is unique in the current approach is that we use a first order logic formalism to represent the correlation between evolutionary events and the input features. Moreover Inductive Logic Programming (ILP) us used to learn event calculus clauses. Event calculus was chosen because it is human understandable, it can be used to model effect of actions in time, and the variation we have adopted can perform ILP in an online fashion which is especially useful in streaming media.

The rest of the paper is organised as follows. In Section 2 we refer to past work on community evolution prediction, in Section 3 we refer to the Event Calculus as a logic formalism but also to Inductive Logic Programming as a way of learning clauses, then in Section 4 we refer to the methodology we followed for community evolution prediction, in Section 5 we present experimental results, conclusions are drawn in Section 6. In appendix A we present samples of Event Calculus clauses, and we elaborate on the Hoeffding bound.

2 Related Work

One of the most interesting problems in social network prediction is the prediction of possible events, at the level of communities, such as growing, shrinking, merging with another community or splitting into more communities The research in community evolution prediction and the events have been proposed is quite extensive.

Patil et al. [14] predicted whether a community will disappear or will survive. They observed that both the level of member diversity and social activities are critical in maintaining the stability of communities. They also found that certain prolific members play an important role in maintaining the community's stability. Goldberg et al. [8] correlated the lifespan of a community with the structural parameters of its early stages. Brodka et al. [7], [2] tried to predict 6 evolutionary events of communities, i.e. growth, shrinkage, continuation, dissolution, merging and split. The used as features the history of the events of a community

in the three preceding timeframes, and the community size in these timeframes. They found that the prediction based on simple input features may be very accurate, while some classifiers are more precise than the others. Kairam et al. [11] tried to understand the factors contributing to the growth and longevity in a social network. They investigated the role that two types of growth (Diffusion and non-diffusion) play during a community's formative stages. Diffusion growth is when a community attracts new members through ties to existing members. Non-diffusion growth occurs with individuals with no prior ties to the network. Diakidis et al. [4] studied on-line social networks as a supervised learning task with sequential and non-sequential classifiers. Structural, content and contextual features as well as the previous states of a community are considered as the features that are involved in the task of community evolution. The evolution phenomena they tried to predict are the continuation, shrinking, growth and dissolution.

Takaffoli et al. [16] quantified the events that may occur in a community as follows: survive:{true, false}, merge:{true, false}, split:{true, false}, size:{expand, shrink}, and cohesion:{cohesive, loose}. First they tried to predict whether a community will survive, followed by a separate predictor for each of the events.

Ilhan et al. [10] proposed a regression ARIMA model to predict values of community features based on the values of the past community instances. Then the predicted community features are used to train a classifier to predict the evolutionary events.

3 Background: Event Calculus and Inductive logic programming

The Event Calculus (EC) is a temporal logic formalism for reasoning about actions and changes. [13] EC that has been used as a basis in event recognition applications, providing among others, direct connections to machine learning, via Inductive Logic Programming (ILP) [3]. Its ontology comprises *time points*, represented by integers; time varying properties known as *fluents*; and actions known as *events*. The events occur in time and may affect the fluents by altering their value. The axioms of the EC incorporate the *law of inertia*, according to which fluents persist over time, unless they are affected by an event. Thus, if an event is initiated at time T, it will persist until another event will fire a termination rule. Also, if an event is terminated at time T, it will remain terminated until another event fires an initiation rule. The basic predicates are presented in Table 1, while the domain-independent axioms are in Table 2. Axiom (1) states that a high level event, represented as fluent F for convenience, is happening at time T if it has been initiated at the previous time point. While Axiom (2) states that F continues to happen unless it is terminated.

Let us examine how the evolution of a community could be represented in terms of EC; an evolutionary phenomenon such as the *growth* will be represented as a fluent, whereas the factors that contribute to the *growth* will be represented as events. For example in Table 3, it means that community $X0$ will grow in time T provided its size was 3 and its density was 4. Likewise, rules can be formed for the rest of the evolutionary phenomena as combinations of features (or events). The rules are also known as *clauses*, whereas the predicates *happensAt* are also known as *literals*. In addition, a rule B is a specialisation of rule A if the instances that satisfy rule B are a subset of the instances that satisfy rule A.

The problem that we try to address is to learn such rules (or clauses) from data, rather than hand encode them; for that we can use inductive logic programming (ILP). In ILP programming first order rules are learnt from relational data under a supervised learning scheme. Thus we have input data and class labels. The input data are often named the *narrative*, and the class label is know as the *annotation*.

Table 1 The basic predicates of the EC

Predicate	Meaning
happensAt(E,T)	Event E occurs at time T
initiatedAt(F,T)	At time T fluent F is initiated
terminatedAt(F,T)	At time T fluent F is terminated
holdsAt(F,T)	Fluent F holds at time T

Table 2 The domain-independent axioms

Axioms
holdsAt(F,T +1) ← (1) initiatedAt(F,T).
holdsAt(F, T+1) ← (2) holdsAt(F,T), not terminatedAt(F,T).

Given an encoding of the known background knowledge and a set of examples represented as a logical database of facts, an ILP system will derive a *hypothesised* logic program which entails all the positive and none of the negative examples. ILP provides various techniques for learning logical theories from examples. In Learning from Interpretations (LfI) [1] setting each training example is an interpretation, i.e. a set of narrative and annotation atoms (see Table 4). Given a set of training interpretations I and some background theory B, which consists of the domain-independent axioms of the EC, the goal in LfI is to find a theory.

In this paper, OLED (Online Learning of Event Definitions) [12] was used for learning rules that perform community evolution prediction. OLED is an online ILP system for learning logical theories from data streams. It has been designed having in mind the construction of knowledge bases for event recognition applications. These applications [5] process sequences of simple events, such as sensor data, and recognize complex events of interest, i.e. events that satisfy some pattern. Logic-based event recognition typically uses a knowledge base of first-order rules to represent complex event patterns and a reasoning engine to detect such patterns in the incoming data. In OLED this knowledge base is in the form of domain-specific axioms in the Event Calculus, i.e. rules that specify the conditions under which simple, low-level events initiate or terminate complex events.

OLED is using an online (single-pass) learning strategy. Online machine learning is a method of machine learning in which data becomes available in a sequential order and is used to update our best predictor for future data at each step, as opposed to batch learning techniques which generate the best predictor by learning on the entire training data set at once. To manage it, the Hoeffding bound [9] for evaluating clauses on a subset of the input stream, is used. With this approach, significant speed-ups are obtained in training time. Table 4 presents an example of input data that is provided to OLED. It consists of a

Table 3 Theory Learnt by OLED

```
initiatedAt(growth(X0),T) ←
        happensAt(size(X0,3),T),
        happensAt(density(X0,4),T).
```

narrative and an annotation list. Narratives are the simple (or low level) events in terms of happensAt/2, expressing the values of communities' features. i.e. happensAt(size(c1,3),1). denotes that community c1 has size 3 in time 1. Annotations are the complex (or high level events) events in terms of holdsAt/2, expressing the ground truth for our training set. i.e. holdsAt(*growth*(c1),2). denotes that community c1 grew in time 2. The non-existence of c1's annotation in time 1 states that growth event is terminated in time 1. Table 3 shows an example of the theory OLED learnt after training. It represents we will begin to have a growth event in time T+1 for any community, which has size 3 and density 4 in time T.

This rule extracted because with these community features in time 1, we had a growth event in time 2 (Table 4).

Table 4 Input of OLED

Timeframe 1	Timeframe 2
Narrative happensAt(size(c1,3),1). happensAt(density(c1,4),1).	Narrative happensAt(size(c1,5),2). happensAt(density(c1,5),2).
Annotation	Annotation holdsAt(*growth*(c1),2).

Learning OLED learns a clause in a top-down fashion, by gradually adding literals to its body. Instead of evaluating each candidate specialization on the entire input, it accumulates training data from the stream, until the Hoeffding bound allows to select the best specialization. The instances used to make this decision are not stored or reprocessed but discarded as soon as OLED extracts from them the necessary statistics for clause evaluation.

OLED relaxes the LfI requirement that a hypothesis H covers every training interpretation, and thus seeks a theory with a good fit in the training data. Let B consist of the domain-independent EC axioms, r be a clause and I an interpretation. We denote by narrative(I) and annotation(I) the narrative and the annotation part of I respectively (Table 4). We denote by Mr^r_I an answer set of $B \cup narrative(I) \cup r$. Given an annotation atom α we say that:

- α is a true positive (TP) atom clause r, iff $\alpha \in annotation(I) \cap Mr^r_I$.
- α is a false positive (FP) atom clause r, iff $\alpha \in Mr^r_I$ but $\alpha \notin annotation(I)$.
- α is a false negative (FN) atom clause r, iff $\alpha \in annotation(I)$ but $\alpha \notin Mr^r_I$.

We define a heuristic clause evaluation function G as follows:

$$G(r) = \begin{cases} \frac{TP_r}{TP_r+FP_r}, & \text{if } r \text{ is an initiatedAt clause} \\ \frac{TP_r}{TP_r+FN_r}, & \text{if } r \text{ is a terminatedAt clause} \end{cases}$$

where TP_r, FP_r and FN_r are the accumulated TP, FP and FN counts of clause r over the input stream and $G \in [0, 1]$.

On the arrival of new interpretations, OLED either expands H, by generating a new clause, or tries to expand (specialize) an existing clause. Clauses of low quality are pruned, after they have been evaluated on a sufficient number of examples. Below there is an example of OLED execution. Initially, processes Linit and Lterm start with two empty hypotheses, Hinit and Hterm. Assume that the annotation in one of the incoming interpretations dictates that the *growth* complex event holds at time 10, while it does not hold at time 9. Since no clause in Hinit yet exists to initiate *growth* at time 9, Linit detects the *growth* instance at time 10 as a FN and proceeds to theory expansion, generating an initiation clause for *growth*. Lterm is not concerned with initiation conditions, so it will take no actions in this case. Then, a new interpretation arrives, where the annotation dictates that *growth* holds at time 20 but does not hold at time 21. In this case, since no clause yet exists in Hterm to terminate *growth* at time 20, Lterm will detect an FP instance at time 21. It will then proceed to theory expansion, generating a new termination condition for *growth*. At the same time, assume that the initiation clause in Hinit is over-general and erroneously re-initiates *growth* at time 20, generating an FP instance for the Linit process at time 21. In response to

that, Linit will proceed to clause expansion, penalizing the over-general initiation clause by increasing its FP count, thus contributing towards its potential replacement by one of its specializations.

In EC the initiation/termination of complex events depends only on the simple events and contextual information of the previous time-point, therefore each interpretation is an independent training instance. This guarantees the independence of observations that is necessary for using the Hoeffding bound, which is defined in Appendix [A]. Let r be a clause and G a clause evaluation function. Assume also that after n training instances, $r1$ is r's specialization with the highest observed mean G-score $\overline{G}$ and $r2$ is the second best one, i.e. $\Delta\overline{G} = \overline{G}(r1) - \overline{G}(r2) > 0$. Then by the Hoeffding bound we have that for the true mean of the scores' difference $\Delta\hat{G}$ it holds $\Delta\hat{G} > \Delta\overline{G} - \varepsilon$, with probability $1 - \delta$. Hence, if $\Delta\overline{G} > \varepsilon$ then $\Delta\hat{G} > 0$, implying that $r1$ is indeed the best specialization to select at this point, with probability $1 - \delta$. In order to decide which specialization to select, it thus suffices to accumulate observations from the input stream until $\Delta\overline{G} > \varepsilon$. Also, because OLED allows to build decision models using only a small subset of the data, by relating the size of this subset to a user-defined confidence level on the error margin of not making a (globally) optimal decision, manages to consume small amounts of memory and time resources.

4 Proposed Methodology

A dynamic social network is time-tamped, and to be analysed it is *segmented* into time frames, with an overlap between them to allow for a smooth transition. The problem we are addressing is to predict the form of a community in the next frame, given some features of the existing form of a community. The model that perform the prediction is learnt through ILP and represented as clauses of Event Calculus.

4.1 Community Features

Pavlopoulou et al. [15] designed two types of features, the *structural* and the *temporal* ones. Structural features represented the physical characteristics of a community such as size, density etc. The temporal features included structural features and evolutionary events that were derived from the past instances of a community, and from relations between past instances of a community. In this work, we use the same features which describe below but first let us introduce some notation: C_t is the set of communities at time frame F_t; C_t^k is the community k of set C_t; $n(F_t) = |V_t|$ is the size of set V_t; $m(F_t) = |E_t|$ is the size of set E_t; and $C_{t_j}^{k_j}$ is the ancestor of $C_{t_i}^{k_i}$, where $j < i$.

The ancestors of a community do not necessarily belong to consecutive time frames. In the current experiments, each community is tracked in each timeframe until its dissolution, thus there are situations in which a community disappears at timeframe t_i but it reappears at timeframe t_j, where $j - i > 1$. Thus, the i-th ancestor of a community is the i-th appearance of the community in the past, counting form the present. Next are the features we used in detail:

Structural Features

Size is the normalized value for the size of a community C_t^k in time frame F_t:

$$Size(C_t^k) = \frac{n(C_t^k)}{n(F_t)}$$

Density is the number of C_t^k edges to the maximum number of edges the community could have:

$$Density(C_t^k) = \frac{m(C_t^k)}{n(C_t^k)(n(C_t^k)-1)/2}$$

Cohesion is defined as:

$$Cohesion(C_t^k) = \frac{2m(C_t^k)(n(F_t)-n(C_t^k))}{m_{out}(C_t^k)(n(C_t^k)-1)}$$

Normalised Association is defined as:

$$NormalizedAssociation(C_t^k) = \frac{2m(C_t^k)}{2m(C_t^k)+m_{out}(C_t^k)}$$

Ratio Association is the average internal degree of a community's members:

$$RatioAssociation(C_t^k) = \frac{2m(C_t^k)}{n(C_t^k)}$$

Ratio Cut is the average external degree of a community's members:

$$RatioCut(C_t^k) = \frac{m_{out}(C_t^k)}{n(C_t^k)}$$

Normalized Edges Number is defined as:

$$NormalizedEdgesNumber(C_t^k) = \frac{m(C_t^k)}{m(F_t)}$$

Average Path Length shows how close on average two random nodes are:

$$AveragePathLength(C_t^k) = \frac{\sum_{v,u\in V_t^k, v\neq u} dist(v,u)}{n(C_t^k)(n(C_t^k)-1)}$$

where $dist(v,u)$ indicates the shortest distance between nodes v and u.

Diameter is the maximum shortest path between all pairs of nodes in community C_t^k:

$$Diameter(C_t^k) = \max_{u,v\in V_t^k, u\neq v} dist(u,v)$$

Clustering Coefficient of community C_t^k shows how often, on average, the neighbours of a node of the community are also connected to each other.

Centrality measures how central each node of a community C_t^k is. We used three centrality measures as features, namely closeness, betweenness and eigenvector centrality [17].

Temporal features

Structural features and Evolutionary events of N ancestors:
One group of temporal features is all the structural features, as described above, as well as the *evolutionary events* for the first n immediate ancestors of community $C_{t_p}^{k_p}$.

Another group of temporal features concerns pairs of communities and depict how a community has evolved compared to its previous instance in time. Using these pairs of communities for a given number of ancestors n to use, we compute the following temporal features:

Similarity of consecutive communities is the fraction between of nodes/edges that are common in both instances of the community and total nodes/edges of two instances.

$$Jacc_{Nodes}(C_{t_i}^{k_i}, C_{t_{i-1}}^{k_{i-1}}) = \frac{|V_{t_i}^{k_i} \cap V_{t_{i-1}}^{k_{i-1}}|}{|V_{t_i}^{k_i} \cup V_{t_{i-1}}^{k_{i-1}}|}, Jacc_{Edges}(C_{t_i}^{k_i}, C_{t_{i-1}}^{k_{i-1}}) = \frac{|E_{t_i}^{k_i} \cap E_{t_{i-1}}^{k_{i-1}}|}{|E_{t_i}^{k_i} \cup E_{t_{i-1}}^{k_{i-1}}|},$$

$$Jacc_{Nodes\&Edges}(C_{t_i}^{k_i}, C_{t_{i-1}}^{k_{i-1}}) = \frac{|V_{t_i}^{k_i} \cap V_{t_{i-1}}^{k_{i-1}}| + |E_{t_i}^{k_i} \cap E_{t_{i-1}}^{k_{i-1}}|}{|V_{t_i}^{k_i} \cup V_{t_{i-1}}^{k_{i-1}}| + |E_{t_i}^{k_i} \cup E_{t_{i-1}}^{k_{i-1}}|}$$

where $E_{t_i}^{k_i}$ is the set of edges and $V_{t_i}^{k_i}$ the set of nodes of community $C_{t_i}^{k_i}$.

Join nodes ratio is the percentage of nodes joining the dynamic community as it evolves.

$$JoinNodesRatio(C_{t_i}^{k_i}, C_{t_{i-1}}^{k_{i-1}}) = \frac{|V_{t_i}^{k_i} \backslash V_{t_{i-1}}^{k_{i-1}}|}{|V_{t_i}^{k_i}|}$$

Left nodes ratio is the percentage of nodes leaving the dynamic community as it evolves.

$$LeftNodesRatio(C_{t_i}^{k_i}, C_{t_{i-1}}^{k_{i-1}}) = \frac{|V_{t_{i-1}}^{k_{i-1}} \backslash V_{t_i}^{k_i}|}{|V_{t_{i-1}}^{k_{i-1}}|}$$

Activeness is the ratio of the number of edges in current community $C_{t_i}^{k_i}$ which also existed in its ancestor community $C_{t_{i-1}}^{k_{i-1}}$, to the number of nodes in current community $C_{t_i}^{k_i}$.

$$Activeness(C_{t_i}^{k_i}, C_{t_{i-1}}^{k_{i-1}}) = \frac{|E_{t_i}^{k_i}| - |E_{t_i}^{k_i} \backslash E_{t_{i-1}}^{k_{i-1}}|}{|V_{t_i}^{k_i}|}$$

The last two temporal features are computed for individual communities instead of pairs.

Lifespan is the ratio of the ancestors of a community based on the corresponding dynamic community M, to the maximum number of ancestors it could have.

$$LifeSpan(C_{t_w}^{k_w}) = \frac{|\{C_{t_p}^{k_p} \in M : p < w\}|}{t_w - 1}$$

Aging of a community $C_{t_w}^{k_w}$ is the average age of the community members. The age of a member is increased by 1 every time it is found to be also a member of an ancestor community of $C_{t_w}^{k_w}$ in the corresponding dynamic community. Aging is normalized by dividing with the maximum possible age of members, which equals w.

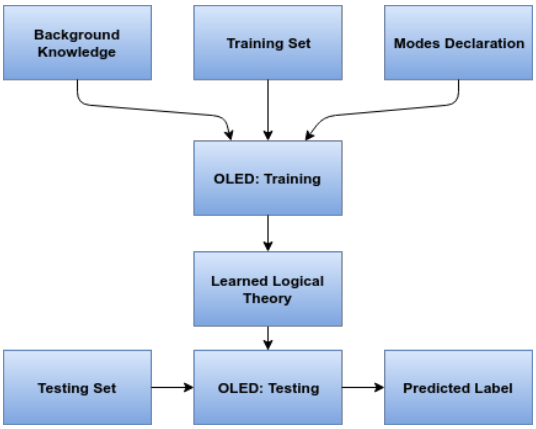
$$Aging(C_{t_w}^{k_w}) = \frac{\sum_{v \in V_{t_w}^{k_w}} |\{C_{t_p}^{k_p} \in M : p \leq w, v \in V_{t_p}^{k_p}\}|}{(|\{C_{t_p}^{k_p} \in M : p < w\}| + 1) n(C_{t_w}^{k_w})}$$

Feature Quantization In OLED the values of variables are discrete thus we implemented two methods to quantize variables. Let q_{value} be the number of quantized values, f_v be the set with values of feature f. In first method, for each feature we split values' total range to q_{value} intervals and the width of each is $\frac{max\{f_v\} - min\{f_v\}}{q_{value}}$. Thus the first interval is $(min\{f_v\}, min\{f_v\} + q_{value})$, the second is $(min\{f_v\} + q_{value} + 1, min\{f_v\} + 2q_{value})$ and so on. The quantized value of each feature is the index of the interval it belongs to. The second method sorts feature's values in a list and creates q_{value} sets. Taking one by one the values from sorted list, we begin to fill the q_{value} sets with consecutive values until each set has $\frac{|f_v|}{q_{value}}$ feature's values. Finally, if at least one feature or tag of community C_k is missing we delete the C_k.

4.2 Community Evolution Prediction

OLED was used to predict four evolutionary events: *growth*, *shrinkage*, *continuation*, *dissolution*. Note that OLED handles two-class problems, so it predicts if a community will sustain or will stop sustaining an evolutionary event. In Figure 1 we present the architecture of the prediction system.

Figure 1 Learning Architecture



The performance of an ILP system may degrade if the background knowledge provided contains large amounts of irrelevant information so experts are required to set the background knowledge they believe to be useful. Table 5 presents an example of background knowledge, where the following types of rules can be defined: Rules for community entity recognition; rules for time entity recognition; facts for features' quantized values recognition; rules for values of ground truth recognition; and rules which represent the inertia of Event Calculus. OLED can produce predicates of many forms. For example, an argument of a predicate can be considered as input or as output. *Modes* declaration is a language that limits the forms a predicate can have. Table 7 presents an example of modes.

Table 5 Example of A OLED's Background Knowledge File

Background Knowledge File	
holdsAt(F,T_e) :- fluent(F), holdsAt(F,T_s), not terminatedAt(F,T_s), $T_e = T_s + 1$, time(T_s),time(T_e).	holdsAt(F,T_e) :- fluent(F), initiatedAt(F,T_s), $T_e = T_s + 1$, time(T_s),time(T_e). Inertia of Event Calculus
fluent(*growth*(X)) :- community(X).	Ground truth recognition
community(X) :- happensAt(size(X,_),_). community(X) :- happensAt(density(X,_),_).	Community entity recognition
time(X) :- happensAt(size(_,_),X). time(X) :- happensAt(density(_,_),X).	Time entity recognition
value(1..5).	Features' quantized values recognition

The form of rules that OLED learns is presented in Table 6. In the head of the rule, *predicted_event* is one of labels we try to predict (*growth*, *shrinkage*, *continuation*, *dissolu-*

Table 6 Rules That OLED Learns

Rules
initiatedAt/terminatedAt(< $predicted\_event$ >(< $community_i$ >),< $time_j$ >) :- happensAt(< $feature_1$ >(< $community_i$ >,< $value_1$ >),< $time_j$ >), ... happensAt(< $feature_n$ >(< $community_i$ >,< $value_n$ >),< $time_j$ >)). (1)/(2)

Table 7 Example of A OLED's Mode Declarations File

Mode Declarations File	
modeh(initiatedAt(*growth*(+community),+time))	
modeh(terminatedAt(*growth*(+community),+time))	The form of the rule's head
modeb(happensAt(size(+community,#value),+time))	
modeb(happensAt(density(+community,#value),+time))	The form of the rule's body

tion). Notice that the $community_i$ and $time_j$ indices are the same in the body and head. Rules can be interpreted as if $feature_1$ of community $community_i$ has $value_1$ at $time_j$ and the same is true for the rest of features then the initiation of event *predicted_event* is fired. This means that the *predicted_event* will start to occur at $time_{j+1}$. The *happensAt* predicates that are required will be discovered by OLED. Likewise, if the body of rule (2) is true then the termination of event *predicted_event* is fired, thus the *predicted_event* will stop to occur at time $time_{j+1}$.

Training and Testing As shown in Figure 1, the dataset was split into training and testing sets according to the Time Series Cross Validation, because it takes into account the temporal relationship between the training and testing sets. Each training comprises only observations that occurred prior to the observation.

Fold 1: Training set includes low events of communities from timeframes F_1, F_2 and high events of communities in timeframe F_2. Testing set includes low events of communities from timeframe F_2 and high events of communities from timeframes F_2, F_3.

Fold 2: Training set includes low events of communities from timeframes F_1, F_2, F_3 and high events of communities from timeframe F_2, F_3. Testing set includes low events of communities from timeframe F_3 and high events of communities from timeframes F_3, F_4.

... ...

Fold T-2: Training set includes low events of communities from timeframes F_1, F_2, ..., F_{T-1} and high events of communities from timeframe F_2, F_3, ..., F_{T-1}. Testing set includes low events of communities from timeframe F_{T-1} and high events of communities from timeframes F_{T-1}, F_T.

T is total number of timeframes in the dataset. Note that the first timeframe has no evolutionary events (high events) since there is no previous timeframe in order to track the communities' evolution of the first timeframe. Respectively, the last timeframe has not features (low events) because there is no next timeframe to predict evolution of its communities. Also, in the training set we comprise the low level events of the timeframe that we are going to predict so that OLED extracts the time variable for high level events. Finally, notice that in the testing set we also comprise the high level events of the previous timeframe than that we are going to predict. This is required by OLED to initiate the inertia

of every community's event.

OLED as an online learner splits its input into chunks. In our experiments, we choose chunks of size 2. Thus, the imported timeframes for the training procedure are split into chunks two by two. We changed the functionality of OLED so that it creates rolling chunks. It means that first chunk contains the timeframes 1,2; the second one the timeframes 2,3; the third the timeframes 3,4 and so on. This necessary because, for example, timeframe 2 has to be in the first and second chunk. In the first chunk, we need the high level events of timeframe 2 for getting the ground truth. While in the second chunk we use the low level events of timeframe 2 as features for our supervised learning classification.

The outline of training process is the following: Initially there is an empty theory. Each time OLED receives a chunk of training examples and transforms the existing theory to satisfy as close as possible the right prediction of the current examples. When the training process is completed, a logical theory is derived as the learnt model. Its form is illustrated in Table 6. Using this theory we predict the evolutionary events of communities which are in testing set.

5 Experiments

Dataset description The data were collected from the Mathematics Stack Exchange forum[1], which is a question and answer site for mathematics. All questions are tagged with their subject areas. The dataset comprises 376,030 posts, 261,600 answers and comments, between 28-09-2009 and 31-05-2013. Each user is represented by a node in a graph and there is an edge between two user nodes if one of them posts an answer or a comment on the other user's post. The dataset was split into 10 equally sized, with respect to the number of posts (questions, answers or comments), timeframes with 60% overlap between them.

Building the ground truth means obtaining community labels per time frame, and then obtaining the evolution of each community across time frames. We considered that a group of users belongs in the same community if they post (questions, answers or comments) about the same topic. In particular, we used tags to determine the communities and since on each post there are multiple tags, thus each user will be assigned to multiple communities. Answer and comment posts inherit the tag of the question they correspond to. Also communities with no more than 3 members were removed. The evolutionary events of each community (*Growth*, *Shrinkage*, *Continuation*, *Dissolution*) were obtained by thresholding. In particular if the size of the community in the next time frame is more (less) than 30 nodes compared to the size in the current frame then the community grows (shrinks).

There are communities which do not appear in each timeframe, although they may not have been dissolved yet. It happens because communities with few members in a timeframe are pruned from dataset. So, we are looking for the evolution of a community in every timeframe of the dataset and consider a community as dissolved only after its last appearance. The evolutionary events of dataset are imbalanced. In particular, the percentage of each class is: *Growth*: 0.5%, *Shrinkage*: 0.2%, *Continuation*: 90% and *Dissolution*: 0.3%. The features were quantized into 5 levels, and represented as low level events. The high level events are the evolutionary events. Experiments were executed with both structural and temporal features, where the number of ancestors was set to 4. At the end, the dataset was split in training and testing sets using Time Series Cross Validation method. Because

[1] https://archive.org/download/stackexchange

the data are highly imbalanced apart from Micro Average measures, we also used Macro Averages.

Survival Experiment First, we conducted an experiment with an event, named *survival* that incorporated all the *growth*, *shrinkage* and *continuation* events. We used both structural and temporal features. In Table 8 we present the results. The micro measures are very high

Table 8 Survival Experiment

	Survival Structural	Survival Temporal
Micro/Macro Precision	0.9737/0.8125	0.9882/0.9941
Micro/Macro Recall	0.9949/0.6289	1.0000/0.6765
Micro/Macro Fscore	0.9842/0.7090	0.9941/0.8051

because the survival events are 97% of the total data. As OLED initialized the inertia of the survival event, it did not find enough negatives examples (dissolution events) to fail in its prediction. Thus the Macro values are much lower.

Table 9 All events Structural features

	Growth	Shrinkage	Continuation	Dissolution
Micro/Macro Precision	0.2358/0.6037	0.1884/0.5832	0.9293/0.8066	0.6512/0.8125
Micro/Macro Recall	0.3027/0.6316	0.1512/0.5671	0.9760/0.6939	0.2629/0.6289
Micro/Macro Fscore	0.2651/0.6174	0.1677/0.5750	0.9521/0.7460	0.3746/0.7090

Experiment with all events The results on all events with *structural* features appear in Table 9. The macro precision is highest in the *dissolution*. The dataset with the *temporal*

Table 10 All events Temporal features

	Growth	Shrinkage	Continuation
Micro/Macro Precision	0.1828/0.5730	0.1882/0.5743	0.9182/0.9222
Micro/Macro Recall	0.1828/0.5730	0.1633/0.5649	0.9955/0.6932
Micro/Macro Fscore	0.1828/0.5730	0.1749/0.5696	0.9553/0.7915

features contains the features of the previous 4 instances of a community, the first timeframe for this dataset is at time 5 (see Table 10). The theory which was derived for the dissolution event was empty. The predictor could not evaluate any rule with high score because there were not many available examples, since the number of timeframes (6, from F_5 to F_{10}) and the number of communities is small. Thus, the dissolution event is not included in the experiments with *temporal features*. The *growth* and *shrinkage* events with temporal features have lower performance than the best corresponding events with *structural* features. But for the *continuation* event, the reverse is true for both micro and macro values.

Experiment with pruning A learnt theory can be pruned to remove clauses whose score is smaller than a quality threshold S_{min}. In the previous experiments S_{min} was 0.9. Next we tried for each event the values 0.5, 0.7, 0.3 as S_{min} and choose the ones with the best performance. With the structural features, the best pruning value for the *growth* event is

Table 11 Best Pruning Experiment with Structural Features

	Growth	Shrinkage	Continuation	Dissolution
Micro/Macro Precision	0.2343/0.6035	0.2047/0.5913	0.9293/0.8066	0.6512/0.8125
Micro/Macro Recall	0.3295/0.6431	0.1512/0.5679	0.9760/0.6939	0.2629/0.6289
Micro/Macro Fscore	0.2739/0.6227	0.1739/0.5794	0.9521/0.7460	0.3746/0.7090

0.7, for *shrinkage* 0.5, for *continuation* 0.9 and for *dissolution* 0.9 (see Table 11). In the temporal features, the best pruning value for the *growth* event is 0.7, for *shrinkage* 0.7, and for *continuation* 0.9 (see Table 12). Overall, pruning had a marginal improvement on the results.

Table 12 Best Pruning Experiment with Temporal Features

	Growth	Shrinkage	Continuation
Micro/Macro Precision	0.1828/0.5730	0.1951/0.5778	0.9182/0.9222
Micro/Macro Recall	0.1828/0.5730	0.1633/0.5656	0.9955/0.6932
Micro/Macro Fscore	0.1828/0.5730	0.1778/0.5716	0.9553/0.7915

Experiment with long range rules We changed the way rules are formed so that they contain features of any of a community's ancestors. This is similar to the temporal features, but we do not have their values as different features but as the same features at different time steps. In order to change the form of the derived rules we changed the modes declaration so that the new form of rules captures long range relationships (see also Section 4.2). The form of the new rules is shown in Table 13, which is the same as in Table 6 except of the <*time*> value in the head can be different from that in the body and the *geqn/3* predicate, which denotes that $time_k$ is num_1 units after $time_l$. Also because now OLED could include more than two timeframes we had to increase the chunk size. If the chunk size equals to $N + 2$, then the derived rules can contain up to N ancestors' features. However, a big chunk size entails greater CPU and memory requirements. In the experiments we selected a chunk of size 3, so we obtained rules that contained features of the first ancestor. The results are presented in Table 14.

Table 13 New Rules That OLED Learns With Long Range Relationships

Rules
initiatedAt/terminatedAt(< $predicted\_event$ >(< $community_i$ >),< $time_j$ >) :-
happensAt(< $feature_1$ >(< $community_i$ >,< $value_1$ >),< $time_1$ >)),
...,
happensAt(< $feature_n$ >(< $community_i$ >, < $value_n$ >), < $time_n$ >)),
geqn($time_k$, $time_l$, num_1),
...
geqn($time_m$, $time_n$, num_1).

Experiment with weighted TPs, FPs, FNs Neither the previous method increased the performance significantly. A problem is that the learnt theories contain more termination than initiation rules, thus the initiation of some events does not happen. It means OLED

Table 14 Long Range Relationships Experiment

	Growth	Shrinkage
Micro/Macro Precision	0.2446/0.6090	0.2016/0.5898
Micro/Macro Recall	0.3487/0.6527	0.1512/0.5678
Micro/Macro Fscore	0.2875/0.6300	0.1728/0.5786

predicts a negative event (i.e. event that does not occur) for a community at next timeframe but in reality it is a positive event (i.e. the event occurs). In this case the FNs frequency of OLED is increased. The numbers of initiation and termination rules are not balanced because OLED evaluates its rules based on TPs, FPs and FNs values. Using these values, it computes a score which evaluates the accuracy of a rule. To control score's value we can add weights on TPs, FPs, FNs values during the training. For example, if the FNs weight is set to 10, it means that the FNs will be considered as ten times more than it really is, in other words the termination rules will overestimate the termination condition. Thus the score of termination rules is getting decreased. With this way we focus more in quality than quantity of termination rules. In Table 15, we present the best weights for each class in a experiment with structural and temporal features. The results are presented in Table 16 and Table 17

Table 15 Best weights for each class with structural/temporal features

	TPs-weight	FPs-weight	FNs-weight
Growth	1/1	1/5	15/1
Shrinkage	20/1	1/1	15/1
Continuation	1/1	1/1	1/15
Dissolution	1	1	15

for the experiment with structural features and with the temporal features respectively.

Table 16 Weights on TPs,FPs,FNs - Experiment With Structural Features

	Growth	Shrinkage	Continuation	Dissolution
Micro/Macro Precision	0.2376/0.6055	0.1127/0.5533	0.9247/0.9623	0.8036/0.8878
Micro/Macro Recall	0.3487/0.6519	0.8023/0.8187	0.9845/0.6772	0.2113/0.6047
Micro/Macro Fscore	0.2826/0.6278	0.1977/0.6603	0.9537/0.7950	0.3346/0.7194

Table 17 Weights on TPs,FPs,FNs - Experiment With Temporal Features

	Growth	Shrinkage	Continuation
Micro/Macro Precision	0.2184/0.5913	0.2459/0.6032	0.9182/0.9222
Micro/Macro Recall	0.2043/0.5857	0.1531/0.5654	0.9955/0.6933
Micro/Macro Fscore	0.2111/0.5885	0.1887/0.5837	0.9553/0.7915

While we were trying various values to weights, we noticed in the results that:

- If TPs's weight is increased then TPs is increased, FPs is increased and FNs is decreased because the number of initiations rules is increased.
- If FPs's weight is increased then TPs is decreased, FPs is decreased and FNs is increased because the number of initiations rules is decreased.

- If FNs's weight is increased then TPs is increased, FPs is increased and FNs is decreased because the number of termination rules is decreased.

We tried to increase the low TPs number by setting appropriate weights, but FPs also increased. OLED overestimated the initiation condition because its initiation rules are not specialised enough to detect correctly in which communities an event will occur. This is a strong indication that with the current features OLED performance could not improve. In Appendix A, we present some of the clauses that derived by above experiments.

6 Conclusions

We tried to predict the evolution of communities in a dynamic social network. The evolution of a community is described as the occurrence of *growth*, *shrinkage*, *continuation* and *dissolution* events. We carried out the prediction using OLED, an Inductive Logic Programming system for learning logical theories from data streams. Initially, we tracked the evolution of communities over the time and obtained the ground truth of evolutionary events. As features we used structural characteristics of communities. Moreover, we also tried temporal features where a preset number of previous instances of each communities were used as well as features that capture change between consecutive instances of a community. Subsequently, the features were quantized. The dataset was obtained from the Mathematics Stack Exchange forum. We presented the micro and macro averages, because the classes (*Growth*, *Shrinkage*, *Continuation* and *Dissolution*) were unbalanced.

We also investigated the best pruning values for the theory in OLED, which did not improve the results. Overall, the experiments with the temporal features had a worse performance than the experiment with the structural features, probably because there were not many timeframes. Then, we execute experiments where OLED learnt rules that represent long range relationships between an evolutionary event and features.

Subsequently, weights were applied to TPs, FPs and FNs values to change rules' scores. This was the experiment with the best results. Finally, we presented the features that were the most influential for each evolutionary event.

Future work could be directed to a range of different fields. Others classifiers can be used to predict the evolution of communities (e.g. SVM, Random Forest) and compare them to our results. Additional, evolutionary events can be added such as merge or split. Other types of features (i.e. topics or context of discussions in social networks) could be studied as well as features that capture the dynamics of communities, such as the rate of change of an existing feature. Also, another quantization algorithm, which will adapt better the quantized values to the distribution of the values of the features might help. Because OLED is an online system, it needs many data to decide if a rule is trusted. However, in our dataset we had only 10 frames. Thus, datasets with more timeframes could be examined. Moreover, a different segmentation of the data stream could be tried, to study its effect on the prediction. Finally, it would be very interesting to test the learnt rules in other datasets to notice how relevant they are at the problem of community evolution prediction.

A Appendix

An advantage of OLED is that the predictive model (Theory) it derives is human-readable. Thus the rules can be read, analyzed and interesting results can be derived from them. Some of the best performing rules are shown in Tables 18 and 19. Transferability to new datasets is also an interesting possibility.

Table 18 Rules learnt in the best experiment: Growth and Shrinkage

initiatedAt(*growth*($X0$),$T1$) :- happensAt(density($X0$,1),$T1$), happensAt(diameter($X0$,2),$T1$).
terminatedAt(*growth*($X0$),$T1$) :- happensAt(ratio_cut($X0$,3),$T1$), happensAt(average_path_length($X0$,3),$T1$), happensAt(normalized_edges_number($X0$,5),$T1$).
terminatedAt(*growth*($X0$),$T1$) :- happensAt(ratio_cut($X0$,3),$T1$), happensAt(closeness_centrality($X0$,3),$T1$), happensAt(normalized_edges_number($X0$,5),$T1$).
terminatedAt(*growth*($X0$),$T1$) :- happensAt(cohesion($X0$,2),$T1$), happensAt(average_path_length($X0$,3),$T1$), happensAt(diameter($X0$,2),$T1$).
terminatedAt(*shrinkage*($X0$),$T1$) :- happensAt(ratio_association($X0$,3),$T1$).
terminatedAt(*shrinkage*($X0$),$T1$) :- happensAt(average_path_length($X0$,2),$T1$).
terminatedAt(*shrinkage*($X0$),$T1$) :- happensAt(closeness_centrality($X0$,2),$T1$), happensAt(ratio_cut($X0$,1),$T1$).
initiatedAt(*shrinkage*($X0$),$T1$) :- happensAt(eigenvector_centrality($X0$,1),$T1$), happensAt(ratio_association($X0$,5),$T1$).

Some features appeared more often in rules of specific evolutionary events than others; while some never appeared. In Tables 20 and 21 we present for each evolutionary event (*growth*, *shrinkage*, *continuation*, *dissolution*), the frequency of the structural features the bodies of rules.

In Table 20 it can be noticed that features like *diameter*, *cohesion*, *ratio_cut* and *average_path_length* affect the prediction of the *growth* event since they represent 54.14% of total features which appeared in the rules. On the contrary features like *betweenness_centrality* and *normalized_association* did not appear at all. For the *shrinkage* event the most used features are *ratio_association*, *ratio_cut*, *cohesion* and *clustering_coefficient*, while the *normalized_association* feature does not appear. In Table 21 the features of the *continuation* and the *dissolution* events are presented. The continuation event seems to be affected mostly from *ratio_cut*, *ratio_association* and the *clustering_coefficient* and not by the cohesion. While for the *dissolution* event, every feature is used in prediction, and especially the *clustering_coefficient*, *cohesion* and the *betweenness_centrality*.

In Tables 22 and 23 we present temporal features which appear in the rules of corresponding experiments. For the *growth* event the temporal features: *ancestor4_average_path_length*, *activeness_ancestor_2_ancestor3*, *aging_ancestor0, ancestor1_diameter*, *ancestor3_closeness_centrality* and the rest that are presented in Table 22, are the equally important. *Shrinkage* event prediction uses the values of *ancestor1_event_is_shrinking*, *ancestor4_clustering_coefficient*, *cohesion*, *eigenvector_centrality*, *joinNodesRatio_currentCommunity_ancestor0*,

Table 19 Rules learnt: Survival

terminatedAt(*survival*(*X*0),*T*1) :- happensAt(ratio_association(*X*0,4),*T*1), happensAt(density(*X*0,2),*T*1).
terminatedAt(*survival*(*X*0),*T*1) :- happensAt(ratio_association(*X*0,3),*T*1), happensAt(clustering_coefficient(*X*0,4),*T*1).
terminatedAt(*survival*(*X*0),*T*1) :- happensAt(diameter(*X*0,3),*T*1), happensAt(ratio_cut(*X*0,3),*T*1), happensAt(ratio_association(*X*0,2),*T*1).
terminatedAt(*survival*(*X*0),*T*1) :- happensAt(ratio_association(*X*0,3),*T*1), happensAt(closeness_centrality(*X*0,3),*T*1).
terminatedAt(*survival*(*X*0),*T*1) :- happensAt(clustering_coefficient(*X*0,1),*T*1), happensAt(closeness_centrality(*X*0,5),*T*1), happensAt(cohesion(*X*0,4),*T*1).
terminatedAt(*survival*(*X*0),*T*1) :- happensAt(normalized_edges_number(*X*0,2),*T*1), happensAt(cohesion(*X*0,2),*T*1), happensAt(average_path_length(*X*0,1),*T*1).
terminatedAt(*survival*(*X*0),*T*1) :- happensAt(size(*X*0,1),*T*1), happensAt(betweenness_centrality(*X*0,1),*T*1), happensAt(cohesion(*X*0,3),*T*1).
terminatedAt(*survival*(*X*0),*T*1) :- happensAt(normalized_edges_number(*X*0,1),*T*1), happensAt(cohesion(*X*0,3),*T*1), happensAt(average_path_length(*X*0,1),*T*1).

ratio_cut. Many temporal features are missing from the bodies of rules for both *growth* and *shrinkage* events. For the *continuation* event only *cohesion* and *ratio_cut* were used.

Hoeffding bound The *Hoeffding bound* is a statistical tool that is used as a probabilistic estimator of the generalization error of a model (true expected error on the entire input), given its empirical error (observed error on a training subset). Given a random variable $X \in [0,1]$ and an observed mean $\overline{X}$ of its values after n independent observations, the Hoeffding Bound states that, with probability $1-\delta$, the true mean $\hat{X}$ of the variable lies in an interval $(\overline{X}-\varepsilon, \overline{X}+\varepsilon)$, where $\varepsilon = \sqrt{\frac{ln(1/\delta)}{2n}}$. In other words, the true average can be approximated by the observed one with probability $1-\delta$, given an error margin ε that decreases with the number of observations n.

Table 20 Structural features frequency: Growth and Shrinkage

Growth	Percentage	Shrinkage	Percentage
diameter	17.68%	ratio_association	20%
cohesion	13.26%	ratio_cut	13.55%
ratio_cut	11.60%	cohesion	11.61%
average_path_length	11.60%	clustering_coefficient	10.97%
density	8.29%	eigenvector_centrality	9.03%
ratio_association	7.73%	density	8.39%
clustering_coefficient	7.73%	average_path_length	7.10%
size	6.63%	closeness_centrality	6.45%
closeness_centrality	6.08%	centrality	3.871%
eigenvector_centrality	3.87%	diameter	3.87%
normalized_edges_number	2.76%	betweenness_centrality	2.58%
centrality	2.76%	size	1.29%
		normalized_edges_number	1.29%

Table 21 Structural features frequency: Continuation and Dissolution

Continuation	Percentage	Dissolution	Percentage
ratio_cut	16.07%	clustering_coefficient	25.93%
ratio_association	15%	cohesion	15.74%
clustering_coefficient	10%	betweenness_centrality	10.19%
density	9.64%	diameter	9.26%
diameter	8.21%	size	8.33%
closeness_centrality	8.21%	normalized_edges_number	6.48%
eigenvector_centrality	7.14%	ratio_association	5.56%
centrality	6.79%	closeness_centrality	3.70%
betweenness_centrality	6.43%	average_path_length	3.70%
normalized_association	4.64%	ratio_cut	2.78%
average_path_length	4.64%	normalized_association	2.78%
normalized_edges_number	2.5%	centrality	2.78%
size	0.71%	density	1.85%
		eigenvector_centrality	0.93%

Table 22 Temporal features frequency: Growth

Growth	Percentage
ancestor4_average_path_length	12.5%
activeness_ancestor_2_ancestor3	6.25%
aging_ancestor0	6.25%
ancestor1_diameter	6.25%
ancestor3_closeness_centrality	6.25%
ancestor3_clustering_coefficient	6.25%
ancestor3_diameter	6.25%
ancestor4_centrality	6.25%
ancestor4_clustering_coefficient	6.25%
ancestor4_diameter	6.25%
jaccardCoefficient_ancestor_0_ancestor1	6.25%
jaccardCoefficient_ancestor_2_Ancestor3	6.25%
joinNodesRatio_ancestor_2_ancestor3	6.25%
joinNodesRatio_currentCommunity_ancestor0	6.25%
leftNodesRatio_ancestor_0_ancestor1	6.25%

Table 23 Temporal features frequency: Shrinkage and Continuation

Shrinkage	Percentage
ancestor1_event_is_shrinking	16.66%
ancestor4_clustering_coefficient	16.66%
cohesion	16.66%
eigenvector_centrality	16.66%
joinNodesRatio_currentCommunity_ancestor0	16.66%
ratio_cut	16.66%
Continuation	**Percentage**
cohesion	50%
ratio_cut	50%

References

1. Hendrik Blockeel, Luc De Raedt, Nico Jacobs, and Bart Demoen. Scaling up inductive logic programming by learning from interpretations. *Data Mining and Knowledge Discovery*, 3(1):59–93, Mar 1999. URL: https://doi.org/10.1023/A:1009867806624, doi:10.1023/A:1009867806624.
2. P. Bródka, P. Kazienko, and B. Kołoszczyk. Predicting Group Evolution in the Social Network. *ArXiv e-prints*, October 2012. arXiv:1210.5161.
3. L. De Raedt. *Logical and Relational Learning*. Cognitive Technologies. Springer Berlin Heidelberg, 2008. URL: https://books.google.gr/books?id=FFYIOXvwq7MC.
4. Georgios Diakidis, Despoina Karna, Dimitris Fasarakis-Hilliard, Dimitrios Vogiatzis, and George Paliouras. Predicting the evolution of communities in social networks. In *Proceedings of the 5th International Conference on Web Intelligence, Mining and Semantics*, WIMS '15, pages 1:1–1:6, New York, NY, USA, 2015. ACM. URL: http://doi.acm.org/10.1145/2797115.2797119, doi:10.1145/2797115.2797119.
5. Opher Etzion and Peter Niblett. *Event Processing in Action*. Manning Publications Co., Greenwich, CT, USA, 1st edition, 2010.
6. Santo Fortunato. Community detection in graphs. *CoRR*, abs/0906.0612, 2009.
7. Bogdan Gliwa, Piotr Bródka, Anna Zygmunt, Stanislaw Saganowski, Przemyslaw Kazienko, and Jaroslaw Kozlak. Different approaches to community evolution prediction in blogosphere. In *Advances in Social Networks Analysis and Mining 2013, ASONAM '13, Niagara, ON, Canada - August 25 - 29, 2013*, pages 1291–1298, 2013. URL: http://doi.acm.org/10.1145/2492517.2500231, doi:10.1145/2492517.2500231.
8. Mark Goldberg, Malik Magdon ismail, Srinivas Nambirajan, and James Thompson. Tracking and predicting evolution of social communities. -, -.
9. Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, 1963. doi:10.1080/01621459.1963.10500830.
10. Nagehan İlhan and Şule Gündüz Öğüdücü. Predicting community evolution based on time series modeling. In *Proceedings of the 2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2015*, ASONAM '15, pages 1509–1516, New York, NY, USA, 2015. ACM. URL: http://doi.acm.org/10.1145/2808797.2808913, doi:10.1145/2808797.2808913.
11. Sanjay Ram Kairam, Dan J. Wang, and Jure Leskovec. The life and death of online groups: Predicting group growth and longevity. In *Proceedings of the Fifth ACM International Conference on Web Search and Data Mining*, WSDM '12, pages 673–682, New York, NY, USA, 2012. ACM. URL: http://doi.acm.org/10.1145/2124295.2124374, doi:10.1145/2124295.2124374.
12. Nikos Katzouris, Alexander Artikis, and Georgios Paliouras. Online learning of event definitions. *Theory and Practice of Logic Programming*, 16(5-6):817–833, 2016. doi:10.1017/S1471068416000260.
13. Robert Kowalski and Marek Sergot. *A Logic-Based Calculus of Events*, pages 23–55. Springer Berlin Heidelberg, Berlin, Heidelberg, 1989. URL: https://doi.org/10.1007/978-3-642-83397-7_2, doi:10.1007/978-3-642-83397-7_2.
14. Akshay Patil, Juan Liu, and Jie Gao. Predicting group stability in online social networks. In *Proceedings of the 22Nd International Conference on World Wide Web*, WWW '13, pages 1021–1030, New York, NY, USA, 2013. ACM. URL: http://doi.acm.org/10.1145/2488388.2488477, doi:10.1145/2488388.2488477.
15. Maria Evangelia G. Pavlopoulou, Grigorios Tzortzis, Dimitrios Vogiatzis, and George Paliouras. Predicting the evolution of communities in social networks using structural and temporal features. In *12th International Workshop on Semantic and Social Media*

Adaptation and Personalization, SMAP 2017, Bratislava, Slovakia, July 9-10, 2017, pages 40–45, 2017. URL: https://doi.org/10.1109/SMAP.2017.8022665, doi:10.1109/SMAP.2017.8022665.

16 Mansoureh Takaffoli, Reihaneh Rabbany, and Osmar R. Zaiane. Community evolution prediction in dynamic social networks. doi:10.1109/ASONAM.2014.6921553.

17 R. Zafarani, M.A. Abbasi, and H. Liu. *Social Media Mining: An Introduction*. Cambridge University Press, 2014. URL: https://books.google.gr/books?id=H9FkAwAAQBAJ.