NATIONAL AND KAPODISTRIAN UNIVERSITY OF ATHENS

SCHOOL OF SCIENCES
DEPARTMENT OF INFORMATICS AND TELECOMMUNICATIONS

UNDERGRADUATE THESIS

Predicting the Evolution of Communities with Online Inductive Logic Programming

Georgios N. Athanasopoulos

Supervisors:

Panagiotis Stamatopoulos, Assistant Professor, NKUA
George Paliouras, Director of Research, NCSR «Demokritos»
Dimitrios Vogiatzis, Collaborating Researcher, NCSR «Demokritos»
Grigorios Tzortzis, Associate Researcher, NCSR «Demokritos»
Nikos Katzouris, Associate Researcher, NCSR «Demokritos»

ATHENS

JANUARY 2018

UNDERGRADUATE THESIS

Predicting the Evolution of Communities with Online Inductive Logic Programming

Georgios N. Athanasopoulos
R.N.: 1115201300002

ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Πρόβλεψη Εξέλιξης Κοινοτήτων με Ακολουθιακό Επαγωγικό Λογικό Προγραμματισμό

Γεώργιος Ν. Αθανασόπουλος

Επιβλέποντες:

Παναγιώτης Σταματόπουλος, Επίκουρος Καθηγητής , ΕΚΠΑ
Γεώργιος Παλιούρας, Διευθυντής Ερευνών, ΕΚΕΦΕ «Δημόκριτος»
Δημήτριος Βογιατζής, Συνεργαζόμενος Ερευνητής, ΕΚΕΦΕ «Δημόκριτος»
Γρηγόριος Τζώρτζης, Ερευνητής, Εξωτερικός Συνεργάτης, ΕΚΕΦΕ «Δημόκριτος»
Νίκος Κατζούρης, Ερευνητής, Εξωτερικός Συνεργάτης, ΕΚΕΦΕ «Δημόκριτος»

ΑΘΗΝΑ

ΙΑΝΟΥΑΡΙΟΣ 2018

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Πρόβλεψη Εξέλιξης Κοινοτήτων με Ακολουθιακό Επαγωγικό Λογικό Προγραμματισμό

Γεώργιος Ν. Αθανασόπουλος
Α.Μ.: 1115201300002

Επιβλέποντες:

Παναγιώτης Σταματόπουλος, Επίκουρος Καθηγητής , ΕΚΠΑ
Γεώργιος Παλιούρας, Διευθυντής Ερευνών, ΕΚΕΦΕ «Δημόκριτος»
Δημήτριος Βογιατζής, Συνεργαζόμενος Ερευνητής, ΕΚΕΦΕ «Δημόκριτος»
Γρηγόριος Τζώρτζης, Ερευνητής, Εξωτερικός Συνεργάτης, ΕΚΕΦΕ «Δημόκριτος»
Νίκος Κατζούρης, Ερευνητής, Εξωτερικός Συνεργάτης, ΕΚΕΦΕ «Δημόκριτος»

ABSTRACT

Recently, research on dynamic social network analysis has risen because of numerous data have been available for investigation. Many researchers have started focusing on predicting events that characterize the evolution of communities over time. Community evolution prediction is an interesting field, since it can contribute to prevention of racism, violence and terrorism, revelation new trends and salvage of information that is in danger to be lost.

In this thesis, we try to predict the evolution of communities in a dynamic social network. For this prediction we use OLED, an Online Inductive Logic Programming method. OLED derives a theory which indicates us when an evolutionary event starts and when it stops to happen. We have adopted the following four evolutionary events in our work: Growth, Shrinkage, Continuation and Dissolution. As features we use the structural characteristics (structural features) of each community and the previous states of this community in time (temporal features). The experiments were executed with real life data from Mathematics StackExchange. We tried and evaluated various parameters settings on OLED, while also we permitted the construction two types of rules. These ones that considers only current time and the other which include elements from previous time in their bodies. Finally, we present which features participate in prediction of each evolutionary event and which of them are unrelated with this prediction.

SUBJECT AREA: Social network analysis

KEYWORDS: social network analysis, community evolution prediction, machine learning, inductive logic programming, event calculus, online learning

ΠΕΡΙΛΗΨΗ

Πρόσφατα, η έρευνα στα δυναμικά κοινωνικά δίκτυα έχει αυξηθεί λόγω των πολυάριθμων δεδομένων που έχουν γίνει διαθέσιμα για εξερεύνηση. Πολλοί ερευνητές έχουν αρχίσει να εστιάζουν στη πρόβλεψη γεγονότων τα οποία χαρακτηρίζουν την εξέλιξη των κοινοτήτων στο χρόνο. Η πρόβλεψη της εξέλιξης των κοινοτήτων είναι ένα ενδιαφέρον πεδίο αφού μπορεί να συνεισφέρει στην εμπόδιση του ρατσισμού, της βίας και της τρομοκρατίας, στην αποκάλυψη νέων τάσεων και στην διάδοση πληροφοριών που κινδυνεύουν να χαθούν.

Σε αυτή τη πτυχιακή εργασία προσπαθούμε να προβλέψουμε την εξέλιξη των κοινοτήτων σε ένα δυναμικό κοινωνικό δίκτυο. Για την πρόβλεψη χρησιμοποιούμε το OLED, μια ακολουθιακή και επαγωγική λογικού προγραμματισμού μέθοδο. Το OLED παράγει μια θεωρία η οποία μας δηλώνει πότε ένα εξελικτικό γεγονός αρχίζει και πότε τελειώνει να συμβαίνει. Έχουμε υιοθετήσει τα ακόλουθα εξελικτικά γεγονότα στην εργασία μας: Αύξηση, Συρρίκνωση, Συνέχιση, και Διάλυση. Σαν χαρακτηριστικά χρησιμοποιούμε τα δομικά χαρακτηριστικά κάθε κοινότητας και τις προηγούμενες καταστάσεις αυτής της κοινότητας (χρονικά χαρακτηριστικά). Τα πειράματα εκτελέστηκαν με πραγματικά δεδομένα από το Mathematics StackExchange. Δοκιμάσαμε και αξιολογήσαμε διάφορες ρυθμίσεις παραμέτρων στο OLED, ενώ επίσης επιτρέψαμε την κατασκευή δύο τύπων κανόνων. Αυτούς που λαμβάνουν υπόψιν τους μόνο την τρέχουσα χρονική στιγμή και του άλλους που περιλαμβάνουν στοιχεία από προηγούμενες χρονικές στιγμές στο σώμα τους. Τελικά παρουσιάζουμε ποια χαρακτηριστικά συμμετέχουν στην πρόβλεψη κάθε εξελικτικού γεγονότος και ποια από αυτά είναι άσχετα με αυτή τη πρόβλεψη.

ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ: Ανάλυση Κοινωνικών Δικτύων

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: ανάλυση κοινωνικών δικτύων, πρόβλεψη εξέλιξης κοινοτήτων, μηχανική μάθηση, επαγωγικός λογικός προγραμματισμός, λογισμός πράξης, ακολουθιακή μάθηση

To my parents, Nikos and Sofia.

ACKNOWLEDGEMENTS

I would like to thank all supervisors who work together for this thesis. Specifically, they are Prof. Panagiotis Stamatopoulos, Dr. George Paliouras, Dr. Dimitrios Vogiatzis, Dr. Grigorios Tzortzis and Dr. Nikos Katzouris. All of them, are individuals with huge scientific experience on the field that they research. They are committed and want the work to be done right, that makes them professionals. This characteristic makes them role models for me. Above all, they have the intention to share their knowledge with others, helping them to improve. Through procedure of this thesis, I learned to approach and research better, unknown objects and to reveal new. I'm grateful for guidance, immense support and excellent collaboration that we had.

Athens, JANUARY 2018
Georgios Athanasopoulos

CONTENTS

LIST OF FIGURES

LIST OF TABLES

PREFACE

This thesis took place in Athens of Greece between MARCH 2017 and JANUARY 2018. This work is consisted by three parts, Reading part, Implementation part and Writing part. During the first part, I had to familiarize with related work and terminology on field of social network analysis and dynamic communities' prediction. The implementation phase was the longer one. It was needed to create a software which takes data of dynamic social networks, create appropriate structures and derive a file in a format that OLED can use. This software is created with a general way, so can support every social network. Our main activity, was the selection of features (structural and temporal) so that we get the best results during prediction. For prediction we used OLED (Online Learning of Event Definitions). Because OLED didn't operate exactly as we wanted, we changed some of its functionalities. All above code is written in Java and Python. Finally, this text was written in order to present current work. It's described detailed both theoretical background and experiments that are executed. So, the reader can be navigated smoothly in all procedure of this thesis.

1. INTRODUCTION

1.1 SOCIAL NETWORKS BACKGROUND

A social network is a social structure which contains individuals called nodes, who are connected with other individuals. The link among them states an interaction which has one or more type of interdependency such as friendship, kinship, common interest, financial exchange. For the studding of these networks, they are represented as graphs. Each user is represented by a node in the graph, and the interaction between two nodes by an edge. There are two types of graphs, static and dynamic graphs. The static graph does not change as time passes, unlike with dynamic ones. However, in real life the most of social networks are dynamic and evolve as time passes.

Social network analysis views social relationships in terms of network theory consisting of nodes and ties. Nodes are the individual actors within the networks, and ties are the relationships between the actors. The resulting graph-based structures are often very complex. There can be many kinds of ties between the nodes. Research in many academic fields has shown that social networks operate on many levels, from families up to the level of nations, and play a critical role in determining the way problems are solved, organizations are run, and the degree to which individuals succeed in achieving their goals

There is a community in graph if the nodes of the network can be easily grouped into sets of nodes such that each set of nodes is densely connected internally. In dynamic networks, the communities are influenced over the time by its users' interaction. This influence causes changes in structure of communities. Many researchers, consider that structure of a community contains important information for network evolution. So, it's highly imperative to model the dynamic behavior in social networks and try to predict their evolution.

In this thesis we study the problem of community evolution prediction in dynamic social networks. For managing it, we use supervised learning task, specifically a classification algorithm which try to predict four types of evolutionary events. They are growth, shrinkage, continuation and dissolution of communities. Various features were investigated in order to understand how they influence the results. Among them, are the structural (recent and past) and temporal characteristics of communities.

We work as follows:

- We create the network's graph, then we segment the graph into time frames, and finally we discover communities per time frame.
- We track the communities through the time by assigning their evolutionary labels to them.
- We design features to represent each community.
- We create chains of communities as they evolve in time.

- We create/extract the temporal label for each community, which indicates what the community will do in the next time frame.
- We train the OLED classifier using the labeled data.

1.2 THESIS CONTRIBUTION

Our contribution in this thesis is:

- Prediction of communities' evolution. Four evolutionary events (growth, shrinkage, continuation and dissolution) will be used.
- Communities' characteristics are represented by structural and temporal features.
- OLED will be used as supervised learning algorithm. This method gives rules that state the conditions for initiation and termination of the above events.
- Long range temporal rules are extracted from OLED, to evaluate how the features of previous time effects on results.
- Real-world datasets are used, such as Mathematics StackExchange.

1.3 THESIS ORGANIZATION

The structure of this text is organized as follows:

- In Chapter 1, we describe in brief the background of social network analysis.
- In Chapter 2, we refer the related work to social network analysis.
- In Chapter 3, we analyze our predicting model, OLED and refer its fundamentals.
- In Chapter 4, we present our methodology, which was followed in order to be work completed.
- In Chapter 5, we present the experiments and results of this thesis.
- In Chapter 6, we discuss some conclusions and what can be done as future work.

2. RELATED WORK

In recent years, there is an interest in the analysis of communities [43], [9], [34], [24] in dynamic social networks. Main research has been on the fields of community detection [44], [2], [32], [15], community tracking [40], [8], [19], [18], and community evolution prediction [26], [17], [4].

In bibliography, the social networks are represented as a graph which is consisted of nodes and edges. Each node is an individual who exists in network. The individuals of network interact with each other. This interaction is represented by an edge. For example, if user v_i interacts with user v_j, then an edge e(i, j) is imported in graph. The graph can be either directed or undirected, depending on whether we need to know who caused the interaction.

One common procedure in the study of the evolution of communities is: 1) Community Detection per time frame and 2) Community Tracking over time. Community Detection is the procedure in which a group of vertices can be naturally grouped into (potentially overlapping) sets such that each set of vertices is densely connected internally. For example, C_i is the i-th community of the graph whose nodes are more densely interconnected relatively to the rest of the network. On the other hand, Community Tracking tries to identify connections between communities in different, usually sequential, timeframes. Communities with many connections among them, can be considered as a snapshot of each other over the time. Noticing the snapshots of communities, we can export the information of how a community of social network is evolving over time. This information could be used as ground truth in community evolution prediction.

At Following, we refer related work has been done on community evolution prediction.

2.1 PREPROCESSING

As first step for community evolution prediction, we should detect communities. A lot of work has been done on this area. Zafarani et al. [45] refers to Modularity Maximization. Modularity is a metric that quantifies the quality of an assignment of nodes to communities by evaluating how much more densely connected the nodes within a community are compared to how connected they would be, on average, in a suitably defined random network. The real communities structure is very different than random structure. The edges in a random structure are less because of randomness. The Modularity Maximization algorithm tries to maximize modularity to find communities far from random structure. Nascimento et al. [31] proposed a Greedy Randomized Adaptive Search Procedure (GRASP) with path relinking, for solving the modularity maximization problem in weighted graphs. GRASP is a randomized multistart local search algorithm which has been applied to a plethora of combinatorial optimization algorithms with favorable computational results. For overlapping structures of communities, there is the Clique Percolation [1], [16], which correspond to complete (fully connected) subgraphs of k nodes. Two k-cliques are considered adjacent if they share k−1 nodes and then belong in same community. This definition of a community permits overlaps between communities. Tong et al. [41] proposed the family of Colibri methods which find low-rank approximations of the adjacency matrix of a graph. The low-rank approximations can be used to find communities. As unsupervised learning, clustering can be used for community detection. Chakrabarti et al. [10] proposed a framework for clustering data points which gradually change over time based on k-means clustering and agglomerative clustering. Chi et al. [11] proposed two frameworks that incorporate temporal smoothness in evolutionary spectral clustering. Tsironis et al. [42] investigated a variant of the spectral clustering which can be efficiently parallelized in MapReduce. MapReduce is a framework for processing parallelizable problems across large datasets using a large number of computers, collectively referred to as a cluster. They studied its effectiveness in finding communities on large-scale social networks. Their evaluation on both real and synthetic large-scale social networks showed promising results. Louvain Method [6], which is a heuristic method that is based on modularity optimization. It is shown to outperform all other known community detection methods in terms of computation time. Also, the quality of the communities detected is very good, as measured by modularity. Pujol et al. [36] used the Louvain Method over a Twitter dataset with 2.4 million nodes and 38 million links. Greene et al. [19] used the Louvain Method in a Mobile Phone Network with 4 million nodes and 100 million links.

The procedure of identifying two communities that are in different timeframes, is called community tracking. The two communities are considered as the same community which evolved over time. This makes possible to observe whether communities grow, shrink, continue, or dissolve with time. Thus, ground truth for prediction is extracted. Much work has been done on community tracking. Specifically, Palla et al. [33] proposed clique percolation method (CPM) that allows to investigate the time dependence of overlapping communities on a large scale and as such, to uncover basic relationships characterising community evolution. Some of these relationships are the: growth, contraction, merge, split, birth and death. Asur et al. [3] developed a framework for capturing and identifying interesting events from non-overlapping snapshots of interaction graphs. Takaffoli et al. [39] presented Modec, a framework for modeling community evolution in social networks by tracking of events (formation, survival, splitting, merging and dissolution). Tracking is done by a similarity metric. The most similar communities between two timeframes are considered as same. The assigning of events is done by a set of rules which determine when formation, survival, splitting, merging or dissolution is happening. Particularity of Modec is the detection of evolutionary events between non-consecutive timeframes. GED (Group Evolution Discovery) [7], [8] reveals 7 events: continuing, shrinking, growing, splitting, merging, dissolving and forming. This method is based on a measure called inclusion, which allows to evaluate the inclusion of one community in another. The higher the inclusion between two communities in different timeframes, the most probable that one is the evolution of the other. The execution is as follows: For each pair of communities <C1, C2> in consecutive timeframes Fi and Fi+1 inclusion I(C1, C2) of C1 in C2 and I(C2, C1) of C2 in C1 is computed. Based on inclusions I(C1, C2) and I(C2, C1) and size, the communities are matched and evolutionary events are detected using the rules below:

Evolutionary Event	Conditions
Growing:	I(C1, C2) ≥ α and I(C2, C1) ≥ β and |C1| < |C2| OR I(C1, C2) ≥ α and I(C2, C1) < β and |C1| ≤ |C2| OR I(C1, C2) < α and I(C2, C1) ≥ β and |C1| ≤ |C2| and there is only one match between C1 and communities in the next timeframe Fi+1
Shrinking:	I(C1, C2) ≥ α and I(C2, C1) ≥ β and |C1| > |C2| OR I(C1, C2) < α and I(C2, C1) ≥ β and |C1| ≥ |C2| OR I(C1, C2) ≥ α and I(C2, C1) < β and |C1| ≥ |C2| and there is only one match between C2 and communities in the previous timeframe Fi
Continuing:	I(C1, C2) ≥ α and I(C2, C1) ≥ β and |C1| = |C2|
Dissolving:	For C1 in Fi and each community C2 in Fi+1, I(C1, C2) < 10% and I(C2, C1) < 10%

Constants α and β, α, β $\in$ [0, 1] are the GED method parameters, which can be used to adjust the sensitivity of the method in identifying particular events.

2.2 COMMUNITY EVOLUTION PREDICTION

Two of the most interesting problems in social network prediction are: the prediction of new interactions among network's members (link prediction) and the prediction of possible events at the level of communities (community evolution prediction) such as growing, shrinking and merging with another community that a community might encounter during its lifetime.

The problem of new interactions prediction is known as link prediction. Its aim is: Given the links in a social network at time t or during a time interval $T_{interval}$, to predict the links that will be added to the network during the later time interval from time $t+i$ where $i > 0$ to a given future time. For prediction different network and group measures are used to determine which unconnected nodes are 'close together' in the topology of the network. Liben-Nowell et al. [29] assign a connection weight score focused on paths and common neighbours pairs of nodes, and then produce a ranked list in decreasing order of score values. Nodes with highest score are predicted to link in the future. Zheleva et al. [46] show that when there are tightly-knit family circles in a social network, we can improve the accuracy of link prediction models. Lichtenwalter et al. [30] focuses on other elements of data such as network observational period, variance reduction, topological causes and degrees of imbalance. Another approach to link prediction is sign prediction. In this case, apart from link prediction, we try to predict if the link we predicted has positive or negative semantic. Symeonidis et al. [37] defined a basic node similarity measure that captures effectively graph features. Thus, they derived a method that apply in signed networks. Kunegis et al. [27] considered signed variants of global network characteristics such as the clustering coefficient, node-level characteristics such as centrality and popularity measures, and link-level characteristics such as distances and similarity measures to predict the sign of the links. Leskovec et al. [28] provided insight into some of the fundamental principles that drive the formation of signed links in networks, shedding light on theories of balance and status from social psychology.

The research in community evolution prediction and the events have been proposed, are extended. In particular, Patil et al. [35] predicted whether a community will disappear or will survive in the future. They observed that both the level of member diversity and social activities are critical in maintaining the stability of groups. They also found that certain 'prolific' members play a more important role in maintaining the group stability. Goldberg et al. [18] correlated the lifespan of a community with structural parameters of its early time of evolution. Brodka et al. [17] [7] tried to predict 6 evolutionary events of communities (grow, shrink, continue, dissolve, merge, split). The features they used are the history of events of the community in the three preceding timeframes, and the community size in these timeframes. They found that the prediction based on the simple input features may be very accurate, some classifiers are more precise than the others. Kairam et al. [22] tried to understand the factors contributing to the growth and longevity of in a social network. They investigated the role that two types of growth (Diffusion growth and non-diffusion growth) play during a group's formative stages from the perspectives of community. Diffusion growth is when a community attracts new members through ties to existing members. Non-diffusion growth is individuals with no prior ties who become members themselves. Diakidis et al. [13] 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. [38] quantified the changes that may occur for a community as follows: survive{true, false}, merge{true, false}, split{true, false}, size{expand,shrink}, and cohesion{cohesive, loose}. All these events and transitions are binary. Since size and cohesion transitions only defined for a survival community, they proposed a technique to detect these two transitions. First, the survival is predicted, then the detection of these transitions is followed. These response variables are not mutually exclusive and may occur together at the same time, where different features may trigger them. Hence, separate models are learned to predict each of them. Ilhan et al. [21] proposed a time series ARIMA model to predict how particular community features will change in the following time. Distinct time windows are examined in constituting and analyzing time series. Furthermore, community feature values are forecasted with an acceptable error rate. Event prediction using forecasted feature values substantially match up with actual events.

3. OLED BACKGROUND

In this thesis, OLED (Online Learning of Event Definitions) [23] was used for community evolution prediction. This chapter will help the reader to be familiar with the system we use and learn its fundamentals. OLED is an online Inductive Logic Programming 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 [14] 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. The Event Calculus [25] is a temporal logic that has been used as a basis in event recognition applications, providing among others, direct connections to machine learning, via Inductive Logic Programming (ILP) [12]. 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 [20] for evaluating clauses on a subset of the input stream, is used. With this approach, significant speed-ups are obtained in training time.

3.1 EVENT CALCULUS

The Event Calculus is a temporal logic for reasoning about events and their effects. Its ontology comprises time points, represented by integers; fluents, i.e. properties which have certain values in time; and events, i.e. occurrences in time that may affect fluents and alter their value. The axioms of the EC incorporate the common sense law of inertia, according to which fluents persist over time, unless they are affected by an event. Specifically, if a high level event is initiated at time T, it will continue happening in the future until a low level will fire a termination rule. Respectively, if a high level event is terminated at time T, it will remain terminated in the future until a low level will fire an initiation rule. The basic predicates are presented in Table 3.1, while the domain-independent axioms are in Table 3.2. Axiom (1) states that a high level event F 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.

Table 3.3 presents an example of input which is required by OLED. It is consisting a narrative and an annotation list. Narratives are the simple 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 events in terms of holdsAt/2, expressing the ground truth for our training set. i.e. holdsAt(growth(c1),2).

Predicate	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 3.1: The basic predicates of the EC

Axioms
holdsAt(F,T +1) ←   initiatedAt(F,T).   (1)

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

Table 3.2: The domain-independent axioms of the EC

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 3.4 shows the theory OLED learned 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 3.3).

Timeframe 1	Timeframe 2
<u>Narrative</u> happensAt(size(c1,3),1). happensAt(density(c1,4),1).	<u>Narrative</u> happensAt(size(c1,5),2). happensAt(density(c1,5),2).
<u>Annotation</u>	<u>Annotation</u> holdsAt(growth(c1),2).

Table 3.3: Input of OLED

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

Table 3.4: Learned Theory by OLED

3.2 INDUCTIVE LOGIC PROGRAMMING

The goal of OLED is to learn a set of domain-specific axioms specifying complex events. It manages this using ILP. ILP is a subfield of machine learning which uses logic programming as a uniform representation for examples, background knowledge and hypotheses.

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. Here is used the Learning from Interpretations (LfI) [5] setting in which each training example is an interpretation. As interpretation we mean the set of narrative and annotation atoms are presented in Table 3.3. Given a set of training interpretations I and some background theory B, which in OLED's case consists of the domain-independent axioms of the EC, the goal in LfI is to find a theory. ILP learners typically employ a separate-and-conquer strategy: clauses that cover subsets of the examples are constructed one by one recursively, until all examples are covered. Each clause is constructed in a top-down fashion, starting from an overly general clause and gradually specializing it by adding literals to its body. The process is guided by a heuristic function G that assesses the quality of each specialization on the entire training set. At each step, the literal (or set of literals) with the optimal G-score is selected and the process continues until certain criteria are met.

3.3 HOEFFDING BOUND

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 with range 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 ε = $\sqrt{ln(1/1-\delta)2n}$. 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. In our EC dialect, 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. Here is how OLED is using Hoeffding bound. Let r be a clause and G a clause evaluation function (which we present in Section 3.4) with range in [0,1]. 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 $1-\delta$, where $\varepsilon = \sqrt{ln(1/1-\delta)2n}$. 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.

3.4 OLED

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 seek for a theory with a good fit in the training data. This implies our theory will have false predictions, so we introduce the notion of True Positive(TP), False Positive(FP) and False Negatives(FN). 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 3.3). 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 $\alpha \notin Mr^r_I$.

Heuristic clause evaluation function G of OLED is the following:

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

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 for a clause r has range 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.

4. OUR METHODOLOGY

In order to predict community evolution using OLED, we need to manipulate the data we have, so that will be transformed in the form which OLED can parse. For this purpose, we created a software system that is called OLED Generator (OG) and can take every social network, analyze it and export a suitable file for OLED. Specifically, OG includes the following operations that are analyzed in detail at next sections:

- Take the social network data segmented into timeframes and detect the communities in each timeframe.
- Track the communities evolution across time using its tags to identify their evolutionary events and construct the ground truth of the data.
- Assign community features
- Quantize community features

OG has been designed with a generic way so that can preprocess every social network. Its components are independent of each other. Thus, it's possible that new components to be imported easily. Besides of OLED exportation file, new file types can be supported to be given as input to other machine learning algorithms.

In section 4.5, we present the steps we followed for community evolution prediction and settings under OLED worked.

4.1 SEGMENTATION INTO TIME FRAMES & COMMUNITY DETECTION

A dynamic social network is timestamped. In order to analyze it, we need to segment its vertices into time frames based the timestamp of each vertex. Two of the approaches which can be used are: First, to set a constant number of time frames and an overlap value between them. Thus, each time frame will have an equal number of nodes and the overlap will allow for a smooth transition between time frames. The other approach is having a specific time duration and an overlap value. In this case, we may have a different number of nodes in each time frame. In the experiments we followed the first approach. OG accept a json file which has already been split into time frames and extract them to structures in order to continue the analysis.

The next step is to detect the communities of dynamic social network. Each timeframe is a graph, where there are vertices and edges. The vertices represent the users of the network and the edges represent the interaction among users who are in the same time frame. During the detection of communities, we try to find subgraphs (communities) which contain densely connected users. There are various algorithms for communities' detection as described in Chapter 2. However, in the current work the community labels are known. In particular, the tags of the original posts effectuated by user, were used to characterize the communities. For example: When the content of a post is related to Linear Algebra, it obtains Linear Algebra tag. The user who created this post and the users who replied to it, belong to a community that has the tag Linear Algebra. Thus, OG recognize communities from its input file, subsequently extract them and finally store them in appropriate structures to manipulate later.

4.2 COMMUNITY TRACKING

After obtaining the community labels, we implement community tracking. Because of the dynamic form of communities, they are evolving across neighbouring time frames. Some of the most usual phenomena that can occur to a community between two neighbouring time frames are:

- New nodes join a community.
- Some nodes may leave.
- The community remains largely intact.
- There is a community C_k in timeframe F_i, but it has dissolved in timeframe F_{i+1} because of all members secession.

Thus, for each community C_k in timeframe F_i, there is a community C_m in timeframe F_j (j>i), which represents the evolution of C_k. To match every community with the corresponding community in next time frames, we implemented a simple community tracking method. Our methodology assumes each community has its own tag(name) that identify the communities over the timeframes. Thus, OG looks for a community with the same tag in every subsequent timeframe of the dataset until the last one. If this community tag is not found in any of the following timeframes of the dataset, then we set the evolutionary event of this community as dissolution. It means that the number of users in a community and their interactions are too reduced to be the community appeared in the timeframe. Thus, we consider that this community no longer exists. On the other hand, if this community tag is found in a following (consecutive or not) timeframe of the dataset, then we have found a matching community that is the evolution of previous community over the time. The evolutionary event of this community is set as follows:

Let us suppose we aim to set the evolutionary event of community C_i at timeframe F_j and we have found the first ancestor of community C_i which is the community C_w at timeframe F_z (where $z > j$). Also, we introduce the $eventthreshold$ that denotes the community members difference between two communities. We represent the number of community members of C_i as $|C_i|$ and C_w as $|C_w|$. The evolutionary event of community C_i is set as described below:

- If $|C_i|$−$|C_w| \geq eventthreshold$, then we set the evolutionary event of community C_i as $shrinkage$.
- Else if $|C_w|$−$|C_i| \geq eventthreshold$, then we set the evolutionary event of community C_i as $growth$.
- Else, we set the evolutionary event of community C_i as $continuation$.

The evolutionary events are the ground truth of the dataset, for the supervised learning classification problem. If a community C_i is tracked at a time frame that is not consecutive, then we assume that C_i and its ground truth persist in all intermediate timeframes. Thus, the intermediate timeframes obtain C_i's ground truth.

4.3 COMMUNITY FEATURES

Many features have been proposed to predict the evolution of dynamic communities in social networks. In this section, we present the features we chose in our experiments which are of two kinds, the structural and the temporal. Structural features represent the physical characteristics of a community such as size, density etc. The temporal features include structural features and evolutionary events that derived from the states of a dynamic community in the past, and relations between properties of a dynamic community and properties of its previous instances in time.

Let $C_{t_i}^{k_i}$ be a community k_i at timeframe t_i. If $C_{t_j}^{k_j}$ is a state of $C_{t_i}^{k_i}$ in the past and $C_{t_j}^{k_j}$ is an ancestor of $C_{t_i}^{k_i}$. The ancestors of a community don't need to belong to consecutive time frames. In the current experiments, each community is tracked in each timeframe until its dissolution, in which there are situations in which a community disappears at timeframe ti but it reappears again at timeframe tj, 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.

First let us introduce some notation,

- $G_t = (V_t, E_t)$ denotes the graph at time frame F_t
- V_t is the vertex set of G_t graph
- E_t is the edge set of G_t graph
- $n(F_t) = |V_t|$ is the size of set V_t
- $m(F_t) = |E_t|$ is the size of set E_t
- C_t is the set of communities at time frame F_t
- C_t^k is the community k of set C_t
- $G_t^k = (V_t^k, E_t^k)$, is a subgraph of G_t which represent community C_t^k
- $V_t^k = \{v_{t1}^k, v_{t2}^k, ..., v_{tn(C_t^k)}^k\}, n(C_t^k) = |V_t^k|$
- $E_t^k = \{(u,v) \in E_t : u, v \in V_t^k\}, m(C_t^k) = |E_t^k|$
- $M = \{C_{t_1}^{k_1}, ..., C_{t_p}^{k_p}, ..., C_{t_m}^{k_m}\}$ is a sequence of instances of a community over time as tracking extracted them. Where:
  - $1 \leq t_1 < t_2 < ... < t_m \leq T$
  - $\forall t_i, t_1 < t_i \leq t_m, \exists t_j < t_i, 1 \leq k_j \leq n_{t_j}, j = 1, ..., m$

- $C_{t_j}^{k_j}$ is the ancestor of $C_{t_i}^{k_i}$, where $j < i$

Below are the features' formulas we used:

4.3.1 STRUCTURAL FEATURES

1. **Size**: The normalized value for the size of a community C_t^k in time frame F_t is formulated as:

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

   where $n(C_t^k)$ is the number of vertices in community C_t^k in frame t and $n(F_t)$ is the number of vertices present in time frame t.

2. **Density**: The density of a community C_t^k is defined as

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

   where the numerator $m(C_t^k)$ is the total number of intra community edges (edges connecting nodes belonging to the community) and the denominator is the maximum number of edges this community could have.

3. **Cohesion**: The Cohesion is defined as

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

   where $m(C_t^k)$ is the number of intra community connections, $m_{out}(C_t^k) = \{(u,v) \in E_t : u \in V_t^k, v \notin V_t^k\}$ is the number of inter community connections, $n(C_t^k)$ is the number of vertices in community C_t^k in frame t and $n(F_t)$ is the number of vertices present in time frame t.

4. **Normalized Association**: The NormalizedAssociation is defined as

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

   where $m(C_t^k)$ is the number of intra community connections and $m_{out}(C_t^k)$ is the number of inter community connections. The denominator equals the sum of the vertex degrees belonging to community C_t^k.

5. **Ratio Association**: The RatioAssociation is defined as

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

   where $m(C_t^k)$ is the number of intra community connections, and $n(C_t^k)$ is the number of vertices in community C_t^k in time frame t.

6. **Ratio Cut**: The RatioCut is defined as

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

   where $m_{out}(C_t^k)$ is the number of inter community connections , and $n(C_t^k)$ is the number of vertices in community C_t^k in time frame t.

7. **Normalized Edges Number**: The NormalizedEdgesNumber is defined as

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

   where $m(C_t^k)$ is the number of intra community connections, and $m(F_t)$ is the number of edges present in time frame t.

8. **Average Path Length**: The AveragePathLength is defined as

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

   where $dist(v,u)$ indicates the shortest distance between vertices v and u and $n(C_t^k)$is the number of vertices in community C_t^k in time frame t. We assume that $dist(v,u)$ is zero if you can't be reached from v. Average Path Length refers to the minimum number of edges two vertices are away from each other, between all pairs of the vertices in a community C_t^k. Hence, it shows how close on average two random vertices are to each other.

9. **Diameter**: The Diameter is defined as

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

   where (u,v) indicates the shortest distance between vertices u and v in C_t^k. Diameter is the maximum shortest path between all pairs of vertices in a community C_t^k and is a measure that shows the upper bound of how far vertices are in that community.

10. **Clustering Coefficient**: Clustering coefficient for a vertex v in a community C$_t^k isdefinedas$

$$ClusteringCoefficient(v) = \frac{2neighE(v)}{neigh(v)(neigh(v)-1)}$$

where $neigh(v) = |\{u : (u,v) \in E_t^k\}|$ is the number of neighbours of vertex v and $neighE(v) = |\{(u,w) \in E_t^k : (u,v) \in E_t^k, (w,v) \in E_t^k\}|$ is the number of edges among the neighbours of vertex v. If the neighbours of a vertex are forming a clique, the clustering coefficient of that vertex would be 1, and if they are not connected at all it is 0. The clustering coefficient of a vertex shows how well neighbours of this vertex are connected to each other. The clustering coefficient of a community C_t^k is the average over all its members:

$$ClusteringCoefficient(C_t^k) = \frac{\sum_{v \in V_t^k} ClusteringCoefficient(v)}{n(C_t^k)}$$

where $n(C_t^k)$ is the number of vertices in community C_t^k in time frame t.

11. **Closeness Centrality**: In complex networks, such as social networks, we are interested in ranking vertices and finding important vertices. Centrality measures have been introduced to help us calculate how much a vertex is central (i.e. centre of importance) according to some criteria.

    The Closeness Centrality for a vertex v in a community C_t^k is computed as

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

where $dist(v,u)$ indicates the distance between vertices v and u in C_t^k and $n(C_t^k)$ is the number of vertices in community C_t^k in time frame t. Closeness centrality shows how much a vertex is close to other vertices in the community graph. It counts the average number of hops a vertex is away from the rest of the graph. Thus if the average distances of a specific vertex with the rest of the graph is small, the closeness centrality of that vertex is high and vice versa. Hence, closeness centrality has an inverse relation with distance in the community graph. The Closeness Centrality of a community C_t^k is the average over all its members:

$$ClosenessCentrality(C_t^k) = \frac{\sum_{v \in V_t^k} ClosenessCentrality(v)}{n(C_t^k)}$$

where $n(C_t^k)$ is the number of vertices in community C_t^k in time frame t.

12. **Betweenness Centrality**: Betweenness Centrality for a vertex v in a community C_t^k is computed as

$$BetweennessCentrality(v) = \frac{\sum_{u,w \in C_t^k, u \neq w \neq v} buw(v)}{(n(C_t^k)-1)(n(C_t^k)-2)}$$

where $buw(v)$ is the probability of v to be on the shortest path between u and w, defined as the fraction of shortest paths between u and w that pass through v, and $n(C_t^k)$ is the number of vertices in community C_t^k in time frame t. Betweenness centrality of a vertex shows the importance of this vertex in controlling the communication between other pairs of vertices in a community graph. The Betweenness Centrality of a community C_t^k is the average over all its members:

$$BetweennessCentrality(C_t^k) = \frac{\sum_{v \in V_t^k} BetweennessCentrality(v)}{n(C_t^k)}$$

where $n(C_t^k)$ is the number of vertices in community C_t^k in time frame t.

13. **EigenvectorCentrality**: Eigenvector centrality is based on the idea that a vertex is more central if it is connected to central vertices. Therefore, in addition to neighbours of a vertex, their centrality value is taken into account. Formulating this concept forms a well-known eigenvalue, eigenvector equation. For a community graph G_t^k with $n(C_t^k)$ vertices let $A \in \{0,1\}^{n(C_t^k)\times n(C_t^k)}$ be the adjacency matrix of the community graph, i.e. $a_u = 1$ if vertex $u_{t_i}^k$ is linked to vertex $v_{t_j}^k$, and $a_u = 0$ otherwise. The eigenvector centrality score of vertex $u_{t_i}^k$ can be defined as

$$xi = \frac{\sum_{j=1}^{n(C_t^k)} a_{ij} x_j}{\lambda}$$

where λ is a constant. The above can be written using matrix notation as A$x = \lambda x$, giving rise to an eigenvalue problem. The eigenvector corresponding to the largest eigenvalue contains the eigenvector centrality scores of the community vertices. The Eigenvector Centrality of a community C_t^k is the average over all its members:

$$EigenvectorCentrality(C_t^k) = \frac{\sum_{v \in V_t^k} EigenvectorCentrality(v)}{n(C_t^k)}$$

where $n(C_t^k)$ is the number of vertices in community C_t^k in time frame t.

4.3.2 TEMPORAL FEATURES

1. **STRUCTURAL FEATURES AND EVOLUTIONARY EVENTS OF FIRST N ANCESTORS**:

   Let us assume we want to compute the temporal features of community $C_{t_p}^{k_p}$, which belongs to time frame t_p and is part of dynamic community M. 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 $C_{t_p}^{k_p}$. In order to represent the ancestors' evolutionary events as temporal features , 1-of-K coding scheme is used. In our case, where the evolution events we try to predict are the continuation, shrinking, growth and dissolution, $K = 4$. If an ancestor community has evolutionary event equal to growth, its evolutionary event as a temporal feature of its descendant is represented as a vector of length $K = 4$, $e = (0, 0, 1, 0)^T$.

Another group of temporal features concerns pairs of communities formed as: Let us assume we want to compute the temporal features of community $C_{t_p}^{k_p}$, which belongs to time frame t_p and is part of dynamic community M. Having the first n immediate ancestors of community $C_{t_p}^{k_p}$ we form the following pairs of communities:

- $C_{t_p}^{k_p}$ and first ancestor in time of $C_{t_p}^{k_p}$
- first ancestor in time of $C_{t_p}^{k_p}$ and second ancestor in time of $C_{t_p}^{k_p}$
- second ancestor in time of $C_{t_p}^{k_p}$ and third ancestor in time of $C_{t_p}^{k_p}$
- ...
- $(n-1)th$ ancestor in time of $C_{t_p}^{k_p}$ and $(n)th$ ancestor in time of $C_{t_p}^{k_p}$

Using the pairs of communities as described above for a given number of ancestors n to use, we compute the following temporal features:

2. **JACCARD SIMILARITY COEFFICIENT OF COMMUNITIES' SET OF VERTICES**:

   Given a pair of communities $C_{t_i}^{k_i}$ and $C_{t_{i-1}}^{k_{i-1}}$, where $C_{t_{i-1}}^{k_{i-1}}$ is the ancestor of $C_{t_i}^{k_i}$, the jaccard coefficient of the set of vertices is defined as

$$JaccCoeff_{Vertices}(C_{t_i}^{k_i}, C_{t_{i-1}}^{k_{i-1}}) = \frac{|V_{t_i}^{k_i} \cap VV_{t_{i-1}}^{k_{i-1}}|}{|V_{t_i}^{k_i} \cup VV_{t_{i-1}}^{k_{i-1}}|}$$

   where $V_{t_i}^{k_i}$ is the set of vertices of community $C_{t_i}^{k_i}$ and $V_{t_{i-1}}^{k_{i-1}}$ is the set of vertices of community $C_{t_{i-1}}^{k_{i-1}}$.

3. **JACCARD SIMILARITY COEFFICIENT OF COMMUNITIES' SET OF EDGES**:

   Given a pair of communities $C_{t_i}^{k_i}$ and $C_{t_{i-1}}^{k_{i-1}}$, where $C_{t_{i-1}}^{k_{i-1}}$ is the ancestor of $C_{t_i}^{k_i}$, the jaccard coefficient of the set of edges is defined as

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

   where $E_{t_i}^{k_i}$ is the set of edges of community $C_{t_i}^{k_i}$ and $E_{t_{i-1}}^{k_{i-1}}$ is the set of edges of community $C_{t_{i-1}}^{k_{i-1}}$.

4. **JACCARD SIMILARITY COEFFICIENT OF COMMUNITIES' SET OF VERTICES & EDGES**:

   Given a pair of communities $C_{t_i}^{k_i}$ and $C_{t_{i-1}}^{k_{i-1}}$, where $C_{t_{i-1}}^{k_{i-1}}$ is the ancestor of $C_{t_i}^{k_i}$, the jaccard coefficient of the set of vertices and edges is defined as

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

where $V_{t_i}^{k_i}$ is the set of vertices of community $C_{t_i}^{k_i}$, $V_{t_{i\text{-}1}}^{k_{i\text{-}1}}$ is the set of vertices of community $C_{t_{i\text{-}1}}^{k_{i\text{-}1}}$, $E_{t_i}^{k_i}$ is the set of edges of community $C_{t_i}^{k_i}$ and $E_{t_{i\text{-}1}}^{k_{i\text{-}1}}$ is the set of edges of community $C_{t_{i\text{-}1}}^{k_{i\text{-}1}}$.

5. **JOIN NODES RATIO**:

   Given a pair of communities $C_{t_i}^{k_i}$ and $C_{t_{i\text{-}1}}^{k_{i\text{-}1}}$, where $C_{t_{i\text{-}1}}^{k_{i\text{-}1}}$ is the ancestor of $C_{t_i}^{k_i}$, the Join Nodes Ratio, is defined as

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

   where $V_{t_i}^{k_i}$ is the set of vertices of community $C_{t_i}^{k_i}$ and $V_{t_{i\text{-}1}}^{k_{i\text{-}1}}$ is the set of vertices of community $C_{t_{i\text{-}1}}^{k_{i\text{-}1}}$. Join Nodes Ratio describes the percentage of new nodes joining the dynamic community as it evolves from time frame $t_{i\text{-}1}$ to time frame t_i.

6. **LEFT NODES RATIO**:

   Given a pair of communities $C_{t_i}^{k_i}$ and $C_{t_{i-1}}^{k_{i\text{-}1}}$, where $C_{t_{i-1}}^{k_{i\text{-}1}}$ is the ancestor of $C_{t_i}^{k_i}$, the Left Nodes Ratio is defined as

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

   where $V_{t_i}^{k_i}$ is the set of vertices of community $C_{t_i}^{k_i}$ and $V_{t_{i-1}}^{k_{i\text{-}1}}$ is the set of vertices of community $C_{t_{i-1}}^{k_{i\text{-}1}}$. Left Nodes Ratio describes the percentage of nodes leaving the dynamic community as it evolves from time frame $t_{i\text{-}1}$ to time frame t_i .

7. **ACTIVENESS**:

   Given a pair of communities $C_{t_i}^{k_i}$ and $C_{t_{i\text{-}1}}^{k_{i\text{-}1}}$, where $C_{t_{i\text{-}1}}^{k_{i\text{-}1}}$ is the ancestor of $C_{t_i}^{k_i}$, the Activeness is defined as

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

   where $V_{t_i}^{k_i}$ is the set of vertices of community $C_{t_i}^{k_i}$, $E_{t_i}^{k_i}$ is the set of edges of community $C_{t_i}^{k_i}$ and $E_{t_{i\text{-}1}}^{k_{i\text{-}1}}$ is the set of edges of community $C_{t_{i\text{-}1}}^{k_{i\text{-}1}}$. Activeness is the ratio of the total number of connections in current community $C_{t_i}^{k_i}$ which also existed in its ancestor community $C_{t_{i\text{-}1}}^{k_{i\text{-}1}}$, to the number of vertices in current community $C_{t_i}^{k_i}$. When there are many connections in current community $C_{t_i}^{k_i}$ which also existed in its ancestor community $C^{k_{i-1}}_{t_{i-1}}$, the value of the activeness for this pair of communities is also high. On the contrary, when the current community $C^{k_i}_{t_i}$ has many new connections which did not exist in its ancestor $C^{k_{i-1}}_{t_{i-1}}$, the value of the activeness for this pair of communities is low.

The last two temporal features are computed for individual communities instead of pairs. Let us assume we want to compute the temporal features of community $C^{k_p}_{t_p}$, which belongs to time frame t_p and is part of dynamic community M. Having the first n immediate ancestors of community $C^{k_p}_{t_p}$ we compute the following temporal features for the current community $C^{k_p}_{t_p}$ and all its first n immediate ancestors. The last two temporal features are described above:

8. **LIFESPAN**:

   Given a community $C^{k_w}_{t_w}$ which is part of dynamic community M and belongs to time frame t_w, the lifeSpan is defined as

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

   LifeSpan is the ratio of the number of time frames between the current community $C^{k_w}_{t_w}$ and the very first instance of the same dynamic community (total number of ancestors of $C^{k_w}_{t_w}$) , to the maximum number of ancestors $C^{k_w}_{t_w}$ could have. The maximum number of ancestors $C^{k_w}_{t_w}$ could have is equal to $t_w - 1$, where t_w is the number of the time frame where $C^{k_w}_{t_w}$ belongs to. In that case there would be an instance of dynamic community M in every time frame from the very first one until time frame $t_w - 1$.

9. **AGING**:

   Given a community $C^{k_w}_{t_w}$ which is part of dynamic community M and belongs to time frame t_w, the Aging is defined as: Age values are assigned to every vertex in $V^{k_w}_{t_w}$ according to the formula above: For every vertex $v \in V^{k_w}_{t_w}$:

   $$AgeValue(v) = |\{C^{k_p}_{t_p} \in M : p \leq w, v \in V^{k_p}_{t_p}\}|$$

   Age values start from zero. Every vertex in $V^{k_w}_{t_w}$ is looked for in all communities of dynamic community M starting from the current community $C^{k_w}_{t_w}$ until its first ancestor. Every time a vertex of set $V^{k_w}_{t_w}$ is found as a member of a community $C^{k_p}_{t_p}$ as shown above, its age value is increased by 1. After computing the Age values for every vertex in $V^{k_w}_{t_w}$, the Age values are divided by the maximum value of age a vertex in $V^{k_w}_{t_w}$ could have. This maximum value is equal to $|\{C^{k_p}_{t_p} \in M : p < w\}| + 1$ (total number of ancestors of $C^{k_w}_{t_w}$ plus 1). In that case a vertex would be present in current community $C_{t_w}^{k_w}$ and all its ancestors. Therefore Aging of a community $C_{t_w}^{k_w}$ is given by the following equation:

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

where $AgeValue(v)$ is the age value of a vertex v in $V_{t_w}^{k_w}$ as described above and $n(C_{t_w}^{k_w})$ is the number of vertices in community $C_{t_w}^{k_w}$ in time frame t_w.

4.4 FEATURES QUANTIZATION

Our prediction method OLED, uses Inductive Logic Programming so given background knowledge and a set of examples, the system will derive hypothesised logic rules. In background knowledge, we define variables to recognize some values such as features' values inside of examples. Because values of variables in logic programming are taken from a finite domain, we should quantize the features' values. We implemented two methods in OG 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. For example, in Figure 4.1 there are four feature's values: 1,60,80,103. If q_{value} is 2 then fist interval will be (1,51) and the second (52,103). Thus the value 1 is quantized to 1, while values 60,80,103 is quantized to 2.

The second method sorts feature's values in a list and creates q_{value} sets. Taking one by one the values from sorted list, begin to fill the q_{value} sets with consecutive values until each set has $\frac{|f_v|}{q_{value}}$ feature's values. In Figure 4.1, if q_{value} is 2 then first set will contain the values 1,60 and second one the values 80,103. So the values 1,60 will be quantized to 1, and values 60,80,103 to 2. The difference between two methods above is that the first creates intervals with constant width but not constant number of elements, while the second creates sets with constant size but not constant length of intervals. Finally, if at least one feature or tag of community C_k is missing then OG delete the C_k to keep the program correctness.

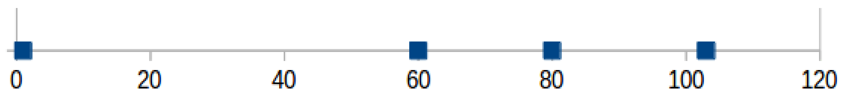

Figure 4.1: Feature's Values Example

4.5 COMMUNITIY EVOLUTION PREDICTION

OG extracted the ground truth for the evolutionary events of the communities, and represented each community as a series of a features. Next, we used OLED that is an online Inductive Logic Programming system for learning logical theories from data streams. We tried to predict four types of evolutionary events: growth, shrinkage, continuation, dissolution. In Figure 4.2 we present the architecture of our prediction system. Initially, training set together with background knowledge and modes declarations, is imported into OLED's training procedure. Subsequently, OLED deduces a logical theory from examples it reads and applies the rules of this theory to a testing set. Finally, for each example of testing set, a label is given. That states whether current example's community is going to grow, shrink, continue as it is or dissolve at the next timeframe. Note that OLED handles two-class problems, so it predicts if a community will begin or stop to exist at next timeframe. Furthermore, OLED estimates its performance using cross-validation technique. Because for our experiments we use Time Series Cross Validation method, OLED's technique isn't appropriate for us. Thus, we changed some of its core's functionalities to support Time Series Cross Validation method. For our experiments, we execute each entity of Figure 4.2 separately. To automate the evaluation procedure of our system, we created a script.

Below every component of Figure 4.2 is discussed detailed.

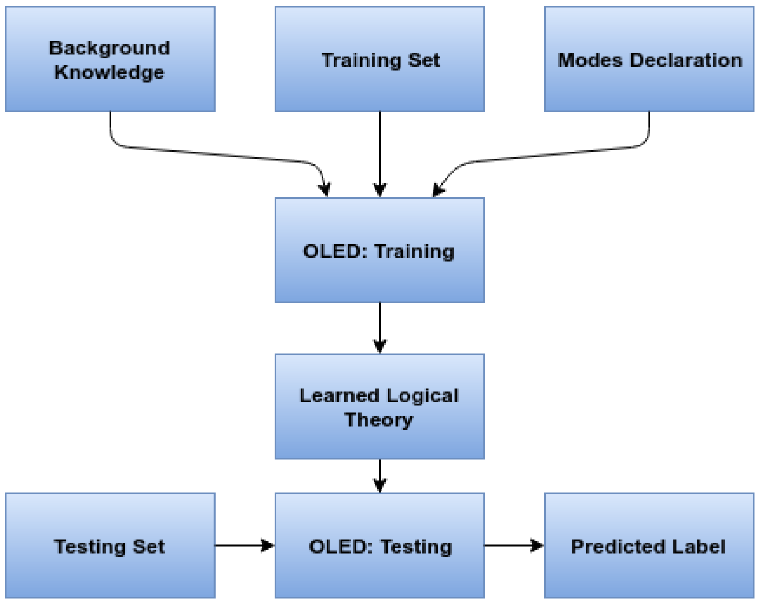

Figure 4.2: Learning Architecture

The background knowledge comes from Inductive Logic Programming field and is provided as a logic theory and its use is to construct explanations for data. Its main target is to take data without any semantics and transform them to objects that have specific meaning in the domain of the problem we study. The system with the background knowledge tries to recognize some patterns in the data in order to derive new facts. Every clause of the theory has the following form: $H : -B_1, ..., B_n$. where $B_i(1 \leq i \leq n)$ is the pattern that is sought for recognition. If each B_i is found, then we deduce that H is true. For example, let $community(X) : -happensAt(size(X, \_), \_).$ be clause. if $happensAt(size(c4, 3), 2).$ is true, then it can be deduced that $c4$ is a community. The performance of ILP system may degrade if the background knowledge provided contains large amounts of irrelevant information so in many realistic problems experts are requested to choose background knowledge they believe to be useful. OLED imports background knowledge through a text file. The Table 4.1 presents an example of this file. Taking into account the importance of background knowledge file, we define the following types of rules in it:

- Rules for community entity recognition.
- Rules for time entity recognition.
- Facts for features' quantized values recognition.
- Rules for values of ground truth recognition.
- Rules which represent the inertia of Event Calculus, as discussed in Chapter 3.

Background Knowledge File
holdsAt(F,T_e) :- fluent(F), initiatedAt(F,T_s), T_e = T_s + 1, time(T_s),time(T_e). 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). 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

Table 4.1: Example of A OLED's Background Knowledge File

All instances of a community across time, have the same id number. So OLED Generator takes over the communities' renaming to be the condition above satisfied. Communities' names are appeared as $c < id >$, where $< id >$ is a number denoting the identity. These names are selected such that don't conflict with other atoms of domain.

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. OLED imports mode declarations through a text file. The Table 4.3 presents an example of this file. This file determines how the rules that OLED will learn, look like. Their form is below:

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

Table 4.2: Rules That OLED Learns

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

Table 4.3: Example of A OLED's Mode Declarations File

The body of rule (1) is a list of $happensAt$ predicates as we described in Chapter 3. $feature_k (1 \leq k \leq n)$ denotes a feature's name. $community_i$ is a community's name and $value_m (1 \leq m \leq n)$ is the quantized value of feature $feature_k$ and community $community_i$. $time_j$ is the integer that represents a specific timeframe. In the head of the rule, $predicted\_event$ is one of labels we try to predict (growth, shrinkage, continuation, dissolution). $community_i$ and $time_j$ are the same with body's ones. Rules are expressed as if feature $feature_1$ of community $community_i$ has value $value_1$ at time $time_j$ and the same is true for the other features of list then is fired the initiation of event $predicted\_event$. This means we predict that $predicted\_event$ will start to occur at next time $time_j + 1$. Rule's body has so many $happensAt$ predicates as the features are considered that are required for prediction.

On the contrary, if the body of rule (2) is true then is fired the termination of event $predicted\_event$. This means we predict that $predicted\_event$ will stop to occur at next time $time_j + 1$.

As shown in Figure 4.2, our dataset was split into two sets, the training and the testing set. This method is common practice for supervised learning. The contents of training and testing file are:

Training Set: It's the dataset we have the training data along with a label. This dataset is usually prepared either by humans or by collecting some data in semi-automated way. In our case the input data are communities represented by structural and temporal features and their evolutionary events (these events are their labels), used for training the OLED.

Testing Set: It's the dataset you are going to apply your model to. It includes the data for which you are interested what your model will predict and thus you don't have any "expected" label here yet. In our case it's a set of communities represented by structural and temporal features along with their evolutionary event (ground truth), used only to evaluate the performance of our system, without participating in the training of OLED.

An issue that exists with training and testing sets is how we should split our dataset into these two files to evaluate our model with best possible way. A well-known method which resolves this issue is the Cross-Validation. It involves partitioning a sample of data into subsets, performing the training on all subsets apart from one, and testing on the remaining subset. To reduce variability, multiple rounds of cross-validation are performed using different subsets for the training, and consequently a different subset for the testing. In our case, we have a timestamped dataset. Thus the classic edition of Cross-Validation isn't suitable. However, there is the Time Series Cross Validation, a variation of Cross-Validation which takes into account the temporal relationship between the training and testing sets it creates. Each training set consists only observations that occurred prior to the observation. As long as we aim to predict the evolutionary events of communities, which are related with the future of every community, it would be unreasonable if communities of a testing set belonged in preceding in time timeframes in comparison to the timeframes of the communities that make up the training set.

OLED imports training and testing sets as json files. Each line of json file represents a time point. Each time point contains the lists of narratives and annotations as they described in Chapter 3. Narratives contains the low level events, while annotations contain high level events which are the ground truth. For example, if we have this json file:

{"narrative":["happensAt(size(c523,5),2)"], "annotation":["holdsAt(growth(c523),2)"], "time":2}

It denotes that at timeframe 2 the community $c523$ has grown and its current $size$ is 5.

Below we present what every subset (Fold) of training and testing set contains after application of Time Series Cross Validation method.

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 we have in datasets. Note that the first timeframe has no evolutionary events (high events) since there is no previous timeframe in the dataset in order to track the communities' evolution of the first timeframe. Respectively, the last timeframe hasn't features (low events) because there is no next timeframe to predict evolution of its communities in the dataset. 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. That's the way the OLED works. 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. It's required by OLED to initiate the inertia of every community's event.

Since the training set is imported into OLED, the training process begins. Because OLED is an online learner, it 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. We did it because we, for example, want the timeframe 2 (and each other timeframe) 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 transform the existing theory to satisfy as good as possible the right prediction of current examples. When the training process is completed, a logical theory is derived as the learned model. Its form is illustrated in Table 4.2. Using this theory we predict the evolutionary events of communities which are in testing set.

To estimate our model, we execute the evaluation procedure where we compare the predicted labels we obtained from Testing with the known ground truth. These labels refer if an example belongs to one class of growth, shrinkage, continuation and dissolution. The comparison above results into four cases:

1. Test set example predicted correctly as belonging to a class. It's considered as True Positive (TP).
2. Test set example predicted mistakenly as belonging to a class. It's considered as False Positive (FP).
3. Test set example predicted mistakenly as not belonging to a class. It's considered as False Negatives (FN).
4. Test set example predicted correctly as not belonging to a class. It's considered as True Negative (TN).

The number of True Positives of a particular class is called True Positives (TPs). Respectively we have False Positives (FPs), False Negatives (FNs) and True Negatives (TNs). To assess the performance for each one of the four types of evolutionary events we examine in this thesis, we calculate using TPs, FPs, FNs and TNs the performance measures below:

$$Micro\_Precision = \frac{TPs}{TPs + FPs}$$

$$Micro\_Recall = \frac{TPs}{TPs + FNs}$$

$$Micro\_Fscore = \frac{2 \times Precision \times Recall}{Precision + Recall}$$

These performance measures are Micro Average. Apart from the we also use the Macro Average. This type of measures is taking into account Precision, Recall and Fscore both positives and negatives so that we take an average value between them. For example, we may get high accuracy value on an imbalanced dataset, but it may be misleading in case where our classifier always predicts correctly the majority class, because there are a lot examples of this. But the classifier couldn't predict correctly the negatives examples because they aren't a lot. In this Macro Average will reduce rightly the accuracy value of our model. Macro-averaging allows to accurately measure performance when a dataset consists of imbalanced classes. So for our imbalanced dataset, Macro Average measures are required.

Below we present Macro Average measures:

$$Macro\_Precision = \frac{\frac{TPs}{TPs+FPs} + \frac{TNs}{TNs+FNs}}{2}$$

$$Macro\_Recall = \frac{\frac{TPs}{TPs+FNs} + \frac{TNs}{TNs+FPs}}{2}$$

$$Macro\_Fscore = \frac{2 \times Macro\_Precision \times Macro\_Recall}{Macro\_Precision + Macro\_Recall}$$

5. EXPERIMENTS

5.1 DATASET DESCRIPTION

The data we have are collected from Mathematics Stack Exchange. Mathematics Stack Exchange is a question and answer site for people studying math at any level and professionals in related fields. It's built and run as part of the Stack Exchange network of Question and Answer sites. Mathematics Stack Exchange was created with a vision of building a library of detailed answers to every question about math, with the help of its users. The primary purpose of Mathematics Stack Exchange is to enable users to post questions, answer questions posted by other users and comment on the users' posts. Users can vote on both answers and questions, and through this process they earn reputation points, which give them the opportunity to unlock new privileges, ranging from the ability to vote and comment on questions and answers, to the ability to moderate many aspects of the site. One of the most important features of Mathematics Stack Exchange is question tagging. All questions are tagged with their subject areas. Each can have up to 5 tags, since a question might be related to several subjects. Users can choose any tag to see a list of questions with that tag or navigate to a tag list to browse for topics that interest them. For our dataset collected 376030 posts from many different topics, 261600 answers and comments between 28-09-2009 and 31-05-2013.

Every Mathematics Stack Exchange user is represented by a vertex in a graph and there is an edge between two user vertices if one of them posts an answer or a comment on the other user's post. The Mathematics Stack Exchange dataset is split into equally sized, with respect to the number of posts (questions, answers or comments). In particular, we had 10 timeframes with some overlap. Thus, we end up having a graph for every timeframe. We should note that there is a single edge added between any pair vertices within the same timeframe and the graph in each timeframe is undirected.

We consider that a group of users belongs in the same community if they make posts (questions, answers or comments) about the same topic. In particular, we use 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. We should also note that communities with no more than 3 members are eliminated. New communities are added in time, and most of the old communities persevere.

We have two types features, structural and temporal. In temporal, the number of ancestors we use is four. In Mathematics Stack Exchange dataset there are communities which aren't appeared in each timeframe, although they may not have been dissolved yet. It means that a community could be appeared in timeframe i, not appeared in timeframe $i + 1$ and appeared again in timeframe $i + 1$. It happens because communities with few members in a timeframe are pruned from dataset. So, we're 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. Specifically, *continuation* event is the majority class (90 percent of examples belong to it). The Table 5.1 shows us in detail how is the percentage of each class in our dataset.

Growth	Shrinkage	Continuation	Dissolution
0,05%	0,02%	0,9%	0,03%

Table 5.1: The percentage of each class in our dataset

5.2 EXPERIMENTAL RESULTS

For the execution of our experiments we followed all the steps as they described in Chapter 4. We created the necessary structures of dynamic social network and deliver the data to OLED in the appropriate format, using OLED Generator. This dataset is split into 10 timeframes and the overlap between consecutive time frames is 60%. The community labels were obtained as explained in the previous sections, i.e. by considering the posts' labels. Communities were tracked over the time and their evolutionary events (Growth, Shrinkage, Continuation, Dissolution) were obtained by a simple threshold. 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). The features were quantized as were assigned as low level event to OLED. The high level events are the evolutionary events. For the quantization, various number of quantized values were tried but we had best performance with quantization of continuous features' values into 5 values. Experiments were executed with both structural and temporal features. At the end, this dataset was split in training and testing sets using Time Series Cross Validation method. Because our data is highly imbalanced apart from Micro Average measures, we also use Macro Average values. Below, we present all experiments that took place.

5.2.1 SURVIVAL EXPERIMENT

Firstly, we conducted an experiment with a simple high level event, named *survival*. As survival event, we considered all the growth, shrinkage and continuation events. We used both structural and temporal features. In Table 5.2 we present the results:

We notice that the micro measures are extremely high. It happens because of data's imbalance. The survival events are 97% of total data, so OLED initializing the inertia of survival event on examples, it doesn't find enough negatives (dissolution events) to fail in its prediction. However, negatives' prediction is considerably lower because the algorithm should derive a more accurate rule which will predict survival terminal correctly. But, negatives' prediction performance isn't observable by Micro values. This is a phenomenon that can mislead us about checking our predictive model. That's why, there are Macro measures. Macro measures are taking into account also the value of TNs, so we have more weighted measures. This appears in our results, since we can see that Macro values are lower.

	Survival Structural	Survival Temporal
Micro Precision	0.9737	0.9882
Micro Recall	0.9949	1.0000
Micro Fscore	0.9842	0.9941
Macro Precision	0.8125	0.9941
Macro Recall	0.6289	0.6765
Macro Fscore	0.7090	0.8051
TPs	5822	1850
FPs	157	22
FNs	30	0
TNs	56	12

Table 5.2: Survival Experiment

5.2.2 EXPERIMENT WITH FIRST QUANTIZATION ALGORITHM

Afterwards, we experimented with the *growth*, *shrinkage*, *continuation* and *dissolution* events. *Dissolution* is an event which has not duration. That is, for an existing community a dissolution event happens, and then it stops to exist. Dissolution event doesn't continue to happen. Thus we are interested for the first presence (initiation) of dissolution event in each community's evolution. This is equivalent to termination of negative event (survival). Consequently, we can evaluate dissolution events by evaluating how accurate are the termination rules of survival because only then a dissolution event occurs.

For this first experiment we quantized our structural features with first algorithm as it referred in Chapter 4. Below (Table 5.3) we represent the results:

	Growth	Shrinkage	Continuation	Dissolution
Micro Precision	0.2365	0.1915	0.9293	0.2675
Micro Recall	0.3027	0.1570	0.9760	0.3052
Micro Fscore	0.2655	0.1725	0.9521	0.2851
Macro Precision	0.6041	0.5848	0.8066	0.6210
Macro Recall	0.6317	0.5698	0.6939	0.6374
Macro Fscore	0.6176	0.5772	0.7460	0.6291

Table 5.3: First Quantization Experiment

5.2.3 EXPERIMENT WITH SECOND QUANTIZATION ALGORITHM

The results of the second feature quantization method appear in Table 5.4.

We notice that second algorithm for quantization goes better in micro precision of the dissolution. We use it in each experiment below. In Table 5.5 results, we use the temporal features with second quantization's algorithm.

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

Table 5.4: Second Quantization Experiment

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

Table 5.5: Second Quantization Experiment with Temporal Features

The dataset with temporal features contains the features of the previous 4 instance of a community, the first timeframe for this dataset is at time 5 because the communities of previous 4 timeframes don't have 4 ancestors. The other features in dataset with temporal features are these we described in Section 4.3.2. The theory which derived from the experiment for dissolution event was empty. The predictor couldn't evaluate any rule with high score because there weren't many available examples, since the number of timeframes (6, from F_5 to F_{10}) and the communities is small. Without rules for dissolution events, we can't predict this event because we can't terminate survival events that fire dissolution of communities. That's why Dissolution results do not appeared in table above. Growth and shrinkage events with temporal features have lower performance than the best corresponding events with structural features. But for continuation event, the reverse is true.

5.2.4 EXPERIMENT WITH BEST PRUNING VALUES

After OLED has derived the learned theory, a pruning method can be applied to it. Specifically, OLED removes the clauses whose score is smaller than a quality threshold S_{min}. In previous experiments, S_{min} was 0.9. Now we try for each event the values 0.5, 0.7, 0.3 as S_{min} and choose them for which we have the best performance. In the dataset with the structural features, best pruning value for growth event is 0.7, for shrinkage 0.5, for continuation 0.9 and for dissolution 0.9. In Table 5.6 we present the results for best pruning values in dataset with structural features.

In the dataset with temporal features, best pruning value for growth event is 0.7, for shrink-

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

Table 5.6: Best Pruning Experiment with Structural Features

age 0.7, and or continuation 0.9. In Table 5.7 we present the results for best pruning values in dataset with temporal features.

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

Table 5.7: Best Pruning Experiment with Temporal Features

5.2.5 EXPERIMENT WITH LONG RANGE RULES

The best pruning value didn't improve the performance. Because the dataset with temporal features can't derive accurate rules for growth and shrinkage events, we tried to change the way rules are formed so that can contain features of communities' ancestors. It's like we use temporal features, but we don't have their values as different features but as the same features at different time. In order to change the form of derived rules, as discussed in Chapter 4, should change modes declaration. So we modified this file so that the new form of rules contains long range relationships. The new rules are illustrated in Table 5.8.

In Table 5.8, the form of rule is the same as in Table 4.2 except of $< time >$ value and geqn /3 predicate. In this form, the rule body's time value isn't required to have the same value with head's time value. With this we manage to have features of community's ancestors. The geqn /3 predicate denotes that the time $time_k$ is num_1 units higher than $time_l$ time. This show us to what ancestor this feature belongs to. Also because now OLED should include more than two timeframes in rule's search, we should increase chunk size. If chunk size equals to $N+2$, then derived rules can contain until N ancestors' features. However, very big chunk size means that domain for rules' searching will be big. Thus CPU and memory consumption will be increased. For our experiments we selected chunk

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).

Table 5.8: New Rules That OLED Learns With Long Range Relationships

size equals to 3, so we obtain rules that contain features of first ancestor. In Table 5.9 are showed the results:

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

Table 5.9: Long Range Relationships Experiment

5.2.6 EXPERIMENT WITH WEIGHTS ON TPs, FPs, FNs

Neither this method increased the performance extremely. A problem is that the learned theories of experiments contains more termination than initiation rules, so we don't trigger the initiation of some events when must do it. It means OLED predicts a negative event (an event that does not occur) for a community at next timeframe but in reality we have a positive event (event occurs). In this case the FNs frequency of OLED is increased. The numbers of initiation and termination rules aren't balanced because OLED evaluate its rules based on TPs, FPs and FNs values. Using these values, it computes a score which refers how accurate is a rule. To control score's value, it's enough to change TPs, FPs and FNs values. So the idea is to add weights on TPs, FPs,FNs values during training process. For example, when we set FNs weight to 10, we mean that each time the number of FNs will be considered as ten times more than it really is. As a result OLED thinks that there are more FNs because termination rules stopped event while they shouldn't. It means that termination rules overestimate the termination condition. Thus score of termination rules is getting decreased. With this way we focus more in quality than quantity of termination rules. This functionality wasn't supported initially by OLED but changing OLED's core code, we added it.

Trying various values for weights, we found these ones which offer best performance in our experiments. In Table 5.10, we present the best weights for each class in experiment with structural features, while in Table 5.11 there are the corresponding values of experiment with temporal features.

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

Table 5.10: Best weights for each class in experiment with structural features

	TPs-weight	FPs-weight	FNs-weight
Growth	1	5	1
Shrinkage	1	1	1
Continuation	1	1	15

Table 5.11: Best weights for each class in experiment with temporal features

The results are presented in Table 5.12 for the experiments with structural features and in Table 5.13 for the experiments with temporal features.

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

Table 5.12: Weights on TPs,FPs,FNs - Experiment With Structural Features

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

Table 5.13: Weights on TPs,FPs,FNs - Experiment With Temporal Features

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

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

We wanted to grow the small TPs number of our results on testing by setting propriate weights. But when we were doing it, FPs was also grown. This means that OLED over-estimates the initiation condition because its initiation rules aren't specialized enough to detect correctly in which communities will occur an event. That's why performance in experiments weren't improved significantly with weights on metrics. Everything shows that OLED couldn't distinguish more accurate positive from negative and thus it has arrived to bound of its predictive ability for this dataset.

An advantage of OLED as prediction method is that the predictive model (Theory) it derives is human-readable. So you can read its rules of theory, analyze and extract interesting results for them. Some of these rules of our experiment with the best performance are shown in Tables 5.14 and 5.15. Transferability is also an interesting possibility. We can evaluate our model (theory) to new datasets without change anything, just importing theory file to OLED. Noticing the theories we obtained from previous experiments, we discovered that some features appeared more often in rules of specific evolutionary events than other. While some of them never appeared in rules' bodies. In Tables 5.16 and 5.17 we present for each evolutionary event (growth, shrinkage, continuation, dissolution), which structural features there are in rules' bodies that predict these events. Also there is the percentage of feature's appearance in corresponding experiment.

Learned Rules
initiatedAt(growth($X0$),$X1$) :- happensAt(density($X0$,1),$X1$), happensAt(diameter($X0$,2),$X1$).
terminatedAt(growth($X0$),$X1$) :- happensAt(ratio_cut($X0$,3),$X1$), happensAt(average_path_length($X0$,3),$X1$), happensAt(normalized_edges_number($X0$,5),$X1$).
terminatedAt(growth($X0$),$X1$) :- happensAt(ratio_cut($X0$,3),$X1$), happensAt(closeness_centrality($X0$,3),$X1$), happensAt(normalized_edges_number($X0$,5),$X1$).
terminatedAt(growth($X0$),$X1$) :- happensAt(cohesion($X0$,2),$X1$), happensAt(average_path_length($X0$,3),$X1$), happensAt(diameter($X0$,2),$X1$).

Table 5.14: Rules that are learned by our best experiment

Learned Rules
terminatedAt(shrinkage($X0$),$X1$) :- happensAt(ratio_association($X0$,3),$X1$).
terminatedAt(shrinkage($X0$),$X1$) :- happensAt(average_path_length($X0$,2),$X1$).
terminatedAt(shrinkage($X0$),$X1$) :- happensAt(closeness_centrality($X0$,2),$X1$), happensAt(ratio_cut($X0$,1),$X1$).
initiatedAt(shrinkage($X0$),$X1$) :- happensAt(eigenvector_centrality($X0$,1),$X1$), happensAt(ratio_association($X0$,5),$X1$).
terminatedAt(survival($X0$),$X1$) :- happensAt(ratio_association($X0$,4),$X1$), happensAt(density($X0$,2),$X1$).
terminatedAt(survival($X0$),$X1$) :- happensAt(ratio_association($X0$,3),$X1$), happensAt(clustering_coefficient($X0$,4),$X1$).
terminatedAt(survival($X0$),$X1$) :- happensAt(diameter($X0$,3),$X1$), happensAt(ratio_cut($X0$,3),$X1$), happensAt(ratio_association($X0$,2),$X1$).
terminatedAt(survival($X0$),$X1$) :- happensAt(ratio_association($X0$,3),$X1$), happensAt(closeness_centrality($X0$,3),$X1$).
terminatedAt(survival($X0$),$X1$) :- happensAt(clustering_coefficient($X0$,1),$X1$), happensAt(closeness_centrality($X0$,5),$X1$), happensAt(cohesion($X0$,4),$X1$).
terminatedAt(survival($X0$),$X1$) :- happensAt(normalized_edges_number($X0$,2),$X1$), happensAt(cohesion($X0$,2),$X1$), happensAt(average_path_length($X0$,1),$X1$).
terminatedAt(survival($X0$),$X1$) :- happensAt(size($X0$,1),$X1$), happensAt(betweenness_centrality($X0$,1),$X1$), happensAt(cohesion($X0$,3),$X1$).
terminatedAt(survival($X0$),$X1$) :- happensAt(normalized_edges_number($X0$,1),$X1$), happensAt(cohesion($X0$,3),$X1$), happensAt(average_path_length($X0$,1),$X1$).

Table 5.15: Rules that are learned by our best experiment

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 5.16: Structural Features Usage For Growth and Shrinkage Events Prediction

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 5.17: Structural Features Usage For Continuation and Dissolution Events Prediction

In Table 5.16 we are noticing that features like diameter, cohesion, ratio_cut and average_path_length affect in prediction of growth event since they are the 54.14 percent of total features which appeared in rules for growth prediction. On the contrary features like betweenness_centrality and normalized_association didn't appear in any rule's body. For shrinkage event the most used features are ratio_association, ratio_cut, cohesion and clustering_coefficient, while there isn't the normalized_association feature. In Table, 5.17 features of continuation and dissolution events are showed. Continuation of an event seems that is affected from ratio_cut, ratio_association and clustering_coefficient features and isn't affected from cohesion feature. While for dissolution event, every feature is used in prediction, especially the usage of clustering_coefficient, cohesion and betweenness_centrality.

In Tables 5.18 and 5.19 we present temporal features which exist in rules of corresponding experiments.

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 5.18: Temporal Features Usage For Growth Event Prediction

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%

Table 5.19: Temporal Features Usage For Shrinkage and Continuation Events Prediction

For growth event prediction the temporal features: ancestor4_average_path_length, activeness_ancestor_2_ancestor3, aging_ancestor0, ancestor1_diameter, ancestor3_closeness_centrality and the other that are illustrated in Table 5.18, are the same important. Shrinkage event prediction uses the values of ancestor1_event_is_shrinking, ancestor4_clustering_coefficient, cohesion, eigenvector_centrality, joinNodesRatio_currentCommunity_ancestor0, ratio_cut. Many temporal features are missing from rules' bodies for both growth and shrinkage event prediction because they didn't help OLED in prediction. That’s why experiments with temporal features didn’t really improved the performance of our prediction. For continuation event prediction features aren’t important, since only two (cohesion and ratio_cut) are used.

6. CONCLUSION

6.1 SUMMARY

In this thesis, we tried to predict the evolution of communities in a dynamic social network. The evolution of a community is consisted of growth, shrinkage, continuation and dissolution events. We carried out the prediction using an online Inductive Logic Programming system (OLED) for learning logical theories from data streams. Initially, we developed a software system (OLED Generator) which manipulated our dataset so that was converted in an appropriate form for OLED. Specifically, we built structures of social network such as timeframes and communities. We tracked the evolution of communities over the time and obtained the ground truth of evolutionary events. As features we used the structural characteristics of network graph that are called Structural features. Moreover, we tried temporal features which also contain the structural features of previous snapshots of social network. Our features were quantized and assigned as features of communities. Our dataset contained real life data from Mathematics Stack Exchange. Micro Average and Macro Average measures of our experiments' performance are presented because of imbalanced classes (Growth, Shrinkage, Continuation and Dissolution). Best theory pruning values were found for every OLED execution. Experiments with temporal features couldn't learn many rules, since there weren't enough timeframes. So, we execute experiments where OLED learned long range rules. For these rules we used every timeframe and ancestors' features could be included in body of them. Subsequently, weights were applied to TPs, FPs and FNs values during training procedure to change rules' scores and try to control TPs, FPs and FNs values during testing procedure. This was the experiment that gave us the best results. There we noticed that OLED couldn't be more accurate. Finally, we analyzed which features are the most influential in growth, shrinkage continuation and dissolution of a community.

6.2 FUTURE WORK

Future work could be directed to a range of different fields. Others machine learning classifiers can be used to predict the evolution of communities (e.g. SVM, Random Forest) to compare our results. More 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. Also, another quantization algorithm, which will adapt better the quantized values into features' values distribution, may help. Because OLED is an online system, it needs too many data streams to decide if a rule is trusted. However, our dataset's timeframes are just 10. So balanced datasets with more timeframes could be examined. Otherwise different segmentation of timeframes is possible to be applied because if each timeframe contains less communities then more timeframes will be created. Finally, it would be very interesting to test our learned rules in other datasets to notice how relevant they are at problem of community evolution in general.

ABBREVIATIONS - ACRONYMS

OLED	Online Learning of Event Definitions
EC	Event Calculus
ILP	Inductive Logic Programming
OG	OLED Generator
GRASP	Greedy Randomized Adaptive Search Procedure
CPM	Clique Percolation Method
GED	Group Evolution Discovery
LfI	Learning from Interpretations
TP	True Positive
FP	False Positive
FN	False Negatives
TN	True Negatives

REFERENCES

[1] Balázs Adamcsek, Gergely Palla, Illés J. Farkas, Imre Derényi, and Tamás Vicsek. Cfinder: locating cliques and overlapping modules in biological networks. *Bioinformatics*, 22(8):1021–1023, 2006.

[2] Hamidreza Alvari, Alireza Hajibagheri, Gita Sukthankar, and Kiran Lakkaraju. Identifying community structures in dynamic networks. *Social Network Analysis and Mining*, 6(1):77, Sep 2016.

[3] Sitaram Asur, Srinivasan Parthasarathy, and Duygu Ucar. An event-based framework for characterizing the evolutionary behavior of interaction graphs. *ACM Trans. Knowl. Discov. Data*, 3(4):16:1–16:36, December 2009.

[4] Michele Berlingerio, Arisitdes Gionis, Bjoern Bringmann, and Francesco Bonchi. Learning and predicting the evolution of social networks. *IEEE Intelligent Systems*, 25:26–35, 2010.

[5] 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.

[6] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(10):P10008, 2008.

[7] P. Bródka, P. Kazienko, and B. Kołoszczyk. Predicting Group Evolution in the Social Network. *ArXiv e-prints*, October 2012.

[8] Piotr Bródka, Stanisław Saganowski, and Przemysław Kazienko. Ged: the method for group evolution discovery in social networks. *Social Network Analysis and Mining*, 3(1):1–14, Mar 2013.

[9] Antoni Calvó-Armengol and Yves Zenou. Social networks and crime decisions: The role of social structure in facilitating delinquent behavior*. *International Economic Review*, 45(3):939–958, 2004.

[10] Deepayan Chakrabarti, Ravi Kumar, and Andrew Tomkins. Evolutionary clustering. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD ’06, pages 554–560, New York, NY, USA, 2006. ACM.

[11] Yun Chi, Xiaodan Song, Dengyong Zhou, Koji Hino, and Belle L. Tseng. Evolutionary spectral clustering by incorporating temporal smoothness. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD ’07, pages 153–162, New York, NY, USA, 2007. ACM.

[12] L. De Raedt. *Logical and Relational Learning*. Cognitive Technologies. Springer Berlin Heidelberg, 2008.

[13] 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.

[14] Opher Etzion and Peter Niblett. *Event Processing in Action*. Manning Publications Co., Greenwich, CT, USA, 1st edition, 2010.

[15] Santo Fortunato. Community detection in graphs. *Physics Reports*, 486(3):75 – 174, 2010.

[16] Wei Gao, Kam-Fai Wong, Yunqing Xia, and Ruifeng Xu. *Clique Percolation Method for Finding Naturally Cohesive and Overlapping Document Clusters*, pages 97–108. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006.

[17] 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.

[18] Mark Goldberg, Malik Magdon ismail, Srinivas Nambirajan, and James Thompson. Tracking and predicting evolution of social communities. ?

[19] Derek Greene, Donal Doyle, and Padraig Cunningham. Tracking the evolution of communities in dynamic social networks. In *Proceedings of the 2010 International Conference on Advances in Social Networks Analysis and Mining*, ASONAM '10, pages 176–183, Washington, DC, USA, 2010. IEEE Computer Society.

[20] Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, 1963.

[21] 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.

[22] 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.

[23] NIKOS KATZOURIS, ALEXANDER ARTIKIS, and GEORGIOS PALIOURAS. Online learning of event definitions. *Theory and Practice of Logic Programming*, 16(5-6):817–833, 2016.

[24] David Kempe, Jon Kleinberg, and Éva 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*, KDD '03, pages 137–146, New York, NY, USA, 2003. ACM.

[25] Robert Kowalski and Marek Sergot. *A Logic-Based Calculus of Events*, pages 23–55. Springer Berlin Heidelberg, Berlin, Heidelberg, 1989.

[26] Ravi Kumar, Jasmine Novak, Prabhakar Raghavan, and Andrew Tomkins. On the bursty evolution of blogspace. *World Wide Web*, 8(2):159–178, Jun 2005.

[27] Jérôme Kunegis, Andreas Lommatzsch, and Christian Bauckhage. The slashdot zoo: Mining a social network with negative edges. In *Proceedings of the 18th International Conference on World Wide Web*, WWW '09, pages 741–750, New York, NY, USA, 2009. ACM.

[28] Jure Leskovec, Daniel Huttenlocher, and Jon Kleinberg. Predicting positive and negative links in online social networks. In *Proceedings of the 19th International Conference on World Wide Web*, WWW '10, pages 641–650, New York, NY, USA, 2010. ACM.

[29] David Liben-Nowell and Jon Kleinberg. The link-prediction problem for social networks. *Journal of the American Society for Information Science and Technology*, 58(7):1019–1031, 2007.

[30] Ryan N. Lichtenwalter, Jake T. Lussier, and Nitesh V. Chawla. New perspectives and methods in link prediction. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '10, pages 243–252, New York, NY, USA, 2010. ACM.

[31] Mariá C.V. Nascimento and Leonidas Pitsoulis. Community detection by modularity maximization using grasp with path relinking. *Computers & Operations Research*, 40(12):3121 – 3131, 2013.

[32] M. E. J. Newman. Detecting community structure in networks. *The European Physical Journal B*, 38(2):321–330, Mar 2004.

[33] Gergely Palla, Albert-Laszlo Barabasi, and Tamas Vicsek. Quantifying social group evolution. *Nature*, 446(7136):664–667, April 2007.

[34] Gergely Palla, Imre Derényi, Illés Farkas, and Tamás Vicsek. Uncovering the overlapping community structure of complex networks in nature and society. *Nature*, 435(7043):814–818, June 2005.

[35] 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.

[36] Josep M Pujol, Vijay Erramilli, and Pablo Rodriguez. Divide and Conquer: Partitioning Online Social Networks. Technical Report arXiv:0905.4918, Jun 2009. Comments: 7 pages, 4 figures.

[37] Panagiotis Symeonidis, Eleftherios Tiakas, and Yannis Manolopoulos. Transitive node similarity for link prediction in social networks with positive and negative links. In *Proceedings of the Fourth ACM Conference on Recommender Systems*, RecSys '10, pages 183–190, New York, NY, USA, 2010. ACM.

[38] Mansoureh Takaffoli, Reihaneh Rabbany, and Osmar R. Zaiane. Community evolution prediction in dynamic social networks.

[39] Mansoureh Takaffoli, Farzad Sangi, Justin Fagnan, and Osmar Zaiane. Modec — modeling and detecting evolutions of communities, 2011.

[40] Mansoureh Takaffoli, Farzad Sangi, Justin Fagnan, and Osmar R. Zäıane. Community evolution mining in dynamic social networks. *Procedia - Social and Behavioral Sciences*, 22(Supplement C):49 – 58, 2011. Dynamics of Social Networks.

[41] Hanghang Tong, Spiros Papadimitriou, Jimeng Sun, Philip S. Yu, and Christos Faloutsos. Colibri: Fast mining of large static and dynamic graphs. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '08, pages 686–694, New York, NY, USA, 2008. ACM.

[42] Serafeim Tsironis, Mauro Sozio, and Michalis Vazirgiannis. Accurate spectral clustering for community detection in mapreduce. 2013.

[43] S. Wasserman and K. Faust. *Social Network Analysis: Methods and Applications*. Structural Analysis in the Social Sciences. Cambridge University Press, 1994.

[44] Zhao Yang, René Algesheimer, and Claudio Tessone. A comparative analysis of community detection algorithms on artificial networks. 6, 08 2016.

[45] R. Zafarani, M.A. Abbasi, and H. Liu. *Social Media Mining: An Introduction*. Cambridge University Press, 2014.

[46] Elena Zheleva, Lise Getoor, Jennifer Golbeck, and Ugur Kuter. *Using Friendship Ties and Family Circles for Link Prediction*, pages 97–113. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.