EPredictor: An Experimental Platform for Community Evolution Prediction Tests

Narimene Dakiche[1][a], Fatima Benbouzid-Si Tayeb[1], Karima Benatchba[1], Yahya Slimani[2], Abdelouahab Khelifati[1] and Hadjer Chabane[1]

[1]*Laboratoire des Méthodes de Conception de Systèmes (LMCS), Ecole Nationale Supérieure d'Informatique (ESI), BP 68M, 16270 Oued Smar, Alger, Algeria*

[2]*Computer Science Department, ISAMM, Institute of Manouba, 2010 Manouba, Tunisia*

Keywords: Dynamic Social Network, Community Detection, Community Evolution, Evolution Prediction.

Abstract: This paper presents "EPredictor" an experimental platform which enables testing, verifying and validating models related to community evolution prediction in dynamic social networks. Community evolution prediction is one of the most interesting issues in the field of social network analysis. It is usually handled by following four main steps: (1) split the network into timeframes; (2) detect communities in each timeframe; (3) track their evolutionary behavior and (4) build a predictive model to forecast the future events. The main objective of EPredictor is to provide a flexible environment that handles the entire process of community evolution prediction with a rich set of literature methods for each step; thus enabling researchers to make valuable comparisons and consistent analysis.

1 INTRODUCTION

Social networks have become ubiquitous and increasingly popular. They have interesting properties that gave rise to the very active field of Social Network Analysis (SNA). SNA exploit network and graph theories in order to understand the relationship between members involved in an interaction for numerous useful purposes. Social networks are usually represented by graphs in which the nodes represent the social entities (e.g., individuals) and the edges describe social interactions (e.g., friendship, collaboration, trust, etc) (Tabassum et al., 2018). A further abstraction of the network concept is the dynamic network in which changes occur over time. These changes happen when new nodes join the network, existing ones leave it, or when existing pairs of nodes establish a new relation or terminate an existing one over time.

One important characteristic of social networks is community structure, i.e., groups of nodes closer to each other in comparison to other nodes of the network. On a community scale, network dynamics result in certain evolutionary events such as growth, merge, split and survive (Bródka et al., 2013). Modeling and detecting community evolution have become a subject of great interest and many researchers have contributed to the understanding of the phenomena (Dakiche et al., 2018). Indeed, the study of community evolution plays a prominent role in understanding the changes of networks and predict their future (Saganowski et al., 2019).

In the related literature, the main issues are how to detect critical events a community can undergo, how to track communities over time and how to predict their future. Existing approaches of predicting community evolution (Bródka et al., 2013; Takaffoli et al., 2014; Diakidis et al., 2015; İlhan and Öğüdücü, 2016; Pavlopoulou et al., 2017; Saganowski et al., 2019; Rajita et al., 2020; Dakiche et al., 2021) are typically addressed through the same main steps: (1) split the network into timeframes also called snapshots; (2) detect communities in each timeframe; (3) track their evolutionary behavior and (4) build a predictive model to forecast the future events.

For each one of the listed steps, there exist several literature methods that can be used. However, to the best of our knowledge, no research tool that handles the entire process for predicting community evolution were proposed.

[a] https://orcid.org/0000-0003-2371-616X

Dakiche, N., Benbouzid-Si Tayeb, F., Benatchba, K., Slimani, Y., Khelifati, A. and Chabane, H.
EPredictor: An Experimental Platform for Community Evolution Prediction Tests.
DOI: 10.5220/0010547502950302
In *Proceedings of the 11th International Conference on Simulation and Modeling Methodologies, Technologies and Applications (SIMULTECH 2021)*, pages 295-302
ISBN: 978-989-758-528-9

In this paper, we present ***"EPredictor"*** an experimental platform that enables testing, verifying and validating models related to predicting community evolution in dynamic social networks. Our challenge is to provide a flexible environment to conduct community evolution prediction with a rich set of literature methods for each step; thus enabling researchers to make valuable comparisons and consistent analysis. Besides, EPredictor can be enriched by adding other options and methods in each step.

The remainder of this paper is organized as follows. Section 2 briefly introduces the community evolution prediction approach. Section 3 presents the EPredictor platform and illustrates its features and functionalities. Finally, Section 4 gives a resume and outlines future developments.

2 COMMUNITY EVOLUTION PREDICTION

Community evolution prediction is usually handled as a supervised learning task, where the history of a community is used to predict its future. Often, three or four past instances are used to train a classifier. It consists of four main steps as shown in Fig.1: (1) The dynamic social network data are segmented into a series of timeframes also called snapshots; (2) Each timeframe is partitioned using a community detection algorithm and for each community a set of features, mostly related to the link structure, such as the size, cohesion, and density are computed; (3) Then evolution events of communities in consecutive timeframes are identified; (4) Finally, a predictive model to forecast community future occurring events is built. The rest of this section deals with a detail description of these steps.

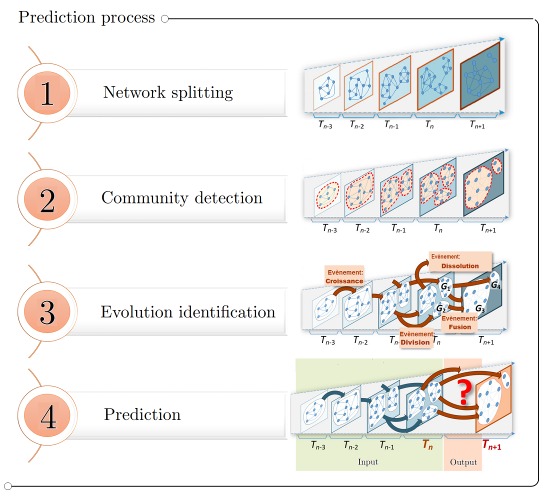


Figure 1: Community evolution prediction process.

2.1 Network Splitting

A dynamic social network $G = (V, E_t)$ is defined by a set of nodes V and a set of time-stamped edges describing the interactions among them. Each edge $e \in E_t$ represents an interaction between two nodes $u, v \in V$ at time t (Holme and Saramäki, 2012).

In order to analyze the dynamic network G, it is split into τ consecutive timeframes, thus obtaining a set of graphs $G = (G_0, ..., G_\tau)$., where $G_i = (V_i, E_i)$ represents a graph with only the set of nodes and edges that appears in the interval (t_i, t_{i+1}).

2.2 Community Detection

Partitioning a network into communities is a difficult task; thus, several methods have been proposed during the last decade, each one of them tailored to extract communities carrying specific characteristics (El Moussaoui et al., 2019).

Formally, given a dynamic social network $G = (G_0, ..., G_\tau)$ as input, for each snapshot, its corresponding communities are detected. The k communities detected in the i^{th} snapshot are denoted by $C_i = (C_i^1, C_i^2, ..., C_i^k)$ where community $C_i^p \in C_i$, $1 \le p \le k$, is also a graph denoted by (V_i^p, E_i^p).

2.3 Community Matching

The crucial goal of this phase is to detect community evolution between consecutive snapshots. It consists in finding series of similar communities in different snapshots. Thus, a dynamic community is represented by its constituent communities ordered by time snapshots. Formally, the dynamic community is denoted by $DC = \{C_{t_0}, C_{t_1}, ..., C_{t_\tau}\}$, where $t_0 < t_1 < ... < t_\tau$ and C_{t_i} represents its instance community at time t_i. Besides, the evolution is represented by the events a community may undergo from one snapshot to another i.e., events like splitting, growing, merging, dissolving and so on. This raises the problem of finding a given community at time t_i among those of time t_{i+1}.

There are several taxonomies in the literature that categorize the changes which are likely to occur to a community. For example, Asur et al. (2009) distinguish five possible events, i.e., the communities may dissolve, form, continue, merge and split; while Bródka et al. (2013), in turn, describe seven noticeable event types: continuing, shrinking, growing, splitting, merging, dissolving and forming. The community evolution can then be defined as a sequence of communities ordered by time, from the timeframe

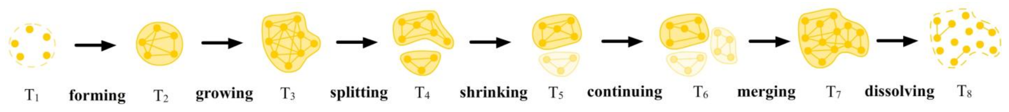


Figure 2: Community evolution in a dynamic network (Bródka et al., 2013).

where it first appears to the timeframe where it is last observed (Fig. 2).

2.4 Prediction

In this step, the predictive model is built. The features previously computed and the identified evolution events are used to create community evolution sequences. A collection of community evolution sequences is the final form of the data used for training and testing the classifiers.

A community evolution sequence contains a community and its several past instances from the previous frames. Formally, it consists of the current community $C^p_{t_i}$, the p^{th} community of timeframe t_i, and its previous instances $C^p_{t_{i-1}}$, $C^p_{t_{i-2}}$, ..., $C^p_{t_{i-n}}$. It looks as follows:

$$input = (C^p_{t_{i-n}}, ..., C^p_{t_{i-2}}, C^p_{t_{i-1}}, C^p_{t_i}) \qquad (1)$$

Each community instance at a specific timeframe is described by its structural features and its identified evolution event. Vector (2) represents the structure of each community instance.

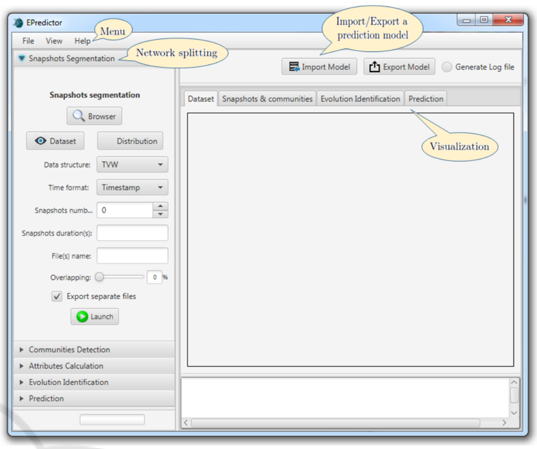
$$C^p_{t_i} = [f^1_{p,t_i}, f^2_{p,t_i}, ..., f^k_{p,t_i}, event] \qquad (2)$$

The goal of classification is to predict the next event for a given community. In what follows, we present in details the EPredictor platform.

3 THE EPredictor PLATFORM

EPredictor[1] is an experimentation platform (Fig.3) which handle the entire prediction process. It is dedicated to test and validate community evolution prediction models. The main objective of EPredictor is to provide a variety of community evolution prediction experimental scenarios for various real social networks. This experimental platform allows to conduct the prediction process passing through its main steps, offering for each step the possibility to use different literature methods. Thus, it offers the possibility to get a significant number of experiments for only one network. Besides, it is possible to enrich the platform by adding other options and methods in each step. Moreover, EPredictor allows to conduct the prediction process by starting from any step, importing and exporting multiple kinds of data and visualizing results of different methods.

[1]https://github.com/NarimeneDakiche/EPredicror



Snapshots segmentation

Data structure: TVW

Time format: Timestamp

Snapshots numb... 0

Snapshots duration(s):

File(s) name:

Overlapping: 0 %

[✓] Export separate files

Figure 3: The EPredictor platform.

The EPredictor architecture is based on the community evolution prediction process steps as shown in Fig.4. It is composed of the following six modules: a data input/output module allowing to start community evolution prediction from any step of the prediction process; a network segmentation module to split the network into several timeframes; a community detection module to identify corresponding communities of each timeframe; an attribute computation module to compute communities structural features; an evolution identification module to provide communities evolution events; a prediction module to forecast communities future events and a visualization module to visualize the results of each one of the previous modules. In the following subsections, we detail the functionalities of each module.

3.1 Data Input/Output Module

EPredictor offers the possibility to load different formats and types of data: dynamic network data, timeframe data, community data, and learning data.

- Loading data from a dynamic network in order to be split into several timeframes.

As it is shown in Fig.5, EPredictor allows to visualize some lines of the data source file in order to observe the information carried on each edge. Indeed, the edges must be time-stamped and this timestamp can have different types. It can be recorded as

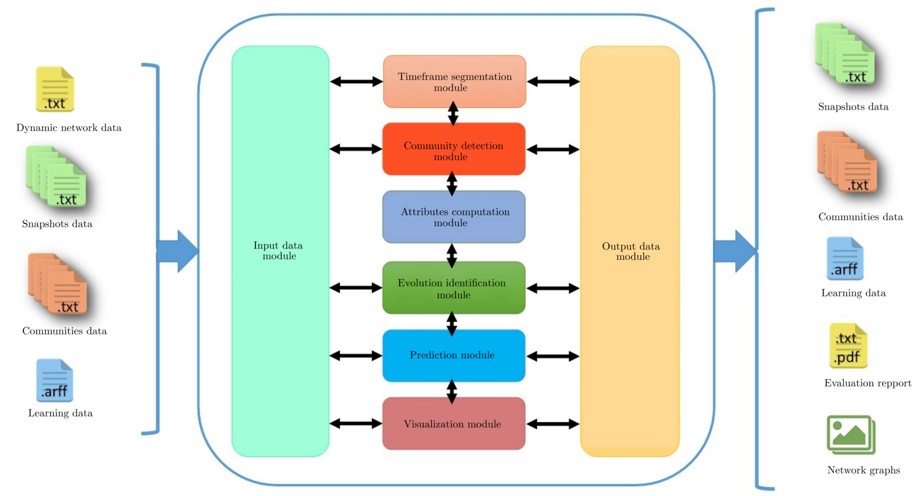
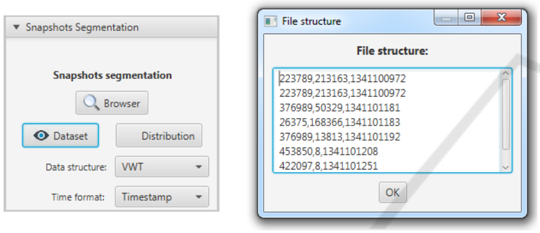


Figure 4: The EPredictor modules.



Data structure: VWT
Time format: Timestamp

File structure:

```
223789,213163,1341100972
223789,213163,1341100972
376989,50329,1341101181
26375,168366,1341101183
376989,13813,1341101192
453850,8,1341101208
422097,8,1341101251
```

Figure 5: Loading dynamic network data.

a unix-timestamp “1284101485” or as a human date “2010-09-10 06:51:25”. It is also possible to add a new type if needed (see Fig.6-(a)). To correctly process the network splitting, the user should specify the adequate type describing the edges. Besides, the order of information carried on each edge is important and should be indicated carefully. For example, an edge between two nodes V and W can be stored according to different formats such as $node1|node2|Stamp$ or $Stamp|node1|node2$. For this reason, the user should specify how these three elements are stored in the file data. The first node is represented by ’V’, the second node by ’ W’, time by ’T’ and any other information by ’X’. It is also possible to add a new structure of data if needed (see Fig.6-(b)).

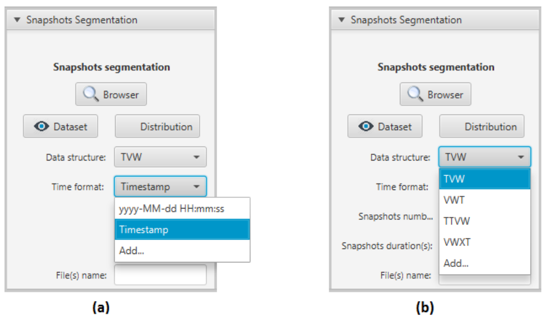
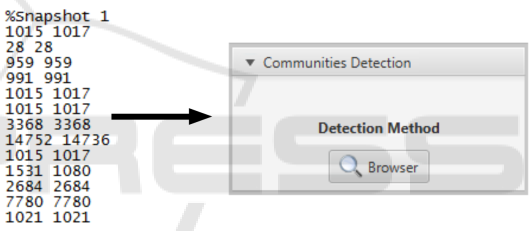


Figure 6: Timestamp and network data types.

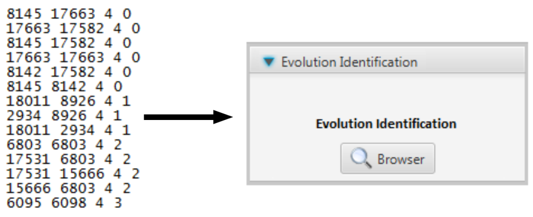
- Loading data from timeframe files (Fig.7): each file contains data of one snapshot. This data is used in the step of community detection.



```
%Snapshot 1
1015 1017
28 28
959 959
991 991
1015 1017
1015 1017
3368 3368
14752 14736
1015 1017
1531 1080
2684 2684
7780 7780
1021 1021
```

Figure 7: Loading timeframe data.

- Loading data from community files (Fig.8): each file contains data of detected communities into one timeframe according to the following format: $node1|node2|community\_id|snapshot\_id$. This data is used to compute community attributes and to identify their evolution events.



```
8145 17663 4 0
17663 17582 4 0
8145 17582 4 0
17663 17663 4 0
8142 17582 4 0
8145 8142 4 0
18011 8926 4 1
2934 8926 4 1
18011 2934 4 1
6803 6803 4 2
17531 6803 4 2
17531 15666 4 2
15666 6803 4 2
6095 6098 4 3
```

Figure 8: Loading community data.

- Loading data from an ARFF file (Hall et al., 2009) containing the learning instances to be used in the prediction model.
- Loading a prediction model (.log file) containing all the methods used in the different steps with all their required parameters (Fig.9).

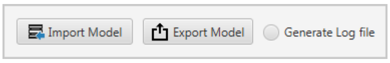


Figure 9: Import / Export of a prediction model.

At the end of each step the results are automatically exported into a folder in the directory of the executable such as timeframe files, detected communities, evolution sequences, learning instances and the final evaluation report.

3.2 The Timeframe Segmentation Module

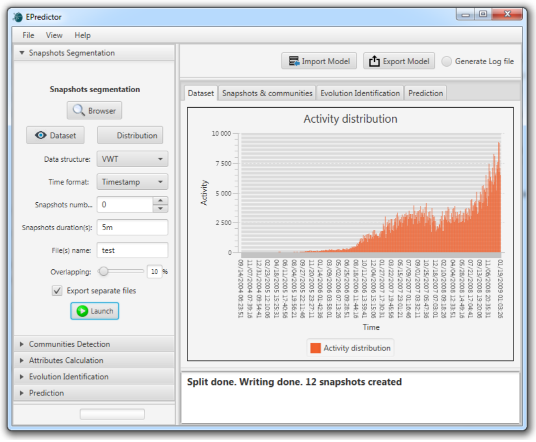
This module allows starting the community evolution prediction from the first step which consists of splitting the network data into several timeframes. EPredictor offers several parameters to perform the segmentation which is done according to one or more of the given parameters as shown in Fig.10. The most important are the following:

- Timeframe number where EPredictor generates a set of timeframes covering equal durations.
- Timeframe duration which is an alternative option to the previous one; indeed, instead of giving the resulting number of timeframes, the segmentation can be done according to the duration of each timeframe. There are two scenarios for using this parameter:
  - One time duration. Giving a single duration as a parameter means that the durations of all timeframes must be equal to this duration.
  - Multiple durations. The network data may not be uniformly distributed, so the segmented timeframes with equal duration can result in a sequence of empty timeframes. To solve this, splitting can be done using unequal durations and thus gathering the successive empty timeframes in a single one.

Furthermore, EPredictor supports different time units. In order to introduce a duration, it should be given in the form "dU" where d is an integer representing the value of the duration and U is the unit of time. The time unit could be {H: Hour, D: day, W: week, M: month, Y: year}.

- The overlapping rate: is the percentage of the intersection between two consecutive timeframes. This parameter is used to create common elements between the various consecutive timeframes and consequently slow down the evolution between them in order to better identify the evolution events. This continuity will be used for the identification phase of the subsequent evolution.

The timeframe data will be exported, following the user's choice, either in separate files or in one file with a supplement field indicating the adequate timeframe. The name of the files in which the data is exported should be given.



Data structure: VWT
Time format: Timestamp
Snapshots numb...: 0
Snapshots duration(s): 5m
File(s) name: test
Overlapping: 10 %
[✓] Export separate files

Split done. Writing done. 12 snapshots created

Figure 10: Timeframe segmentation.

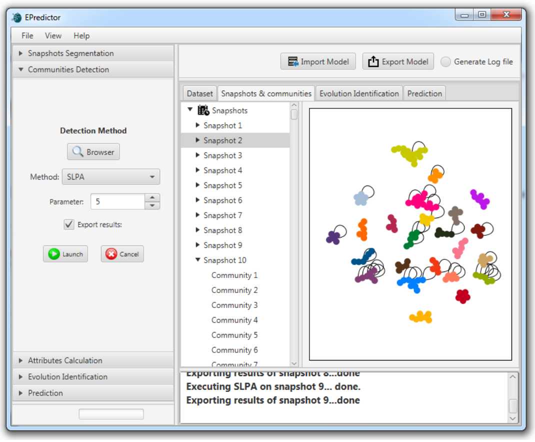
3.3 The Community Detection Module

The EPredictor platform offers the possibility to choose one among several community detection methods. Indeed, in order to compare the performance of different community detection algorithms on prediction of the community evolution, we included various algorithms commonly used in social network analysis studies (El Moussaoui et al., 2019), which are presented in Table 1. Two of them detect disjoint communities, while the others can find communities that overlap which is more realistic in real social networks.

Table 1: Community detection algorithms.

Method	Overlap
CPM (Clique Percolation Method)	✓
CONGA (Cluster-Overlap Newman Girvan Algorithm)	✓
COPRA (Community Overlap PRopagation Algorithm)	✓
CONCLUDE (COmplex Network CLUster DEtection)	✗
CM (CliqueMod)	✗
SLPA (Speaker-listener Label Propagation Algorithm)	✓

No matter which algorithm is used, at the end of the community detection on all timeframes, the results are displayed as shown in Fig.11. In this step, communities of all timeframes are detected. It is possible to locate the timeframe data on the disk directly, without going through the segmentation step. Then the method of detection is selected as well as its required parameter, if it exists, otherwise the parameter field is not editable. The user could choose whether or not to export the detection results.



Method: SLPA

Parameter: 5

Export results: ✓

```
Exporting results of snapshot 8...done
Executing SLPA on snapshot 9... done.
Exporting results of snapshot 9...done
```

Figure 11: Community detection.

3.4 The Attributes Computation Module

The next step of EPredictor is to compute a set of community features that may be important to determine community evolution and measure the communities from structural aspects (Tabassum et al., 2018) (Fig.12). These features encompass many structural properties; they can describe the community itself such as size, average density, cohesion, leadership, reciprocity, and so on. Alternatively, they can define the nodes belonging to a community such as the total degree of nodes, the incoming and outgoing degrees, the intermediate centrality, and so on. The explanations for the features/structural measures, available in EPredictor, are presented below:

- Size: the number of nodes in the community (n).
- Average Degree: the ratio of the sum of degrees of the nodes in the community to the number of nodes in the community:

$$\frac{1}{n}\sum_{u=1}^{n} d_u$$

where d_u means the degree value of node u in the community.

- Clustering Coefficient: a measure of how complete a node's neighborhood is to the number of nodes in the community.

$$CC_u = \frac{2t_u}{d_u(d_{u-1})}$$

where t_u is the the number of triangles in which node u participates.

- Density: a measure expressing how many connections between nodes are present in the community in relation to all possible connections between them:

$$\frac{2 \cdot E_{in}}{n(n-1)}$$

where E_{in} is the number of edges within the community.

- Diameter: the greatest shortest paths between any pair of nodes in the community:

$$max(\sigma(u,v)) \qquad \forall u,v \in V$$

where function $\sigma(u,v)$ is the shortest path from node u to v.

- Betweenness Centrality: a node measure describing the number of the shortest paths from all nodes to all others that pass through that node.

$$B_w = \sum_{u \neq v \neq w} \frac{\sigma_{uv}(w)}{\sigma_{uv}}$$

where σ_{uv} align is the total number of the shortest paths from node u to v and $\sigma_{uv}(w)$ is the number of those paths that pass through w.

- Cohesion: a measure characterizing strength of connections inside the group in relation to the connections outside the group:

$$\frac{2 \cdot E_{in}(n' - n)}{E_{out}(n-1)}$$

where E_{out} is the set of outer edges of the community and n' is the number of nodes in the network.

- Leadership: a measure describing centralization in the graph or community:

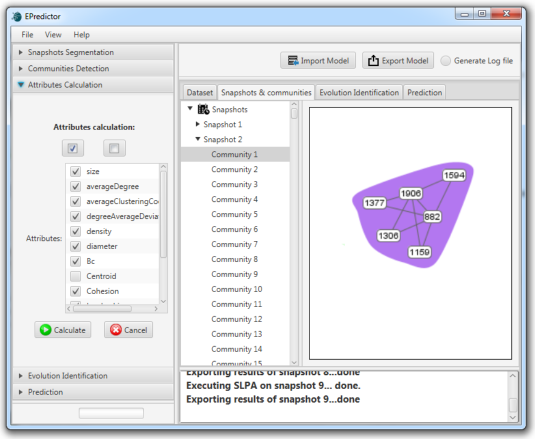
$$L = \sum_{u=1}^{n} \frac{d_{max} - d_u}{(n-2)(n-1)}$$

where d_{max} means the maximum value of degree in the community and n the number of nodes in the community.

- Closeness Centrality: a node measure defined as the inverse of the farness, which in turn, is the sum of distances to all other nodes:

$$C_u = \sum_{u \neq v} \frac{1}{d(u,v)}$$

where function $d(u,v)$ is distance from node u to v.



Attributes:
- [✓] size
- [✓] averageDegree
- [✓] averageClusteringCo
- [✓] degreeAverageDevia
- [✓] density
- [✓] diameter
- [✓] Bc
- [ ] Centroid
- [✓] Cohesion

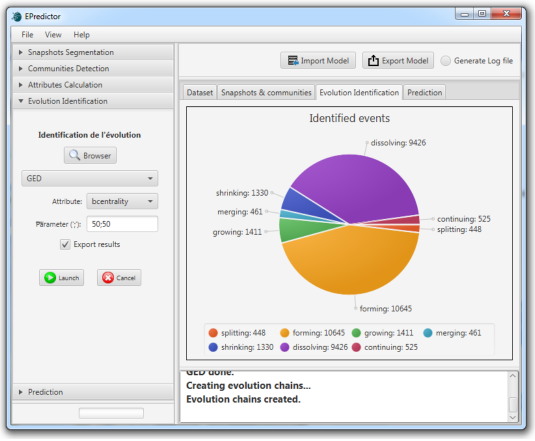
```
Executing SLPA on snapshot 9... done.
Exporting results of snapshot 9...done
```

Figure 12: Attribute computation.

3.5 The Evolution Identification Module

This step consists in applying a community tracker in order to determine the events depicting the evolutionary behavior of each detected community. Right now, there are two methods implemented in the EPredictor platform: GED (Group Evolution Discovery) (Bródka et al., 2013) and Asur's method. For both, it is possible to introduce the required parameters. For example GED depends on two parameters (α and β) as well as a node's importance measure that could be chosen among the list of attributes previously computed.

The results are displayed in a chart depicting the numbers of each identified event as shown in Fig. 13. If the user starts the prediction process from this step, community data is loaded directly from the disk. In this case, the attributes will be automatically computed before starting the evolution events identification. According to the user's choice, the results will be exported to a database file containing the detected community evolution sequences.



GED
Attribute: bcentrality
Parameter (';'): 50;50
[✓] Export results

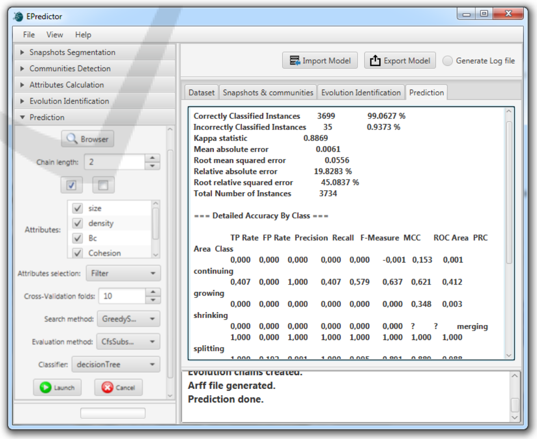
```
Creating evolution chains...
Evolution chains created.
```

Figure 13: Evolution event identification.

3.6 The Prediction Module

In the last step, users have the possibility to choose the classifier to be used for the prediction task. EPredictor includes several Weka classifiers: *NaiveBayes*, *BayesNet*, *J48*, *SMO*, *randomForest*, *decisionStump*, *perceptron*, *Logistic*, *RandomTree*, *iBk*, *oneR* and *Bagging* (Hall et al., 2009). Given the fact that classification algorithms are sensitive to data, it is pertinent to use several classifiers to perform credible comparisons.

As previously explained, community evolution sequences are used to train the classifier. Users could specify the length of community evolution sequences as well as the attributes describing a community instance from the list of the computed attributes (Fig.14). For the classification, it is possible to choose the attribute selection technique; it could be either manual, by filtering or by encapsulation. The manual selection considers the attributes notched by the user. For the selection by filtering or encapsulation, the user has to choose a research method that defines a subset of attributes to be used for the prediction. Then, the predictive capacity of this subset of attributes is evaluated. To export the results, if notched, an evaluation report will be exported in Pdf containing a synthesis of the built model of prediction. It is also possible to start the prediction process from this step by loading the community evolution sequences stored in an .arff file directly from the disk.



Chain length: 2
Attributes:
- [✓] size
- [✓] density
- [✓] Bc
- [✓] Cohesion

Attributes selection: Filter
Cross-Validation folds: 10
Search method: GreedyS...
Evaluation method: CfsSubs...
Classifier: decisionTree

```
Correctly Classified Instances      3699          99.0627 %
Incorrectly Classified Instances      35           0.9373 %
Kappa statistic                    0.8869
Mean absolute error                  0.0061
Root mean squared error                0.0556
Relative absolute error            19.8283 %
Root relative squared error          45.0837 %
Total Number of Instances            3734

=== Detailed Accuracy By Class ===

              TP Rate  FP Rate  Precision  Recall  F-Measure  MCC      ROC Area  PRC
Area  Class
              0,000    0,000    0,000      0,000    0,000      -0,001   0,153     0,001
continuing
              0,407    0,000    1,000      0,407    0,579      0,637    0,621     0,412
growing
              0,000    0,000    0,000      0,000    0,000      0,000    0,348     0,003
shrinking
              0,000    0,000    0,000      0,000    0,000      0,000    ?         ?        merging
              1,000    0,000    1,000      1,000    1,000      1,000    1,000     1,000
splitting
```

```
Arff file generated.
Prediction done.
```

Figure 14: Evolution prediction results.

3.7 Results Visualization

EPredictor allows to visualize the results of the different treatments of the community evolution prediction process. More specifically, the distribution of network activity over time (Fig.10), the detected communities and their timeframes (Fig.11, Fig.12), evolution identification results (Fig.13) and prediction results (Fig.14). It also allows to export an evaluation report containing all user entries and statistics of different results throughout the prediction process.

4 CONCLUSION

In this paper, the EPredictor platform, a research tool, was described and its usability was presented in detail for the community evolution prediction issue. To the best of our knowledge, no other research tools, that handle the entire community evolution prediction process, were proposed for predicting community evolution in dynamic social networks. We have presented step by step all the options offered by the platform in order to make visible the range of functionalities which can be involved for one particular network. Several experiments were conducted by students for their Master dissertation, which proved the usability of the current version of the EPredictor platform for educational purposes. We meant, through this paper, to present the EPredictor platform to the researchers in the field of social network analysis in order to prove its usability for research purposes. Yet, the platform is still being developed and new functionalities are constantly incorporated in order to enrich the set of methods proposed by the platform for each step. The EPredictor platform is now available for academic and other non-commercial purposes.

REFERENCES

Asur, S., Parthasarathy, S., and Ucar, D. (2009). An event-based framework for characterizing the evolutionary behavior of interaction graphs. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 3(4):1–36.

Bródka, P., Saganowski, S., and Kazienko, P. (2013). Ged: the method for group evolution discovery in social networks. *Social Network Analysis and Mining*, 3(1):1–14.

Dakiche, N., Benbouzid-Si Tayeb, F., Benatchba, K., and Slimani, Y. (2021). Tailored network splitting for community evolution prediction in dynamic social networks. *New Generation Computing*, pages 1–38.

Dakiche, N., Benbouzid−Si Tayeb, F., Slimani, Y., and Benatchba, K. (2018). Tracking community evolution in social networks: A survey. *Information Processing & Management*.

Diakidis, G., Karna, D., Fasarakis-Hilliard, D., Vogiatzis, D., and Paliouras, G. (2015). Predicting the evolution of communities in social networks. In *Proceedings of the 5th International Conference on Web Intelligence, Mining and Semantics*, pages 1–6. ACM.

El Moussaoui, M., Agouti, T., Tikniouine, A., and El Adnani, M. (2019). A comprehensive literature review on community detection: Approaches and applications. *Procedia Computer Science*, 151:295–302.

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., and Witten, I. H. (2009). The weka data mining software: an update. *ACM SIGKDD explorations newsletter*, 11(1):10–18.

Holme, P. and Saramäki, J. (2012). Temporal networks. *Physics reports*, 519(3):97–125.

İlhan, N. and Öğüdücü, Ş. G. (2016). Feature identification for predicting community evolution in dynamic social networks. *Engineering Applications of Artificial Intelligence*, 55:202–218.

Pavlopoulou, M. E. G., Tzortzis, G., Vogiatzis, D., and Paliouras, G. (2017). Predicting the evolution of communities in social networks using structural and temporal features. In *2017 12th International Workshop on Semantic and Social Media Adaptation and Personalization (SMAP)*, pages 40–45. IEEE.

Rajita, B., Ranjan, Y., Umesh, C. T., and Panda, S. (2020). Spark-based parallel method for prediction of events. *Arabian Journal for Science and Engineering*, pages 1–17.

Saganowski, S., Bródka, P., Koziarski, M., and Kazienko, P. (2019). Analysis of group evolution prediction in complex networks. *PloS one*, 14(10):e0224194.

Tabassum, S., Pereira, F. S., Fernandes, S., and Gama, J. (2018). Social network analysis: An overview. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 8(5):e1256.

Takaffoli, M., Rabbany, R., and Zaïane, O. R. (2014). Community evolution prediction in dynamic social networks. In *2014 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pages 9–16. IEEE.