Monitoring Network Evolution using MDL

Jure Ferlež [#1], Christos Faloutsos [*2], Jure Leskovec [*3], Dunja Mladenić[#4], Marko Grobelnik[#5]

[#]*Department of Knowledge Technologies, Jožef Stefan Institute*
Jamova 39, 1000 Ljubljana, Slovenia
[1]jure.ferlez@ijs.si
[4]dunja.mladenic@ijs.si
[5]marko.grobelnik@ijs.si

[*] *Machine Learning Department, School of Computer Science, Carnegie Mellon University*
5000 Forbes Ave, Pittsburgh, PA 15213, USA
[2]christos.faloutsos@cs.cmu.edu
[3]jure.leskovec@cs.cmu.edu

Abstract— **Given publication titles and authors, what can we say about the evolution of scientific topics and communities over time? Which communities shrunk, which emerged, and which split, over time? And, when in time were the turning points? We propose TimeFall, which can automatically answer these questions given a social network/graph that evolves over time. The main novelty of the proposed approach is that it needs no user-defined parameters, relying instead on the principle of Minimum Description Length (MDL), to extract the communities, and to find good cut-points in time when communities change abruptly: a cut-point is good, if it leads to shorter data description. We illustrate our algorithm on synthetic and large real datasets, and we show that the results of the TimeFall agree with human intuition.**

I. Introduction

Time evolving graphs appear in a wide range of disciplines and application domains. These datasets from various domains have in common that they can be represented as a network which changes over time. Examples of such time-evolving graphs are publication databases, (e.g. PubMed MEDLINE) and the dynamic on-line social networks (e.g. Orkut and Facebook).

For example, in the setting of scientific community evolution we start with a set of papers (authors, abstracts and time of publication) which define a time evolving network. We would like to answer the following questions: Which are the coauthoring communities of researchers that are growing in size over the years. Which communities are diminishing in size but gaining in density? Once we know answers to these questions we can also explore: What are the characteristics of the communities that have split? Can we predict such splits or merges of the communities? Can we associate topics with communities of researchers?

Here we propose TimeFall ("time waterfall"), an approach for analysis of network evolution. TimeFall addresses several of the above questions simultaneously, by enabling analysis of the evolution of network clusters through time. An example of the TimeFall visualization is displayed in Figure 1.

In Figure 1 each box represents a community of words – a user profile topic. Each topic is thus described by a set of keywords that are characteristic for the corresponding topic. Each line (horizontal group of topics) represents a time step, and arrows between the topics from the adjacent time steps represent the evolution of the topics. Notice the splits and merges of the topics over time.

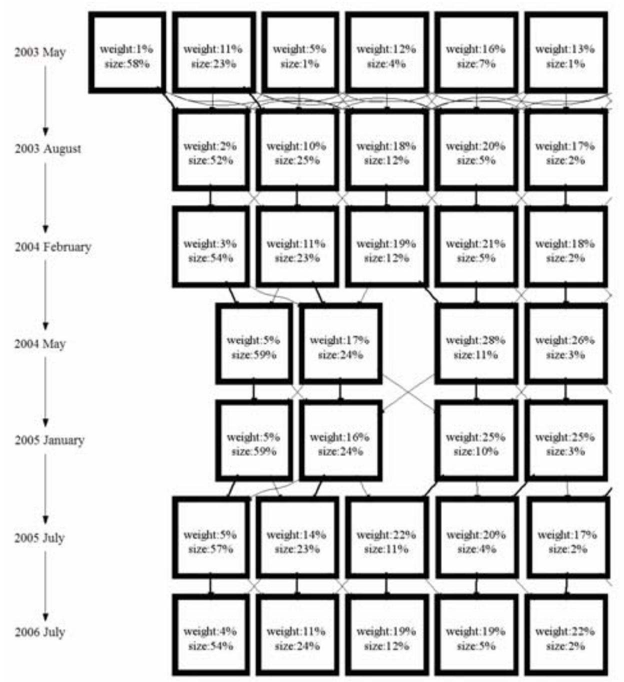

Fig. 1 TimeFall visualization of the evolution of user profile descriptions inside a large on-line social network. We analyse and visualize the topic evolution of 7.5 million textual profiles over the period of 3 years. The analysis reveals the profile topics (keywords omitted for confidentiality) and cut-points in time when evolution is especially abrupt.

Let us now more formally define the problem at hand:

PROBLEM: Parameter-free network evolution tracking

Given: n time-stamped events (like, e.g., papers published), each related to several of m items (like, title-words, and/or author-names, and/or publisher names)

Find: simultaneously (a) the communities, that is, item-groups (e.g., research topics and/or research communities) and (b) describe how the communities evolve over time (e.g., appear, disappear, split, merge) and (c) select the appropriate cut-points in time when existing community structure change abruptly.

Without: any user defined parameters.

The rest of the paper is organized as follows. We continue in Section 2 by introducing the proposed TimeFall algorithm for monitoring and visualizing network evolution. Section 3 presents the experimental results. We conclude this paper in Section 4.

II. Proposed Method

We begin by introducing the TimeFall algorithm. For simplicity reasons we will assume a bipartite graph with time stamps, and produce a visualization of the evolving network clusters (topics, communities) over time. TimeFall can be trivially extended to handle multi partite graphs.

Our goal is to operate on such a time-stamped graph, to automatically spot communities, their evolution and cut-points between epochs of stable community evolution. The intuition behind our approach is to organize the time-stamped adjacency matrix in such a way that it is easy to compress. This exactly fits the Minimum Description Length (MDL) principle [2] of using some data description language to produce as short data description as possible. In general, the idea is to treat the problem as a compression problem, because it will guide us to find patterns (natural cut-points, natural communities), eliminating the need for the user-defined parameters.

We designed the TimeFall description language, which facilitates a short temporal graph description by providing efficient means of communicating the most probable temporal patterns observed in a graph like merging, splitting and evolution of the communities. The language provides means to efficiently describe the difference between the clustering of two matrices (graphs). When the matrices describe the network at two consecutive time points, then the difference in clustering represents the network community evolution. We show that the differences in clustering of two matrices can be efficiently described by using mutual information between the matrices, i.e., we use the clusters (communities) at time t to efficiently describe the communities at time t+1. This way we can detect patterns like splits and merges of communities, and also birth of new and diminishing of old communities. We used a simple hill climbing algorithm for searching for these patterns and the corresponding short matrix (graph) description.

The designed network description language and the short description search algorithm extend the existing Cross Association [2] community membership description language and search algorithm. The extension enables efficient modelling of temporal changes in the network at the same time preserving the efficiency and the absence of parameters of the original algorithm. What is more, both the original Cross Associations algorithm and the developed TimeFall algorithm can be effectively parallelized to achieve linear speedups. An overview of the TimeFall approach to detecting network evolution can be presented on an example of monitoring topic development in research papers as depicted in Figure 2.

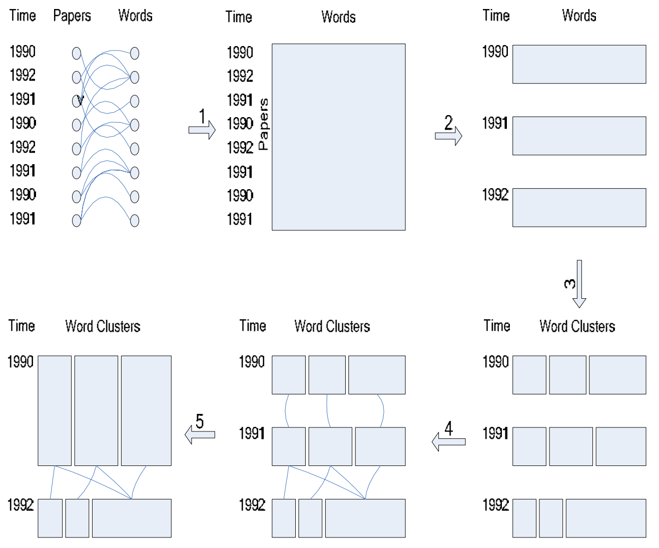

Fig. 2 TimeFall approach illustrated on a temporal bipartite graph representing papers published over time and words from their content. The approach first represents the graph in the form of an adjacency matrix (step 1). The approach then splits the rows according to their time stamps (step 2) and uses the CA algorithm to cluster the columns of the connection matrices (step 3). It then utilizes the MDL principle to connect the column clusters of the matrices (step 4) and reduce the unimportant time points in the graph evolution history (step 5).

III. Experiments

We evaluated our algorithm on synthetic datasets and on a real-world dataset obtained from a large social network. The synthetic data was used to test the algorithm under the controlled conditions. The real-world dataset was used to show the algorithm's ability to identify important time points in the history of the social networking web site.

A. Experiments on synthetic data

Our fist experiment on a synthetic dataset aims at testing the the TimeFall algorithm on producing expected results in a controlled environment, The synthetic dataset has an underlying structure that can be shown by a binary matrix displayed in the centre of Figure 3. The synthetic dataset we used is a binary matrix with 6000 rows and 10000 columns. The matrix contains different blocks of ones (marked with black) while the rest of the matrix contains zeros. Rows of the matrix are marked with time stamps spanning from 1 to 6. We can observe that the rightmost cluster (*B*) is fairly stable over time, mainly keeping the same columns (more precisely, from time 2 to time 3 it grows a bit from size 25% (*B***)** to size 31% (*G*) and then gets back to 25% (*B'*) at time 5). The bigger cluster (*A*) at time 1 and 2 remains the same and then at time 3 it splits into several smaller clusters (*a*,*b*,*c*,*d*,*e*,*f*) that merge back in time 5 (*A'*).

After permuting the rows and columns of this matrix (left hand side of Figure 3) to get a realistic dataset and applying the TimeFall algorithm we get the visualization displayed on the right of Figure 3. We see that the algorithm succeeded in identifying the main structure of the column clusters and the changes over time and the correct pattern of evolution. What is more it correctly merged the time points with similar clusters (time 1 and 2, 3 and 4, 5 and 6) producing an effective summary of the temporal development of the graph.

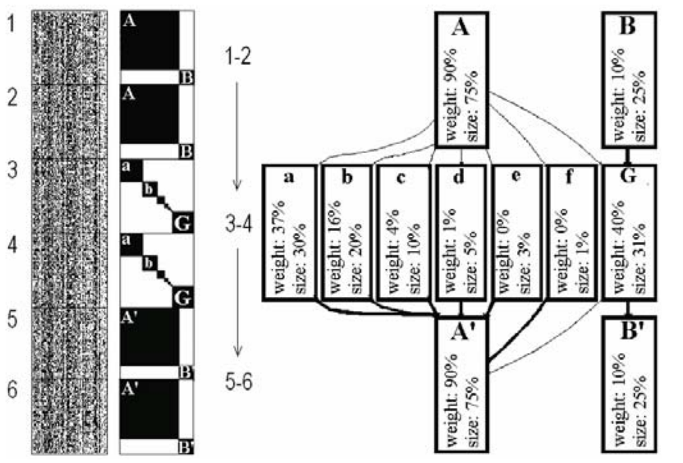

Fig. 3 Permuted synthetic binary matrix of 6000 rows and 1000 columns (left) and the nicely blocked version of the same matrix (centre) represents a time evolving graph from time 1 to time 6. Black parts represent ones and white parts represent zero values. TimeFall analysis of the permuted matrix yields a correct visualization of the network evolution (right).

In the second experiment on synthetic data we present the scalability of our approach. We created a random binary matrix with 1000 columns and varying number of rows. We varied the number of rows from 1000 to 128,000. The matrix contained 30 percent of non-zeros and 70 percent zeros. We created random time stamps for every row of the matrix. We run the base and the parallelized version of the TimeFall on a server machine with 8 logical processors and recorded the running time and memory requirements in Figure 11.

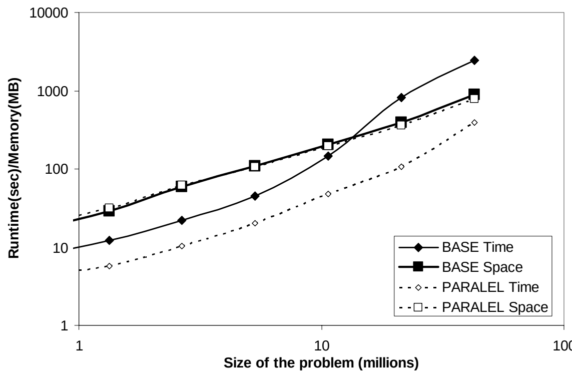

Fig. 11 Time and space requirements of the algorithm. Notice that parallelized version of the algorithm requires same memory but runs up to 8 times faster.

We observe that the time and space requirement is linearly dependent on the problem size. We noticed that the parallelized version of the algorithm takes the same amount of memory as baseline single threaded implementation. However, the parallel version runs 2 to 8 times faster than single threaded implementation of the algorithm. This gives a significant speedup over the single threaded version and allows scaling to large datasets.

B. Experiments on a Large Social Network

We obtained the evolving social network from anonymized real-world on-line social networking service, like Orkut, or Facebook. The network contains 7.5 million people with 53 million time stamped links among them. Each link signifies a personal tie established at a given point in time, and all nodes are labelled with textual profiles. The data spans a period from May 2003 to October 2006.

The goal of the analysis was to get an insight into the growth of this social network through time. We were especially interested in recognizing which word groups (topics) in user profiles are most suitable for the description of the network expansion.

In order to use the TimeFall algorithm to answer this question we constructed an invitation matrix. Rows of the matrix correspond to invitations from existing users of the network to the members outside of the network. The columns of the matrix describe every invitation by words from the profile of the inviter. Every row of the matrix corresponds to one invitation and is thus associated with a date stamp.

Applying the TimeFall algorithm to this matrix yields the diagram in Figure 1. From the diagram one gets the impression that earlier in time the profile topics changed more rapidly (we found many intervening links between first two epochs of stable development between May and August 2003). This behaviour settles down later in time when fluctuation of words between the topic clusters very much diminishes between consecutive epochs. Most importantly, the extracted cut-points in time between consecutive epochs were recognized to be in correspondence with the important changes in the functionality of the social networking site.

IV. Conclusion

Analysis of large time graphs can provide an insight into temporal development of a graph, which can represent different phenomena, such as evolution of research collaboration and research topics. Our work provides an efficient parallelizable approach that addresses this task in a principled MDL way and without a single parameter provided by the user. Our experiments show that the proposed approach reveals the correct (synthetic dataset) and intuitive patterns (social network dataset) in evolution of networks.

Acknowledgment

This work was supported by SRA - Analysis of Dynamic Networks with Graph and Text Mining Methods, EC - NeOn (IST-4-027595-IP) and the U.S.A. NSF (IIS-0534205, IIS-0705359)

References

[1] D. Chakrabarti, S. Papadimitriou, D. S. Modha, and C. Faloutsos. Fully automatic cross-associations. *In KDD*, 2004.

[2] A.Barron, J.Rissanen, and B.Yu, The minimum description length principle in coding and modeling. IEEE Trans. Information Theory, vol. 44 (1998), no. 6, pp. 2743-2760.