Link prediction with sequences on an online social network

Benjamin Fasquelle[1]
supervised by
Renaud Lambiotte[2]

[1] École Normale Suprieure de Rennes,
Campus de Ker lann, Avenue Robert Schuman, 35170 Bruz, France
http://www.ens-rennes.fr
[2] UNIVERSITY OF NAMUR,
rue de Bruxelles 61, B-5000 Namur, Belgium
https://www.unamur.be/en

May 15 - August 1, 2017

Abstract. During this internship, I studied a dynamic social network and modelled it to understand its growth. Thus I propose a new model of graph, the sequence model, which is a model with growth rules corresponding to observations on the real graph. I study the properties of the model by experiments, then I use it for prediction link, and compare its results with the classic methods for link prediction. Finally, I propose a hybrid model for link prediction, using the sequence model and the Common Neighbors model.

Keywords: Social network, link prediction, graph modelling

Table of Contents

1 Introduction

In several scientific fields, objects of interest can be studied as a network: a set of points, the nodes (or vertices), joined together in pairs by lines, the edges. The edges could be directed if they run in one specific direction, or undirected if they run in both directions. For example, there are the Internet in technological networks, biochemical networks in Biology, online social networks like Twitter or Facebook networks, or also other social networks like co-authored papers networks.

The structure of a system can have big effects on the behavior of the system. The connections in a social network affect information spreading, or also the spread of disease. The structure of the Internet network affects the routes followed by the data, and the efficiency of the data transportation.

In this report, I work only on online social networks, in particular on a Facebook network. In such a network, the nodes are people and there is an edge between two people if they are friends (this is an undirected network here). This network is dynamic: we know when the edges appear, so we have a snapshot of the network at each time step (here, it is one second).

First, I will present some of the most famous graph models and classic link prediction methods. Second, I will describe the dataset and its properties. Then I will propose a new graph model, based on sequences of new edges. Finally, I will use this model for link prediction, and compare it with the classic methods.

1.1 Real-world online social networks properties

First, I present some properties and measures on networks which will be useful thereafter.

Some real-world networks seem to have a power-law degree distribution, or at least for the tail of the distribution (the nodes with high degree). A perfect power-law distribution is the distribution where the probability p_k that a node has a degree k is given by:

$$p_k = Ck^{-\alpha}$$

where C and α are constants. Networks with power-law degree distribution are often called *scale-free networks*, because scaling the degree k by a constant factor c causes only a proportionate scaling (by the factor $c^{-\alpha}$) of the degree distribution itself:

$$p_{k\times c} = C(k \times c)^{-\alpha} = c^{-\alpha}Ck^{-\alpha}$$

To detect power-law distribution, we can plot the distribution on logarithmic scales. A power-law distribution histogram has a straight-line shape with these scales. Indeed, we have:

$$log(p_k) = log(Ck^{-\alpha}) = log(k^{-\alpha}) + log(C) = -\alpha log(k) + log(C)$$

Figure 1 shows such a distribution, with the example of the Internet graph at the level of autonomous systems. A lot of real-world networks seem to be scale-free, but we have to be very cautious when interpreting that, and we should not give them a sense of universality [12].

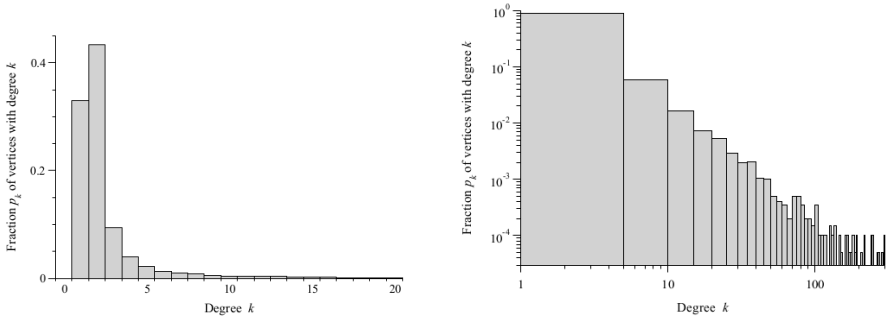


Fig. 1: The power-law degree distribution of the Internet, from [9]. On the right, logarithmic scales are used, and we can see the straight-line shape of the histogram.

If we define the distance between two nodes as the size of one of the shortest path between them, the diameter of a network is the size of the biggest distance between two nodes in the network. In many real networks, the diameter is very small (not more than 10, even for networks with thousands of nodes). Consequently, the mean distance between two nodes is also surprisingly small. This phenomenon is called the *small-world effect*.

The clustering coefficient is a measure to know how the nodes tend to cluster together. Precisely, it is the average probability that two neighbors of a node are connected. In other words, it is the density of triangles in the network. In real-world networks, and in particular in the social networks, the clustering coefficient is quite high: it is often between 0.2 and 0.5.

Another measure with high values in social networks is the modularity, which reveals how the network can be divided into groups. To get this measure, we need to divide the network into separate communities. For a given set of communities, the modularity is the fraction of the edges that are within the communities minus the expected fraction if the edges were distributed at random. Some community detection algorithms work by searching the set of communities which maximize the modularity. With communities detected by algorithms, real-world social networks have a high modularity, often more than 0.4.

1.2 Graph models

One of the best ways to understand real-world networks is to build mathematical models. It could explain how a network had formed and how it develops.

We can classify the network models in two groups: those where all the nodes are there and edges are chosen following specified rules, and generative models in which the network grows (nodes coming in one by one) against growth rules.

Random graphs

Random graphs point out models in which a specific set of parameters take fixed values, but the rest is random. For example, one of the simplest random graph is the one where the number of nodes and the number of edges are fixed. To build a graph following this model with n nodes and m edges, one just has to choose m edges uniformly at random among all possible pairs of nodes, within a set of n nodes. There is a property of such a random graph that is easy to compute: the mean degree. Indeed, the mean degree of nodes in such a graph is $\frac{2\times m}{n}$ (each edge is counted two times because it joins two nodes). Unfortunately, other properties of this model can not be computed easily.

Another simple random graph is the one in which we fixe the number of nodes n and the probability p of edges. To build a graph, each possible pair of nodes has an idependent probability p to be joined by an edge. For this model, we have interesting analytical results. I don't give details of how we obtain them, but they follow our intuition. First, the mean number of edges in such a graph is $\binom{n}{2}p$, that is the number of all possible pairs of nodes multiplied by p. With this result, we are able to compute the mean degree: $(n-1)p$, which is the number of possible neighbors for each node multiplied by p. Another result is the degree distribution which is a Poisson degree distribution (in the large-n limit). This is different from real networks degree distribution, that is the big problem of this model.

To solve this problem, we can use models which can have the degree distribution we want, like the configuration model. In this model, the degree distribution is fixed: each node in the network has a fixed degree. Consequently, the number of nodes and the number of edges are fixed. To create a graph, each node with degree k get k stubs of edge, then pairs of stubs are chosen randomly and connected, until it doesn't remain any stub. The problem of this model is the clustering coefficient which, for a given degree distribution, decreases linearly against the number of nodes n, while some real-world networks, in particular the social networks, have a quite high clustering coefficient.

Generative models

We have seen models which can have the degree distribution we want, but these models don't explain why the network should have this degree distribution. Generative models are models in which the network grows, so they offer an explanation of the properties of real-world networks. Indeed, if a model with a set of fixed growth rules gives networks with the similar structures than real networks, it suggests that similar generative mechanisms may work in real networks.

The most famous generative model is the preferential attachment model. A first version was proposed by Price [10] to understand the network of citations of scientific papers, and gives directed and acyclic networks. I detail here a second version, independently discovered by Barabási and Albert [1], which gives undirected networks.

In this model, nodes are added one by one to a growing network and each node connects, when it is added, to a randomly chosen set of previously existing nodes. The number c of new connections is fixed and is the same for all the nodes. It implies that all the nodes have a degree bigger than c. When a new node is added, the probability there is a connection with a previously existing node is proportional to its current degree. One can show that this model generates networks with a degree distribution with a power-law tail with an exponent $\alpha = 3$. Thus, the BarabásiAlbert model is a simple model which has a power-law degree distribution, like most of the real networks. However, the exponant of the power-law is fixed at 3, and it is not the exponant of all the concerned real networks.

1.3 Link prediction

The link prediction problem is the following: given a snapshot of a social network, which new edges are likely to appear in the future? Most of the link prediction methods are based on measures of the proximity, or similarity, of the nodes [7]. I present here some of these measures, which will be used.

Suppose we have a graph $G = (V, E)$, where V is the node set and E the edge set. Let $\overline{G} = (V, \overline{E})$ where $\overline{E} = \{(u, v) | (u, v) \notin E\}$. We want to find edges in $\overline{E}$ which will appear in G. Thus, the methods assign a connection weight $score(u, v)$ to each edge (u, v) in $\overline{E}$, and create a ranked list of edges in decreasing order of $score(u, v)$. Let $\Gamma(u)$ denote the set of neighbors in G of the node $u \in V$.

Common Neighbors: This is the simplest measure, which only counts the number of common neighbors: $score(u, v) = |\Gamma(u) \cap \Gamma(v)|$.

Adamic/Adar index: This index refines the simple counting of common neighbors by assigning the less connected neighbors more weights: $score(u, v) = \sum_{w \in \Gamma(u) \cap \Gamma(v)} \frac{1}{log|\Gamma(w)|}$.

Preferential attachment: This measure assumes that the probability that a new edge involves node u is proportional to $|\Gamma(u)|$ (as in the preferential attachment generative model presented previously). On the co-authored paper network, empirical evidence shows that the probability of co-authorship of u and v is correlated with the product of the number of collaborators of u and v. Because of this, the measure is defined as: $score(u, v) = |\Gamma(u)| \times |\Gamma(v)|$.

Jaccards coefficient: The Jaccard's coefficient is a statistic used for comparing the similarity of sets. More precisely, it is defined as the size of the intersection divided by the size of the union of the sets. We can use it to define a measure, comparing the similarity of the neighbor sets: $score(u, v) = \frac{|\Gamma(u) \cap \Gamma(v)|}{|\Gamma(u) \cup \Gamma(v)|}$.

Resource allocation index: The resource allocation process is originally proposed to explain the correlation between transportation capacity and connectivity of airports [15]. Considering two nodes u and v in $\overline{G}$ (they are not connected in G), u can send some resource to v by transmitters, which are their common neighbors. Each transmitter has a unit of resource, and averagely distribute it to all its neighbors. The measure for two nodes u and v is defined

as the amount of resource v received from u (or from v to u, it is symmetric): $score(u,v) = \sum_{w \in \Gamma(u) \cap \Gamma(v)} \frac{1}{|\Gamma(w)|}$.

2 Dataset study

The dataset is a Facebook graph collected in the SensibleDTU project, which used smartphones to measure and understand social behavior. The data collection took place at the Technical University of Denmark, from October 2013 to June 2014, on 1,000 newly started students. Thus, we have a dynamic network of real world person-to-person interactions between approximately 1,000 individuals (densely-connected population). With all the measure of the experiment (phone call, text messages, Facebook data, how people move around in space (via GPS)), social groups can be identified [11]. Here, we only have the dynamic Facebook network of the experiment.

2.1 Network properties

At the end of the collection, the network has a total of 874 nodes and 8,952 edges. The mean degree is around 20.5. Figure 2 shows the degree distribution of the graph at the end of the collection. This distribution is not usual, because it seems the network is not a scale-free one. Indeed, it fits better with a log-normal distribution (which correspond to the multiplicative product of many independent random variables) than with a power-law one.

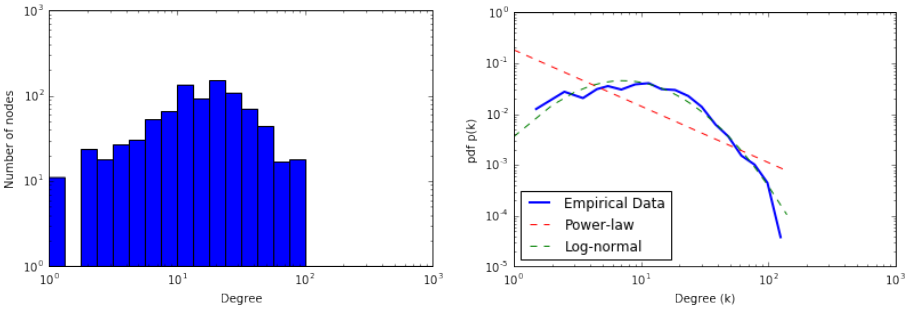


Fig. 2: Degree distribution of the network

The diameter of the graph is 6, that is small, as expected for social networks. The clustering coefficient is around 0.25, that is usual for social networks, which have usually higher clustering coefficient than other networks. With the Louvain algorithm [3], which is a community detection algorithm, we obtain a modularity score of 0.45, that shows there are clusters in the network.

Excepted for the degree distribution, which is not a power-law one, the properties of the network are usual.

2.2 How the new links appear

We have seen the properties of the network at the end of the data collection, but it is also interesting to observe how the network evolves over time. Such observations could permit to make more realistic models, like the forest fire model [6].

Figure 3 shows the evolution of the number of nodes and edges in the network during the year. We can see that most of the nodes have been already in the network before the beginning of the data collection, and most of the new edges and new nodes appear at the beginning of the data collection, probably because of the start of the school year. Indeed, as people are in new classes, they could meet unknown people, who are potential new friends, at this time.

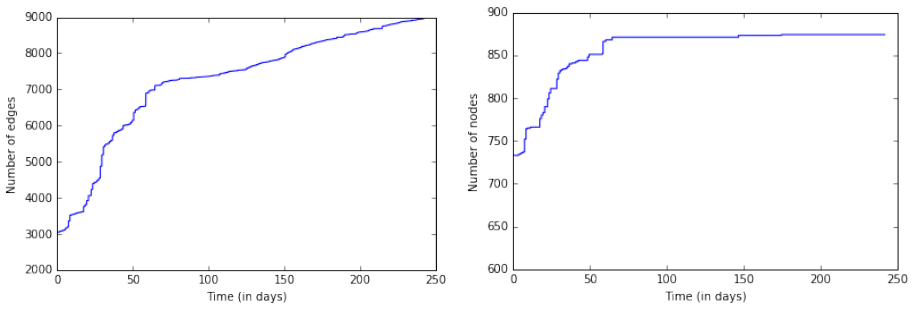


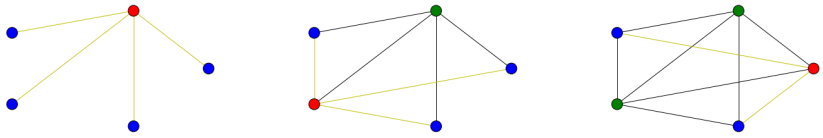
Fig. 3: Number of edges (on the left) and nodes (on the right) in the graph in the time

When we look the new links in arrival order, we can see that they are arranged in sequences: successively, there is a node (I call it the active node) which becomes friend with several other nodes, which constitute a set I call sequence. We can also see that there is a link between the successive sequences: when a node adds new friends, those new friends have an inclination towards becoming themselves friends. Moreover, when a node adds new friends, it is often one of its new friends that becomes active. Figure 4 shows this phenomenon.

We can see on Figure 5 the sizes of the sequences in the time. We can observe large sequences at the beginning of the year, then the sizes decrease. This decline reinforces the idea that sequences rely on previous ones. A sequence with a given size seems to create new sequences with lower sizes, which are subsets of the initial sequence.

3 Sequence model

I propose here a new graph model, based on the sequences of the previous section. The main idea is that a node adds new links with several others, which



(a) Step 1 (b) Step 2 (c) Step 3

Fig. 4: Example of sequences: at each step, the red node is the active one, the blue nodes are the sequence (the new friends of the active node) and other nodes are in green. New edges at each step are in yellow.

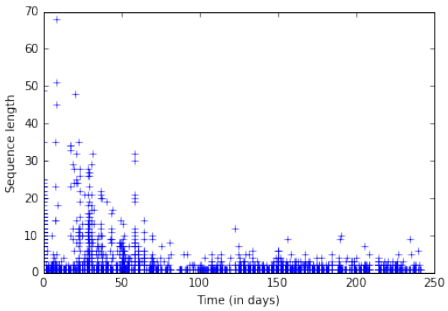


Fig. 5: Size of the sequences in the time

constitute a sequence, and this sequence implies other similar sequences. This model is closer to generative models than others because, although all the nodes are present at the beginning, the edges are added step by step, sequences by sequences. This model is based on the closure of recently opened triangles, and thus combines ingredients of triadic closure [2] [5] and burstiness observed in temporal networks [4].

3.1 Model and algorithm

The model to build graph works by iteration. All the nodes have an activation rate and a potential sequence of nodes. At each iteration, a node becomes active (chosen randomly with the activation rate: the higher the activation rate is, the higher the probability that the node is chosen is), and it creates a link with nodes which constitute a new sequence: all the nodes in the potential sequence have a high probability to be in the new sequence, others a low one. For all nodes in the new sequence, their own potential sequence becomes the new one except the node itself, and their activation rate increases.

The entry variables of the model are:

- N: Number of nodes
- I: list of initiators and their first sequences
- a: initial activation rate
- a_{seq}: activation rate when a node appears in a sequence the first time, bigger than a
- m: multiplicative factor of the activation rate for nodes that appear in several sequences
- p: When a node becomes active, it has a probability p to create a link with each node in his potential sequence
- ϵ: When a node becomes active, it has a probability ϵ to create a link with each node that is not in his potential sequence
- K: Iteration number (at each iteration, a node is activated)

The list of the initiators and their first sequences can be chosen randomly. In my experiments, I only give the number of initiator and the average size of the first sequences, then the algorithm builds them randomly.

Algorithm:

First, we have to initialize the graph and the first sequences. For each node, we initialize the potential sequence (empty) and the activation rate (a). Then we initialize the graph, and add links between initiators and their sequences. For nodes in those sequences, we increase the activation rate and update their own potential sequence.

After that, we do K times the following process:

- We choose the node to activate (randomly, with the activation rate)
- We choose the new links for the active node against its potential sequence, and update activation rates and potential sequences of new friends

The more detailed algorithm is presented in Appendix.

3.2 Properties

Unfortunately, the model is too complicated to have interesting analytical results. Because of this, I present here only experimental results about the model. For my experiments, I use the following parameters: 1000 nodes, 50 initiators with initial sequence of random size between 10 and 80, $a = 10^{-6}$, $a_{seq} = 10^{-2}$, $m = 1.2$, $p = 0.7$, $\epsilon = 10^{-7}$, and $K = 5000$ iterations.

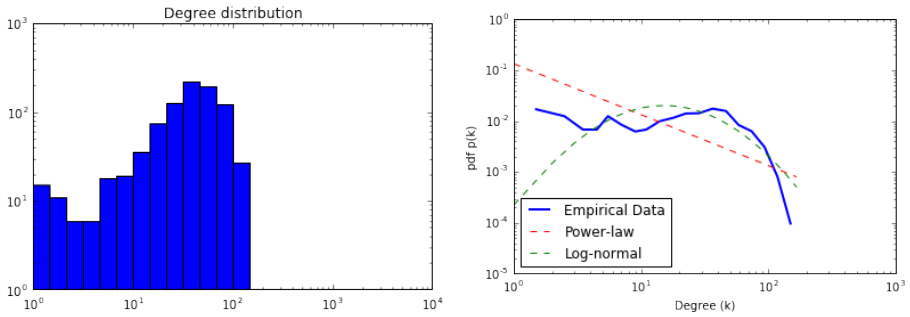


Fig. 6: Degree distribution of a graph built with the model

Figure 6 shows the degree distribution of a graph obtained with the model. We can see that it is similar to the real graph one, and it fits better with a log-normal distribution than with a power-law one. However, we can see quite a large number of nodes with low degree (less than 2), which seem to be nodes that were not in one of the initial sequences.

Figure 7 shows the sizes of the sequences in the time. In average, they decrease, like the dataset sequences sizes we observed previously.

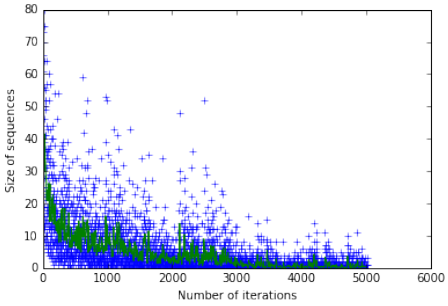


Fig. 7: Size of the sequences by arrival order (local mean in green)

Others properties are presented in the Figure 8. I obtained them building 100 graphs with the model, and for each of them, the initiators and initial sequences

are chosen randomly. We can notice that the standard deviation is always around 10% of the average. All the values are similar to those of the real graph. This model is interesting because it has all the properties of the real graphs. Even if we have not analytical results, we can give an intuition for the obtained properties. Indeed, as sequences rely on previous local ones, it encouraged the creation of clusters, that explains high clustering coefficient and modularity. The low diameter can be explained by the fact that some nodes belong to several clusters (those which are in several sequences at the beginning), that reduces the distance between nodes in the clusters, and the fact that nodes in a same cluster are very close.

	Number of edges	Diameter	Modularity	Clustering coefficient	Build time
Average	25,114	5.1	0.372	0.337	10.5s
Standard deviation	2094	0.50	0.02	0.012	0.6

Fig. 8: Properties of the model

The clustering coefficient is also similar to the real graph one, and as we can see one Figure 9 it relies on the probability p to make new links with the nodes in the potential sequence. The clustering coefficient increases with p, as expected.

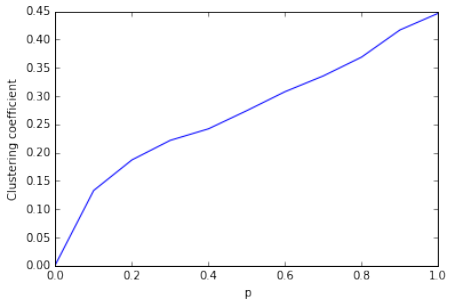


Fig. 9: Clustering coefficient against p

This model has some drawbacks. On one hand, it has a lot of parameters. In comparison, models presented in the introduction have not more than 2 parameters. On the other hand, build a graph with this model is quite expensive in time (the complexity of the algorithm is $O(K * N)$, in comparison the complexity in time of the configuration model presented in the introduction is $O(N+MlogM)$, where M is the number of edges [8]), but also in space (we have to save two variables by nodes, so the space complexity is $O(N)$). In spite of this, we have

networks with the same properties than real-world networks, and which are built with a similar mechanism that the dataset network. Such a mechanism could be the reason of observed properties of real online social networks like Facebook ones.

4 Link prediction

We can give at the sequence model particular initiators and initial sequences as entry variables. Thus, if we have real sequences, we can use the model for link prediction. It should be underlined that such a prediction doesn't follow the definition given in the introduction: here we don't only have a snapshot of the network, but we have also temporal data, the sequences.

4.1 How to use the model for link prediction ?

We can use this model to predict the future links in a network. We could only define an observation period, then save the sequences (and the associated active nodes) during this period, and use the sequence model algorithm with this data. However, there would be all the parameters to choose (we could use the Approximate Bayesian Computation [13] for example), and such a prediction would not always give the same results, because of the probabilities in the model.

As the classic prediction link models work with measures, I use the model to compute a score for each potential link. I defined an observation period, and save the sequences during this period. For each potential link, I initialized the score to zero. Then, for each couple of nodes within a sequence and not linked at the end of the observation period, I increased the score of the associated potential link by 1. Thus, each potential link in the network has a score, which is a positive integer.

I did it on the dataset network with an observation period of one month, and it gives around 200 sequences.

4.2 Comparison with other link prediction methods

I compare the results of the prediction using the sequence model with some of the classic link prediction methods that I presented in the introduction. For each method, using a threshold, we can select the potential links with the highest scores. But the number of predicted links relies on the threshold. Thus, to compare the results of the different methods, I display the percentage of good prediction against the total number of predicted links.

Results are presented in the Figure 10. We can see that the sequence method is better than others for small numbers of predicted links (less than 2,000). For bigger numbers, all the methods have similar results, except the preferential attachment one which is clearly the worst.

As I use the dynamic of the network, I also tried the classic methods on the graph where there are only the links that appear during the observation month

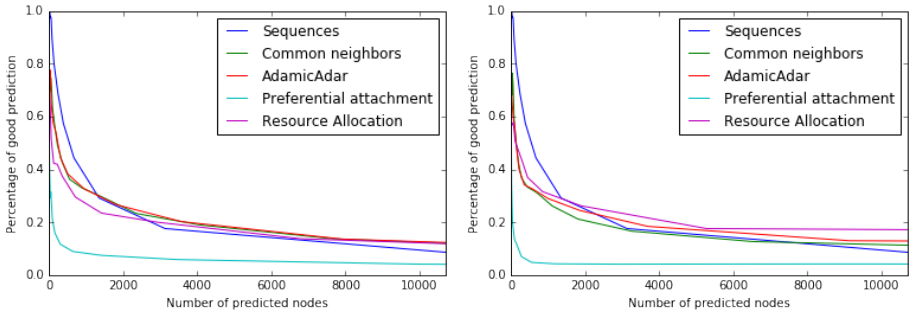


Fig. 10: Results of link prediction methods. On the left, classic methods use the snapshot of the network at the end of the observation month. On the right, they use only the graph with the links that appear during the observation month.

(again, it doesn't follow the definition given previously), because it seems that the recent activity of the network impacts on its future (it is what we notice with the sequences). Indeed, we can see on Figure 10 the results of those predictions, and the Resource allocation method is better than before. The sequence method is again the best for small numbers of predicted links. Notice that here, the Common Neighbors method is closed to the sequence one: when two nodes are in a same sequence, they have a common neighbor which is the initiator of the sequence. But they are not exactly the same, because you can have a same initiator with several sequences, and in this case, all the nodes in the sequences have a common neighbor (the initator), but they are not in the same sequence. This is a little difference but we can see that results of the sequence model is here better than common neighbors model results (except for more than 8,000 predicted nodes).

In the real network, there are around 3,000 new edges that appear after the observation month. For this value of predicted links, the sequence model is not the best. However, most of the links predicted by the sequence method appear shortly after the observation month, so the sequence method seems better for short time prediction (indeed, in this case there is less links to predict, and the sequence model is the best).

4.3 Hybrid method

An interesting property of the sequence method is that it gives different predicted links than classic methods, while all classic methods give similar sets of predicted links. Indeed, classic methods give sets of predicted links with 90% of the links in common, while only 20% of them are in common with the sequence method prediction set .Thus, we can use it to make a hybrid prediction method, combining a classic method and the sequence one.

I used the Common Neighbors method to make it, because its measure has the same scale that the sequence method one, and their meanings are closed. Thus,

I simply defined the score of the hybrid method as the sum of the score of the sequence method (used on the observation month), and the score of the Common Neighbors method used on the snapshot of the network before the observation month, because the sequence model can't predict the links that don't appear because of the sequences.

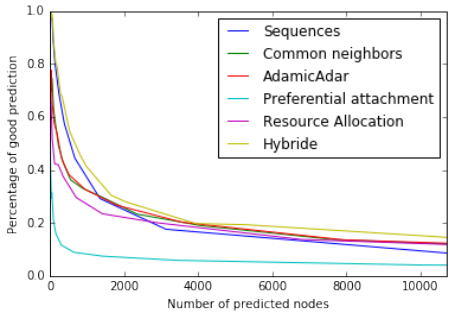


Fig. 11: Results of link prediction methods, in comparison with the hybrid method. Classic methods used the snapshot of the network at the end of the observation month

Figure 11 shows the results of the hybrid method, in comparison with other methods, and we can see that the hybrid method is the best one. Nevertheless, this method needs more information than the others, because it needs a snapshot of the network and an observation period.

This method is efficient, but has some drawbacks: she needs a lot of data on the network, and only works on online social networks where we can observe sequences, like Facebook networks. On such networks, we could combine this method with the classic ones in other ways. For example, it is possible to use several methods and machine learning to improve the performance of the link prediction, as the *RankMerging* do [14].

5 Conclusion

During my internship, I studied a Facebook network and its evolution during one year. Based on observations of the arrival of the new links, I proposed a new graph model: the sequence model. Then I adapted this model for link prediction, and I compared it with the classic link prediction methods. Finally, I made a hybrid model, using the sequence and the Common Neighbors models, which outperforms the classic link prediction models.

The sequence model is an interesting one because it has most of the properties of the real graphs: a high clustering coefficient, a low diameter, and a high modularity. The observed mechanism of sequences seems to be an explanation

of these properties. However, the model is too complicated to have interesting analytical results and is expensive.

On online social networks like Facebook, the link prediction could be used for the friend recommendations, combined with information people shares, like their work or their location. The use of this model for the link prediction problem is possible: not only it has good results, particularly for short time prediction, but also it gives different links that classic methods, so it permits to make a hybrid method with the classic ones (here the Common Neighbors method), that improves the efficiency of the prediction. However, predictions with the sequence model only work on online social networks like the studied one, and seems efficiency only at the beginning of the setting up of the network, because it is at this time there are the biggest sequences.

References

[1] Albert-László Barabási and Réka Albert. "Emergence of scaling in random networks". In: *science* 286.5439 (1999), pp. 509–512.

[2] Ginestra Bianconi et al. "Triadic closure as a basic generating mechanism of communities in complex networks". In: *Physical Review E* 90.4 (2014), p. 042806.

[3] Vincent D Blondel et al. "Fast unfolding of communities in large networks". In: *Journal of statistical mechanics: theory and experiment* 2008.10 (2008), P10008.

[4] Petter Holme and Jari Saramäki. "Temporal networks". In: *Physics reports* 519.3 (2012), pp. 97–125.

[5] Jussi M Kumpula et al. "Emergence of communities in weighted networks". In: *Physical review letters* 99.22 (2007), p. 228701.

[6] Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. "Graphs over time: densification laws, shrinking diameters and possible explanations". In: *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*. ACM. 2005, pp. 177–187.

[7] David Liben-Nowell and Jon Kleinberg. "The link-prediction problem for social networks". In: *journal of the Association for Information Science and Technology* 58.7 (2007), pp. 1019–1031.

[8] Naoki Masuda and Renaud Lambiotte. *A guide to temporal networks*. Vol. 4. World Scientific, 2016.

[9] Mark Newman. *Networks: an introduction*. Oxford university press, 2010.

[10] Derek J De Solla Price. "Networks of scientific papers". In: *Science* (1965), pp. 510–515.

[11] Vedran Sekara, Arkadiusz Stopczynski, and Sune Lehmann. "Fundamental structures of dynamic social networks". In: *Proceedings of the national academy of sciences* 113.36 (2016), pp. 9977–9982.

[12] Michael PH Stumpf and Mason A Porter. "Critical truths about power laws". In: *Science* 335.6069 (2012), pp. 665–666.

[13] Mikael Sunnåker et al. “Approximate bayesian computation”. In: *PLoS computational biology* 9.1 (2013), e1002803.

[14] Lionel Tabourier, Anne-Sophie Libert, and Renaud Lambiotte. “RankMerging: Learning to rank in large-scale social networks”. In: *DyNakII, 2nd International Workshop on Dynamic Networks and Knowledge Discovery (PKDD 2014 workshop)*. 2014.

[15] Tao Zhou, Linyuan Lü, and Yi-Cheng Zhang. “Predicting missing links via local information”. In: *The European Physical Journal B-Condensed Matter and Complex Systems* 71.4 (2009), pp. 623–630.

Appendix

Here is the detailled algorithm to build graph with the sequence model. The function *random*() return a random floating number between 0 and 1, and the function *choose_node*() return a node of the graph, chosen at random against the activation rate of the nodes.

Algorithm 1 Sequence model

$G = Graph()$
for $node = 1$ *to* N **do**
  $G.add\_node(node)$
  $node.potential\_sequence = \{\}$
  $node.activation = a$
end for
for i *in* I **do**
  for $node$ *in* $S[i]$ **do**
    $G.add\_edge(i, node)$
    $node.potential\_sequence = S[i] \setminus \{\{node\} \cup G.neighbors(n)\}$
    $node.activation = max(a_{seq}, m \times node.activation)$
  end for
end for
for $k = 1$ *to* k **do**
  $node = choose\_node()$
  $new\_sequence = [\,]$
  for n *in* $node.potential\_sequence$ **do**
    if $random() < p$ **then**
      $new\_sequence.append(n)$
    end if
  end for
  for n *in* $G.nodes()$ **do**
    if n *not in* $node.potential\_sequence$ AND $random() < \epsilon$ **then**
      $new\_sequence.append(n)$
    end if
  end for
  for n *in* $new\_sequence$ **do**
    $G.add\_edge(node, n)$
    $n.potential\_sequence = new\_sequence \setminus \{\{node\} \cup G.neighbors(n)\}$
    $n.activation = max(a_{seq}, m \times node.activation)$
  end for
end for
return G