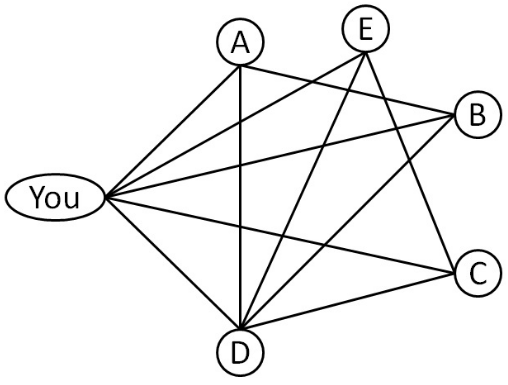
Link Prediction in Social Networks

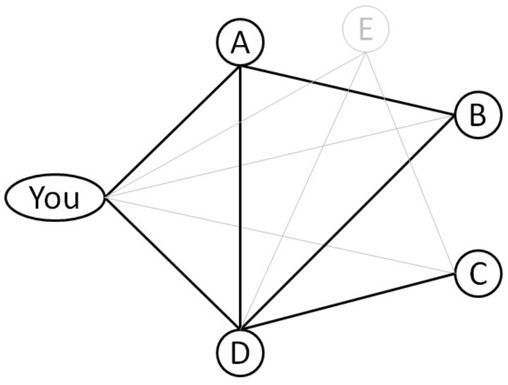
1 Introduction

With the advent of social media, everyone should be familiar with the concept of a *social network*. A *social network* is a graphical representation of your relationships to other people. A toy example is shown in Figure 1a. In this social network, each node represents a person, say yourself and several friends named A, B, C, D, and E. Each edge signifies a relationship between two people. For example, an edge could signify a friend, a family member, an acquaintance, or a colleague.

In general, the nodes could store information such as age, location, and interests. Edges could be encoded with the time that they were created, the type of relationship, or the number of times an interaction has happened.



(a) *The whole network, $G(V, E)$.*



(b) *The observed network, $G'(V', E')$.*

Figure 1: *An example of an entire social network, and how it might look to an observer.*

Social media sites like Facebook, however, only see a portion of the social network. Not everyone will join their site, and those that do join may not take the time to upload all of their data. The portion of the network that they do know about is the *observed network*. This concept is illustrated in Figure 1b.

It is in the interest of these social media sites to find efficient ways to predict the links that they cannot or have not seen. The ability to predict these interactions provides a more meaningful experience for their users and will keep them coming back to the site.

Social networks are not the only networks that can benefit from link prediction methods. Table 1 shows a few examples of different networks and the types of interactions that could be inferred from link prediction methods.

The general problem in all of these examples is to find the meaningful interactions that exist between two nodes based on only the observed network.

2 Problem definition

Let $G'(V', E')$ be an undirected, possibly weighted graph that represents the observed network, and let the whole network be denoted $G(V, E)$. Then G' is a subgraph of G and the set of

Type of network	**Type of interaction predicted**
Social	Friendships
	Collaborations
	Collusion
Biological	Protein-protein interactions in biological processes
	Food webs - how different organisms interact with each other and their environment
Information Systems	User-item interactions - recommender systems

Table 1: *Link prediction applied to different types of networks.*

missing edges, $E - E'$, will be denoted E^*. For an edge $e \notin E'$, it is our job to estimate the likelihood that $e \in E^*$.

The estimation that $e \in E^*$ will be called the *score* of that edge and will be denoted $S_{x,y}$. A score can be computed for all of the $V' \times V' - E'$ edges that are missing in the observed graph. Note that the scoring function is particular to each method that will be discussed, and that scores computed with different methods cannot be compared against each other.

Notice that information encoded in the nodes and edges has been left out of our problem definition. For this lecture we will investigate methods that only look at the topology of the graph. Higher level methods that combine the basic techniques described here with more sophisticated approaches to include this encoded information may be found in the references.

This lecture will also be slightly different than previous lectures. The methods that we will describe today are all heuristics. We will not be able to offer any proofs of correctness. We will, however, discuss the running times and basic implementations of each method.

3 Why is this a difficult problem?

For our social network $G'(V', E')$, there are $V' \times V' - E'$ possible edges to choose from, if we were picking a random edge to predict for our existing social network.

If G' is dense, then $E' \approx {V'}^2 - b$ where b is some constant between 1 and V'. Thus, we have a constant number of edges to choose from, and $O(\frac{1}{c})$ probability of choosing correctly at random.

If G' is sparse, then $E' \approx V'$. Thus, we have a ${V'}^2$ edges to choose from, and $O(\frac{1}{V'^2})$ probability of choosing correctly at random. This is shown pictorially in Figure 2 and in Table 2.

Fortunately for us as humans, but unfortunately for us as computer scientists, social networks are **sparse**, so picking at random is a terrible idea [2]. For example, Facebook has 1.06 billion members as of April 2013 [4]. The average person has on the order of 100 friends. So, Facebook is extremely sparse, and if we picked a new edge at random to suggest that two people become friends, we would have a $O(\frac{1}{10^{18}})$ shot at getting the right answer.

So, we want to find a way to narrow this down and make it a more feasible problem. The goal is to take advantage of the fact that social networks exhibit topological features such as grouping and clustering and friend neighborhoods, and use these artifacts to narrow down our prediction. If the social network were just as random as an Erdős-Renyi graph, we would

	E'	Possible Edges to Choose From	Pr Choosing at Random
Dense	$E' \approx V'^2 - b$	$V'^2 - (V'^2 - b) = b$	$O(\frac{1}{c})$
Sparse	$E' \approx V'$	$V'^2 - V' = O(V'^2)$	$O(\frac{1}{V'^2})$

Table 2: *Comparison of dense and sparse graphs and the probability of guessing a new edge correctly at random*

be out of luck for the heuristics described in this lecture.

3.1 Why this problem is special

Since there is no one correct solution to this problem, there is also no way to computationally brute force a solution. Even though we can compute all the possible missing edges on Facebook, the only way to ever know if an edge is correct is to ask the two users who are the endpoints of that edge. And sometimes, even if they do know each other in real life, perhaps they do not want to be Facebook friends. Maybe they are coworkers wanting to keep personal and professional life separate, or perhaps one person is angry at the other for stealing her secret link prediction heuristic method. The human factor involved in a social network means that these methods cannot be precise.

But, Facebook has a monetary interest in getting people to become friends. It is better for their ad system if they have more data about friendships. And more data about friendships will help with link prediction. It would be very expensive in terms of user patience to ask a user, "Do you know any of the following billion people?" So, it is in Facebook's best interest to shrink the pool.

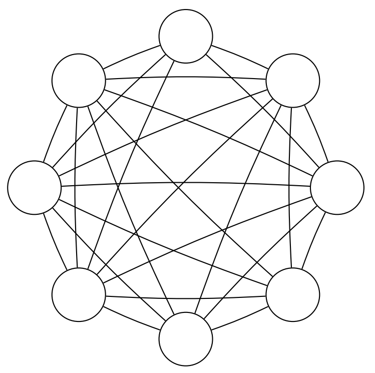
For other link prediction applications, it would also be expensive to test all V'^2 links. If a lab is studying protein-protein interaction, it would be financially prudent to save time and testing materials by calculating a set of interactions that may be more likely than the majority of the others, and working on the likely set before trying the rest.

4 Generic Algorithm

We want our link prediction heuristic to give us a list of possible edges. Since most prediction heuristics will rank a large number of the possible vertex pairs, we will choose the highest scoring pairs as our predicted edges. We can choose to take k of the top rated edges.

Since we have no way of determining if our link-prediction heuristic is correct, aside from asking all of our users or running some lab experiments, it would be prudent to test the heuristics on a known network before trying its recommendations on our observed network $G'(V', E')$.

We will therefore remove some of the E' edges of G' at random. Perhaps we could use a reverse Erős -Renyi generation method to do this. This will give us a graph $G''(V', E'')$,


(a) A dense graph with 2 edges to predict from

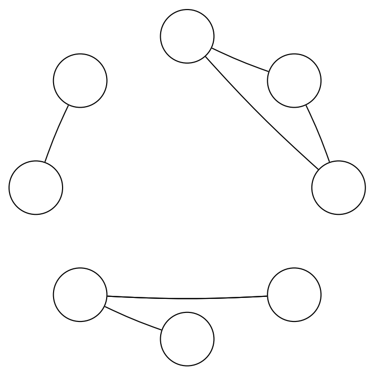

(b) A sparse graph with 58 edges to predict from

Figure 2: *The difference between predicting the correct edge in a sparse graph versus a dense graph. The dense graph has a 50% chance of picking the correct link, while the sparse graph has a 1.7% chance* [3]

which has all the same vertices we observed in G', but has fewer edges. Our goal is to try many different heuristics and find the one that best predicts which links we removed.

Algorithm 1 *Basic Experiment for Testing Heuristics*

```
Input: Observed network $G(V', E')$
1: $G''(V', E'') = G(V', E')$ - random edges
2: Score some or all of ${V'}^2 - E''$ edges using a heuristic method
3: $E_{new}$ = pick $k$ top ranked edges
4: Evaluate prediction method: effectiveness= $|E_{new} \cap (E' - E'')|$
Output: Effectiveness of the heuristic used
```

The graphic in Figure 3 visualizes what we are doing in this experiment.

5 Link Prediction Methods

The heuristics described in this section are described in a local manner. Each method described looks to make predictions about some link from source node x to some other node y. Therefore, as they are described here, the methods could be called for each vertex $v \in V'$. The running time of each method will be discussed under the assumption that the graph G' is sparse and that the average number of connections each node has is n. l_{max} will be defined as the maximum path length between the two nodes can influence the likelihood of an edge's existence. Paths longer than this will be assumed to have zero influence on the likelihood that the edge exists.

Table 3 summarizes the methods that we will cover, their scoring function, the basic implementation of each method and their running times on the entire graph G'. The following subsections describe each method in detail.

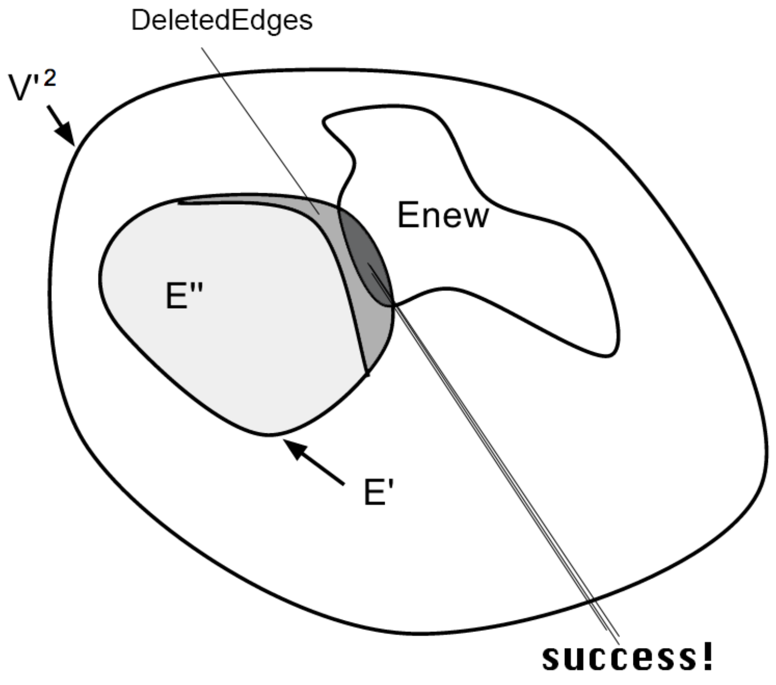


Figure 3: *The V'^2 possible edges in G''. E' contains both E'' and the edges we deleted. E_{new} represents the k top scoring edges resulting from running a link prediction heuristic. The intersection of $E' - E''$ and E_{new} represents the set of successful edge predictions. We wish to maximize the size of this set. The upper size limit for this success area is $|E' - E''|$*

5.1 Shortest Paths

Recall the *small worlds* experiment that we discussed in the Network Flow lectures. In this experiment Milgram selected some random people in Nebraska and Kansas and sent them a letter. In the letter he asked them to participate in his experiment. They were asked to forward a letter to a specific person in Boston, using only people they knew on a first name basis. When Milgram looked at his results, he found that it took about six contacts to reach the target and concluded that, on average, people in the US are separated by about six people.

So, in the shortest paths method, we simply look at the distance between two nodes and estimate the likelihood that a link exists as the negative of the shortest path length between the two nodes.

$$S_{x,y}^{SP} = -d_{x,y} \tag{1}$$

A basic implementation of this strategy could be breadth first search, and under the assumption that every node has on average n neighbors, this would take $O(V' \cdot n^l)$ time estimate predictions over the entire graph G'.

In the example graph G' shown in Figure 1b, both B and C are located at a distance of 2 from You, so each would be predicted with equal likelihood.

Heuristic	Scoring Function	Basic Implementation	Running Time
Shortest Paths	$-d_{x,y}$	BFS	$O(V' \cdot n^l)$
Common Neighbors	$\|\Gamma(x) \cap \Gamma(y)\|$	List comparison	$O(V'^2 \cdot n \lg n)$
Katz	$\sum_{l=1}^{\infty} \beta^l \cdot \|paths_{x,y}^{\langle l \rangle}\|$	DFS	$O(V' \cdot n^l)$
SimRank	see Equation 6	Fixed-Point Iteration	$O(KV'^2 n^2)$

Table 3: *A comparison of the basic link prediction methods.*

5.2 Common Neighbors

Extending the "small worlds" concept leads us to the conclusion that a node's immediate neighbors hold some valuable information about the likely new edges for node x. So let $\Gamma(x)$ denote the set of neighbors of node x.

Recall the lecture on Graph Clustering where we discussed the concept of *homophily.* This was the concept that the interests of two people, x and y, are similar if a friendship exists between them. Looking at this from the link prediction perspective, if two people, x and y, have many common friends, then it is very likely that x and y are also friends. This concept is the basis for the *common neighbors* method of link prediction.

Examining the degree of overlap between two nodes neighbors, we can score the likelihood that a link exists between two nodes as:

$$S_{x,y}^{CN} = |\Gamma(x) \cap \Gamma(y)| \tag{2}$$

If the graph G' is stored as an adjacency list, this is simply a list comparison operation between each nodes list and can be done in $O(n \lg n)$ time if the two lists are sorted, or $O(n)$ time if the lists are hashable.

Again, by examining the example graph in Figure 1b we can see that You has more neighbors in common with node B than with node C, so the link between You and B will be predicted to exist with a higher likelihood than the link between You and C.

5.3 Katz

Again extending the *small worlds* idea and also building on the common neighbors approach, if one short path between two nodes indicates that a link might exist, then many short paths between two nodes should indicate a stronger likelihood that a link exists between these two nodes.

An examination of the graph G' in Figure 1b shows that there are a number of paths that exist between You and nodes B and C. These observations are detailed in Table 4 for the possible different path lengths and arbitrary damping values that put less emphasis on paths of longer length.

Path Length	$\|paths_{You,B}\|$	$\|paths_{You,C}\|$	Damping (β^l)
2	2	1	$\frac{1}{2}$
3	2	1	$\frac{1}{4}$
4	0	1	$\frac{1}{8}$

Table 4: *Calculating a score based on all of the paths that exist between You and nodes B and C in G'.*

Using this table we could compute a score for each possible edge as shown in Equations 3 and 4.

$$\langle You, B\rangle = \frac{1}{2}\cdot 2 + \frac{1}{4}\cdot 2 + \frac{1}{8}\cdot 0 = 1.500 \tag{3}$$

$$\langle You, B\rangle = \frac{1}{2}\cdot 1 + \frac{1}{4}\cdot 1 + \frac{1}{8}\cdot 1 = 0.875 \tag{4}$$

More formally, we can describe this scoring function as shown in Equation 5.

$$\begin{aligned} S^K_{x,y} &= \sum_{l=1}^{\infty} \beta^l \cdot |paths^{\langle l\rangle}_{x,y}| \\ &= \beta \cdot |paths^{\langle 1\rangle}_{x,y}| + \beta^2 \cdot |paths^{\langle 2\rangle}_{x,y}| \cdots \end{aligned} \tag{5}$$

Where β is weighting factor and is typically chosen so that longer paths do not influence the likelihood of a link existing between two nodes, and $|paths^{\langle 2\rangle}_{x,y}|$ means the weight of all paths from x to y of length l.

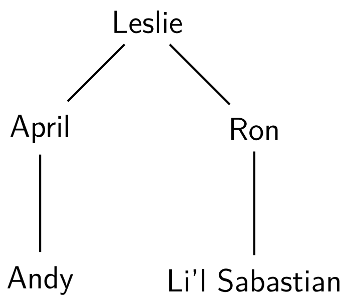
This method could be implemented using a form of Depth First Search. As each edge is identified as a Tree, Back, Forward, or Cross edge, you could keep track of how many paths to each node there are and what their lengths are. At a minimum, this method would take $O(V' \cdot n^l)$ time for the entire graph G'.

5.4 SimRank

SimRank is a general technique for ranking the degree of similarity between objects. It is based on the idea that "two objects are similar if they are related to similar objects." [1] We can tailor this for link prediction by saying that two nodes are similar if they have similar neighbors. So, a and b are similar if they are connected to c and d, and c and d are themselves similar [1]. The base case is that an object is similar to itself, ie. a is completely similar to a.

We can measure similarity on a scale of $[0, 1]$, where 1 is maximally similar, and 0 is completely dissimilar. A node that was not connected to any nodes at all would have a similarity of 0 with all the other nodes in the network.

As an example of how similarities may propagate from a base case is seen in Figure 4. There is a network of friends with some edges that that we know about. Leslie is maximally similar to herself, and so we can infer that April and Ron are perhaps similar to each other, but to a lesser degree. Similarly, since April and Ron are similar, then we can infer that their respective friends, Andy and Li'l Sebastian, will also be similar, but to a lesser degree than April and Ron. Of course, this series of similarities would extend to include all n^2 pairs of people in the network, and other pair's similarities will affect the once that propagated outward from Leslie.



(a) A network of happy-go-lucky friends

$$\begin{aligned} s(\mathit{Leslie}, \mathit{Leslie}) &= 1 \\ &\downarrow \\ s(\mathit{April}, \mathit{Ron}) &= 0.4 \\ &\downarrow \\ s(\mathit{Andy}, \mathit{Li'lSebastian}) &= 0.33 \end{aligned}$$

(b) Leslie's is maximally similar to herself. Since she is friends with April and Ron, they may be similar to each other, but less similar than Leslie is to herself. Similarly, since April and Ron are similar, their friends Andy and Li'l Sebastian may also be similar, but less similar than April and Ron.

Figure 4: *An example of how Leslie being maximally similarity to herself, $s(Lesley, Lesley) = 1$, propagates to affect the similarities of her friends and friends of friends.*

This idea of a base case similarity and the notion that similarities depend on neighboring similarities, which depend on neighboring similarities lends itself to a recursive definition. For for a measure of similarity between two nodes, denoted as $s(a, b)$:

$$s(a,b) = \begin{cases} 1 & \text{if } a = b, \\ \frac{C}{|\Gamma(a)|\cdot|\Gamma(b)|} \sum_{z\in\Gamma(a)} \sum_{z'\in\Gamma(b)} s(z,z') & \text{if } a \neq b. \end{cases} \tag{6}$$

Further, we will define the score between two nodes for the purpose of our heuristic ranking to be equal to the similarity between the two nodes:

$$S_{a,b} = s(a,b)$$

In equation (6), the base case describes a node's similarity to itself. The recursive case iterates over all neighbor pairs (z, z') of (a, b), and sums up the similarity $s(z, z')$ of each of these pairs. Then we divide by the total number of neighbor pairs, $|\Gamma(a)||\Gamma(b)|$, to normalize the sum. So, this is basically averaging the similarities of the neighbors of a and the neighbors of b. Note that this is symmetrical, so $s(a, b) = s(b, a)$.

The constant C can be viewed as a damping factor or confidence factor. For example, in the previous example from Figure 4, when we note that April and Ron are similar due to the fact that Leslie is friends with them, we could say that $s(\mathit{April}, \mathit{Ron}) = s(\mathit{Leslie}, \mathit{Leslie}) = 1$, but that would probably be too confident of a statement in terms of the degree of similarity between April and Ron. So, it would be better to temper this with an constant so that we have $s(April, Ron) = C \cdot s(Leslie, Leslie)$ for some $C = [0, 1]$. In practice, people often use $C = 0.8$.

5.4.1 Random Surfer-Pairs:

Jeh and Widom, authors of the 2002 SimRank paper [1], describe a model called Random Surfer-Pairs which provides a deeper intuition about what the SimRank score is calculating. The surfer terminology relates to the idea of using SimRank to relate two similar webpages using the hyperlinks on their pages. So, a pair of surfers would be randomly surfing over hyperlinks in that scenario.

Jeh and Widom show that the SimRank score $s(a, b)$ measures how soon two random surfers are expected to meet at the same node if they started at nodes a and b and randomly walked the graph in lockstep. They provide a full proof that the Random Surfer-Pairs reduces exactly to the SimRank score that we defined above.

5.4.2 Iterative Solution

An approach to finding the similarities of all vertex pairs is to solve by iteration to a fixed-point, where we improve the quality of our approximation of $s(a, b)$ for each successive iteration. Let V' be the number of nodes in G''. We will create an array with ${V'}^2$ entries $R_k(*, *)$, with $R_k(a, b)$ corresponding to the similarity between a and b on iteration k.

The first iteration calculates $R_0(*, *)$, which corresponds to the similarities at the base case:

$$R_0(a, b) = \begin{cases} 1 & \text{if } a = b, \\ 0 & \text{if } a \neq b. \end{cases}$$

If other similarities are known before running the algorithm, they could be specified in the base case as well (this would be an example of a way to combine this method with node information or other information external to the network structure). Then we will compute $R_{k+1}(a, b)$ for all successive iterations using equation (6):

$$R_{k+1}(a, b) = \begin{cases} 1 & \text{if } a = b, \\ \frac{C}{|\Gamma(a)| \cdot |\Gamma(b)|} \sum_{z \in \Gamma(a)} \sum_{z' \in \Gamma(b)} R_k(z, z') & \text{if } a \neq b. \end{cases}$$

Jeh and Widom prove that for all $a, b \in V'$, $lim_{k \to \infty} R_k(a, b) = s(a, b)$. So, we can assume there is convergence. In addition, they found experimentally that $K \approx 5$ iterations usually achieves fairly stable similarities. This would need to be verified for any given social network.

This method will take $O({V'}^2)$ space for our input graph to store the vertex pairs. To calculate the running time, recall from above that vertices have an average of n neighbors. Therefore, we can approximate the average number of calculations required to compute a single score $R_k(a, b)$ to be $|\Gamma(a)||\Gamma(b)| \approx n^2$. There are K iterations which each calculate ${V'}^2$ scores. The running time will be $O(K{V'}^2 n^2)$. For most networks, n^2 will be a constant relative to V', since the network will be sparse, and the average number of neighbors $\ll V'$.

6 Comparison of Approaches

[2] compares the performances of these simple prediction methods when applied to some real social networks. The study examines how well these methods predict co-authorship in some active subfields of the physics community, namely Condensed Matter and Astrophysics.

Table 5 shows an excerpt of these results. The results that we have chosen to display show the performance of each prediction method when compared to a random prediction of edges. The authors note that while these raw results are pretty poor, a factor of improvement greater than 5 is not possible by chance and chance alone. This implies that the topology alone does give us some useful information to use when determining missing edges in a network.

The results also show that the performance of a given prediction method highly depends on the type of network being analyzed. [2] identifies some specific peculiarities inherent to each physics community that influences the results and why some methods work better than others in different subfields, but the conclusion remains: you must test many prediction methods to determine the right heuristic for your particular network of interest.

Prediction Method	Physics Community	
	Condensed Matter	Astrophysics
Number of Authors	1253	1561
Random Prediction	0.147%	0.475%
Shortest Path	25.1	9.4
Common Neighbors	40.8	18.0
Katz	41.4	10.9
SimRank	39.0	14.5

Table 5: *An excerpt from [2] showing the performance of the link prediction methods covered in this lecture when applied to real social network in the physics community.*

References

[1] Glen Jeh and Jennifer Widom. Simrank: a measure of structural-context similarity. pages 538–543, 2002.

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

[3] Jean-Noï£¡l Quintin. Example: A complete graph. http://www.texample.net/tikz/examples/complete-graph/, February 2012. accessed 2013-April-19, modified code.

[4] Craig Smith. (april 2013) how many people use the top social media, apps & services? http://expandedramblings.com/index.php/resource-how-many-people-use-the-top-social-media/, April 2013. accessed 2013-April-24.