CHAPTER

Entity Resolution and Link Prediction

9

Once a network is constructed, there is often missing and duplicated information. There may be multiple nodes representing the same person, or there may be missing edges. For example, consider Figure 9.1.

All pairs of nodes but one are connected in this network. While it is possible that nodes A and E do not know one another, it is extremely unlikely. *Link prediction* is a method of analysis that detects where missing links should be present in the network.

Similarly, consider Figure 9.2. In this network the nodes on the left and right—John Smith and J. Smith—have very similar names, share all the same connections, and have no connection between one another. It could be that John and J. Smith are actually the same person. *Entity resolution* is a technique for merging nodes that represent the same person, as we might do here.

Methods for doing link prediction and entity resolution can range from simple to very complex. This chapter will introduce the fundamentals of each technique and illustrate their application through several case studies.

Link prediction

Formally, the goal of link prediction is to analyze a network at a set time and predict the edges that may appear in the network in the future. However, it can be used in many applications. This can include "cleaning" a dataset. Data often has errors in it, including missing links, and link prediction could identify places where an analyst might want to check to confirm that there is no edge between a pair of nodes. It can also be used to identify people.

For example, consider Figure 9.3. This network shows how often Alice, Bob, Chuck, and Frank attend meetings together. The weight on the edges indicates the number of meetings each pair of users has attended together. If we know that Alice and Bob attended a meeting with a third person, but we do not know who the third person is, we can consult the graph to make a guess about what links are likely missing in the graph of the new meeting. While it is possible that the third person is Frank or someone not pictured, the graph in Figure 9.3 suggests that it is most likely Chuck.

Many of the network features discussed so far in this book come into play when considering link prediction. Figure 9.1 above illustrates a simple case where

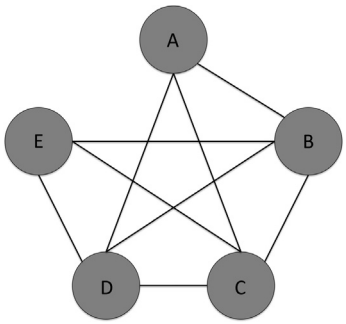


FIGURE 9.1

A network where all pairs of nodes but one are connected.

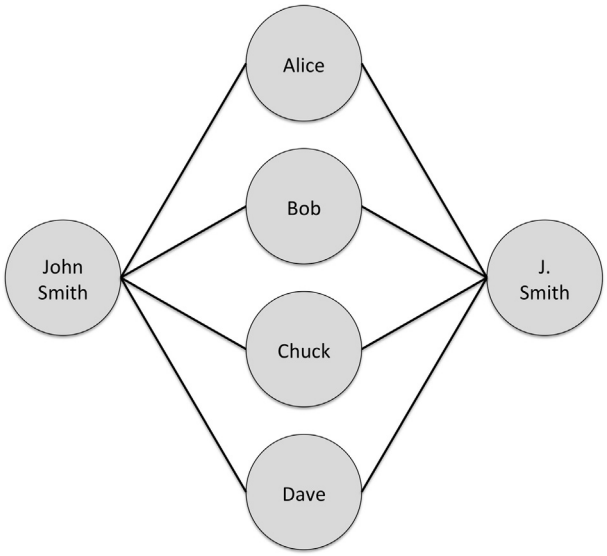


FIGURE 9.2

A network with two nodes, John Smith and J. Smith, who have similar names and acquaintances with no connection to one another. This could suggest that they are actually the same person.

we might conclude that a link should be present. If we consider tie strength, the case can be clearer. For example, in Figure 9.4, the thickness of the edges indicates the tie strength. In Figure 9.4(a), nodes A and E have strong ties with all the other graphs. This forms many forbidden triads, as discussed earlier in the book. It is very rare to have two nodes that share strong ties with another node but have no tie

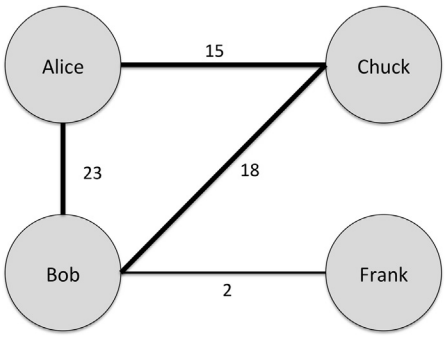


FIGURE 9.3

A network showing the frequency with which Alice, Bob, Chuck, and Frank attend meetings together.

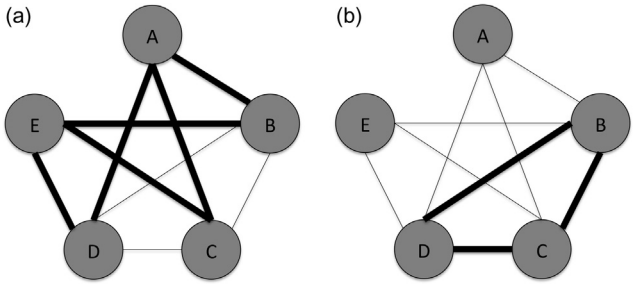


FIGURE 9.4

Two variations of the graph in Figure 9.1 where edge thickness indicates tie strength. In (a), nodes A and E have many shared strong ties, while in (b) they only share weak ties.

with one another. However, in Figure 9.4(b), there are only weak ties between A and E's common neighbors, so it is less likely that they share a tie when compared with graph (a).

These examples provide anecdotes that illustrate what link prediction can do. The next step is to create systematic methods for predicting links.

There are many ways to do link prediction, but all of the algorithms generate a score for each pair of nodes. If we have two nodes, A and B, then the score (A,B) indicates how closely connected A and B are in the graph. After computing the score for every pair of nodes, the algorithm returns a ranked list. The pairs with the highest score are predicted to be the most likely new edges.

As a running example to illustrate how each of the scoring methods works, we will use a simple undirected graph shown in Figure 9.5.

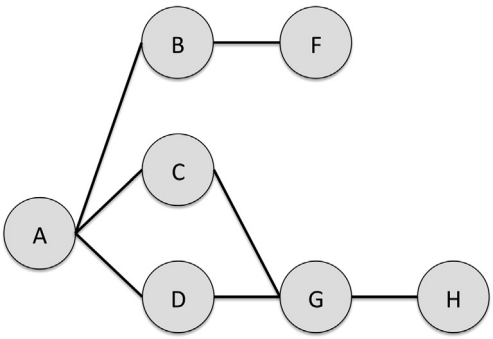


FIGURE 9.5

An example graph with eight nodes.

Mathematical notation

Before looking at the equations for computing scores, we will review some basic mathematical terminology and notation so that we can write the equations concisely. A *set* is a collection of items. In graphs, the neighbors of a node are a set. For example, the neighbors of node A in Figure 9.5 are {B, C, D}. This is a set. Let *Neighbors(A)* indicate the set of A's neighbors. If a set is written with vertical bars on either side, that refers to the size of the set. So |*Neighbors(A)*| means the size of the set of A's neighbors. Since A has three neighbors, |*Neighbors(A)*| = 3. Note that the size of the set of a node's neighbors is equivalent to its degree; that is, *Neighbors(A)*| = *degree(A)*.

Sets can overlap. For example *Neighbors(A)* = {B, C, D} and *Neighbors(G)* = {C, D}. The *intersection* of two sets is the items they have in common. In this example, the intersection of *Neighbors(A)* and *Neighbors(B)* is {C,D} since those nodes are in both sets. The intersection is indicated with the $\cap$ symbol. Thus, to get the neighbors that A and G share in common, we write $Neighbors(A) \cap Neighbors(G)$. The *number* of nodes they have in common is indicated with the vertical bars on either side $|Neighbors(A) \cap Neighbors(G)| = 2$.

The *union* of two sets is the set of all items. The union of *Neighbors(A)* and *Neighbors(G)* is {B,C,D,H}. Note that we do not duplicate nodes C and D.

The symbol Σ is used to indicate that we are taking the sum of values. When working with sets, we might want to take the sum of a value for each item in the set. For example, we may want to add up the degree of each node who is neighbors with node A. To do this, we need to specify that we want each element for a set. In this case *Neighbors(A)* is our set. Then we need to indicate that we want each element in that set. We do this by saying $x \in Neighbors(A)$. That means x represents each item from the set. To show that we are adding these values up,

we put this notation below the Σ. So, to sum the degree of each node who is neighbors with A, we would write:

$$\sum_{x \in Neighbors(A)} degree(x)$$

The important thing to remember with this notation is that by putting the $x \in Neighbors(A)$ underneath the Σ , it means to add up the value after the Σ for each of the items x represents. Although the notation may be a bit complex if you have not seen it before, breaking it down will make it easy to understand.

Computing score

One of the simplest ways to score the similarity or closeness of two nodes is to use the shortest path length between them. Nodes that are close to one another are more likely to create a relationship. This is especially true for nodes that have a mutual friend. However, as the average shortest path length increases, we want the score to decrease because nodes that are far apart (with a high average shortest path length) are less likely to be connected. Thus, we can use the negative value of the shortest path, so closer nodes have higher scores.

$$score(A, B) = -shortestPath(A, B)$$

So for Figure 9.5, the ranked list of scores is as follows:

Pair	Score: −Shortest Path Length
A,F	−2
A,G	−2
B,C	−2
B,D	−2
C,D	−2
C,H	−2
D,H	−2
A,H	−3
B,G	−3
C,F	−3
D,F	−3
B,H	−4
F,G	−4
F,H	−5

Note that since this is an undirected network, each pair appears only once in the list. For example, A,F appears and F,A is not listed since it would have the same value. If the network were directed, node pairs representing an edge in either direction would be listed.

In this first example with the shortest path length, many nodes are tied with a high score of -2. If a simple rule is used to predict that edges will occur between nodes with the highest scores, then all these pairs—(A,F), (A,G), (B,C), (B,D), (C,H), and (D,H)—would have predicted edges between them. Indeed, when using this method, any nodes that have at least one common neighbor will have a predicted edge added between them.

Another way of computing scores that uses more information from the network structure is to count the number of common neighbors between the two nodes in a pair. For the pair (A,B), we can represent this as the intersection of the set of nodes that are neighbors of A and the set of nodes that are neighbors of B.

$$score(A, B) = Neighbors(A) \cap Neighbors(B)$$

The result of this equation is the number of neighbors that the two nodes share. For the graph in Figure 9.5, the results are as follows:

Pair	Score: Common Neighbors
A,G	2
C,D	2
A,F	1
B,C	1
B,D	1
C,H	1
D,H	1
A,H	0
B,G	0
B,H	0
C,F	0
D,F	0
F,G	0
F,H	0

The result here is quite different. Two pairs, (A,G) and (C,D), have the high score. Thus, these would be the only edges predicted when we apply this algorithm.

The number of common neighbors makes social sense, too. The more friends two people have in common, the more likely they are to be introduced to one another.

However, number of common friends is not the whole story. Some people have an abnormally high number of connections in social networks, particularly in social media. For example, celebrities may have millions of friends on Facebook, but the fact that, for example, a popular singer and a politician have many friends in common may not mean much since they are connected to so

many people in the first place. A common statistic called the *Jaccard Index* can account for this.

The Jaccard Index counts the total number of friends in common and divides that by the total number of people who are friends of either node. So, in our simple graph in Figure 9.5, nodes A and G have two friends in common. The total number of nodes who are friends with either A or G is four: nodes B, C, D, and H. Note that we do simply add the number of nodes who are friends with A (3) to the number of nodes who are friends with G (3), because this would count their mutual friends twice (nodes C and D). Instead, we are taking the *union* of their friends.

Note that the size of the union is always the sum of the degrees of the two nodes minus the size of the intersection. For nodes A and G, the sum of their degrees is 6 (3 + 3) and the size of the intersection (number of common friends) is 2, so the size of the union is $6-2=4$, as we counted above. This will be useful later.

Thus, the formula for the Jaccard Index used to compute a score between nodes is:

$$score(A,B) = \frac{|Neighbors(A) \cap Neighbors(B)|}{|Neighbors(A) \cup Neighbors(B)|}$$

For the graph in Figure 9.5, the scores are as follows.

Pair	Score: Jaccard Index
C,D	1
C,H	0.50
D,H	0.50
A,G	0.50
A,F	0.33
B,C	0.33
B,D	0.33
A,H	0
B,G	0
B,H	0
C,F	0
D,F	0
F,G	0
F,H	0

To clarify further, here are the calculations for a few of these pairs. As mentioned earlier, nodes A and G have two common friends and four total nodes in the union of their neighbors. Thus, their score is $2/4 = 0.5$. Nodes C and H also have a score of 0.5, but they share only one friend in common. Since there are only two nodes in the union of their neighbors (nodes A and G), their score is

$1/2 = 0.5$. Node A and F have one friend in common also, but there are three nodes in their union (B, C, and D), so their score is $1/3 = 0.33$. Nodes C and D have two common neighbors (A and G). Since these are the only neighbors of C and D, their score is $2/2 = 1$.

Thus, in this network, we would predict that the next edge appears between nodes C and D.

The value of the Jaccard Index becomes clearer in a big network. Say we have four nodes: Alice, Bob, Chuck, and Dave. Let Alice and Bob be celebrities, each with 1 million friends. Chuck and Dave are average users with 100 friends each. Now say Alice and Bob have 2,000 friends in common while Chuck and Dave have only 20 friends in common.

Although Alice and Bob may seem to be more strongly connected than Chuck and Dave, since they have 100 times more common friends, the Jaccard Index indicates this is not the case. Remember: The size of the union is the sum of the degrees minus the size of the intersection. Thus, the Jaccard scores for these two pairs is as follows:

$$score(Alice, Bob) = \frac{2,000}{(1,000,000 + 1,000,000) - 2,000} = \frac{2,000}{1,998,000} = 0.001$$

$$score(Chuck, Dave) = \frac{20}{(100 + 100) - 20} = \frac{20}{180} = 0.11$$

So, although the number of friends in common is 100 times higher for Alice and Bob, the Jaccard Index is over 100 times higher for Chuck and Dave because they do not have as many total friends. Stepping back from the math, it makes sense that people who have 20 real friends in common are likely to be closer than celebrities who have lots of common "friends" but also far more friends that are not shared.

This example brings up another problem. What if the 20 people Chuck and Dave know in common are also celebrities? That is much less meaningful than if they are other people with a smaller number of friends. Adamic and Adar (2003) came up with a method for dealing with this issue. They look at common friends and assign a score that gives more weight to people who have a few friends.

The formula is as follows:

$$score(A, B) = \sum_{x \in Neighbors(A) \cap Neighbors(B)} 1/\log(|Neighbors(x)|)$$

The formula looks a bit complicated at first, but it is quite simple. For every node who is a neighbor of both A and B (call this node x), we add a value to the total. That value is 1 over the log of the number of neighbors x has. For a node with 100 neighbors, the value would be $1/\log(100) = 1/2 = 0.50$. For a node with

2,000 neighbors, the value would be $1/\log(2,000) = 1/3.3 = 0.30$. As the number of neighbors increases, the value decreases. A node with 1,000,000 neighbors would only have a value of 0.17.

The values for our sample network using the Adamic/Adar method is as follows:

Pair	Score: Adamic/Adar
A,G	6.64
C,D	4.19
A,F	3.32
B,C	2.10
B,D	2.10
C,H	2.10
D,H	2.10
A,H	0.00
B,G	0.00
B,H	0.00
C,F	0.00
D,F	0.00
F,G	0.00
F,H	0.00

The clear winner here is (A,G). This method predicts that the next link to be added is between these nodes. Note that in the Jaccard measure, the pair (C,D) came out ahead. They are lower on the list here because their neighbors, A and G, both have the highest degrees in the network and thus they do not count as strongly.

While it is different from the Jaccard Index rankings, the ranking here is the same as when we used the number of common neighbors. This is because, in our sample network in Figure 9.5, all the nodes have a small degree. Thus, the method here will not have a large impact on the scores. However, when there are large differences in the degrees of nodes, as is expected in most networks since the degree follows a power law distribution, there will be larger effects from using this approach.

One final technique for predicting links is to consider *preferential attachment*. This network principle states that nodes with a high degree are more likely to gain new links. Popular nodes are more likely to gain new friends than less popular nodes. When predicting edges, preferential attachments suggest that nodes with high degree are more likely to gain new edges. The formula for this scoring method is relatively simple:

$$score(A,B) = |Neighbors(A)|^*|Neighbors(B)| = degree(A)^*degree(B)$$

For the example in Figure 9.5, the scores are as follows:

Pair	Score: Preferential Attachment
A,G	9
B,G	6
B,C	4
B,D	4
C,D	4
A,F	3
A,H	3
F,G	3
B,H	2
C,F	2
C,H	2
D,F	2
D,H	2
F,H	1

With this measure, we would predict that the next edge to appear will be between nodes A and G.

Advanced link prediction techniques

The examples covered so far are relatively straightforward link prediction techniques, and there are many ways to make the approach more sophisticated. One could begin by combining the measures above. For example, we could take the average ranking of each node pair from each measure and rank by that value. The result would be a ranking that considers all the factors described above.

There are also probabilistic models for link prediction that are very successful. These often rely on a technique called Markov Networks . Some approaches consider nodes' attributes in addition to network structure. They can also work with weighted and directed graphs. Machine learning has also been effective when applied to this problem.

While these techniques are beyond the scope of this book, many references can be found online. Good overviews are also provided in Getoor and Diehl (2005) and Liben-Nowell and Kleinberg (2007).

Entity resolution

Entity resolution is a technique that tries to identify nodes that represent the same entity and then to merge them together. For example, in Figure 9.2, the two nodes

"John Smith" and "J. Smith" may represent the same person. How do we determine if they are the same or not?

Just as there are many techniques for doing link prediction, there are a number of methods for entity resolution. Most of them involve looking at the data about the nodes, including their attributes and relationships.

Table 9.1 contains sample data for four people. Before getting into network connections, we can look at this information alone.

The simplest approach to merge duplicated nodes is used when we have unique identifiers for each node. For example, each person in the United States has a unique Social Security Number (SSN). Each person has only one SSN, and each SSN is used for only one person at a time. In Table 9.1, nodes "J Smith" and "John Smith" have the same SSN, so we know they must represent the same person. If their SSNs were different, we would know they are definitely not the same person.

Other attributes can allow us to make similarly definitive conclusions, but not as often. For example, the birthday of a person should be consistent. In this case, "John Smith" and "JA Smith" have the same address, but their birthdays are different. Thus, we can conclude either that they are not the same person or that there is an error in the data. We will assume the data is correct in these examples to more easily illustrate our points.

Not all attributes need to match. For example, "J Smith" and "John Smith" have different addresses. People move or have work and home addresses, so the mismatch on that point does not indicate that the nodes are or are not the same person.

Similarly, first names do not need to match. People may use nicknames, they may go by their first and middle names in different contexts, and their initials may be used. Similarity in names can suggest that two people are the same, but that is not totally conclusive.

For example, in Table 9.2 we have "J Smith" and "Robert Smith." Their names are similar in that they have the same last name. The differences in the first name could be because the same person is using his first initial in some cases and his middle name in others. Or they simply may be different people. If we look at the other data, we see that they have the same address and birthday. Those shared attributes provide evidence that they may indeed be the same person.

Table 9.1 Sample Personal Data for Four People

First Name	Last Name	Address	Birthday	SSN
J	Smith	123 Main St	1/6/68	123-12-1234
John	Smith	54 Elm St	January 1968	123-12-1234
Robert	Smith	123 Main St	1/6/1968	
JA	Smith	54 Elm St	March 1968	

Table 9.2 Values for each Example Node Pair and the Associated Similarity Measures

	Node Pair		
	A,J	**B,D**	**E,I**
Common Neighbors	3	0	1
Jaccard Index	0.38	0	0.33
Adamic/Adar	9.97	0	1.43
Preferential Attachment	25	1	4

There is uncertainty at this point. How can we deal with that? We use a similar approach to that from link prediction: scoring. For each pair of nodes, we can compute a score that represents the likelihood that they are the same node. Then, we can set a threshold value. Any pair of nodes with a score above that threshold will be merged, and any nodes below the threshold will not be merged.

Scoring techniques

There are many approaches for creating scores, and these can become quite sophisticated, using machine learning algorithms and data mining. In this chapter, we will present one of the simpler approaches that you could apply on your own.

To create a score for a pair of nodes, we will consider similarity on a set of attributes. Those can include data such as that in Table 9.2 and similarity in the network structure. For each pair of nodes, we will know if their values match or not on a given attribute. For example, in Table 9.2, "J Smith" and "John Smith" have the same SSN. Thus, we can record a score of 1 for that attribute indicating a match. However, their first names do not match. In that case, we record a score of 0.

As discussed earlier, however, matching on some attributes is more important than on others. Two different people may have the same name, but a match on the SSN is much more definitive. Thus, we would want to give more weight to a SSN match than we do to a name match. Similarly, we want a weight for when nodes do not match. For example, if nodes do not match on their SSN, we want to subtract a lot of value from the score since that is a strong indicator that the nodes represent different people.

To create a score for a pair of nodes, we will determine that they match on a given attribute and we will have a weight for each attribute. Then, we add the positive weights for each attribute where the nodes match (i.e., receive a score of 1), and subtract the negative weights when they do not match (i.e., receive a score of 0).

Then, we are left to find a method for computing the weights for each attribute.

We want weights to be higher for attributes that are more definitive, like the SSN, and lower for attributes that are more commonly shared, like the month of birth. There is a common method for computing these weights for the entity resolution. This is done with values called *u* and *m* probabilities. The *u* probability is that two nodes will match on an attribute by chance. For example, the probability that two nodes have the same birth month is 1/12. Thus the *u* probability for birth month equals 1/12 or 0.083. The probability of two nodes having the same last name is more complex to compute because the probability varies based on the last name itself. For example, "Smith" is the most common last name in the United States, representing about 1% of all citizens' last names. Thus, the *u* probability for matching on the last name "Smith" is 0.01. However, for an uncommon last name in the United States, like "Himmelblau," which is used by only 0.00004% of the population, the *u* probability would be 0.0000004. When computing *u* probabilities, we can have a single value, like for birth month, or a set of values for each possible attribute value, like last name.

The *m* probability is the probability that two nodes that represent the same person will have the same value. Often we expect this value will be 1. For example, two nodes that are the same should have the same birthday, gender, SSN, and so on. However, the *m* value is not always 1. In some cases, like address or phone number, two nodes may indeed represent the same person but have different values. For example, one node could have personal/home information, and the other could have work information. Also, there may be missing attribute data. For example, in Table 9.2, several nodes are missing SSNs. Thus, they could represent the same person, but if one has an SSN and the other does not, the values will not match.

Setting the *m* probabilities will depend on the data you have. For our data, we could say the *m* probability for SSN is 0.95 (assuming there is more data than what is shown in Table 9.2 and we know how good it is), the *m* probability for address is 0.6, and the *m* probability for birth month is 0.98.

Once we have the *u* and *m* probabilities, we need to turn them into weights. There will actually be two weights for each attribute. The first is how much weight we add to the score if there is a match, and the second is how much weight we subtract from the score if there is no match. The common formulas are as follows.

For a match:

$$w = \ln(m/u)/\ln(2)$$

For a nonmatch:

$$w = \ln\left(\frac{1-m}{1-u}\right)/\ln(2)$$

Using the values we discovered above, the weight for a match on birth month would be:

$$\ln(0.98/0.083)/\ln(2) = 2.469/0.693 = 3.56$$

The weight for a no-match on birth month would be:

$$\ln\left(\frac{1-0.98}{1-0.083}\right)/\ln(2) = \ln\left(\frac{0.02}{0.917}\right)/\ln(2) = -3.825/0.693 = -5.520$$

We would perform this calculation for every attribute in the table. Then, we would check for matches and add the appropriate weights for a match or non-match to compute a final score.

Incorporating network data

The scoring above works with fixed attributes for a set of nodes. It does not look at the network structure at all, and that can be very important. For example, in Figure 9.2, John Smith and J Smith share many friends in common, but they are not connected to one another. If John and J are different people, we would probably expect them to know one another since they have so many common acquaintances.

Relational data is useful for enhancing the attribute-based similarity discussed above. Consider a more sophisticated graph than the one in Figure 9.2.

We could compute similarities between all pairs of nodes in the network. For simplicity, we will consider three pairs of nodes as examples: A and J, B and D, and E and I.

Before considering any formulas and just by observing the network, some features emerge. Nodes A and J both share several common neighbors, and they also have the highest degrees in the network. Nodes E and I have a common neighbor but have much lower degrees. Nodes B and D are far apart in the network. Without any mathematical work, we might consider that A and J seem most similar and B and D seem most distant.

To quantify how similar these nodes are to one another structurally, we want to examine their egocentric networks and compare them. Specifically, we want to compare the neighbors of one node to the neighbors of the other. This is exactly the same comparison we made when performing link prediction above. Thus, we can use many of the same scoring mechanisms from link prediction to quantify how similar a pair of nodes are to one another.

For entity resolution, the number of common neighbors, the Jaccard Index, the Adamic/Adar method, and preferential attachment all compared the neighbors of one node with those of another. We will use those same measures here. Table 9.2 gives the values for each measure to each pair of nodes in our example.

To review, the common neighbors simply counts how many nodes are neighbors of both nodes in the pair. Nodes A and J have three common

neighbors: nodes E, F, and G. Nodes B and D have no common neighbors. Nodes E and I have one node, J, as a common neighbor.

The Jaccard Index divides the size of the intersection by the size of the union. Nodes A and J have three nodes in their intersection (E, F, and G), and eight nodes in the union (B, C, E, F, G, H, I, and K). Thus, the Jaccard Index is $3/8 = 0.38$. Since nodes B and D have no neighbors in common, the Jaccard Index is 0. For nodes E and I, they have one node in common and three nodes in their union (A, J, and K). This yields a Jaccard Index of $1/3 = 0.33$.

The Adamic/Adar method sums up the inverse log of each neighbor's degree, giving more weight to nodes with fewer edges. For nodes A and J, this results in the following sum:

$$1/\log(degree(E)) + 1/\log(degree(F)) + 1/\log(degree(G)) =$$
$$1/\log(2) + 1/\log(2) + 1/\log(2) = 3/\log(2) = 3/0.301 = 9.97$$

For nodes E and I, they only have node J in common. That gives a simpler result:

$$1/\log(5) = 1.43$$

Finally, preferential attachment is the product of the degree of the two nodes being considered. For nodes A and J, that product is $5*5 = 25$. For nodes B and D, the product is $1*1 = 1$. This is the only measure that is nonzero for this pair. Nodes E and I have a product of $2*2 = 4$.

The results from these similarity measures can be used in addition to attribute data. For example, if two nodes are very similar in their attribute data but have very little similarity in the network, we can reduce the similarity score. A high similarity on the network may make up for lower similarity in attribute data as well. Network and attribute data can be considered as separate steps, or the network data score can receive its own weight for use in the sum above.

More sophisticated entity resolution

As with link prediction, there are many more sophisticated methods for doing entity resolution that are beyond the scope of this book. These use machine learning techniques, Bayesian networks, and statistical models. A good overview is available in Brizan and Tansel (2006).

However, there are some ways to iterate on even the relatively simple methods introduced here. One approach is to allow for partial matches. Returning to the "John Smith" and "J Smith" example, while their first names are not an exact match, they are close. Since "J" is the correct first initial for "John," we may label this a partial match. Then, instead of adding the weights for items that match, we can add part of the weight for a partial match. For example, if we say "J" is a 0.3 match for "John," then we could add 0.3 times the weight for a name match to the score. We would also have the option of subtracting 0.7 times the non-match score.

In this approach, the formula for an attribute that is a partial match with value p, we would add the following to the score:

$$p^{*}w_{match} + (1 - p)w_{non-match}$$

Say the weight for a matching first name is 5.5 and the weight for a non-matching first name is −3.2. If we did not give any credit for a partial match, then we would simply subtract 3.2 from the score. But if there is a 0.3 match on the first name, then the score becomes:

$$0.3^{*}5.5 + 0.7^{*} - 3.2 = 1.65 - 2.24 = -0.59$$

This partial match allows us to give much more credit to the pair, subtracting only 0.59 instead of 3.2.

A second, more sophisticated step is that we can do repeated iterations of entity resolution. For example, say we merge nodes A and J in the graph in Figure 9.6. After they are merged, the new graph that results is shown in Figure 9.7.

Now, node H has many similar neighbors with the merged node A/J. In fact, if we recompute the measures of similarity on the network, the scores for nodes H and A/J are as high or higher than they were for A and J in the previous round. This is shown in Table 9.3.

The network data suggests a lot of similarity between nodes H and A/J. If the attribute data supported a decision to merge node H with node A/J, we would produce a second new graph, shown in Figure 9.8.

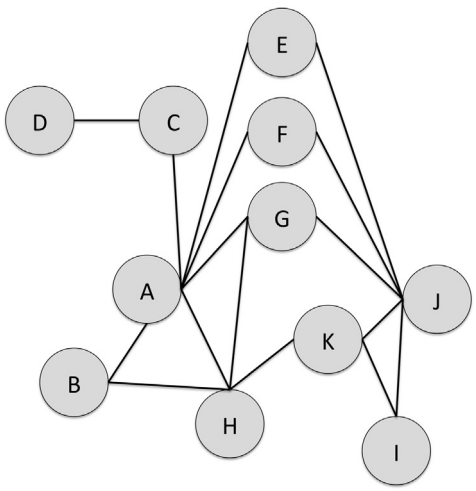


FIGURE 9.6

A graph in which we will consider whether or not to merge nodes. The examples will consider merging A and J, B and D, and E and I.

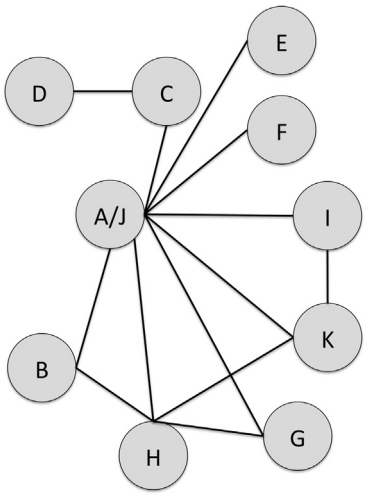


FIGURE 9.7

The network from Figure 9.6 after nodes A and J are merged.

Table 9.3 Measures of Network Similarity for Nodes on the Merged Network Shown in Figure 9.7

	Node Pair			E,I
	Previous A/J	**A/J, H**	**B,D**	
Common Neighbors	3	3	0	1
Jaccard Index	0.38	0.43	0.00	0.50
Adamic/Adar	9.97	9.97	0.00	1.11
Preferential Attachment	25	32	1	2

Note that the values for the pair E and I have also changed because of the merger in the network.

Attribute data should be considered in this iterative process as well, and similarities between other nodes in that respect may lead to further merges.

Link prediction: Case study—Friend recommendation

Link prediction is an important and widely studied problem, but what are the applications of a good link prediction algorithm? There are many, and one case particularly relevant to the topics in this book is for friend recommendation.

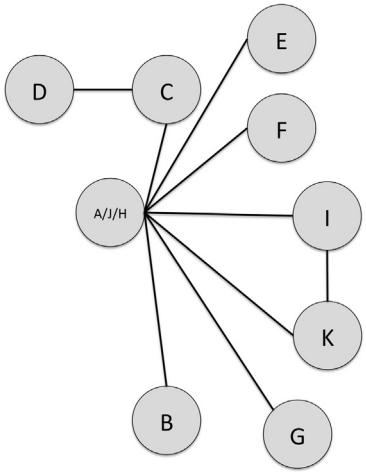


FIGURE 9.8

The network from Figures 9.6 and 9.7 with nodes A, J, and H all merged.

FIGURE 9.9

A suggestion about people to follow made by Twitter.

Many social networking and social media websites have a feature that recommends friends. For example, Figure 9.9 shows Twitter's "Who to follow" recommendation.

How does a system go about recommending people to friend or follow? There are many techniques, and link prediction is one way to do it.

Recall that link prediction, as described earlier, considers all unconnected pairs of nodes in the network and generates a score for each. Those scores can be

used to add the top-scoring link to the graph, or they can be considered a ranked list of potential edges to add. For friend recommendation, we do not necessarily want to consider all edges in the graph, but rather all possible edges for a specific user.

When that user logs in, the system can compute a score for each pair comprising the user and every other node in the network. Then, the pairs can be sorted from highest to lowest score, and the other node in the top-scoring edges becomes a recommended friend.

For large networks, however, computing scores for every pair of nodes can be computationally expensive and take a long time. For example, Facebook has over a billion users. Running 1 billion calculations takes a long time, especially if the system needs to get the friend list for every person. Fortunately, the process can be optimized. Recall that for most of the link prediction techniques, the nodes needed to share at least one common neighbor. If the system uses this as a limit, then the only nodes that need to be considered as candidate friends for the user are the users' friends' friends. That greatly cuts back on the number of possible pairs to score, making the computation much faster.

Note that link prediction results are not necessarily the only thing to consider when recommending friends. Looking at similarity of node attributes can add valuable information. While the interesting attributes will be different from those in entity resolution, the techniques for using them may be similar. For example, when recommending friends, we might look for people with matches on interests, educational background, favorite sports teams, and so on. We can create weights for matches on each of those attributes and use them in a score, just as we used weights on matching personal information to conduct entity resolution. Combining these attribute-based insights with the link prediction results will often lead to better friend suggestions.

Entity resolution: Case study—Finding duplicate accounts

When people sell things online on sites such as eBay, Etsy, or Amazon, the transaction requires that the buyers trust them. Even with insurance and seller protection systems in place, few buyers want to go through the hassle of receiving a bad item or no item and then filling out forms and dealing with a system's bureaucracy to receive a refund. Thus, the seller's reputation is extremely important for the transaction to go well.

When sellers develop a poor reputation, a common "solution" is to open another account. Having no reputation is often better than having a poor one. In more sinister cases, some sellers will develop good reputations in many accounts by selling small items, leveraging that reputation to sell a few big items at which point they defraud the buyers, absconding with the money and closing the now-worthless account.

To protect buyers, companies that host online sales want to ensure that people are not maintaining multiple accounts without an obvious link between them. Knowing which accounts belong to the same person allows the company to track the good and bad actions of each unique person and to have the power to suspend *all* the accounts if the seller does something very bad on one of them, or if the sum of bad behavior across the accounts crosses some threshold.

When sites have millions of sellers, as many do, how can a company track which accounts actually belong to the same person? Entity resolution works well for this task. User attributes, like financial information and addresses, are often very distinctive and can help identify the accounts' owner. Network information can also be included, especially when the accounts are linked to the same products or customers.

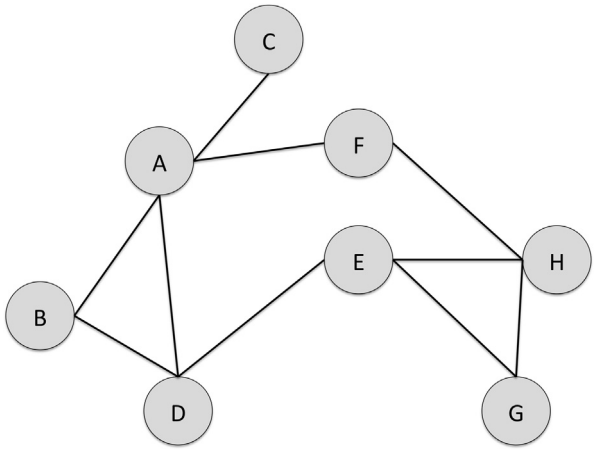
Conclusion

Link prediction and entity resolution are two ways to identify missing information in networks. Link prediction helps identify edges that are likely to appear in the future, if they do not exist already. Entity resolution uses attributes and network structure data to link nodes that represent the same individual.

These techniques take advantage of many network features covered earlier in this text, including degree, clustering, and path lengths. This chapter introduced some of the basic methods for both tasks. As described above, many more sophisticated computational approaches to both link prediction and entity resolution exist, and those will make excellent further reading for computer scientists interested in this topic.

The results can be applied in many areas. Two short case studies discussed how link prediction can be used to recommend connections in social media and how entity resolution is useful for identifying duplicate accounts belonging to the same person. Link prediction is also particularly useful for network forecasting. Knowing which people in an organization are likely to connect can be used in many ways. Within companies, for example, this could be leveraged to make introductions and get collaborations moving faster. Within a criminal or terrorist organization, the predicted links could provide interesting intelligence about how the group will evolve. Entity resolution also has many other applications in social media and online. It is often applied to Census records, where data about people in multiple locations should be connected. It has similar anticrime and antiterrorism applications, linking aliases with true identities. Other applications include merging duplicate products in online shopping, merging duplicate web search results, and detecting spam.

Exercises



1. In the graph above, there are 28 pairs of nodes. Ten of those pairs already have edges between then (e.g., A and B, E and H). The remaining pairs have no direct connection. For each of the indirectly connected pairs, complete the following table with scores for the indicated link prediction formula.

Pair	Shortest Path	Common Neighbors	Jaccard Index	Adamic/ Adar	Preferential Attachment
A,E					
A,G					
A,H					
B,C					
B,E					
B,F					
B,G					
B,H					
C,D					
C,E					
C,F					
C,G					
C,H					
D,F					
D,G					
D,H					
E,F					
F,G					

2. For each of the formulas in the table for Exercise 1, list the pair of nodes or pair between which you would add an edge based on the scores. Assume only the top-rated pair is selected.
3. Are there any patterns that emerge in your analysis of the graph in exercises 1 and 2? Are there certain pairs that frequently receive high or low ratings? Can you explain that by analyzing their position in the graph structure?
4. Compute the *u* probability for the following attributes:
   a. Gender
   b. Marital status. Assume the options are married, single, divorced, and widowed and for simplicity, assume there are the same number of people in each category, like with gender or birth month.
5. On the website companion for the book, you will find a file with the 2010 U.S. Census data showing the population data for each zip code in Washington, D.C. The total population of D,C. in that dataset is 601,723. Compute the *u* probability for the following zip codes based on that dataset:
   a. 20010
   b. 20045
   c. 20535
   d. 20002
6. If we know our dataset is 100% correct with no errors and no missing data, what is the *m* probability for matching nodes to have the same birthdate?
7. Again, assuming the data is 100% correct with no errors or missing data, is the *m* probability for phone number 1? Why or why not?
8. Assume the following table describes the nodes in the graph above.

Node	First Name	Last Name	SSN	City	State	Marital Status
A	John	Doe	123-23-1324	Chicago	IL	Married
B	Jane	Doe	234-32-4321	Chicago	IL	Married
C	Robert	Donovan				Divorced
D	John	Smith	132-13-1321	Washington	DC	Single
E	William	Smith		Washington	DC	Single
F	Jeannette	D.	234-32-4321	Madison	WI	Married
G	Jeannette	Doe		Seattle	WA	Single
H	JB	Doe	123-23-1324	Madison	WI	Widowed

Using the same list of node pairs from exercise 1, compute the scores and fill in the table below using the following match and nonmatch scores. Use only matches or nonmatches, not partial matches in your scoring. Show your work.

Score	First Name	Last Name	SSN	City	State	Marital Status
Match	2.1	3.4	6.1	4.9	1.6	1.8
Nonmatch	−1.3	−4.8	−3.3	−2.7	−1.9	−2

Pair	Score
A,E	
A,G	
A,H	
B,C	
B,E	
B,F	
B,G	
B,H	
C,D	
C,E	
C,F	
C,G	
C,H	
D,F	
D,G	
D,H	
E,F	
F,G	

9. Repeat exercise 8 but allow for partial matches. You can assign your own values between 0 and 1 for a partial match. For example, if one node's first name is "Michael" and another node has a first name listed as "M," you may decide to use a value of 0.3 as a partial match because the initial "M" could represent "Michael."

List each partial match, your value for it, and your reasoning for the value. Then, recompute the score for each pair of nodes listed above. Show your work.

10. For the following node pairs, compute their similarity for entity resolution using the specified methods.

Pair	Common Neighbors	Jaccard Index	Adamic/ Adar	Preferential Attachment
A,E				
A,B				
A,H				
B,C				
B,E				
C,F				
D,E				
D,H				
E,F				
F,H				

11. Give a weight of 7.3 to the Jaccard Index as you computed it in exercise 10. For the following pairs, give a new score that incorporates the Jaccard Index in addition to the attribute data. Do this by adding the Jaccard Index times its weight to the existing scores. Do this for the full matches (from exercise 8) and partial matches (from exercise 9).

Pair	Full Match Score	Partial Match Score
A,E		
A,H		
B,C		
B,E		
C,F		
D,H		
E,F		

12. Set the similarity threshold equal to 4.0.

- **a.** Which nodes will be merged based on your calculations in exercise 8?
- **b.** Which nodes will be merged based on your calculations in exercise 9?
- **c.** Which nodes will be merged based on your calculations in exercise 11?
- **d.** How does incorporating the network similarity data change the results of the entity resolution decision?

13. The conclusion listed several applications of entity resolution and link prediction. Pretend you are running a social network. Come up with your own new applications—one for link prediction and one for entity resolution. Describe the problem you would solve and how you would use the techniques presented in this chapter.