

---

# Social Relationship Analysis with Data Mining

John C. Hancock

Microsoft Corporation

[www.johnchancock.net](http://www.johnchancock.net)

November 2005

**Abstract:** The data mining algorithms in Microsoft® SQL Server™ 2005 can be used as a platform to develop powerful applications, such as helping to understand complex relationships between people. This paper describes how to build the data mining model, implement the CLR stored procedures and display the results using Reporting Services.

The sample files for this paper can be downloaded from

[http://www.johnchancock.net/downloads/Social Relationships and Data Mining.zip](http://www.johnchancock.net/downloads/Social%20Relationships%20and%20Data%20Mining.zip)



# Contents

<b>Social Relationship Analysis with Data Mining .....</b>	<b>1</b>
Introduction .....	1
Building the SQL Server Database .....	1
Data Mining for Relationships.....	2
Creating the Analysis Services Project .....	2
Editing the Mining Model .....	4
Exploring the Mining Model.....	5
Building Stored Procedures for Data Mining .....	6
Context.....	6
Debugging .....	7
Social Relationships Stored Procedures .....	7
Reporting on Data Mining Models .....	9
Social Relationships Reports .....	10
Conclusion .....	10
Resources .....	10



# Introduction

Studying the complex interactions between people is an area that is getting an increasing amount of attention. Sociologists often turn to concepts from the field of network analysis to help them understand and quantify these ideas, such as how central a particular person is within their social network, or which subgroups of people are closely related. This paper will describe how to use the data mining platform in SQL Server 2005 to build an application that can be applied to a set of data and begin to answer some of these questions.

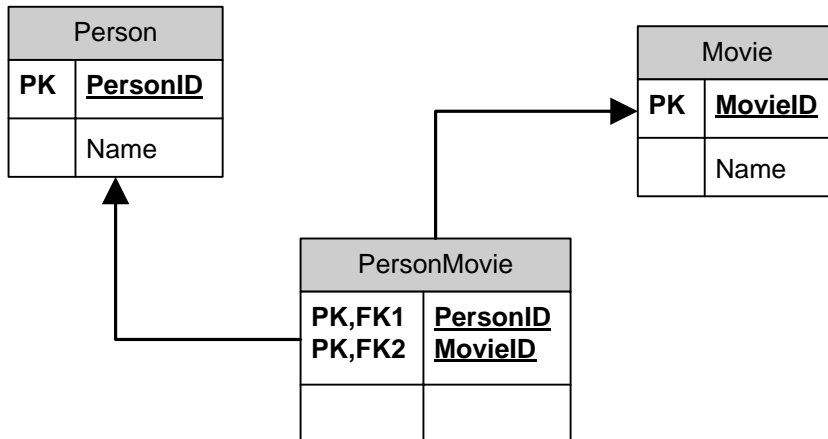
There are many examples of networks of people working together, such as collaboration between researchers who publish scientific papers together, or the loose connections between members of a terrorist organization. A very complex network for which there is (fortunately) a lot of data available is the film industry. Actors and directors who work together on different movies have relationships of differing strengths. For example, some pairs of actors have only worked on a single movie together, while others have worked together in many of the same movies.

Even actors who have not worked directly with each other may be related by a third party who has worked with both of these actors. Mathematicians often refer to the length of this shortest path as the distance between two people, but the commonly used concept of **degrees of separation** represents the same idea; that is, two people who have worked directly together have 1 degree of separation, while two people who are only related by a third person have 2 degrees of separation. For a given person, the list of people that they have worked directly with (1 degree of separation) is known as their **neighbourhood**, and the number of these direct connections to the person is an important measure of their centrality known as their **degree**.

This paper describes how to approach the above questions by loading some sample data (see Resources section) on actors and movies into relational tables, and then building a data mining model. The paper also describes how to implement the stored procedures to calculate some of the measures described above, and display the results to users using Reporting Services.

## Building the SQL Server Database

Since the sample data that we are using is in XML format, we will need to build a relational database and load the information into it. Each person could be a director or cast member for multiple movies, and each movie has multiple people starring in it. The schema of the database for our example is simple, see Figure 1.



**Figure 1: Hollywood Database Schema**

When the tables have been created, the data can be loaded using SQL Server Integration Services (SSIS), which in SQL Server 2005 has great support for loading complex XML documents.

## Data Mining for Relationships

The data mining algorithms in SQL Server 2005 include the Microsoft Association Rules algorithm, which identifies combinations of items that tend to appear together. The algorithm is primarily used for market-basket analysis, but since it identifies significant relationships between nodes, we can use it for social relationship analysis too.

Using data mining to build the set of associations has a lot of benefits over the alternative approach of writing code directly against relational or other data structures. The mining model provides the flexibility to connect to many different schemas, and does the work of creating the rules. In the sample, I built the model directly against the list of actors in movies, rather than having to translate this into a set of nodes (actors) and links.

Also, the algorithm parameters can be used to control the number of relationships that are identified in very large data sets, for example by restricting the model to only consider relationships that occur a minimum number of times in the data (this is known as support). Finally, stored procedures can be written against this model so that applications such as Reporting Services can query the relationship logic using standard, server-based mechanisms.

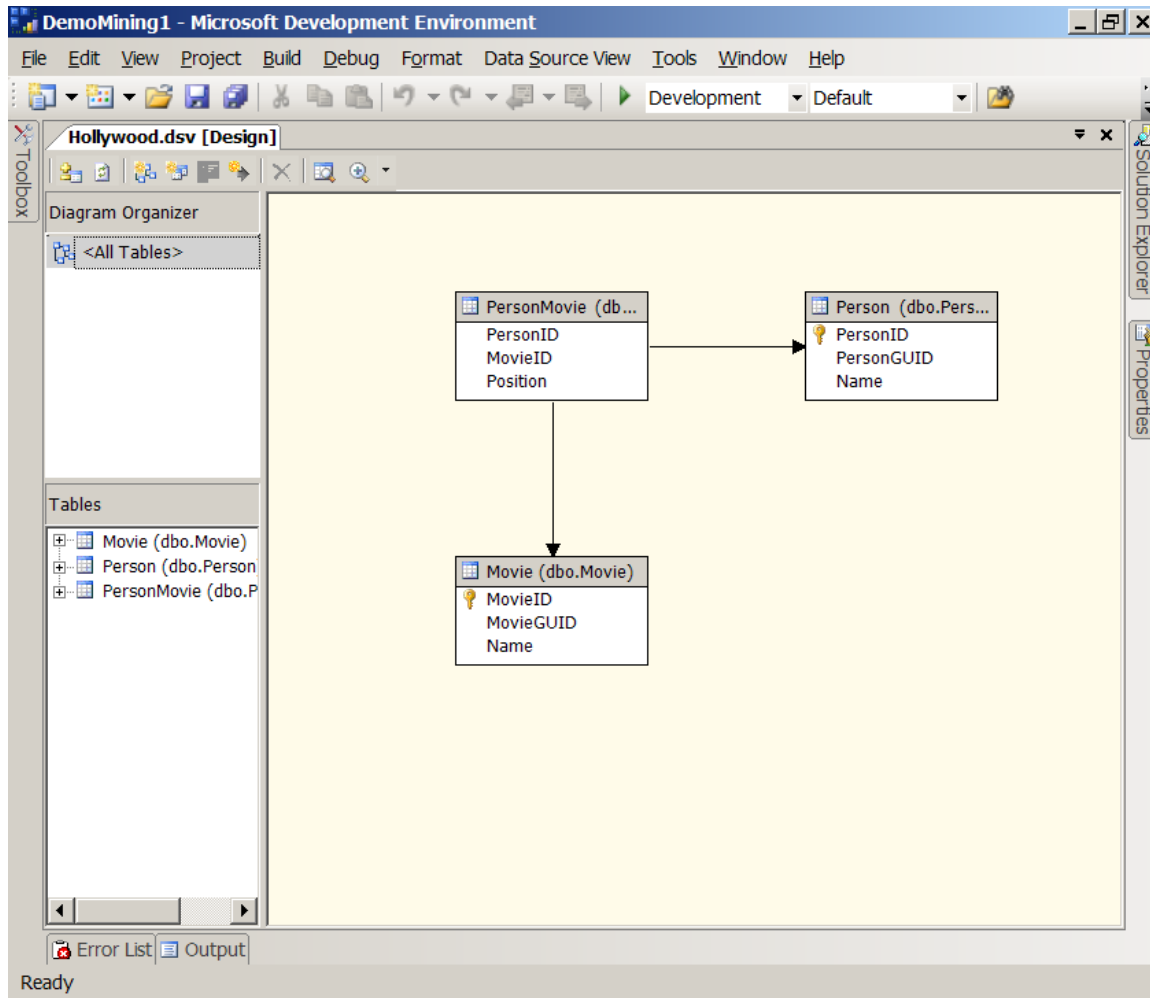
For a more detailed introduction to data mining with SQL Server 2005, my book "Practical Business Intelligence with SQL Server 2005" has a chapter on data mining which is available as a free download from Addison-Wesley – see the link on [www.johnchancock.net](http://www.johnchancock.net).

## Creating the Analysis Services Project

The data mining model is part of an Analysis Services project that is created using the Business Intelligence Development Studio.

## To create the Analysis Services project

1. Open Business Intelligence Development Studio, and create a new Analysis Services project named **Hollywood Mining**.
2. Create a new data source against the Hollywood SQL Server database, using the **SQL Native Client**.
3. Create a new data source view, including all three tables (**Person, Movie, and PersonMovie**). The relationships between these tables are automatically detected because of the foreign key constraints that are defined in the SQL Server database.



**Figure 2: Data source view**

Now that the project has been created and the data source defined, we can move on to creating the data mining model.

## To create the Mining Model

1. In Solution Explorer, right-click **Mining Structures**, and then click **New Mining Structure**.

The Data Mining Wizard opens.

2. On the Welcome page, click **Next**
3. Click **From existing relational database or data warehouse**, and then click **Next**.
4. Under **What data mining technique do you want to use?**, click **Microsoft Association Rules**.
5. Click **Next**.  
By default, Hollywood is selected in the **Select data source view** window.
6. Select the **Case** check box next to the **Movie** table and the **Nested** check box next to the **PersonMovie** table, and then click **Next**.
7. Check the **Key** check box next to **MovieID**, and the **Key, Input** and **Predictable** check boxes next to the **PersonID** column, and click **Next**.
8. Click **Next**.
9. In the **Mining structure name** and **Mining model name** boxes, type *Person Associations By Movie*, and then click **Finish**.

## Editing the Mining Model

The data mining model editor is displayed with the new model. Since we selected numeric ID's in the wizard, we need to edit the model to use the correct names for people and movies.

### To change the Mining Model to use names

1. On the Mining Structure tab, select the MovieID column on the left hand side, right-click and show Properties.
2. For the Name Column property, select (new).
3. Select Name from the list of source columns, and click OK.
4. Now select the PersonID column on the left hand side, right-click and show Properties.
5. For the Name Column property, select (new).
6. Select Person as the source table, then select Name from the list of source columns, and click OK.

Before processing the model, the default values of three of the parameters need to be changed: Minimum\_Support, Minimum\_Probability and Maximum\_Itemset\_Size. Support defines the percentage of cases in which a rule must exist before it is considered to be a valid rule. Probability defines how likely an association must be before it can be considered valid. The itemset size defines how many items are involved in the association.

### To adjust Association model parameters

1. Switch from the Mining Structure tab to the **Mining Models** tab.
2. Right-click **Microsoft\_Association\_Rules**, and select **Set Algorithm Parameters**.

The **Algorithm Parameters** dialog box opens.

3. Set the following parameters:

Parameter	Value
MAXIMUM_ITEMSET_SIZE	2
MINIMUM_PROBABILITY	0.01
MINIMUM_SUPPORT	0.0001*

\* Note that for this system, we will be using a minimum\_support parameter value that is **much** lower than it normally would be. This is so that all relationships between people are identified, even if they have only starred in a single movie together. The default parameter values are much more sensible for most cases. Also, the maximum itemset size is set to 2 so that only pair relationships are considered.

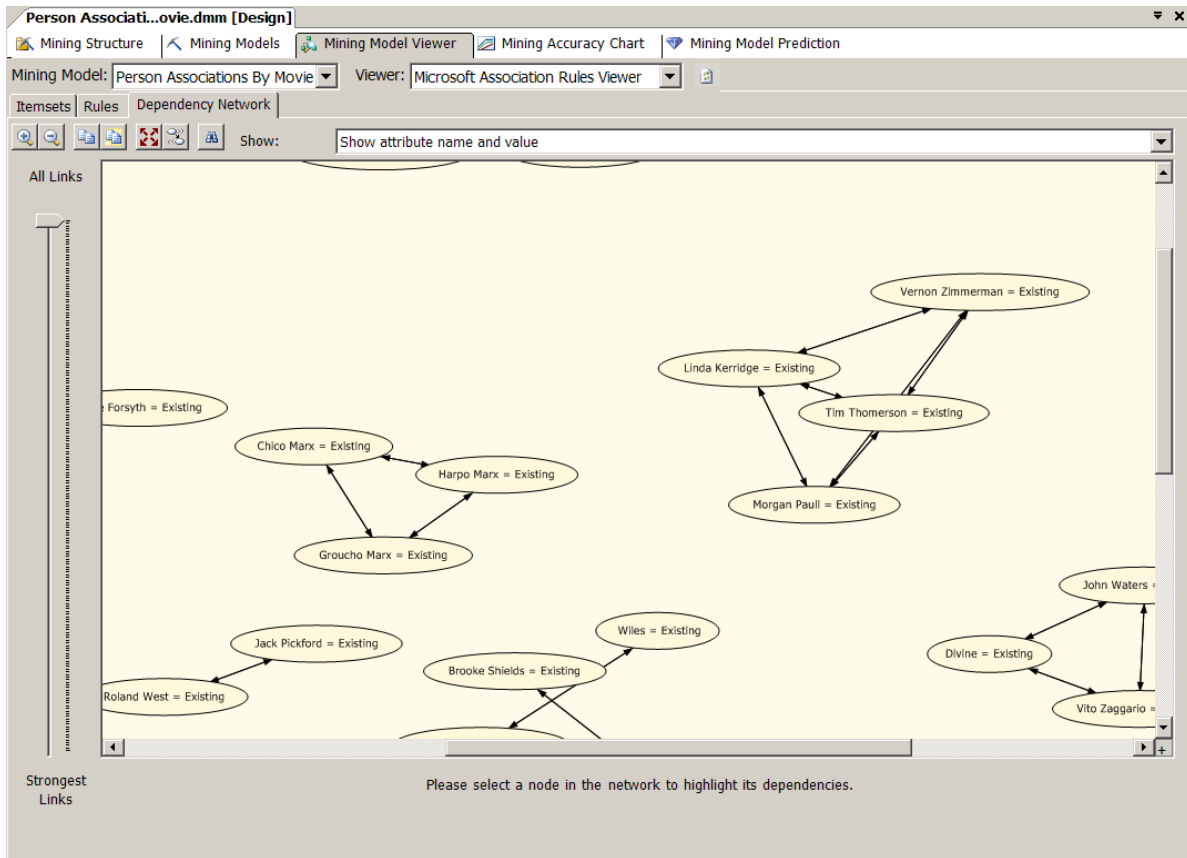
## Exploring the Mining Model

Now that you have created the model, you can deploy the project to the server, load the data by processing the model, and then view the results. Click on the Mining Model Viewer tab, and you will be prompted to deploy to the server and then process – you can use the default settings for these steps.

The **Itemsets** tab shows the list of all the pairs of actors that were identified (those with a size of 2) as well as the individual actors (size 1). This list can be restricted to pairs only by changing the minimum itemset size at the top of the tab from 0 to 2. You can also search for pairs that include a specific actor by typing the actor’s name in the Filter Itemset text box. The list is easier to read if you select “Show attribute name only”.

The Hollywood sample data includes some interesting pairs; the list is sorted by support so it shows the pairs of people who have been in the most movies together at the top of the list. For example, Woody Allen with Mia Farrow has a very high support – movie buffs will recognize that these two people are a famous Hollywood couple, so it’s no real surprise that these two show up together! There are also some other interesting pairs in this list, such as Stan Laurel with Oliver Hardy, and Thomas Chong with Cheech Marin.

The **Dependency Network** tab contains a graphical view of the relationships. By looking at the network in this way we can start to get the feeling that groups of people are highly related to each other, and also have connections to other groups. For example, one cluster in the sample data shows three people that are fairly well connected to each other – closer examination shows that these are the three Marx brothers.



**Figure 3: Dependency Network Viewer**

## Building Stored Procedures for Data Mining

Stored procedures are implemented as public methods on a class in a .NET assembly, which is then registered with Analysis Services. Methods that return a DataTable or DataSet can be executed using the CALL syntax in DMX queries, while methods that return value types like strings or numbers can be used as functions. The Microsoft.AnalysisServices.AdomdServer namespace is used to access data mining objects (as well as OLAP objects) in server-side code.

### Context

Since assemblies can be registered so that they are available for all databases on the server, your stored procedures will often need to obtain information such as the current database. This information is supplied by the Context class in the Microsoft.AnalysisServices.AdomdServer namespace, which includes static methods such as CurrentDatabaseName and CurrentMiningModel.

The Context class also includes a Boolean property, ExecuteForPrepare, which will be set to True when your code is called by a client that is not executing the stored

procedure, but is simply looking for a schema definition (such as a reporting tool that needs to display a list of fields to the user). You should check this at the beginning of your stored procedure, and not perform any long-running tasks or raise any errors if it is set to True. A good practice is to create an empty DataSet or DataTable with the appropriate columns at the beginning of your stored procedure, and then return this when ExecuteForPrepare is set. You will need to add the SafeToPrepare attribute to your method to indicate that it supports this behaviour.

## Debugging

The great news about building CLR stored procedures is that they are very easy to debug. After you have compiled your assembly and registered it with the server, you can simply attach to the msmdsrv.exe process, set a breakpoint, and you will be stepping through your code. If you change the code and recompile, you will need to de-register the assembly and then register it again.

## Social Relationships Stored Procedures

The stored procedures for this application return a list of people who are related to other people. In this case, the people are related by having starred in the same movie together, but since the data mining model contains only the relationships we can write code that works regardless of the underlying data.

Since the mining model only contains a set of rules for these relationships, the main task for the program to solve is how to derive lists of neighbours for individual nodes from the association rules. The PairwiseAssocList class in the sample code supplies this functionality, and can return an ArrayList of related nodes for a specified node.

## Building the PairwiseAssocList class

Each model has a Content property which contains a collection of nodes; for AssociationRules models, these nodes can be either itemsets or association rules. The rules will be used to build the list of relationships, but while we are iterating through the collection the itemsets can also be used to create a dictionary that relates the name of a node (an actor's name in our example) to their corresponding unique identifier. This will be useful because the user will be able to issue a stored procedure call using a name that they will know about (such as Kevin Bacon), even though the API only provides methods to retrieve nodes by their UniqueName, which is an internal numeric value.

### ItemSets

As you saw in the data mining model viewer earlier, itemsets can have a length of 1, 2 or more. The single itemsets have a Description property that contains the name of the actor along with the rule, such as "Kevin Bacon = Existing"; we can extract the names from this string to build a dictionary that relates the UniqueName to the actor's name.

### Rule length

Although we are only interested in relationships between two people, association rules can relate to multiple people, so one of the tasks is to identify the rule length and

restrict the code to rules with a length of 2. Each node of the AssociationRule type has a NodeRule property, which is a string describing the rule. For example,

```
<AssocRule support="1" confidence="1.000000" length="2" LHS_ID="1908"
rule="Brooke Shields = Existing -> Leo McKern = Existing"/>
```

The rule length can be extracted from this string by searching for "length=" and extracting the text up between the opening and closing quotes.

## Working with rules

Now that we have ensured that we are working with rules with a length of two, we can determine the information about the left and right nodes that make up the rule. The property that we need to know about these nodes is the UniqueName of each. For the left node, this information can once again be extracted from the NodeRule string, by searching for "LHS\_ID".

The unique name of the right node is a little trickier to obtain. Since the rule can relate more than 2 nodes, the NodeRule string does not contain an equivalent name for the other node. However, the Distribution property of the node contains a collection of statistical properties (MiningDistribution objects), one of which has a ValueType of 10 – the Value of this object is the unique name of the right hand node.

So, now that we know the names of both of the two nodes in the rule, we can retrieve the actual nodes themselves from the mining model using the GetNodeFromUniqueName method. The right hand node is then added to the list of related nodes that is stored for each left hand node, and this can then be used in our algorithm to find the neighbourhood of directly related nodes for any node in the model.

## Caching Data

Loading a PairwiseAssocList object requires iterating through the entire collection of nodes in a model, and can take several minutes for large models. In order to respond quickly to queries, the object needs to be cached once it has been loaded. We can use static variables to accomplish the caching, since the variables persist between stored procedure calls.

Each PairwiseAssocList contains the list for a single model, so a hashtable of these lists needs to be maintained for every model for which a stored procedure call has been made. When models are processed, the data that is cached in the static variable becomes out of date, so the PairwiseAssocList stores the LastProcessedDate of the model when it is loaded so that the cache can be cleared if the data is stale. Note that this data is only loaded when the first stored procedure call is made after the model has been re-processed, so this call will take much longer than usual (a cache warming mechanism could be used to avoid impacting users, such as scripting the model processing to issue a query to the model).

As with all static variables, care needs to be taken to avoid threading issues. When the data is being loaded for the first call, other calls are blocked from proceeding using the lock construct. The static variable persists until either Analysis Services is re-started, or the assembly is de-registered.

## GetSubGroup Stored Procedure

With the PairwiseAssocList loaded, the task of retrieving the subgroup that is related to a person is relatively simple. The GetSubGroup stored procedure retrieves the list of people who are directly related to the selected person, as well as the people who are in turn related to them. This is repeated until the MaxDegreeOfSeparation is reached.

The stored procedure also calculates the shortest path from the root node (or "ego") to every other person that is found. Since multiple paths of the same length can exist between 2 people, the stored procedure also picks the "best" shortest path, either by selecting the path through the person with higher support (which represents the highest level of relationship) or through the person with the higher degree (which is the person who has the most other relationships).

The results are sorted before returning the DataTable from the stored procedure. Since the stored procedure must return either a DataTable or DataSet rather than a DataView (which would allow sorting), the results either need to be loaded in the correct order in the first place, or sorted and copied into a new table.

## Testing the stored procedure

You can test the stored procedure using SQL Server Management Studio, by connecting to the Analysis Services database and creating a new DMX query with the following syntax:

```
CALL SNADemo.GetNeighbourhood('Person Associations By Movie', 'Kevin Bacon')
```

Remember that if this is the first time you are calling the stored procedure, it may take several minutes to cache the results if you are using a data set with thousands of movies and people.

## Reporting on Data Mining Models

Since the stored procedures above return information as a DataTable, the complex logic in the model can easily be used with Reporting Services (or other tools) using DMX queries. In the wizard's query builder, you will need to right-click anywhere in the dialog and choose "Query" rather than the default "Design", in order to specify stored procedure calls.

## Parameterized DMX queries

You can specify parameters for a DMX query by using a string expression for the query, such as

```
"CALL SNADemo.GetNeighbourhood('Person Associations By Movie', '" + Parameters!PersonName.Value + "')
```

To generate the list of fields that is available, it is easier to specify a query with hard-coded parameters when you initially create the report, and then replace this with an expression as above.

# Social Relationships Reports

The following reports make up the user interface for the social relationships application.

## Person Relationships

This report uses the GetSubGroup stored procedure to return a list of related people, within a parameterized number of degrees of separation. Users can click on the name of any person in the report, and a hyperlink action will display the Person Relationships report for that person. Also, users can click on any path to show the Movies in Common report.

## Movies In Common

For a specified path such as Kevin Bacon\Tom Cruise\Martin Scorsese, this report uses the underlying relational tables to show which movies each pair of actors has starred in. Since the relational query needs the list of actors in the format required for an IN clause, i.e. 'Kevin Bacon', 'Tom Cruise', 'Martin Scorsese', custom code is used in the report to translate the string to the correct format.

## Conclusion

This application could be extended in several interesting ways, for example by using more sophisticated algorithms to calculate measures of centrality other than just the degree, or by adding reports that present the data from the stored procedures in interesting ways, such as grouping the records by degree of separation to calculate any actor's "average Bacon number" (see Resources)

In summary, the combination of the data mining features in SQL Server 2005 with CLR stored procedures forms a powerful platform for solving problems that require extensive data manipulation.

## Resources

- The sample data for this paper was obtained from the Stanford University Database Group's [web site](#). Some manipulation of the XML data was required in order to load it correctly, and a lot of the data was not included.
- My book "Practical Business Intelligence with SQL Server 2005" has a chapter on the data mining features in SQL 2005, which is available as a free download from Addison-Wesley – see the link on [www.johnchancock.net](http://www.johnchancock.net).
- The [International Network for Social Network Analysis](#) has links to many important papers on the subject of SNA.
- "Social Network Analysis, A Handbook" by John Scott (Sage 2000) is a great introduction to the theory and practice of network analysis in the social sciences.
- For information on average Bacon numbers, see [Who is the Center of the Hollywood Universe?](#) at the University of Virginia.
- The [Erdős Number Project](#) studies research collaboration among mathematicians.