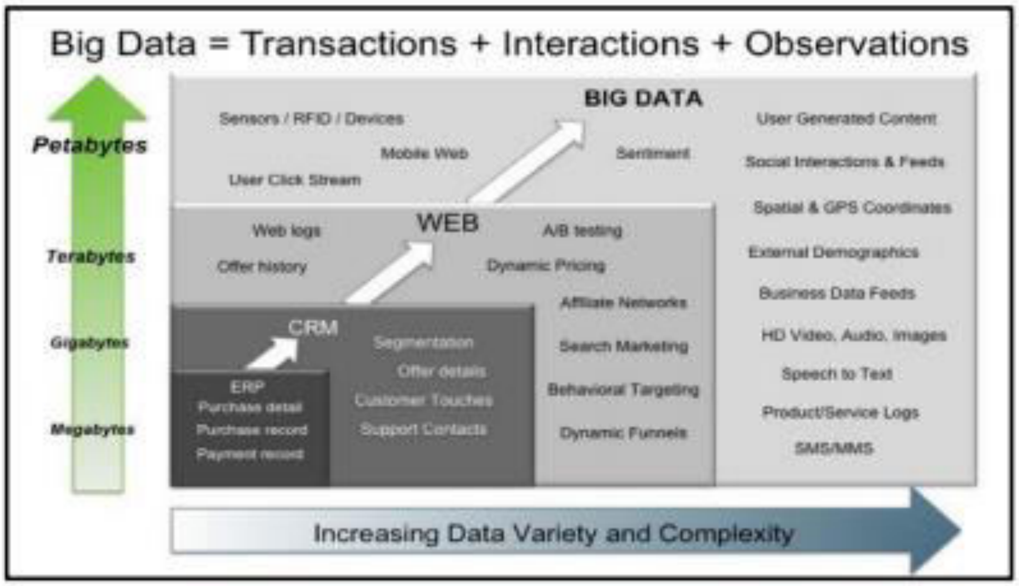
Unit -5

Medium data-Tradition

Relational SQL-based databases have traditionally been used to handle massive and dynamic datasets. Not only does this allow various components of the device to access the same data in one central location, but it also expands the size boundary to provide a lot of cheap hard drive space vs limited and costly RAM. Over the years, countless software programmes and applications have been created that link to a SQL database. There is almost always a SQL backend in whatever setting the analyst finds herself in. Of course, this includes Django, our personal favourite. It's only natural to handle social network data the same way we manage anything else: by saving and editing the graph directly in the database.

Big Data: The Future

Big Data is a collection of **data** that is **huge** in volume, yet growing exponentially with time. It is a **data** with so **large** size and complexity that none of traditional **data** management tools can store it or process it efficiently.



.

Figure 5.1: Big Data – Transactions, Interactions, Observations

Big Data Characteristics

The three Vs of Big data are Velocity, Volume and Variety.

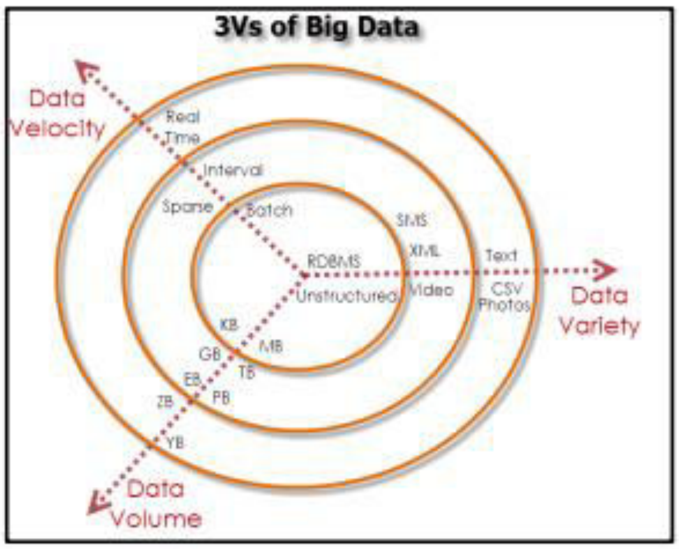


Figure 5.2: Big Data – Characteristics

VOLUME

The exponential growth in the data storage as the data is now more than text data. The data can be found in the format of videos, music's and large images on our social media channels. It is very common to have Terabytes and Petabytes of the storage system for enterprises. As the database grows the applications and architecture built to support the data needs to be re-evaluated quite often. Sometimes the same data is re-evaluated with multiple angles and even though the original data is the same the new found intelligence creates explosion of the data. The big volume indeed represents Big data.

VELOCITY

The data growth and social media explosion have changed how we look at the data. There was a time when we used to believe that data of yesterday is recent. The matter of the fact newspapers is still following that logic. However, news channels and radios have changed how fast we receive the news. Today, people reply on social media to update them with the latest happening. On social media sometimes a few seconds old messages (a tweet, status updates etc.) is not something interests users. They often discard old messages and pay attention to recent updates. The data movement is now almost real time and the update window has reduced to fractions of the seconds. This high velocity data represent Big Data.

VARIETY

Data can be stored in multiple format. For example database, excel, csv, access or for the matter of the fact, it can be stored in a simple text file. Sometimes the data is not even in the traditional format as we assume, it may be in the form of video, SMS, pdf or something we might have not thought about it. It is the need of the organization to arrange it and make it meaningful. It will be easy to do so if we have data in the same format, however it is not the case most of the time. The real world have data in many different formats and that is the challenge we need to overcome with the Big Data.

Big Data Layout

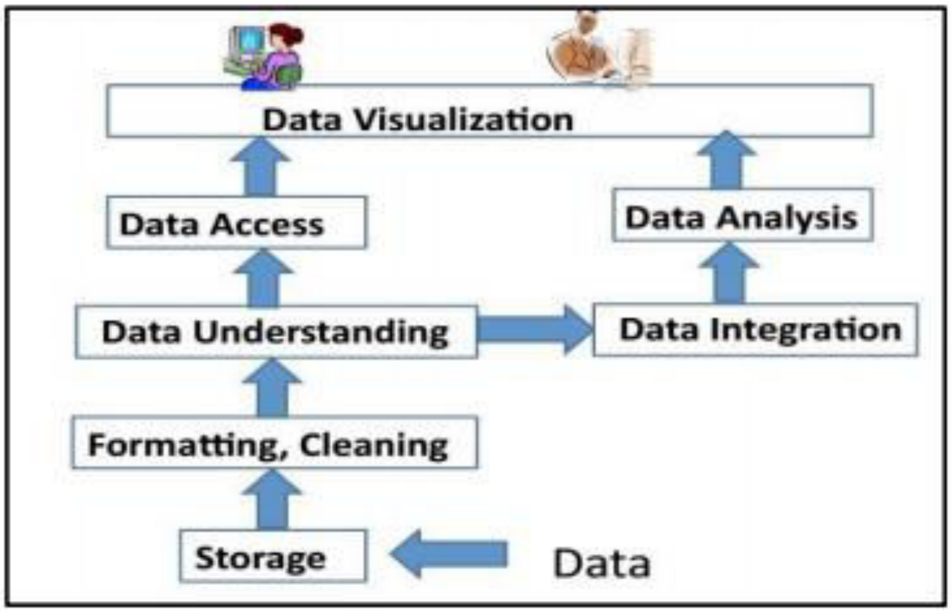


Figure 5.3: Big Data Layout

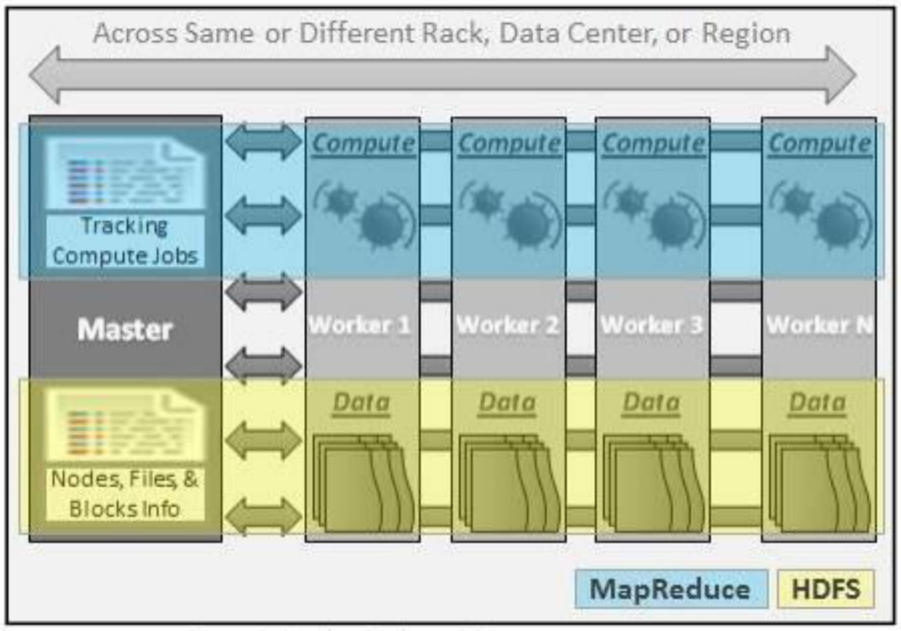


Figure 5.4: Big Data –Physical Architecture

1.APACHE HADOOP

Apache Hadoop is one of the main supportive element in Big Data technologies. It simplifies the processing of large amount of structured or unstructured data in a cheap manner. Hadoop is an open source project from apache that is continuously improving over the years. "Hadoop is basically a set of software libraries and frameworks to manage and process big amount of data from a single server to thousands of machines. It provides an efficient and powerful error detection mechanism based on application layer rather than relying upon hardware."

2.MAP REDUCE

MapReduce was introduced by google to create large amount of web search indexes. It is basically a framework to write applications that processes a large amount of structured or unstructured data over the web. MapReduce takes the query and breaks it into parts to run it on multiple nodes. By distributed query processing it makes it easy to maintain large amount of data by dividing the data into several different machines. Hadoop MapReduce is a software framework for easily writing applications to manage large amount of data sets with a highly fault tolerant manner. More tutorials and getting started guide can be found at Apache Documentation.

3.HDFS(Hadoop distributed file system)

HDFS is a java based file system that is used to store structured or unstructured data over large clusters of distributed servers. The data stored in HDFS has no restriction or rule to be applied, the data can be either fully unstructured of purely structured. In HDFS the work to make data senseful is done by developer's code only. Hadoop distributed file system provides a highly fault tolerant atmosphere with a deployment on low cost hardware machines. HDFS is now a part of Apache Hadoop project, more information and installation guide can be found at Apache HDFS documentation.

4. HIVE

Hive was originally developed by Facebook, now it is made open source for some time. Hive works something like a bridge in between sql and Hadoop, it is basically used to make Sql queries on Hadoop clusters. Apache Hive is basically a data warehouse that provides ad-hoc queries, data summarization and analysis of huge data sets stored in Hadoop compatible file systems. Hive provides a SQL like called HiveQL query based implementation of huge amount of data stored in Hadoop clusters.

5. PIG

Pig was introduced by yahoo and later on it was made fully open source. It also provides a bridge to query data over Hadoop clusters but unlike hive, it implements a script implementation to make Hadoop data access able by developers and business persons. Apache pig provides a high level programming platform for developers to process and analyses Big Data using user defined functions and programming efforts

Types of Data

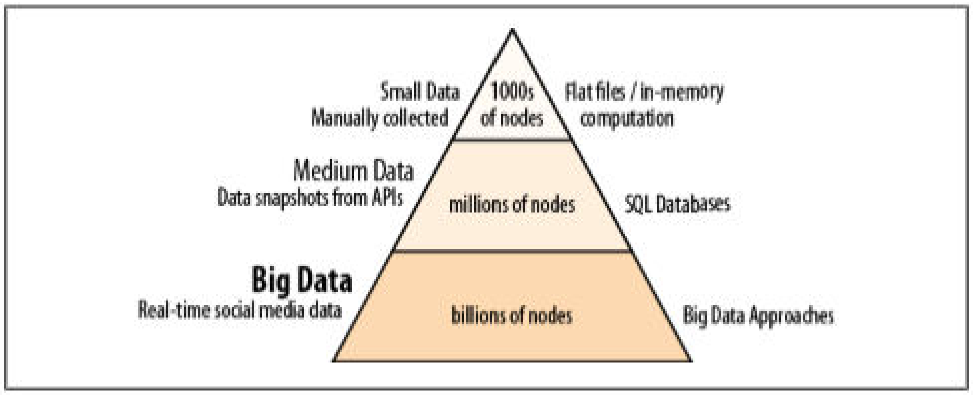


Figure 5.5 The realms of data sizes

Small data-flat files

- Flat files are great for quick analysis: they are portable and mostly human-readable.
- NetworkX is a tool to process graphs
- The graph we want to study has to fit entirely in memory.
- In general, memory usage is bounded by O(n)=n +n2=n(n+1).

NetworkX --supporting a number of different formats;

EdgeList Files

Perhaps the simplest way to store graph data is the Edgelist format. Its main advantage —besides simplicity—is the ability to easily import and export files into Excel or other spreadsheets.

- Edgelist files do not carry attribute data about nodes—but can carry arbitrary amounts of data concerning edges. The file format is exceedingly simple:
- <from_id> <to_id> <data1> <data2> ... <dataN>
- where from_id and to_id are string name or ID of the nodes that specify a graph edge, and columns <data1>...<dataN> are an arbitrary list of values that are assigned to this edge.
- freeform data can be kept in a Python dict format: <from_id> <to_id> <dict> foo bar {'weight':1,'color':'green' }

<u>.net Format</u>

_.net files represent a very expressing network data using ASCII text. This format was first used by a tool called Pajek, and is now something like a lingua franca of network data interchange. The files look something like this:

```
*Vertices 3
1 "Node1" 0.0 0.0 0.0 ic Green bc Brown
 2 "Node2" 0.0 0.0 0.0 ic Green bc Brown
    3  "Node3" 0.0 0.0 0.0 ic Green bc Brown
```

*Arcs 1 2 3 c Green 2 3 5 c Black *Edges 1 3 4 c Green The file is separated by sections, each section name starting with an asterisk (*):

*Vertices

- This section contains nodes' names and attributes.
- The header *Vertices is immediately followed by the number of vertices expected. Pajek expects this number to be accurate, although it is entirely optional elsewhere. Each of the lines contains the following columns:
- Numeric ID of the node (in sequence)
- Name of the node in quotes
- Node coordinates (x, y, z) and colors (background and foreground). These are non-zero only if the Pajek layout has been precomputed; they are optional for use in NetworkX .

*Arcs and *Edges

Arcs are directed edges, while Edges are undirected. If the graph is undirected, the Arcs section will not be present; if the graph is directed, the Edges section can be omitted. Each of the rows contains:

• from_id (a numeric ID of a node from the Vertices section)

• to_id (another numeric ID of a node)

• weight (value of the edge or arc)

• color (only if doing the layout in Pajek)

GML, GraphML, and other XML Formats

GML (Graph Modeling Language) and GraphML are two distant-cousin XML-based formats. A variety of tools support one or both of them—ranging from software for analysis of protein interactions, to project planning and supply chain planning systems. GraphML is more expressive, as it allows for hierarchical graphs, multigraphs, and other special cases .

A GraphML file looks something like this:

- <?xml version="1.0" encoding="UTF-8"?>

- <graphml> <key id="d0" for="node" attr.name="color" attr.type="string">yellow</key> <key id="d1" for="edge" attr.name="weight" attr.type="double"/> <graph id="G" edgedefault="undirected"> <node id="n0">
- <data key="d0">green</data> </node> <node id="n1"/> <node id="n2"> <data key="d0">blue</data> </node> <node id="n3"> <data key="d0">red</data> </node>

Ancient Binary Format—##h

- Files UCINET, a commercial Windows-based tool for SNA, has been considered “the Excel of SNA” for almost 20 years.
- It is a GUI-driven system encompassing hundreds of different modules, and is arguably the most comprehensive SNA tool on the planet.

“Medium Data”: Database Representation

A SQL-backed representation. Not only that, we will be able to add, remove, and modify nodes and edges in the stored graph directly in the database, without having to load the graph to memory

- def _prepare_tables(conn, name):
- c = conn.cursor()
- c.execute('drop table if exists "%s_edges"' % name)
- c.execute('drop table if exists "%s_nodes"' % name)
- c.execute('create table "%s_nodes" (node, attributes)' % name)
- c.execute('create table "%s_edges" (efrom, eto, attributes)' % name)
- conn.commit()
- c.close()

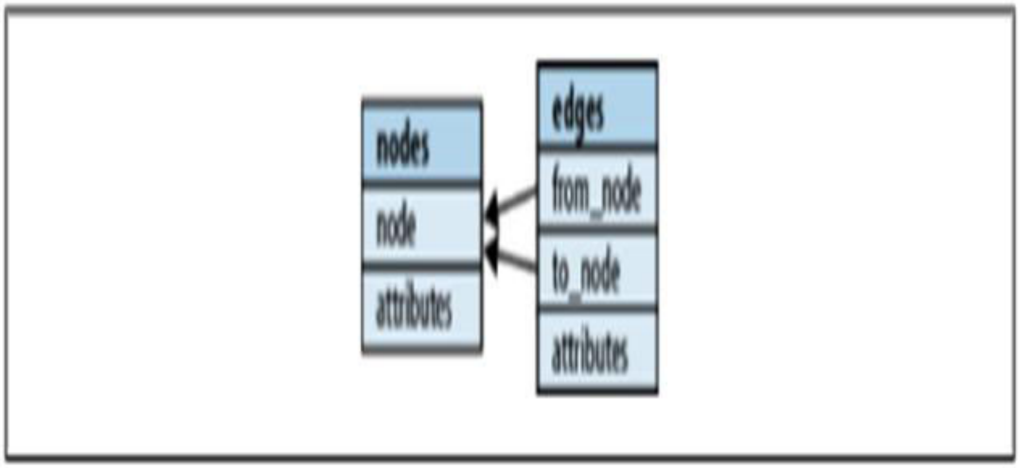


Figure 5.6 shows the database schema

Cursors

- Cursors are special database objects used to traverse tables.
- When a cursor executes a query, it returns another special object called a result set: a pointer into a set of results not yet returned by the server. The client then iterates over the result set, asking for one or several rows at a time, and the server only has to fetch them in small batches. This is partial lazy evaluation.

Nodes as Data, Attributes as ?

- In NetworkX a node is an object—it can be anything. Similarly, node/edge attributes are dictionaries of objects

Transactions

- Transactions are a safety feature that help prevent query errors from leaving data in a corrupt state, by executing a batch of operations atomically. Once a transaction is started, no changes are made to the actual data. It can then be either committed if no errors have occurred; or otherwise, rolled back. Committing a transaction commits the changes to the database, while a rollback discards them.

Names

- Aside from nodes and edges, NetworkX graphs have one more important piece of data: a name. In a database we can store multiple objects, so let's assume that a name uniquely identifies a graph.
- For simplicity, we'll use the graph name as part of the table name.

Functions and Decorators

- Before we proceed to methods that delete nodes and add/delete edges, we can observe an emerging pattern:
- 1. Get or create cursor
- 2. Execute a query
- 3. connection.commit()
- 4. cursor.close()

Function example

- def generalize(func):
- def generic(foo):
- cursor = connection.cursor()
- func(cursor, foo) # <- calling the wrapped function!
- cursor.close()

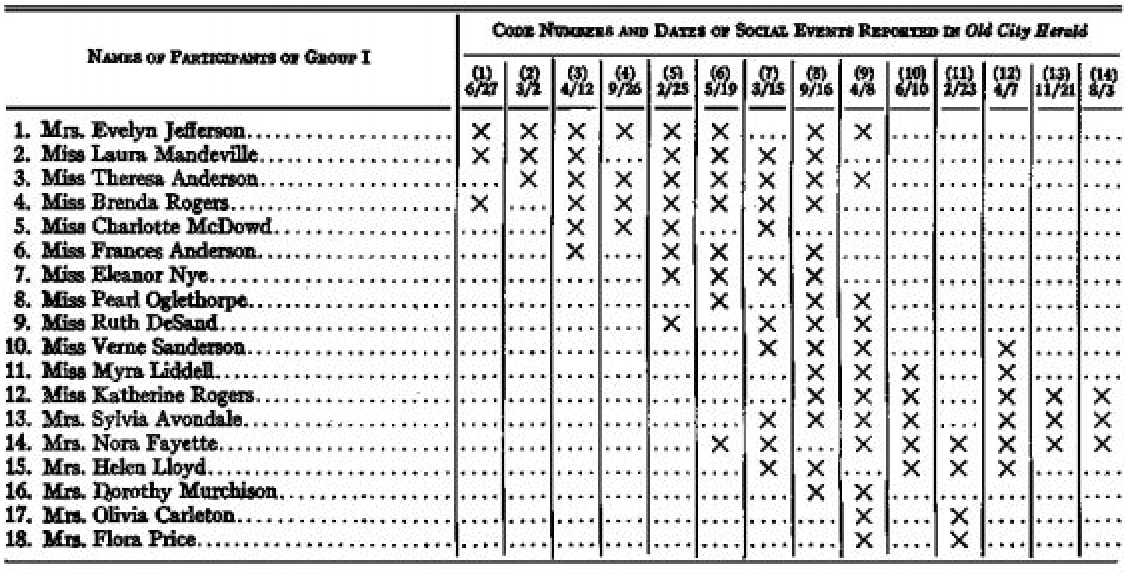
Decorator notation

- The technique of “wrapping” functions is commonly known as the Decorator Pattern. Python has special syntax for this:
- def add_node(cursor, node): cursor.execute('insert into my_nodes (node) value(?)', (foo,))
- add_node = generalize(add_node)

Working with 2 mode data

In social network analysis, 2-mode data refers to data recording ties between two sets of entities. In this context, the term “mode” refers to a class of entities – typically called actors, nodes or vertices – whose members have social ties with other members (in the 1- mode case) or with members of another class (in the 2-mode case). Most social network analysis is concerned with the 1-mode case, as in the analysis of friendship ties among a set of school children or advice-giving relations within an organization. The 2-mode case arises when researchers collect relations between classes of actors, such as persons and organizations, or persons and events. For example, a researcher might collect data on which students in a university belong to which campus organizations, or which employees in an organization participate in which electronic discussion forums. These kinds of data are often referred to as affiliations. Co-memberships in organizations or participation in events are typically thought of as providing opportunities for social relationships among individuals (and also as the consequences of pre-existing relationships). At the same time, ties between organizations through their members are thought to be conduits through which organizations influence each other.

Perhaps the best known example of 2-mode network analysis is contained in the study of class and race by Davis, Gardner and Gardner (henceforth DGG) published in the 1941 book Deep South. They followed 18 women over a nine-month period, and reported their participation in 14 events, such as a meeting of a social club, a church event, a party, and so on. Their original figure is shown in Figure 5.7



Names of Participants of Group I	Code Numbers and Dates of Social Events Reported in *Old City Herald*													
	(1) 6/27	(2) 3/2	(3) 4/12	(4) 9/26	(5) 2/25	(6) 5/19	(7) 3/15	(8) 9/16	(9) 4/8	(10) 6/10	(11) 2/23	(12) 4/7	(13) 11/21	(14) 8/3
1. Mrs. Evelyn Jefferson	×	×	×	×	×	×		×	×					
2. Miss Laura Mandeville	×	×	×		×	×	×	×						
3. Miss Theresa Anderson		×	×	×	×	×	×	×	×					
4. Miss Brenda Rogers	×		×	×	×	×	×	×						
5. Miss Charlotte McDowd			×	×	×		×							
6. Miss Frances Anderson			×		×	×		×						
7. Miss Eleanor Nye					×	×	×	×						
8. Miss Pearl Oglethorpe						×		×	×					
9. Miss Ruth DeSand					×		×	×	×					
10. Miss Verne Sanderson							×	×	×			×		
11. Miss Myra Liddell								×	×	×		×		
12. Miss Katherine Rogers								×	×	×		×	×	×
13. Mrs. Sylvia Avondale							×	×	×	×		×	×	×
14. Mrs. Nora Fayette						×	×		×	×	×	×	×	×
15. Mrs. Helen Lloyd							×	×		×	×	×		
16. Mrs. Dorothy Murchison								×	×					
17. Mrs. Olivia Carleton									×		×			
18. Mrs. Flora Price									×		×			

Figure 5.7 shows the DGG women-by-events matrix.

DGG used the data to investigate the extent to which social relations tended to occur within social classes

A typical data matrix has two dimensions or ways, corresponding to the rows and columns of the matrix. The number of ways in a matrix X can be thought of as the number of subscripts needed to represent a particular datum, as in xij. If we stack together a number of similarly sized 2-dimensional matrices, we can think of the result as a 3- dimensional or 3-way matrix. The modes of a matrix correspond to the distinct sets of entities indexed by the ways. In the DGG dataset described above, the rows correspond to women and the columns to a different class of entities, namely events. Hence, the matrix has two modes in addition to two ways; it is 2-way, 2-mode. In contrast, a persons-by-persons matrix A, in which aij = 1 if person i is friends with person j, is a 2-way, 1-mode matrix, because both ways point to the same set of entities.

In a sense, what constitutes different modes is up to the researcher. If we collect romantic ties among a group of people of both genders, we could construct a 2-mode men-bywomen matrix X in which xij = 1 if a romantic tie was observed between man i and woman j, and xij = 0 otherwise. Or, one could construct a larger 1-mode person-by-person matrix B also consisting of 1s and 0s in which it just happens that 1s only occur in cells where the row and column correspond to persons of different gender. Use of the men-bywomen matrix would imply that same-gender relations were impossible, whereas use of the person-by-person matrix would suggest that same-gender relations were logically possible, even if actually not observed.

Matrices recording relational information such as romantic ties can be represented as mathematical graphs as well. A graph G(V,E) consists of a set of nodes or vertices V together with a set of lines or edges E that connect them. An edge is simply an unordered pair of nodes (u,v). (In directed graphs or digraphs we use ordered pairs to indicate direction of the tie.) To indicate a tie between two nodes u and v, we simply include the pair (u,v) in the set E. The number of nodes in a graph is denoted by |V| or n.

A bipartite graph is a graph in which we can partition all nodes into two sets, V1 and V2, such that all edges include a member of V1 and a member of V2. The number of nodes in each vertex set is denoted n1 and n2, respectively.

Two-Mode Data in Social Network Analysis

Most social networks are conceived of as relations among a set of nodes, and therefore represented as a 1-mode matrix (typically of 1s and 0s) or a simple graph or digraph. For example, we might collect data on who is friendly with whom within an organization, or who injects drugs with whom in a neighborhood. However, 2-mode data are common in social network contexts as well. Typical examples include, actor-by-event attendance (as in the DGG data), actor by group membership (such as managers sitting on corporate boards), and actor by trait possession (such as adjective checklist data), and actor by object possession (such as material style of life scales in which inventories are made of household possessions). In many cases when 2-mode data are collected, the analytical interest is focused on one mode or the other. For example, in the DGG dataset, person-by-event attendances were collected in order to understand social relations among the women, specifically, whether women tended to have social relations primarily within their own social classes.

In the interlocking directorate literature, membership of executives on corporate boards is collected mainly in order to understand how corporations are intertwined, and how the structure of this connectivity affects corporate control of society. However, it can also occur that neither mode dominates our analytical focus and the primary interest is in the correspondence of one mode to the other. For example, a university might ask its faculty which courses they prefer to teach. Here, the objective is typically not to understand how faculty are related to each other through courses, nor how courses are related via faculty, but in the optimal assignments of persons to courses so that courses are staffed and faculty are not complaining.

Social networks and big data

Our data transmission, processing, and storage capacities have become practically limitless and inexpensive. As they currently exist, social networks are essential to us in one primary way: as massive data collectors of human behaviour patterns, enthusiastically powered by the examined themselves, and functioning on a global scale with near-instant response times. The end result is massive amounts of data.

NOSQL

NoSQL is a class of databases that is a departure (sometimes radical), from the classic RDBMS. NoSQL databases typically do not support a query language and lack a fixed schema. They exist in different variations: document stores, key-value stores, object stores, graph databases, and so on. Each variation is designed differently, with a particular context in mind. They are important to us for three common reasons: structure, size, and computation—the same reasons why relational databases are often ill-suited for our purposes. Here are some NoSQL examples:

BigTable

Google's implementation of a document store, designed to scale across assorted inexpensive hardware. They use it to store the whole Internet. The whole Internet.

HBase

Open source clone of BigTable, by the Apache Software Foundation. A work in progress, but moving quickly.

Hadoop

Not a database itself, but the underlying framework for distributed storage and processing, by ASF. The Hadoop project is very impressive and is one of the established standards for Big Data and cloud computing.

MongoDB

A prominent scalable high-performance document store, with support for indexes, queries and more. Uses JSON documents with dynamic schemas.

CouchDB

Another distributed document-oriented database, by ASF. Simple to install and operate, and supports MapReduce style indexing (more on MapReduce below).

Neo4j

A high-performance graph database written in Java.

Hive

A SQL-like database that can operate on plan text files in a variety of formats.

Structural Realities

The world is a fickle and ever-changing place. The structural coherence of Big Data is not always assured, owing to the fact that the data comes from a variety of different sources. Social media platforms are no exception. And more so, as the market progresses, the data structure evolves at a breakneck pace. Social networking sites are constantly updating their functionality and data standards. We must contend with not only inconsistencies between providers, but also changes in data from the same provider over time. Because of this structure flux, any attempt to coerce, say, a Twitter stream into a relational schema is futile.

Plain Text

The plain-text aspect is of great practical value to us. Suppose we were collecting Twitter data in the format shown above, from an HTTP stream.

With a SQL database, a data ingestion flow may look like this:

1. First, learn the data structure and possibly design a fitting schema.

2. Read the stream. For every record: a. Parse the record. b. Map record fields to an INSERT query. c. Execute query, handle unique constraint violations. d. Optionally update other tables.

With a text-file-based NoSQL database, the flow becomes this:

1. Read stream.

2. Write record to file.

Computational Complexities

Distributed computing is a technique of breaking a big task into smaller subtasks, distributing them across multiple machines, and combining the results. One can have a hardware array in-house or turn to the cloud and rent virtual machines en masse, on demand, for pennies per hour (each). Horizontal scaling means NoSQL databases are perfectly suited for the cloud.

Big Data at Work

Distributed computing

Distributed computing is a model in which components of a software system are shared among multiple computers. Even though the components are spread out across multiple computers, they are run as one system. This is done in order to improve efficiency and performance.

Hadoop, S3 and Map Reduce

A distributed cluster of computational nodes needs some way to access the source data. Hadoop by ASF makes that easy with HDFS (Hadoop file system) and features like data streaming, for chaining processes. Hadoop automatically distributes chunks of source data between nodes, controls their operation, and monitors their progress via a supplied Web interface

Amazon S3 (Simple Storage Service) is an online storage service by Amazon. It is distributed and scalable, though the implementation is kept under wraps and to the user it appears as a gigantic disk in some contexts and as a key-value store in others (the key is the filename and the value is the file content). S3 objects can be accessed over HTTP, making it a great Web storage medium. More interestingly, S3 can be accessed as a Hadoop file system.

MapReduce is a computational framework invented by Google and supported by Hadoop. A MapReduce job consists, not surprisingly, of a map step and a reduce step— both names of common functions in functional programming, though not exactly the same. Map step takes the source input, one record at a time, and outputs some meaningful data about it. Multiple copies of the map step work each on a different subset of the source data. Reduce step takes the output of several (or all) map steps and combines them.

Hive

Hive is another project by ASF. It is unique for its effort to bring SQL back into a NoSQL world. Hive operates by transforming SQL queries into MapReduce jobs and presenting their results back in SQL form. Hive has the important ability to read and write flat files as SQL tables, using what they call a SerDe (serializer/deserializer) to map file contents to columns. There are SerDes provided for CSV, JSON, and regexp formats, and more can be added. The best part is that it can read a collection of files on S3.

2-Mode Networks in Hive- a recipe

```
create external table hashtags ( tweet_id string, text string, indices string )

row format serde 'com.amazon.elasticmapreduce.JsonSerde'

with serdeproperties ('paths'='id, entity.text, entity.indices')

location 's3://our-json-data/output/hashtags/';
```

2-mode networks are just as easy with Hive as any other SQL server.

Using this table as an example, this will produce an edge list, weighted by cooccurrence counts, of the hashtag network:

```
hive> select x.text, y.text, count(x.tweet_id) from hashtags x full outer join hashtags y on (x.tweet_id=y.tweet_id) where x.text != y.text group by x.text, y.text;
```

Visualising Online Social networks

Visualization of social networks has a rich history, particularly within the social sciences, where node-link depictions of social

relations have been employed as an analytical tool since at least the 1930s. Linton Freeman documents the history of social network visualization within sociological research, providing examples of the ways in which spatial position, color, size, and shape can all be used to encode information . For example, networks can be arranged on a map to represent the geographic distribution of a population. Alternatively, algorithmically generated layouts have useful spatial properties: a force-directed layout can be quite effective for spatially grouping connected communities, while a radial layout intuitively portrays network distances from a central actor. Color, size, and shape have been used to encode both topological and non-topological properties such as centrality, categorization, and gender.

Vizster –Visualisation tool

Vizster was to build a visualization system that end-users of social networking services could use to facilitate discovery and increased awareness of their online community. We wanted to support the exploratory and playful aspects of Friendster while also giving users easier access to search and group patterns. While users regularly explored the network on Friendster, the linear format limited such explorations. This led us to develop richer network views and exploratory tools, while maintaining a local orientation. We also learned that the use of imagery was indispensable for identifying people and establishing a presentation of self, and so must play a central role in the visualization. In addition to helping support the current practices, we wanted to make sure that Vizster did not eliminate the data that helped users get a sense of people through their profiles. One example is the use of re-appropriated profile fields (e.g., inverting ages to identify teenagers) for coded communication within a subpopulation. For this reason, we realized that we must make searchable profile data very present and accessible in the visualization. These goals position Vizster differently from

traditional social network visualizations used as analysis tools by social science researchers. The following description includes the implications this approach has had for our design decisions, both in terms of presentation and the level of technical sophistication exposed by the visualization. Vizster presents social networks using a familiar node-link representation, where nodes represent members of the system and links represent the articulated "friendship" links between them (Figures 5.8). In this view, network members are presented using both their self-provided name and, if available, a representative photograph or image. The networks are presented as egocentric networks: networks consisting of an individual and their immediate friends. Users can expand the display by selecting nodes to make visible others' immediate friends as well. To the right of the network display is a panel presenting a person's profile. As discussed later, the profile panel also provides direct manipulation searches over profile text.

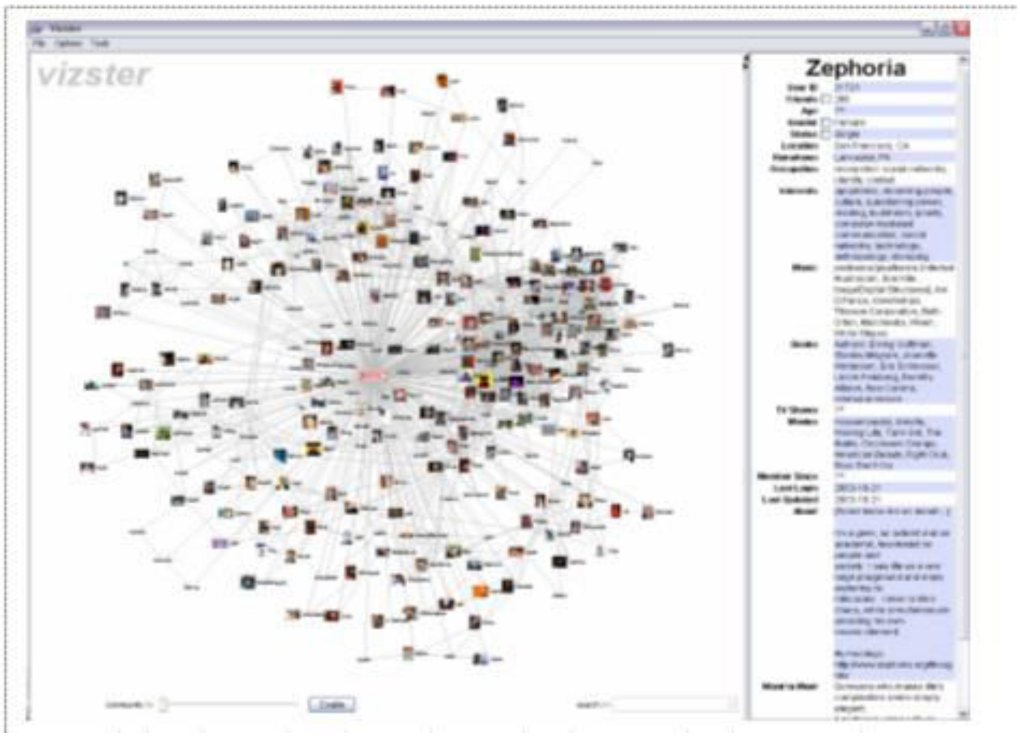


Figure 5.8 shows the screen short of Vizster visualization system

Applications of social network analysis

Social network analysis is used extensively in a wide range of applications and disciplines. Some common network analysis applications include data aggregation and mining, network propagation modeling, network modeling and sampling, user attribute and behavior analysis, community-maintained resource support, location-based interaction analysis, social sharing and filtering, recommender systems development, and link prediction and entity resolution.[56] In the private sector, businesses use social network analysis to support activities such as customer interaction and analysis, information system development analysis,[57] marketing, and business intelligence needs (see social media analytics). Some public sector uses include development of leader engagement strategies, analysis of individual and group engagement and media use, and community-based problem solving.

Security applications

Social network analysis is also used in intelligence, counter-intelligence and law enforcement activities. This technique allows the analysts to map covert organizations such as an espionage ring, an organized crime family or a street gang. The National Security Agency (NSA) uses its electronic surveillance programs to generate the data needed to perform this type of analysis on terrorist cells and other networks deemed relevant to national security. The NSA looks up to three nodes deep during this network analysis. After the initial mapping of the social network is complete, analysis is performed to determine the structure of the network and determine, for example, the leaders within the network This allows military or law enforcement assets to launch capture-or-kill decapitation attacks on the high-value targets in leadership positions to disrupt the functioning of the network. The NSA has been performing social network analysis on call detail records (CDRs), also known as metadata, since shortly after the September 11 attacks.

Textual analysis applications

Large textual corpora can be turned into networks and then analysed with the method of social network analysis. In these networks, the nodes are Social Actors, and the links are Actions. The extraction of these networks can be automated by using parsers. The resulting networks, which can contain thousands of nodes, are then analysed by using tools from network theory to identify the key actors, the key communities or parties, and general properties such as robustness or structural stability of the overall network, or centrality of certain nodes.This automates the approach introduced by Quantitative Narrative Analysis, whereby subject-verb-object triplets are identified with pairs of actors linked by an action, or pairs formed by actor-object. In other approaches, textual analysis is carried out considering the network of words co-occurring in a text (see for example the Semantic Brand Score). In these networks, nodes are words and links among them are weighted based on their frequency of co-occurrence (within a specific maximum range).

Internet applications

Social network analysis has also been applied to understanding online behavior by individuals, organizations, and between websites.Hyperlink analysis can be used to analyze the connections between websites or webpages to examine how information flows as individuals navigate the web. The connections between organizations has been analyzed via hyperlink analysis to examine which organizations within an issue community.

Social media internet applications

Social network analysis has been applied to social media as a tool to understand behavior between individuals or organizations through their linkages on social media websites such as Twitter and Facebook.

In computer-supported collaborative learning

One of the most current methods of the application of SNA is to the study of computer-supported collaborative learning (CSCL). When applied to CSCL, SNA is used to help understand how learners collaborate in terms of amount, frequency, and length, as well as the quality, topic, and strategies of communication. Additionally, SNA can focus on specific aspects of the network connection, or the entire network as a whole. It uses graphical representations, written representations, and data representations to help examine the connections within a CSCL network When applying SNA to a CSCL environment the interactions of the participants are treated as a social network. The focus of the analysis is on the "connections" made among the participants – how they interact and communicate – as opposed to how each participant behaved on his or her own.