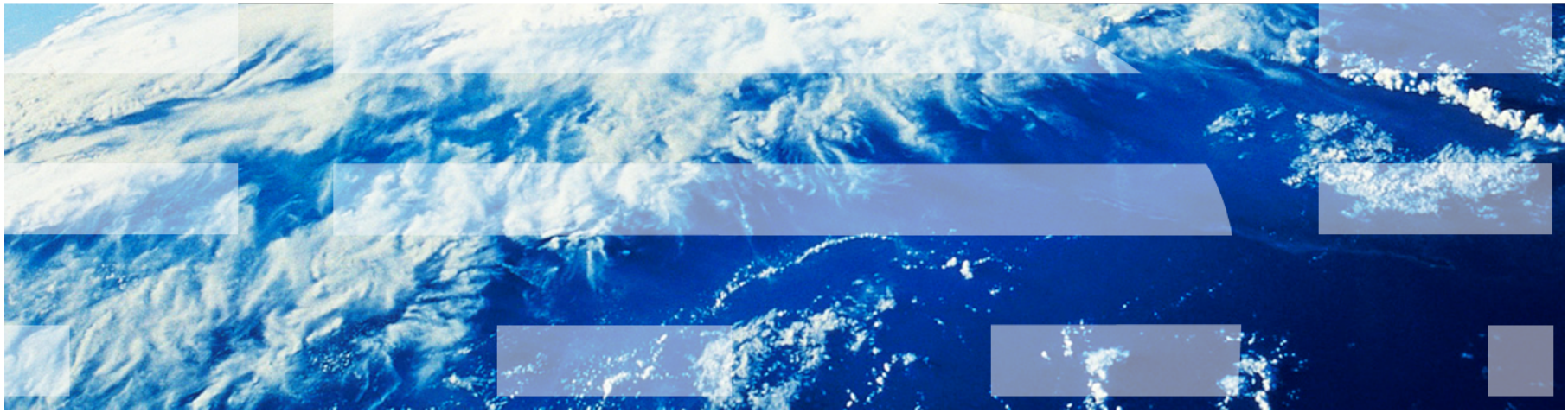
Columbia University

E6893 Big Data Analytics Lecture 6:

Graph Database and Analytics Use Case

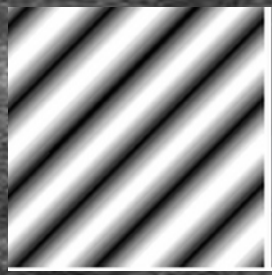
Ching-Yung Lin, Ph.D.

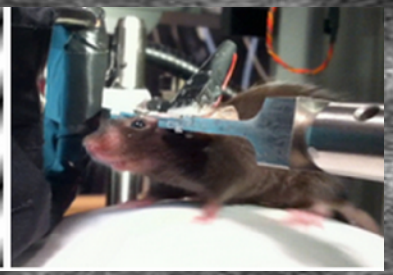
Adjunct Professor, Dept. of Electrical Engineering and Computer Science



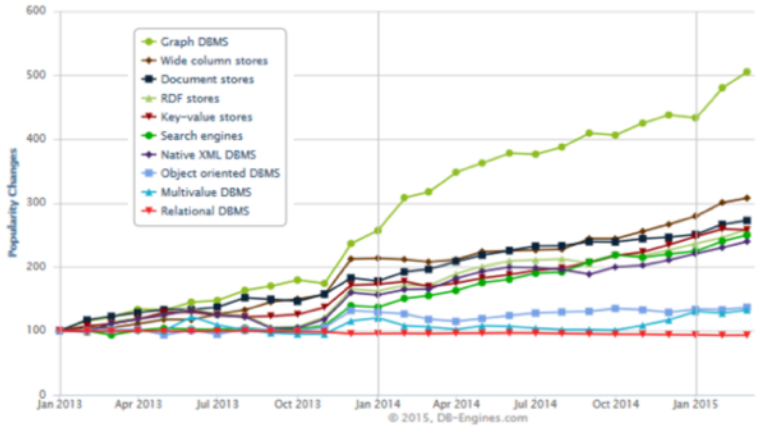
October 11, 2019

Network / Graph is the way we remember, we associate, and we understand.





Example: Graph Technology for Financial Service Sectors



- **Graph Database is much more efficient than traditional relational database**

- **How does FINRA analyze ~50B events per day TODAY?** - *Build a graph of market order events from multiple sources [ref]*

- **How did journalists uncover the Swiss Leak scandal in 2014 and also Panama Papers in 2016?** -- *Using graph database to uncover information thousands of accounts in more than 20 countries with links through millions of files [ref]*

RDF and SPARQL

not provided by spark

Columbia University

WHAT DO RDF AND SPARQL BRING TO BIG DATA PROJECTS?

Bob DuCharme
TopQuadrant, Charlottesville, Virginia

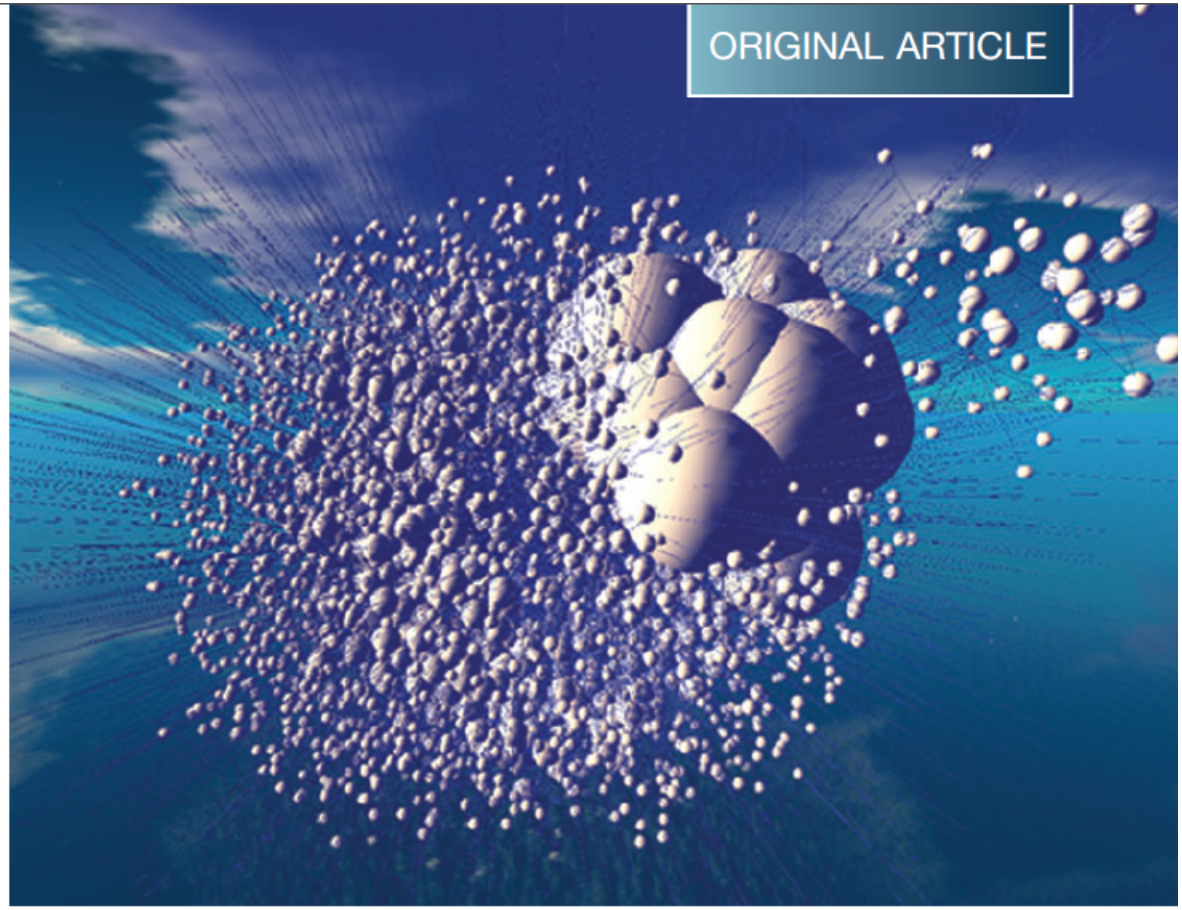


Photo Credit, Erich Bremer: http://www.ebremer.com/nexus/2011-05-15

Resource Description Format (RDF)

- A W3C standard sicne 1999
- Triples try define entities and their relationship
- Example: A company has nince of part p1234 in stock, then a simplified triple rpresenting this might be {p1234 inStock 9}.
- Instance Identifier, Property Name, Property Value.
- In a proper RDF version of this triple, the representation will be more formal. They require uniform resource identifiers (URIs).

```
@prefix fbd: <http://foobarco.net/data/>.
@prefix fbv: <http://foobarco.net/vocab/>.

fbd:p1234 fbv:inStock "9".
fbd:p1234 fbv:supplier "Joe's Part Company".
```

An example complete description

```
@prefix fbd:<http://foobarco.net/data/>.
@prefix fbv:<http://foobarco.net/vocab/>.
fbd:p1234 fbv:inStock "9".
fbd:p1234 fbv:name "Blue reverse flange".
fbd:p1234 fbv:supplier fbd:s9483.
fbd:s9483 fbv:name "Joe's Part Company".
fbd:s9483 fbv:homePage "http://www.joespartco.com".
fbd:s9483 fbv:contactName "Gina Smith".
fbd:s9483 fbv:contactEmail "gina.smith@joespartco.com".
```

Advantages of RDF

- Virtually any RDF software can parse the lines shown above as self-contained, working data file.
  - You can declare properties if you want.
  - The RDF Schema standard lets you declare classes and relationships between properties and classes.
  - The flexibility that the lack of dependence on schemas is the first key to RDF's value.
- Split trips into several lines that won't affect their collective meaning, which makes sharding of data collections easy.
  - Multiple datasets can be combined into a usable whole with simple concatenation.
- For the inventory dataset's property name URIs, sharing of vocabulary makes easy to aggregate.

SPARQL – Query Langauge for RDF

The following SPQRL query asks for all property names and values associated with the fbd:s9483 resource:

```
PREFIX fbd: <http://foobarco.net/data/>

SELECT ?property ?value
WHERE {fbd:s9483 ?property ?value.}
```

输出s9483的property&value

The heart of any SPARQL query is the WHERE clause, which specifies the triples to pull out of the dataset. Various options for the rest of the query tell the SPARQL processor what to do with those triples, such as sorting, creating, or deleting triples. The above query's WHERE clause has a single triple pattern, which resembles a triple but may have variables substituted for any or all of the triple's three parts. The triple pattern above says that we're interested in triples that have fbd:s9483 as the subject and—because variables function as wildcards—anything at all in the triple's second and third parts.

The SPAQRL Query Result from the previous example

property	value
<http://foobarco.net/vocab/contactEmail>	"gina.smith@joespartco.com"
<http://foobarco.net/vocab/contactName>	"Gina Smith"
<http://foobarco.net/vocab/homePage>	"http://www.joespartco.com"
<http://foobarco.net/vocab/name>	"Joe's Part Company"

Another SPARQL Example

What is this query for?

```
PREFIX fbd:<http://foobarco.net/data/>
PREFIX fbv:<http://foobarco.net/vocab/>

SELECT ?flangeContactEmail
WHERE
{
   ?part fbv:name "Blue reverse flange".
   ?part fbv:supplier ?supplier.
   ?supplier fbv:contactEmail ?flangeContactEmail.
}
```

输出gina.smith@joespartco.com.
先找到id是p1234, 然后supplier是s9483.
再找到supplier的contactEmail

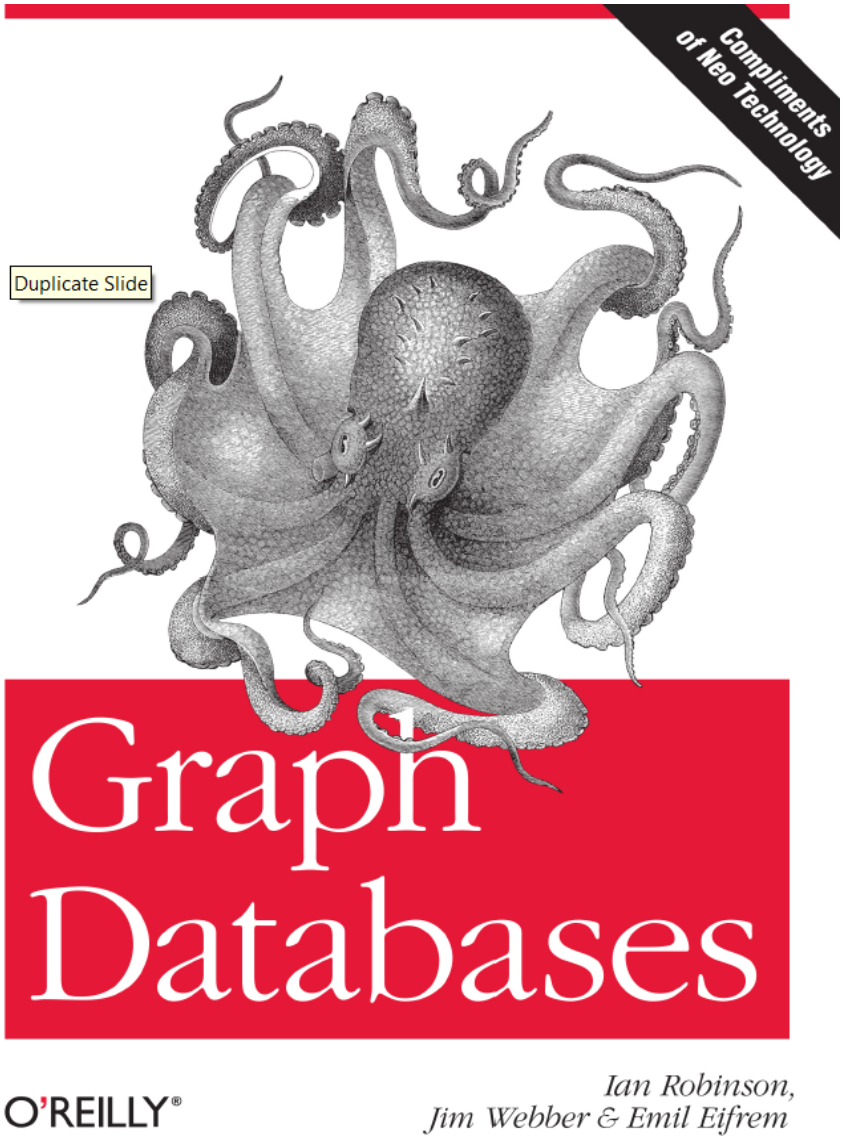
Data

```
@prefix fbd:<http://foobarco.net/data/>.
@prefix fbv:<http://foobarco.net/vocab/>.
fbd:p1234 fbv:inStock "9".
fbd:p1234 fbv:name "Blue reverse flange".
fbd:p1234 fbv:supplier fbd:s9483.
fbd:s9483 fbv:name "Joe's Part Company".
fbd:s9483 fbv:homePage "http://www.joespartco.com".
fbd:s9483 fbv:contactName "Gina Smith".
fbd:s9483 fbv:contactEmail "gina.smith@joespartco.com".
```

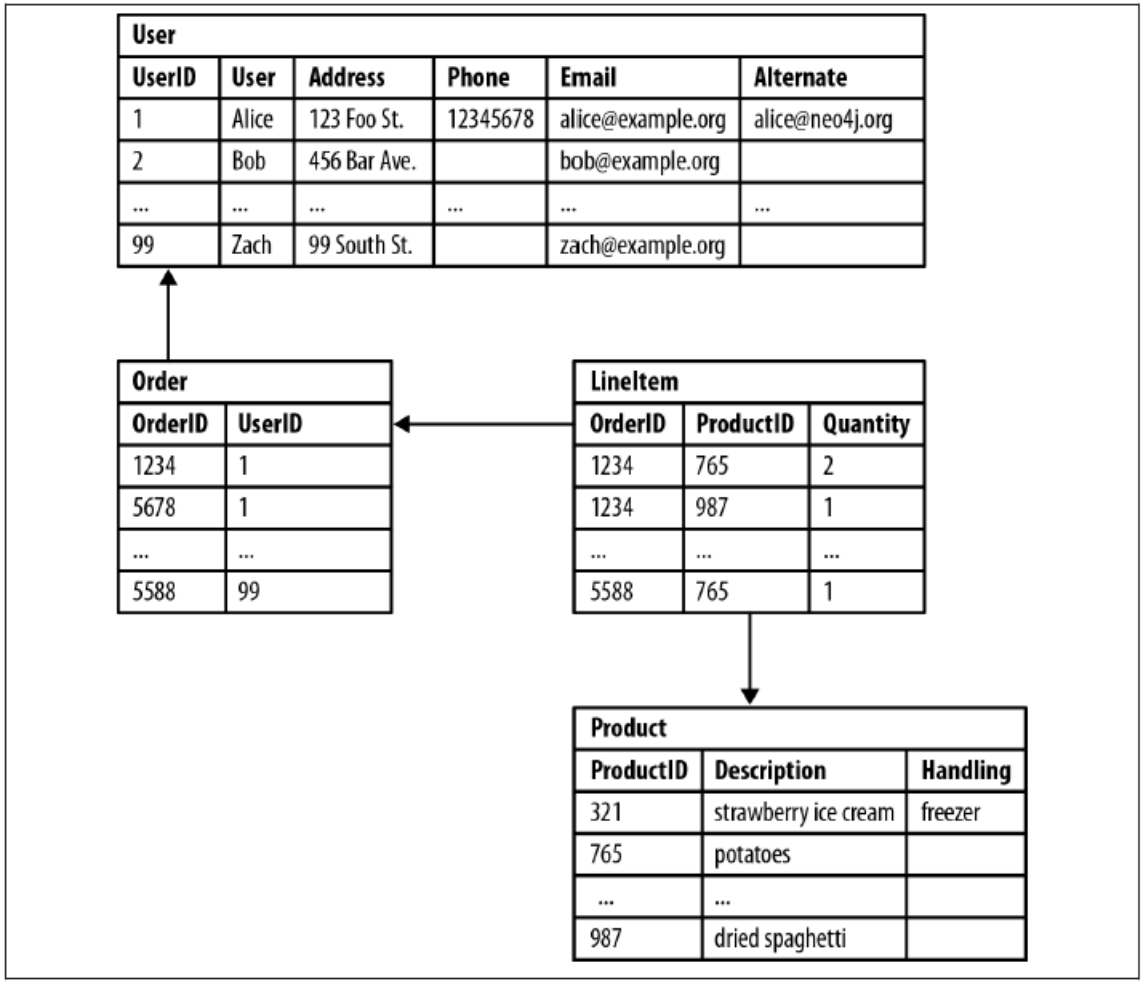
Open Source Software – Apache Jena

Property Graphs

Reference



A usual example



User

UserID	User	Address	Phone	Email	Alternate
1	Alice	123 Foo St.	12345678	alice@example.org	alice@neo4j.org
2	Bob	456 Bar Ave.		bob@example.org	
...	...	...	...	...	...
99	Zach	99 South St.		zach@example.org	

Order

OrderID	UserID
1234	1
5678	1
...	...
5588	99

LineItem

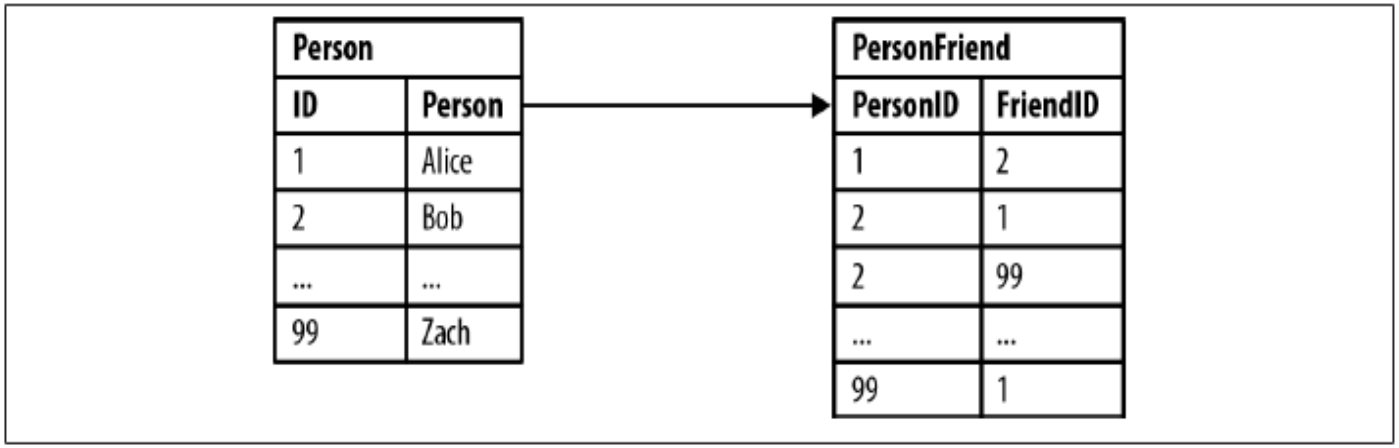
OrderID	ProductID	Quantity
1234	765	2
1234	987	1
...	...	...
5588	765	1

Product

ProductID	Description	Handling
321	strawberry ice cream	freezer
765	potatoes	
...	...	
987	dried spaghetti	

Figure 2-1. Semantic relationships are hidden in a relational database

Query Example – I



Person	
ID	Person
1	Alice
2	Bob
...	...
99	Zach

PersonFriend	
PersonID	FriendID
1	2
2	1
2	99
...	...
99	1

Figure 2-2. Modeling friends and friends-of-friends in a relational database

Asking "who are Bob's friends?" is easy, as shown in Example 2-1.

Example 2-1. Bob's friends

```
SELECT p1.Person
FROM Person p1 JOIN PersonFriend
  ON PersonFriend.FriendID = p1.ID
JOIN Person p2
  ON PersonFriend.PersonID = p2.ID
WHERE p2.Person = 'Bob'
```

Query Examples – II & III

Example 2-2. Who is friends with Bob?

```
SELECT p1.Person
FROM Person p1 JOIN PersonFriend
  ON PersonFriend.PersonID = p1.ID
JOIN Person p2
  ON PersonFriend.FriendID = p2.ID
WHERE p2.Person = 'Bob'
```

Example 2-3. Alice's friends-of-friends

```
SELECT p1.Person AS PERSON, p2.Person AS FRIEND_OF_FRIEND
FROM PersonFriend pf1 JOIN Person p1
  ON pf1.PersonID = p1.ID
JOIN PersonFriend pf2
  ON pf2.PersonID = pf1.FriendID
JOIN Person p2
  ON pf2.FriendID = p2.ID
WHERE p1.Person = 'Alice' AND pf2.FriendID <> p1.ID
```

Computational intensive

Graph Database Example

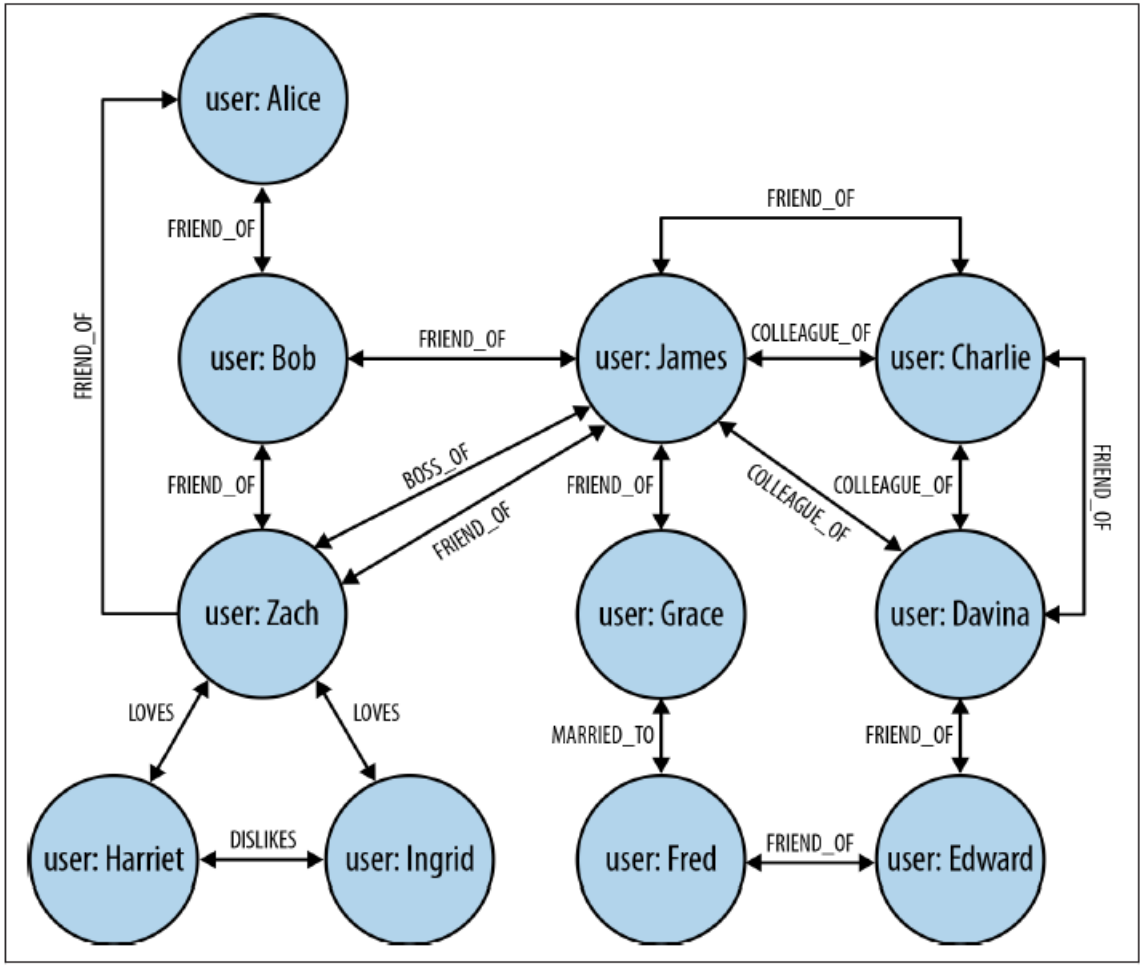


Figure 2-5. Easily modeling friends, colleagues, workers, and (unrequited) lovers in a graph

Execution Time in the example of finding extended friends (by Neo4j)

Partner and Vukotic's experiment seeks to find friends-of-friends in a social network, to a maximum depth of five. Given any two persons chosen at random, is there a path that connects them that is at most five relationships long? For a social network containing 1,000,000 people, each with approximately 50 friends, the results strongly suggest that graph databases are the best choice for connected data, as we see in Table 2-1.

Table 2-1. Finding extended friends in a relational database versus efficient finding in Neo4j

Depth	RDBMS execution time (s)	Neo4j execution time (s)	Records returned
2	0.016	0.01	~2500
3	30.267	0.168	~110,000
4	1543.505	1.359	~600,000
5	Unfinished	2.132	~800,000

Modeling Order History as a Graph

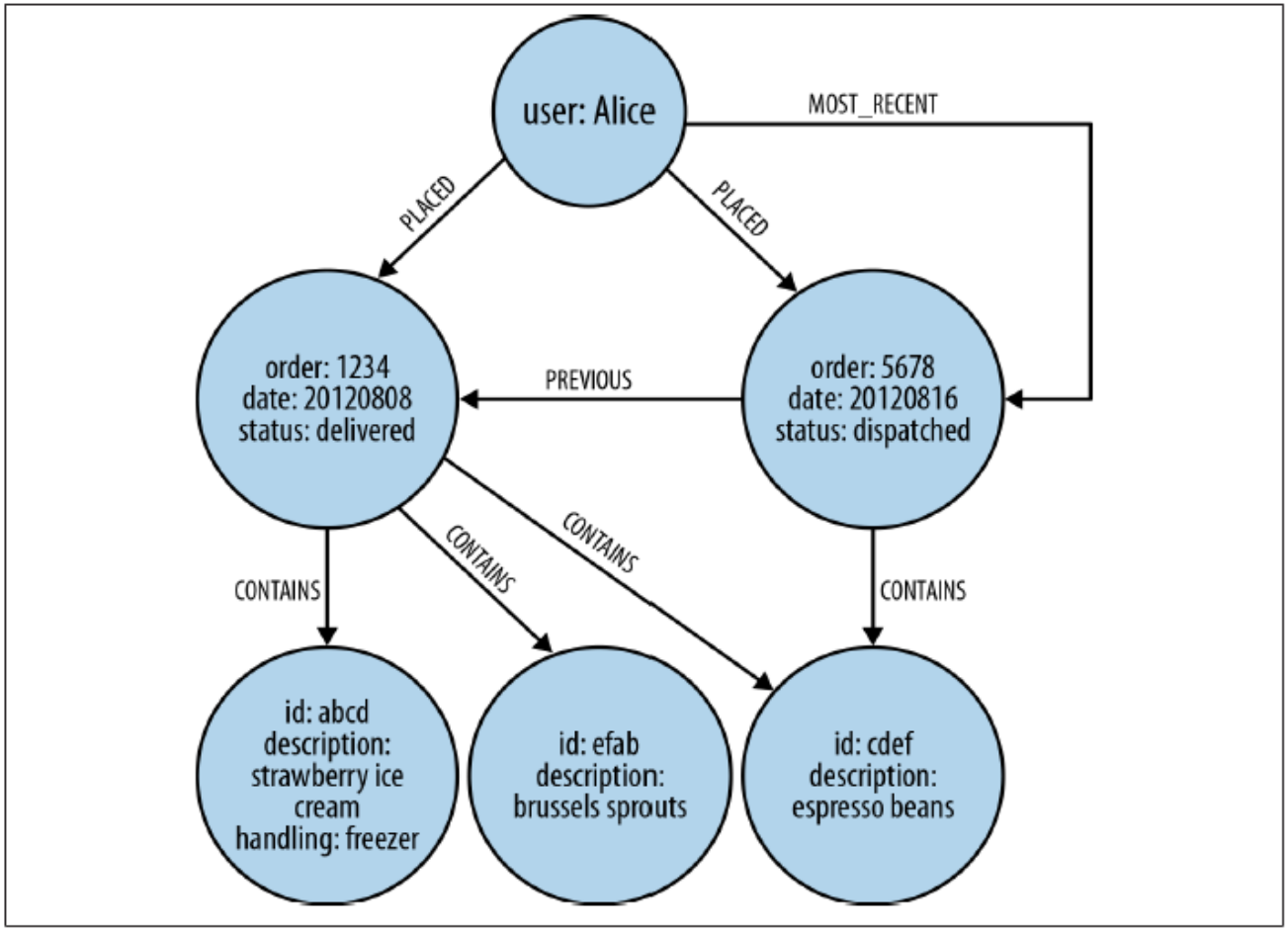


Figure 2-6. Modeling a user's order history in a graph

A query language on Property Graph – Cypher

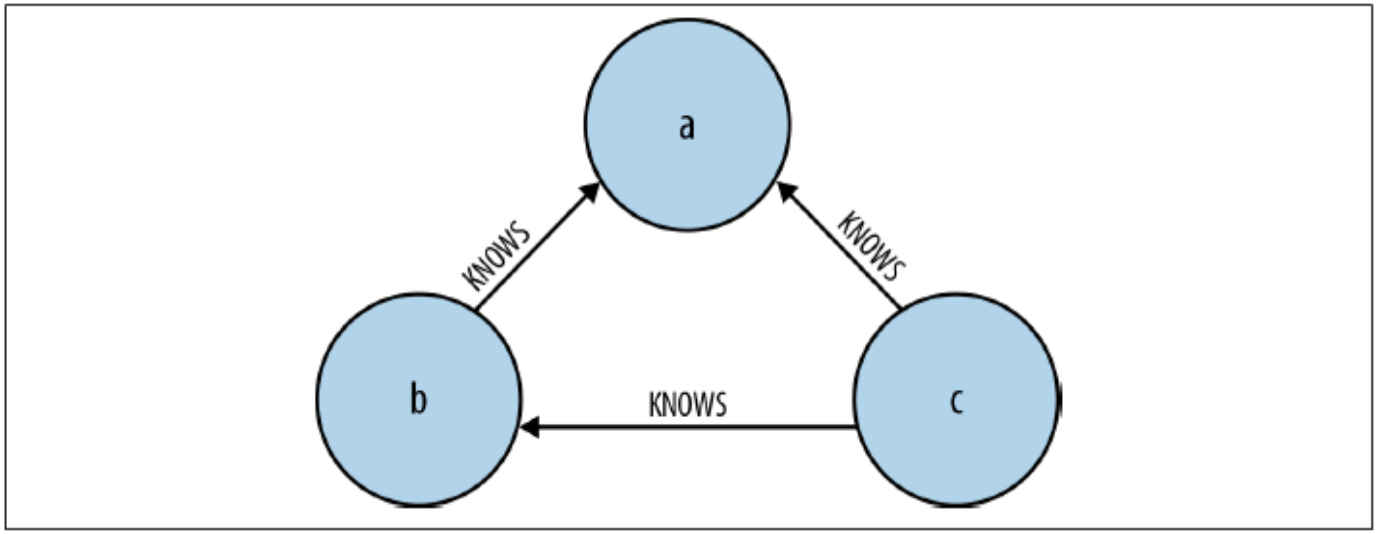


Figure 3-1. A simple graph pattern, expressed using a diagram

This pattern describes three mutual friends. Here's the equivalent ASCII art representation in Cypher:

```
(a)-[:KNOWS]->(b)-[:KNOWS]->(c), (a)-[:KNOWS]->(c)
```

Cypher Example

Like most query languages, Cypher is composed of clauses. The simplest queries consist of a `START` clause followed by a `MATCH` and a `RETURN` clause (we'll describe the other clauses you can use in a Cypher query later in this chapter). Here's an example of a Cypher query that uses these three clauses to find the mutual friends of user named *Michael*:

```
START a=node:user(name='Michael')
MATCH (a)-[:KNOWS]->(b)-[:KNOWS]->(c), (a)-[:KNOWS]->(c)
RETURN b, c
```

Other Cypher Clauses

`WHERE`
Provides criteria for filtering pattern matching results.

`CREATE` *and* `CREATE UNIQUE`
Create nodes and relationships.

`DELETE`
Removes nodes, relationships, and properties.

`SET`
Sets property values.

`FOREACH`
Performs an updating action for each element in a list.

`UNION`
Merges results from two or more queries (introduced in Neo4j 2.0).

`WITH`
Chains subsequent query parts and forward results from one to the next. Similar to piping commands in Unix.

Property Graph Example – Shakespeare

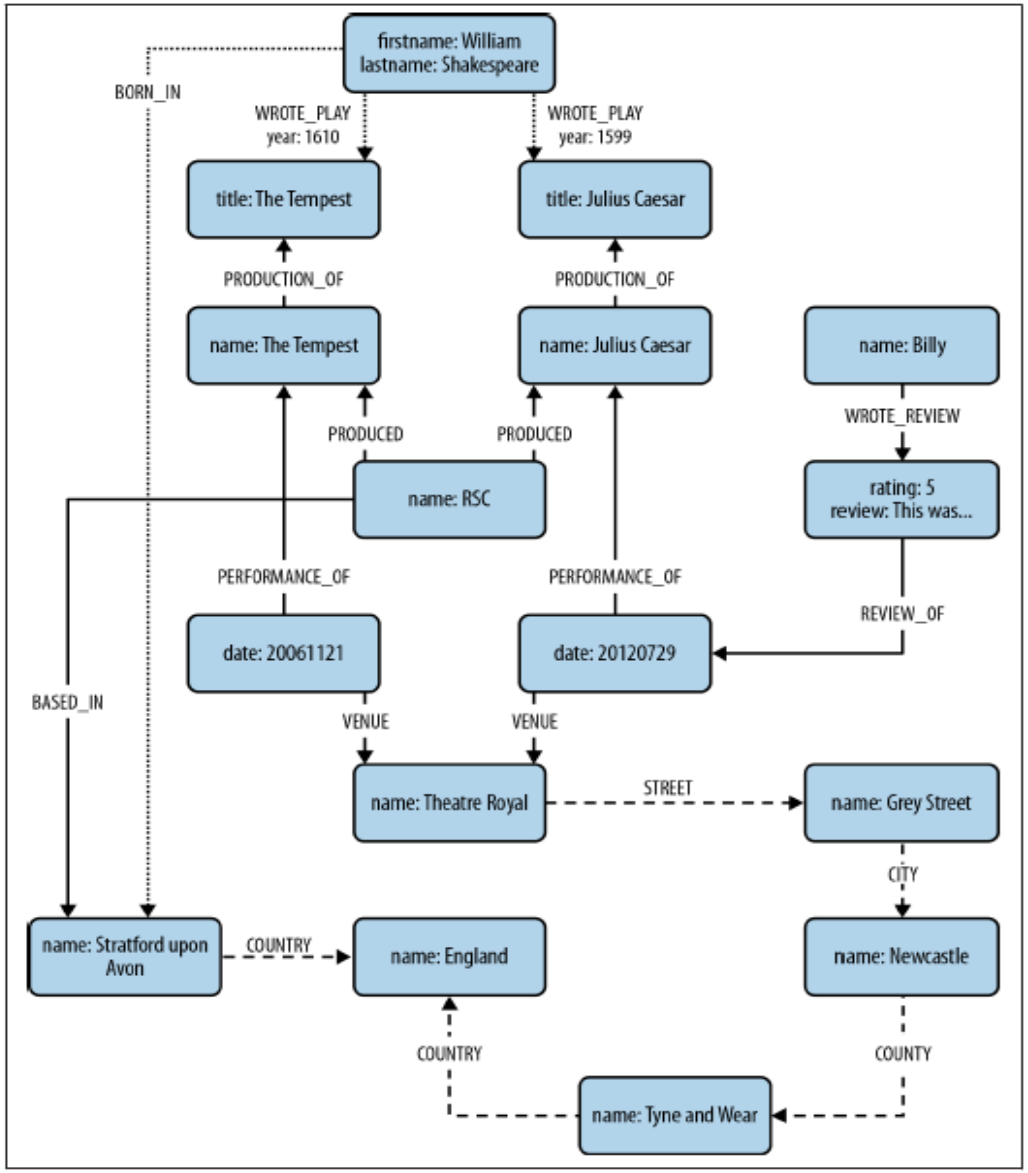


Figure 3-6. Three domains in one graph

Creating the Shakespeare Graph

```
CREATE (shakespeare { firstname: 'William', lastname: 'Shakespeare' }),
       (juliusCaesar { title: 'Julius Caesar' }),
       (shakespeare)-[:WROTE_PLAY { year: 1599 }]->(juliusCaesar),
       (theTempest { title: 'The Tempest' }),
       (shakespeare)-[:WROTE_PLAY { year: 1610}]->(theTempest),
       (rsc { name: 'RSC' }),
       (production1 { name: 'Julius Caesar' }),
       (rsc)-[:PRODUCED]->(production1),
       (production1)-[:PRODUCTION_OF]->(juliusCaesar),
       (performance1 { date: 20120729 }),
       (performance1)-[:PERFORMANCE_OF]->(production1),
       (production2 { name: 'The Tempest' }),
       (rsc)-[:PRODUCED]->(production2),
       (production2)-[:PRODUCTION_OF]->(theTempest),
       (performance2 { date: 20061121 }),
       (performance2)-[:PERFORMANCE_OF]->(production2),
       (performance3 { date: 20120730 }),
       (performance3)-[:PERFORMANCE_OF]->(production1),
       (billy { name: 'Billy' }),
       (review { rating: 5, review: 'This was awesome!' }),
       (billy)-[:WROTE_REVIEW]->(review),
       (review)-[:RATED]->(performance1),
       (theatreRoyal { name: 'Theatre Royal' }),
       (performance1)-[:VENUE]->(theatreRoyal),
       (performance2)-[:VENUE]->(theatreRoyal),
       (performance3)-[:VENUE]->(theatreRoyal),
       (greyStreet { name: 'Grey Street' }),
       (theatreRoyal)-[:STREET]->(greyStreet),
       (newcastle { name: 'Newcastle' }),
       (greyStreet)-[:CITY]->(newcastle),
       (tyneAndWear { name: 'Tyne and Wear' }),
       (newcastle)-[:COUNTY]->(tyneAndWear),
       (england { name: 'England' }),
       (tyneAndWear)-[:COUNTRY]->(england),
       (stratford { name: 'Stratford upon Avon' }),
       (stratford)-[:COUNTRY]->(england),
       (rsc)-[:BASED_IN]->(stratford),
       (shakespeare)-[:BORN_IN]->stratford
```

Query on the Shakespeare Graph

```
START theater=node:venue(name='Theatre Royal'),
      newcastle=node:city(name='Newcastle'),
      bard=node:author(lastname='Shakespeare')
MATCH (newcastle)<-[:STREET|CITY*1..2]-(theater)
      <-[:VENUE]-()-[:PERFORMANCE_OF]->()-[:PRODUCTION_OF]->
      (play)<-[w:WROTE_PLAY]-(bard)
WHERE w.year > 1608
RETURN DISTINCT play.title AS play
```

Adding this `WHERE` clause means that for each successful match, the Cypher execution engine checks that the `WROTE_PLAY` relationship between the Shakespeare node and the matched play has a `year` property with a value greater than 1608. Matches with a `WROTE_PLAY` relationship whose `year` value is greater than 1608 will pass the test; these plays will then be included in the results. Matches that fail the test will not be included in the results. By adding this clause, we ensure that only plays from Shakespeare's late period are returned:

```
+---------------+
| play          |
+---------------+
| "The Tempest" |
+---------------+
1 row
```

Another Query on the Shakespeare Graph

```
START theater=node:venue(name='Theatre Royal'),
      newcastle=node:city(name='Newcastle'),
      bard=node:author(lastname='Shakespeare')
MATCH (newcastle)<-[:STREET|CITY*1..2]-(theater)
      <-[:VENUE]-()-[p:PERFORMANCE_OF]->()-[:PRODUCTION_OF]->
      (play)<-[:WROTE_PLAY]-(bard)
RETURN   play.title AS play, count(p) AS performance_count
ORDER BY performance_count DESC
```

The RETURN clause here counts the number of PERFORMANCE_OF relationships using the identifier p (which is bound to the PERFORMANCE_OF relationships in the MATCH clause) and aliases the result as performance_count. It then orders the results based on performance_count, with the most frequently performed play listed first:

```
+-------------------------------------+
| play           | performance_count |
+-------------------------------------+
| "Julius Caesar" | 2                |
| "The Tempest"   | 1                |
+-------------------------------------+
2 rows
```

Building Application Example – Collaborative Filtering

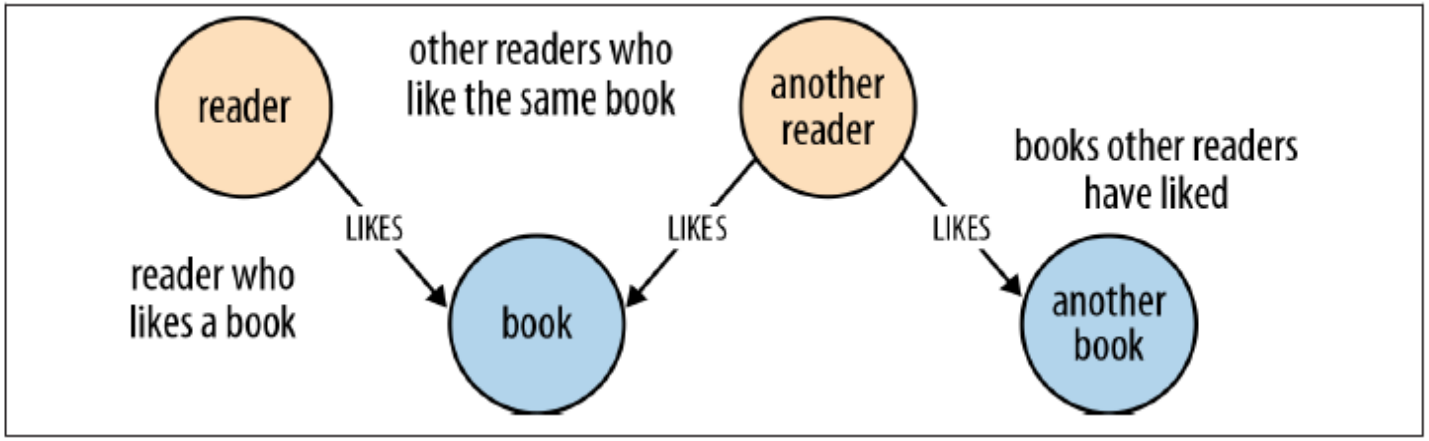


Figure 4-1. Data model for the book reviews user story

Because this data model directly encodes the question presented by the user story, it lends itself to being queried in a way that similarly reflects the structure of the question we want to ask of the data:

```
START reader=node:users(name={readerName})
      book=node:books(isbn={bookISBN})
MATCH reader-[:LIKES]->book<-[:LIKES]-other_readers-[:LIKES]->books
RETURN books.title
```

Chaining on the Query

```
START bard=node:author(lastname='Shakespeare')
MATCH (bard)-[w:WROTE_PLAY]->(play)
WITH play
ORDER BY w.year DESC
RETURN collect(play.title) AS plays
```

Executing this query against our sample graph produces the following result:

```
+---------------------------------+
| plays                           |
+---------------------------------+
| ["The Tempest","Julius Caesar"] |
+---------------------------------+
1 row
```

Example – Email Interaction Graph

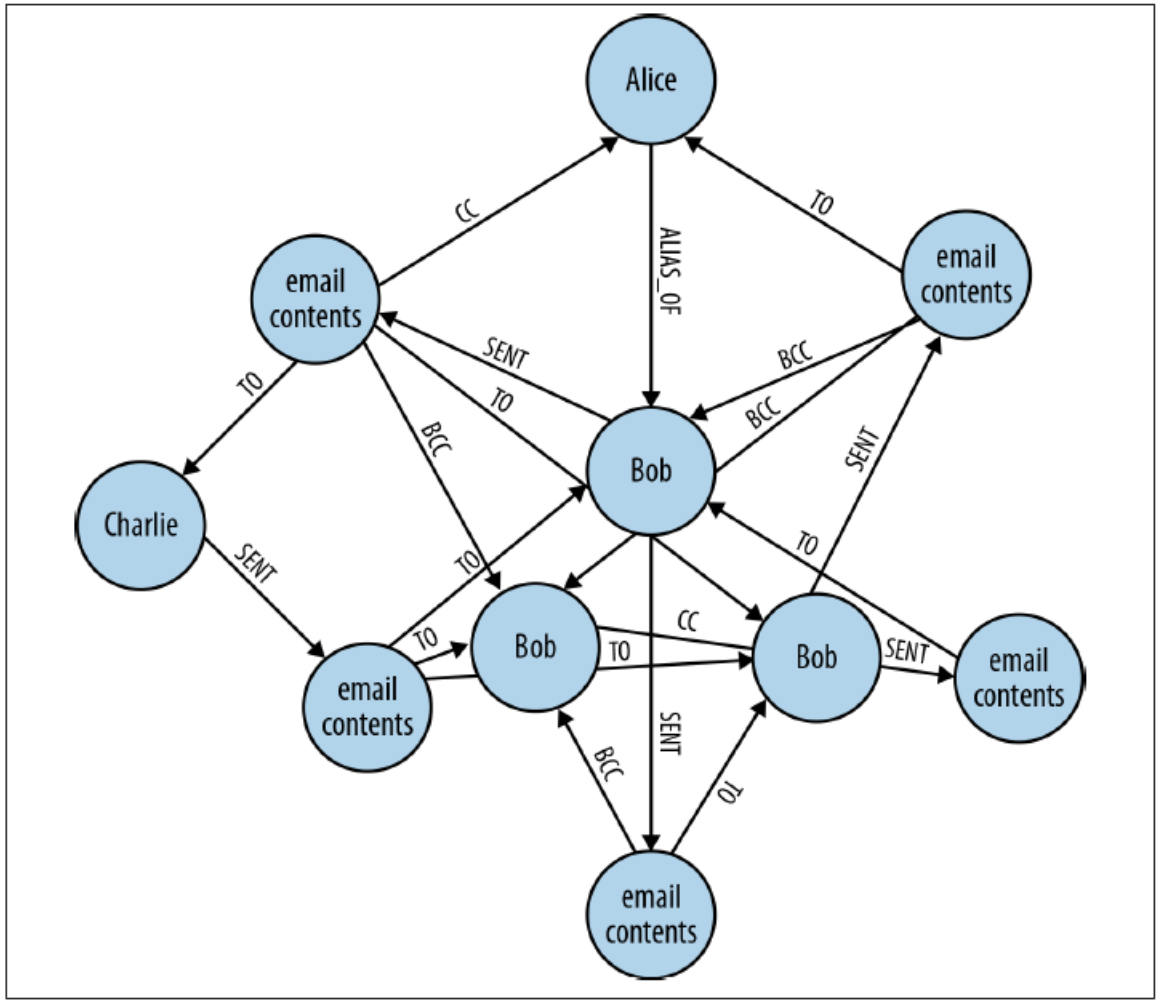


Figure 3-10. A graph of email interactions

```
START bob=node:user(username='Bob')
MATCH (bob)-[:SENT]->(email)-[:CC]->(alias),
      (alias)-[:ALIAS_OF]->(bob)
RETURN email
```

What's this query for?

How to make graph database fast?

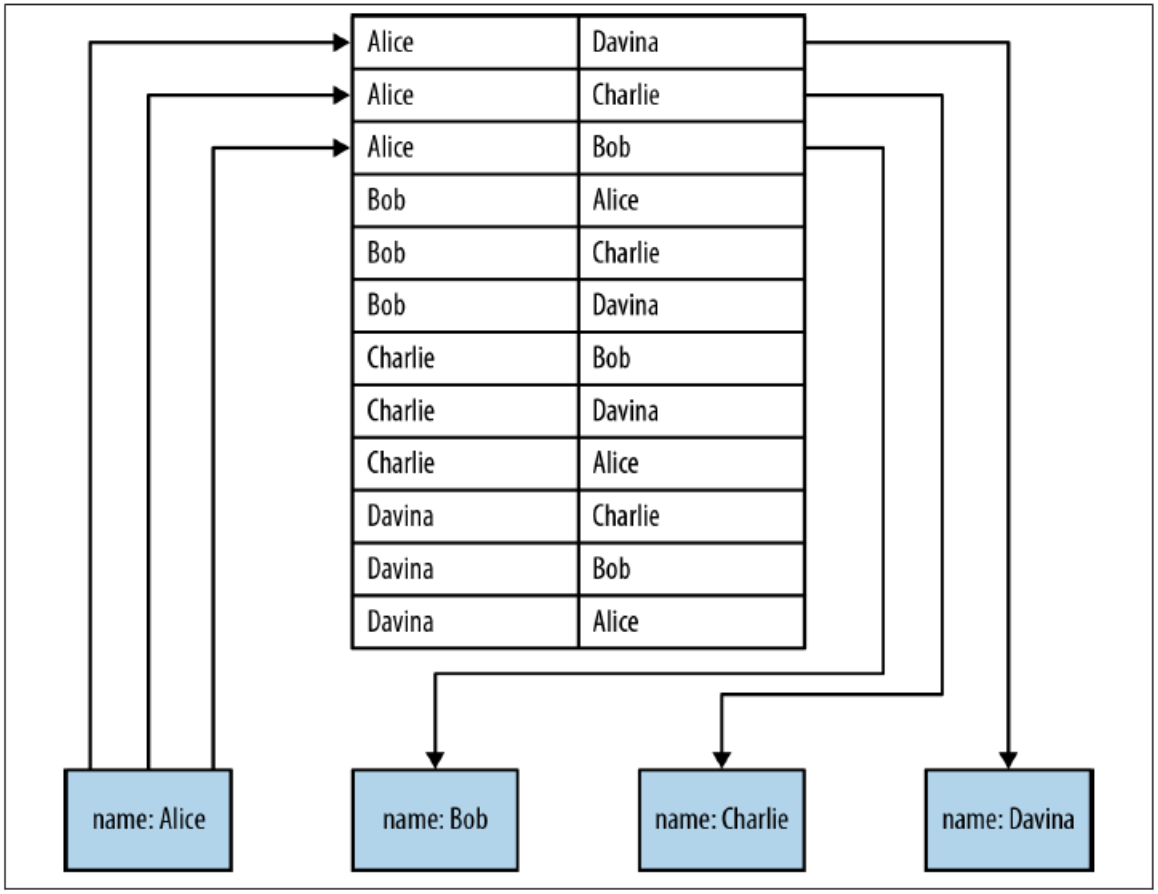
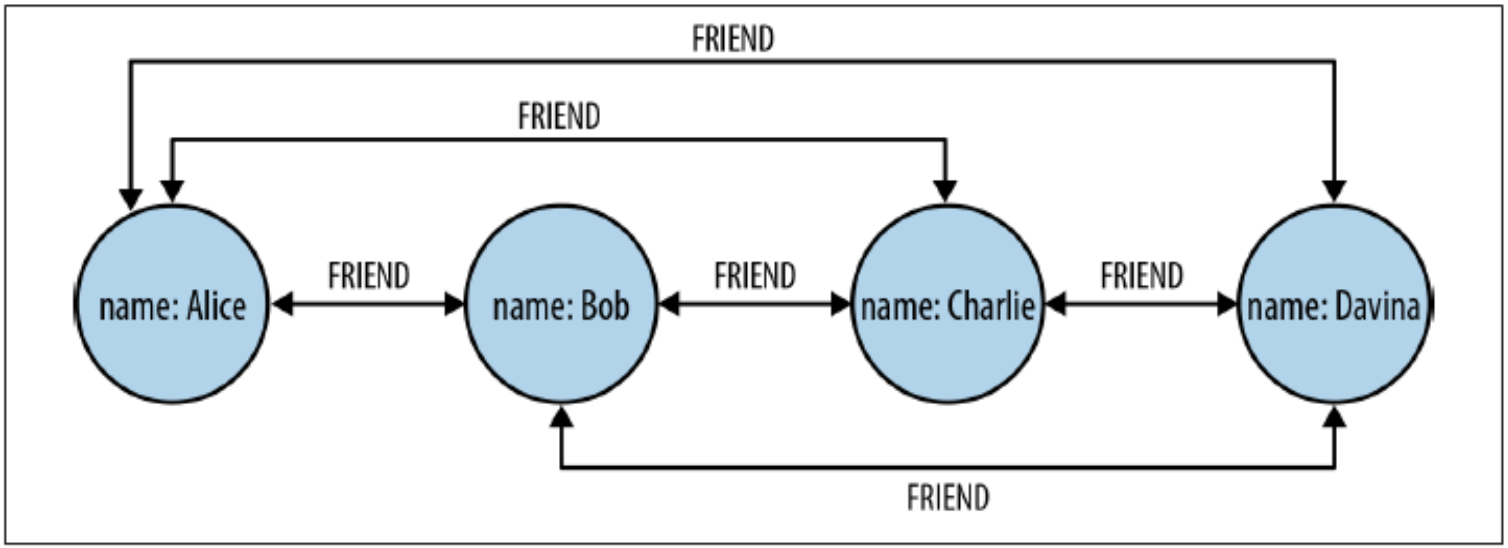


Figure 6-1. Nonnative graph processing engines use indexing to traverse between nodes

Use Relationships, not indexes, for fast traversal



Storage Structure Example

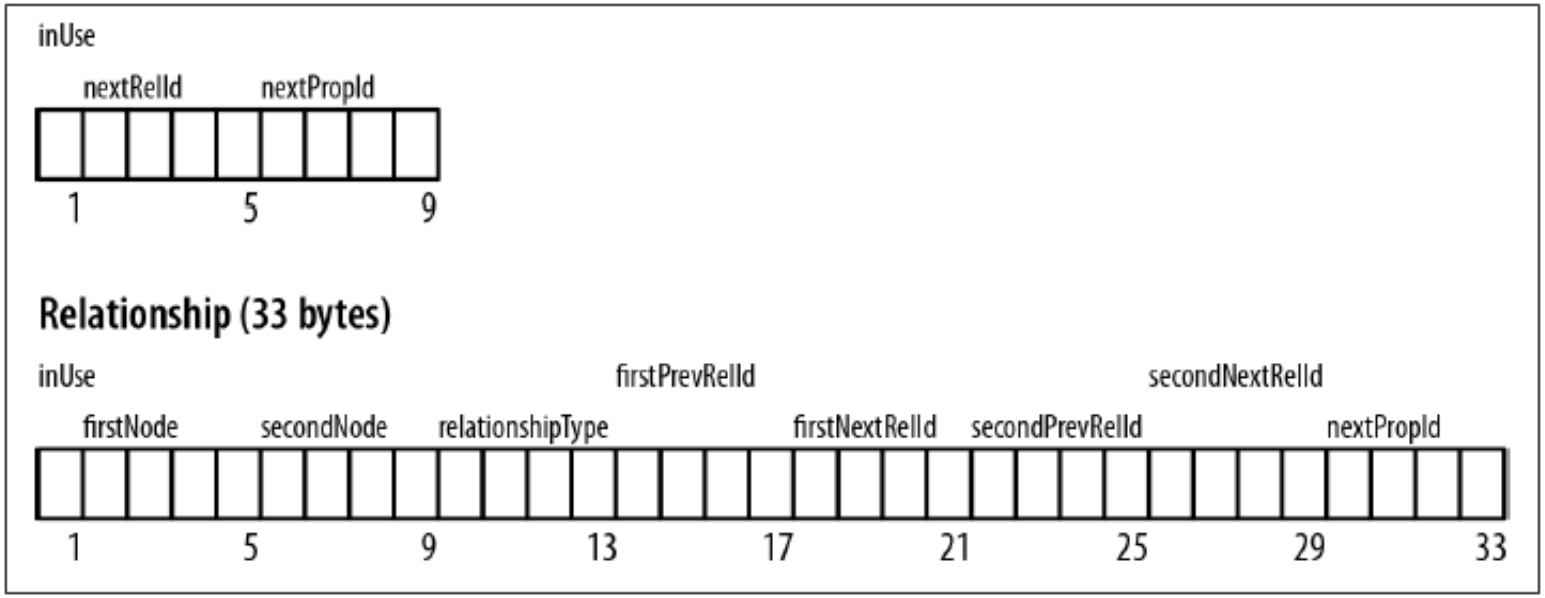


Figure 6-4. Neo4j node and relationship store file record structure

An experiment

Dataset: 12.2 million edges, 2.2 million vertices

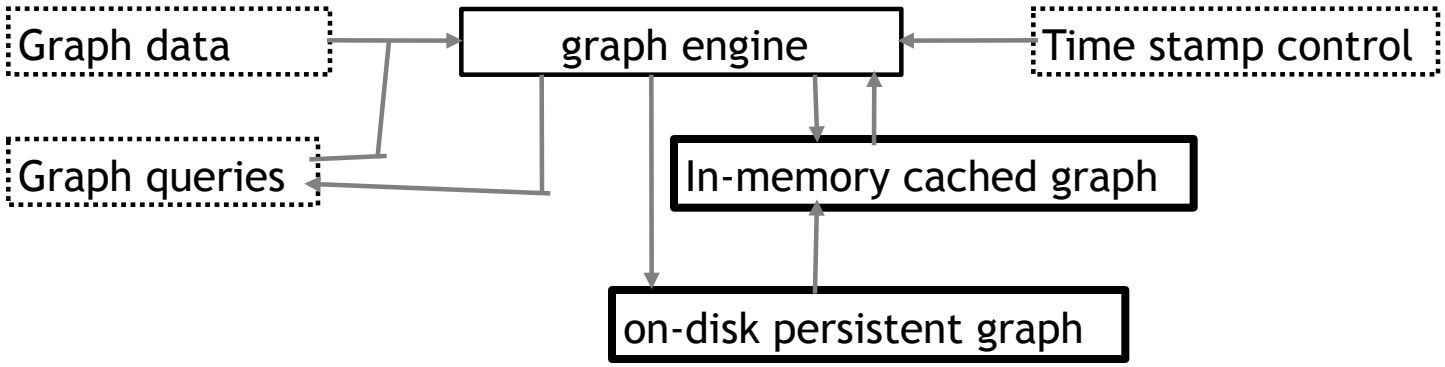
Goal: Find paths in a property graph. One of the vertex property is call TYPE. In this scenario, the user provides either a particular vertex, or a set of particular vertices of the same TYPE (say, "DRUG"). In addition, the user also provides another TYPE (say, "TARGET"). Then, we need find all the paths from the starting vertex to a vertex of TYPE “TARGET”. Therefore, we need to 1) find the paths using graph traversal; 2) keep trace of the paths, so that we can list them after the traversal. Even for the shortest paths, it can be multiple between two nodes, such as: drug->assay->target , drug->MOA->target

		Avg time (100 tests)	
	First test (cold-start)	Requested depth 5 traversal	Requested full depth traversal
NativeStore C++	39 sec	3.0 sec	4.2 sec
NativeStore JNI	57 sec	4.0 sec	6.2 sec
Neo4j (Blueprints 2.4)	105 sec	5.9 sec	8.3 sec
Titan (Berkeley DB)	3861 sec	641 sec	794 sec
Titan (HBase)	3046 sec	1597 sec	2682 sec

First full test - full depth 23. All data pulled from disk. Nothing initially cached.
Modes - All tests in default modes of each graph implementation. Titan can only be run in transactional mode. Other implementations do not default to transactional mode.

Native Store Overview

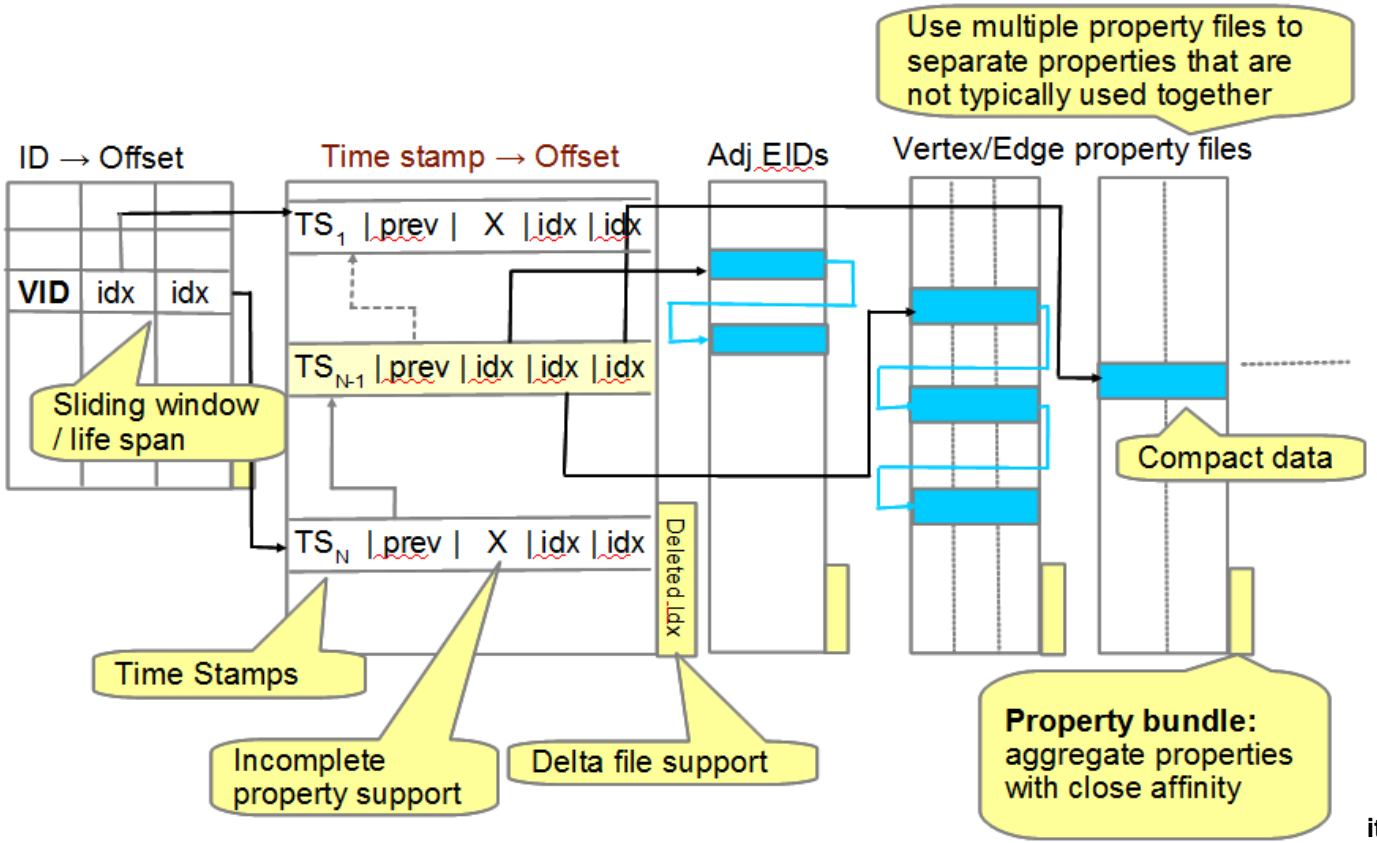
- **Native store represents graphs in-memory and on-disk**
  - Organizing graph data for representing a graph that stores both graph structure and vertex properties and edge properties
  - Caching graph data in memory in either batch-mode or on-demand from the on-disk streaming graph data
  - Accepting graph updates and modifying graph structure and/or property data accordingly and incorporating time stamps
    - Add edge, remove vertex, update property, etc.
  - Persisting graph updates along with the time stamps from in-memory graph to on-disk graph
  - Performing graph queries by loading graph structure and/or property data
    - Find neighbors of a vertex, retrieve property of an edge, traverse a graph, etc.



On-Disk Graph Organization

- **Native store organizes graph data for representing a graph with both structure and the vertex properties and edge properties using multiple files in Linux file system**
  - Creating a list called ID → Offset where each element translates a vertex (edge) ID into two offsets, pointing to the earliest and latest data of the vertex/edge, respectively
  - Creating a list called Time_stamp → Offset where each element has a time stamp, an offset to the previous time stamp of the vertex/edge, and a set of indices to the adjacent edge list and properties
  - Create a list of chained block list to store adjacent list and properties

On-disk persistent graph:

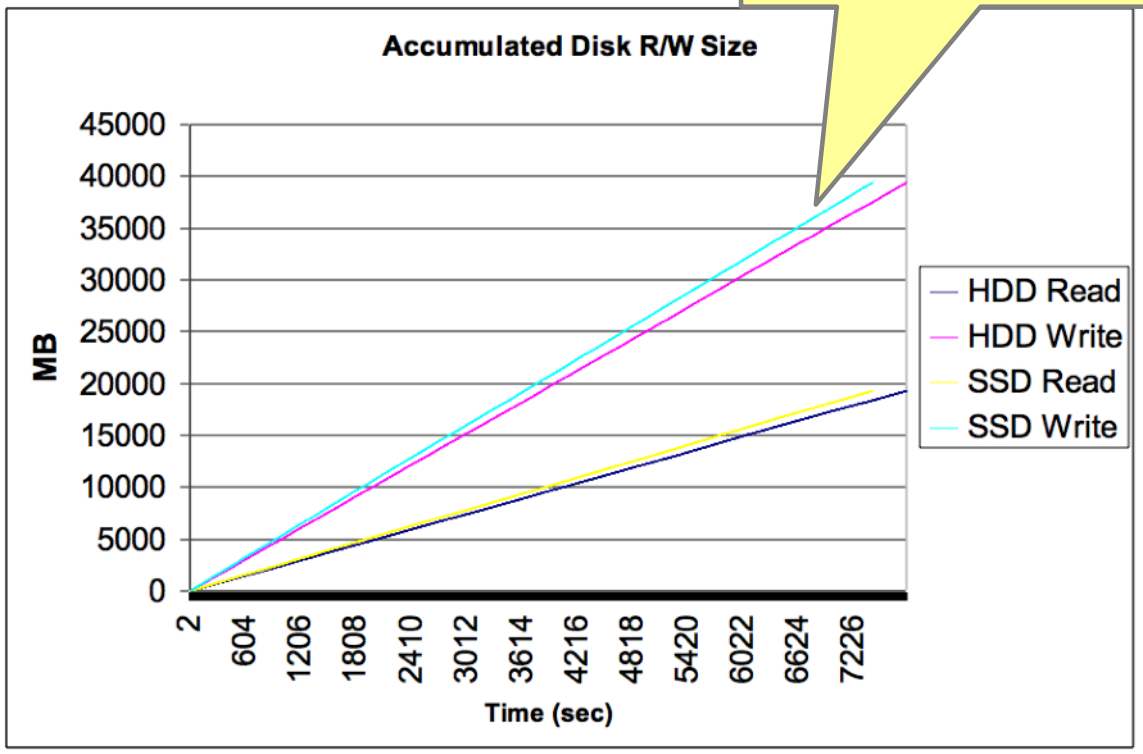


Impact from Storage Hardware

- Convert csv file (adams.csv 20G) to datastore
  - Similar performance: 7432 sec versus 7806 sec
  - CPU intensive
    - Average CPU util.: 97.4 versus 97.2
  - I/O pattern
    - Maximum read rate: 5.0 vs. 5.3
    - Maximum write rate: 97.7 vs. 85.3

Ratio HDD/SDD	TYPE1	TYPE2	TYPE3	TYPE4
	13.79	6.36	19.93	2.44

SSD offers consistently higher performance for both read and write

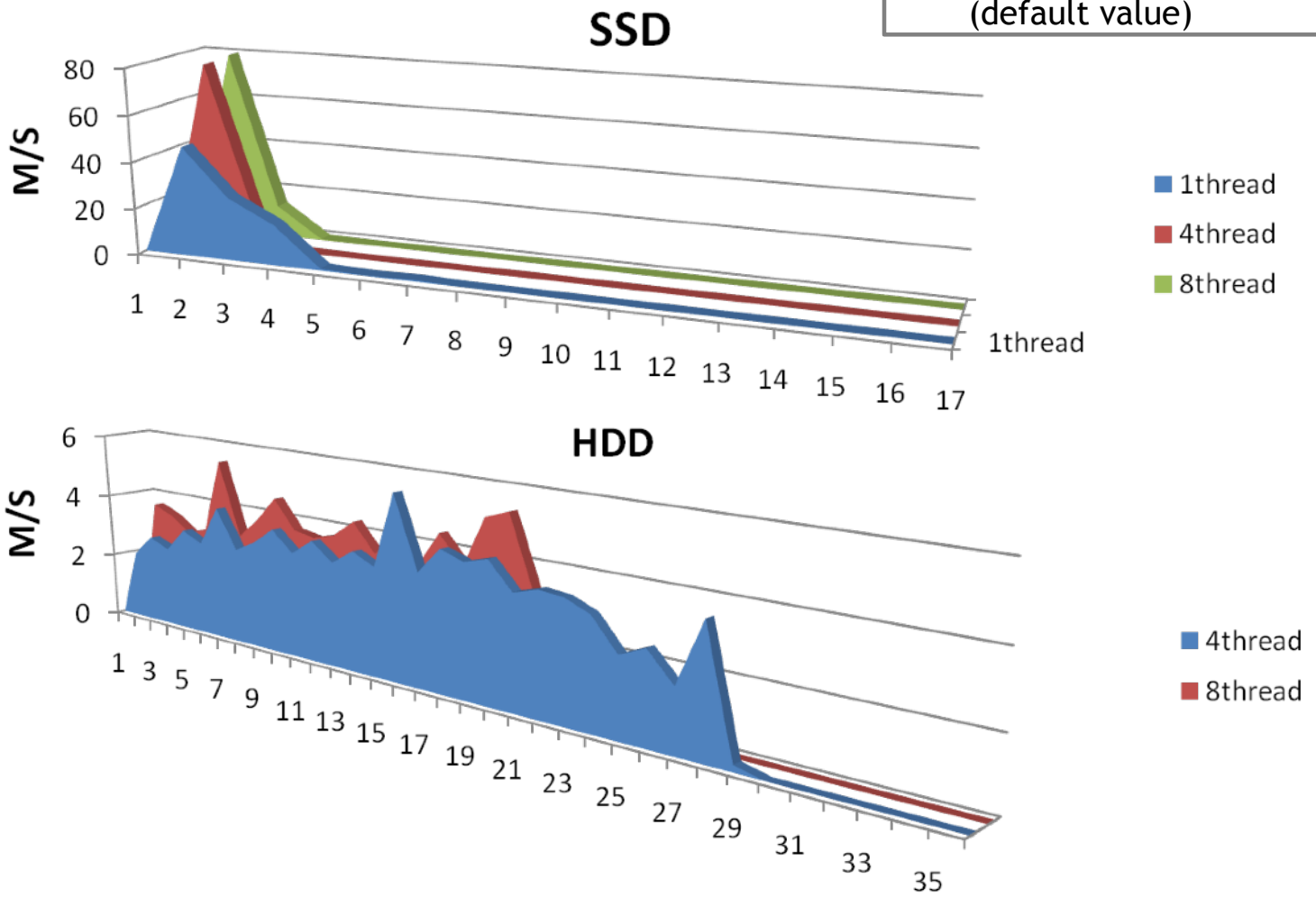


Queries

- Type 1: find the most recent URL and PCID of a user
- Type 2: find all the URLs and PCIDs
- Type 3: find all the most recent properties
- Type 4: find all the historic properti

Impact from Storage Hardware — 2

- Dataset: Knowledge Repository
  - 138614 Nodes, 1617898 Edges
- OS buffer is flushed before test
- Processing 320 queries in parallel
- In memory graph cache size: 4GB (default value)



Big Data Analytics Use Case:

Company Network and Value Analysis

Are we able to find out answers for these questions?

Finding answers of,

- Is it possible to predict a company's profit or revenue changes based on dynamic company networks?
- How can we infer evolutionary company networks from public news?
- How accurate can network characteristics help predicting profit/ revenue changes?
- What are the most important – positive or negative – feature measures of networks to predict profit/revenue?

Social Network Analysis

- **An Analytics research field since 1920s.**
- **Social Networks (SNs)**

  Nodes : Actors (persons, companies, organizations etc.)

  Ties : Relations (friendship, collaboration, alliance etc.)

- **Network properties**
  - Degree, distance, centrality, and various kinds of positional and equivalence
- **Application of SNs**
  - Social psychology: analyzing social phenomena
  - Economics: consulting business strategy
  - Information science: Information sharing and recommendation, trust calculation, ontology construction

Example of Company Value Analysis

Accounting-based financial statement information

Fundamental values:

ROE(Return On Equity), ROA(Return On Asset), PER(Price Earnings Ratio), PBR(Price Book-value Ratio), Employee Number, Dividend Yield, Capital Ratio, Capital, etc.

E.g. *"Fundamental Analysis, Future Earnings and Stock Prices",* [Abarbane&Bushee97]

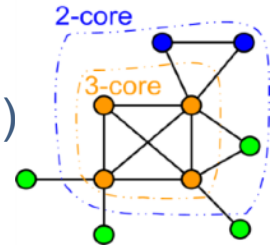
Applying historical trends to predict stock market index (Heng Seng Index in Hong Kong)

E.g. *"Support Vector Machine Regression for Volatile Stock Market Prediction"* [H.Yang02]

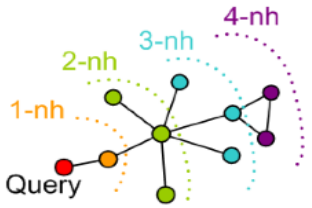
$$\hat{I}_t = f(I_{t-w} + \ldots + I_{t-1})$$

Example of Analytical Tools

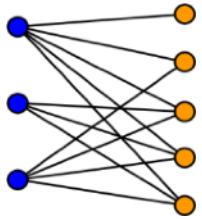
- ***Network topological analysis*** tools
  - •Centralities (degree, closeness, betweenness)
  - •PageRank
  - •Communities (connected component, K-core, triangle count, clustering coefficient)
  - •Neighborhood (egonet, K-neighborhood)
- ***Graph matching and search*** tools
  - •Graph search/filter by label, vertex/edge properties (including geo locations)
  - •Graph matching
  - •Collaborative filtering
- ***Graph path and flow*** tools
  - •Shortest paths
  - •Top K-shortest paths
- ***Probabilistic graphical model*** tools
  - •Bayesian network inference
  - •Deep learning

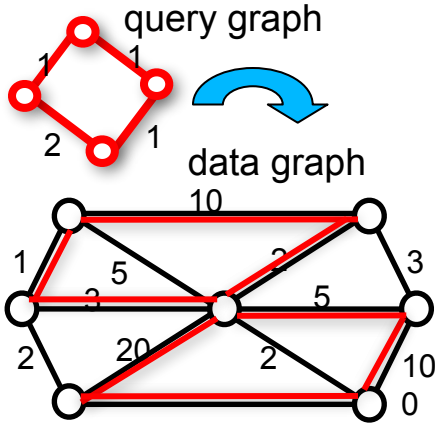


K-core

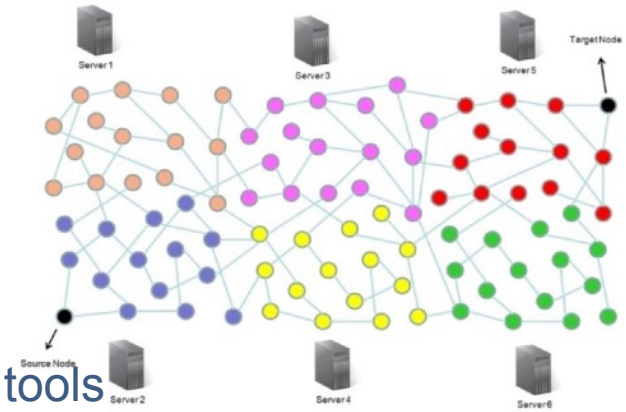


K-neighborhood


Collaborative filtering
Bipartite weighted graph matching



Graph matching



Top k-shortest paths

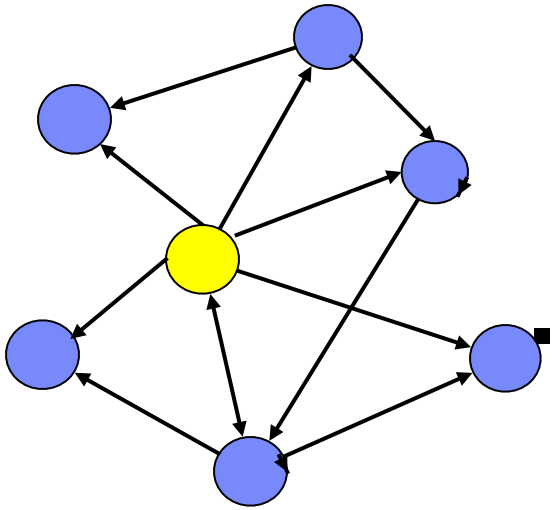

Bayesian network inference

Are Social Networks of Companies related to Companies' Value?

Outline

- Background and Study goal
- Infer Company Networks from Public News
- Network Feature Generation & Selection
- Predict Company Value
- Conclusion and Future work

Company Relationship Detection



- **Specific Relation**
  Cooperation, competition, acquisition, incorporation, supply chain, stock share…
  "*Extracting Inter-business Relationship from World Wide Web*" [Jin08]
  "*Temporal Company Relation Extraction*" [Changjian09]
  - Focus on details, deeper NLP
  - Rare events, sparse, ad-hoc
- **Generic Relation**
  - Who give me more impact [in a period]? (maybe positive or negative)
  - Comprehensive, dynamic relations (like Google rank)
  - Shallow NLP, easy to get weighted and directed networks, much more events.

  ➔ THIS WORK!

Generic Relation Extraction

Article (document)

I.B.M. Will Buy a Maker of Data Analysis Software

By STEVE LOHR
Published: July 28, 2009

I.B.M. took a big step to expand its fast-growing stable of data analysis offerings by agreeing on Tuesday to pay $1.2 billion to buy SPSS Inc., a maker of software used in statistical analysis and predictive modeling.

Sentence

software. In the last couple of years, I.B.M., Oracle, SAP and Microsoft have collectively spent more than $15 billion buying makers of such software.

- Basic Idea:
  - For each company x, we extract companies who
    - Frequently co-appear with x in x's important financial articles
    - Frequently mentioned together with x in important sentences

    In a period of time (e.g. one year)

Example (from NYT 2009 articles about I.B.M)

About 300 articles mentioned I.B.M.
International Business Machines (84 articles), *I.B.M.* (277 articles)

- **I.B.M. -- Microsoft** (55 articles, 264 sentences, weight=85.85455)
  http://www.nytimes.com/2009/03/06/business/06layoffs.html
  Two days after I.B.M.'s report, Microsoft said that its quarterly profits were disappointing.
  http://www.nytimes.com/2009/05/07/technology/07iht-telecoms.html
  ... the world's largest software makers, including Microsoft, SAP and I.B.M., which...
  http://www.nytimes.com/2009/01/31/business/31nocera.html
  Caterpillar, Kodak, Home Depot, I.B.M., even mighty Microsoft they are all cutting jobs.
  http://www.nytimes.com/2009/03/23/technology/companies/23mainframe.html
  More recently, Sun Microsystems, Hewlett-Packard and Microsoft have made mostly unsuccessful attempts to have made mostly unsuccessful attempts to pull mainframe customers away from I.B.M. by ...
- **I.B.M. -- SPSS** (1 articles, 9 sentences, weight=13.675)
  http://www.nytimes.com/2009/07/29/technology/companies/29ibm.html
  I.B.M.to Buy SPSS, a Maker of Business Software
  I.B.M.'s $50-a-share cash offer is a premium of more than 40 percent over SPSS's closing stock price on Monday.
  I.B.M.took a big step to expand its fast-growing stable of data analysis offerings by agreeing on Tuesday to pay $1.2 billion to buy SPSS Inc.,...
- **I.B.M. -- Nike**. (4 articles, 9 sentences, weight=8.212)
  http://www.nytimes.com/2009/01/22/business/22pepsi.html
  … companies that have taken steps to reduce carbon emissions includes I.B.M., Nike, Coca-Cola and BP, the oil giant.
  http://www.nytimes.com/2009/11/01/business/01proto.html
  Others are water-based shoe adhesives from Nike and a packing insert from I.B.M.

Generic Relation Extraction

For target company "x", first download NYT articles for a year, and select candidate companies Y={y1,y2,…} appeared on the articles, then calculate each candidate company's relation strength with "x".

Document Weight

Title: x……
…x.. …y1. .. ..
…….y3……
……y4..

- |Y|: How many companies on the article?
- Σtf_{xY}: How many times companies appeared?
- tf_xHow many times "*x*" company appear?
- *w*: Does names appeared on the title?

Sentence Weight

S1: x ….. y1 …
S2: x… y3…..y5
S3: y3..x…, y4…y1…

- $|Y_1|$: the number of company names appeared in the same sentence.
- $|Y_2|$: the number of company names appeared between "x" and "y"

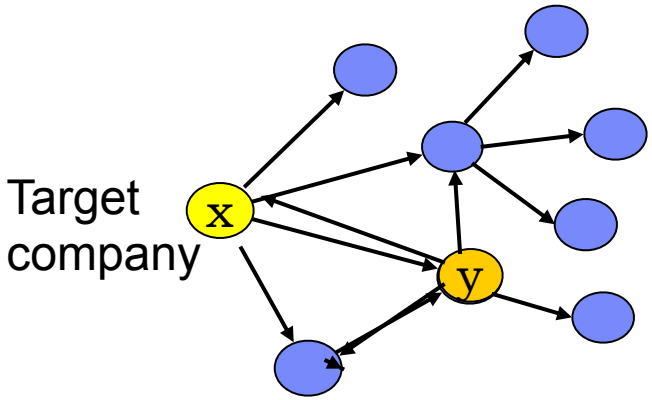
$$w_d = \log(1+\frac{1}{|Y|}) \times \frac{w * tf_x}{\sum_{y_i \in Y} w * tf_{x,y_i}}$$

$$w_s = \log(1+\frac{1}{|Y_1|}+\frac{1}{|Y_2|})$$

$$w = a \cdot \sum df \times w_d + b \cdot \sum sf \times w_s$$

Data and Network

- Data Source:
  - Relationships among companies from public articles
    - New York Times (NYT) articles: 1981 ~ 2009 http://www.nytimes.com/
    - 7594 companies http://topics.nytimes.com/topics/news/business/companies/index.html
  - Company Values: profit, revenue, etc.
    - Fortune 500: 1955-2009 http://money.cnn.com/magazines/fortune/fortune500/2009/full_list/
- Target companies:
  - 308 companies (from NYT & Fortune500)
  - 656,115 articles about target companies:



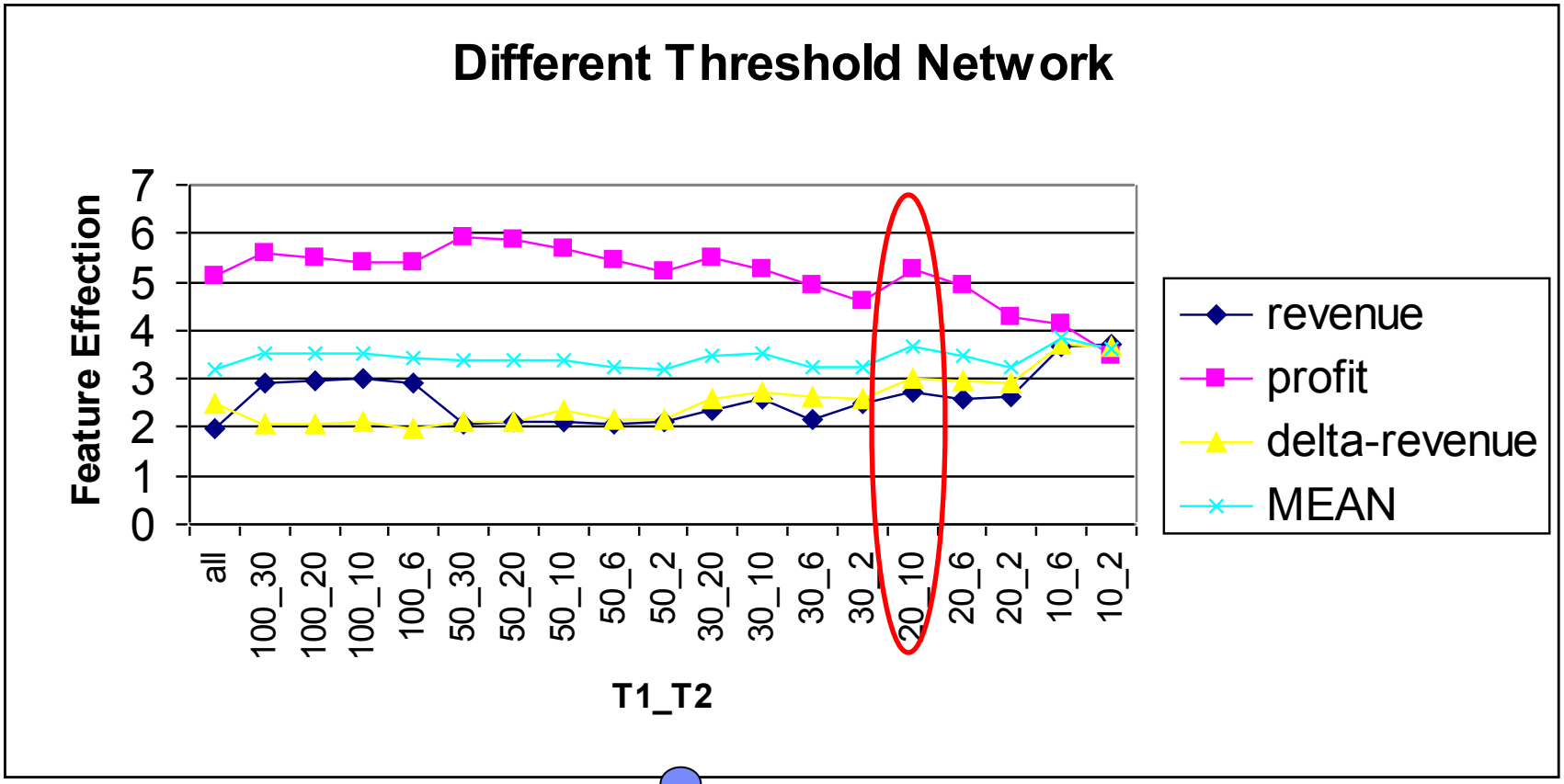
Bootstrap approach

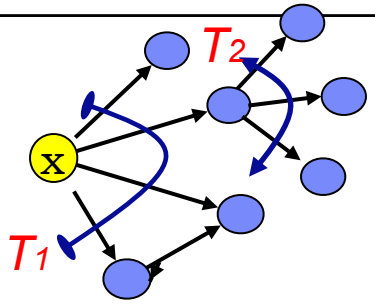
Network size (all)

year	#nodes	#edges	year	#nodes	#edges
1981	463	4030	1996	1202	46265
1982	478	4457	1997	1266	45650
1983	477	4546	1998	1312	51362
1984	484	4606	1999	1379	53653
1985	546	6606	2000	1534	59079
1986	565	6680	2001	1496	55801
1987	**941**	**124326**	2002	1487	54713
1988	1015	108075	2003	1504	54173
1989	1066	132906	2004	1461	51801
1990	1070	177022	2005	1193	43944
1991	1080	107973	2006	1355	51896
1992	1125	53625	2007	1280	44501
1993	1133	136807	2008	1260	43340
1994	1147	130975	2009	1203	37921
1995	1134	52855			

Financial Crisis 1987

Thresholding of Networks

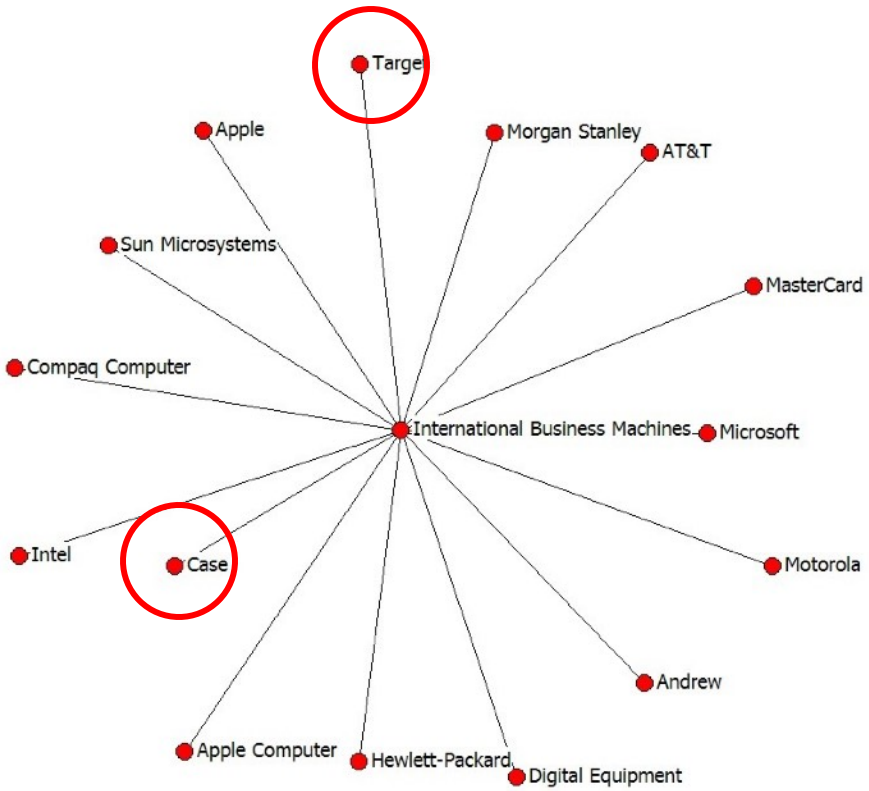




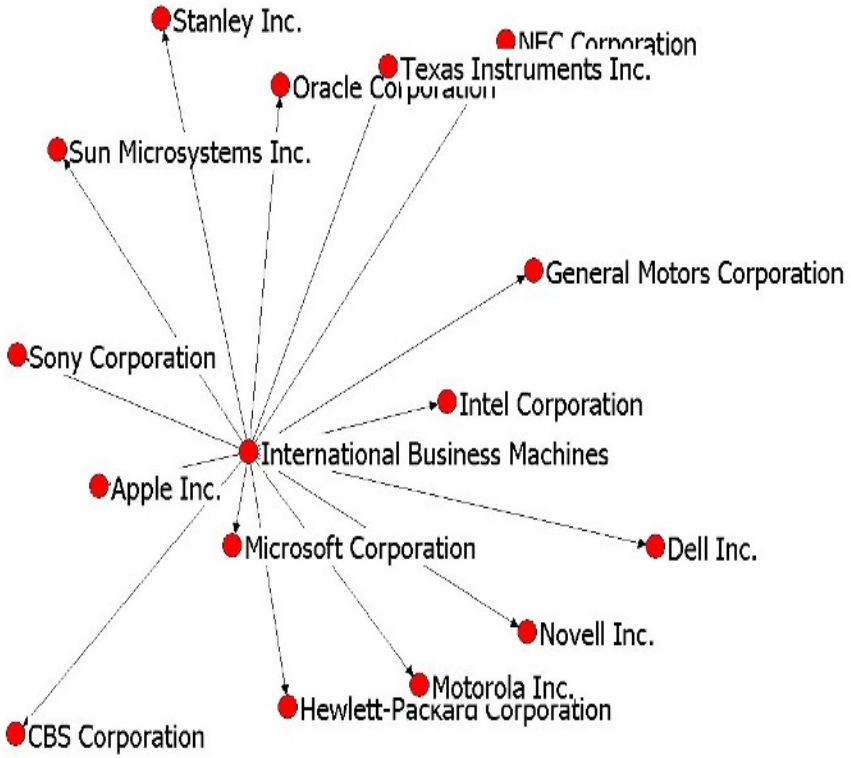
T1=20
T2=10

Comparison of Naïve co-occurrence and the proposed method

- IBM 1995 (doc coocurrence)



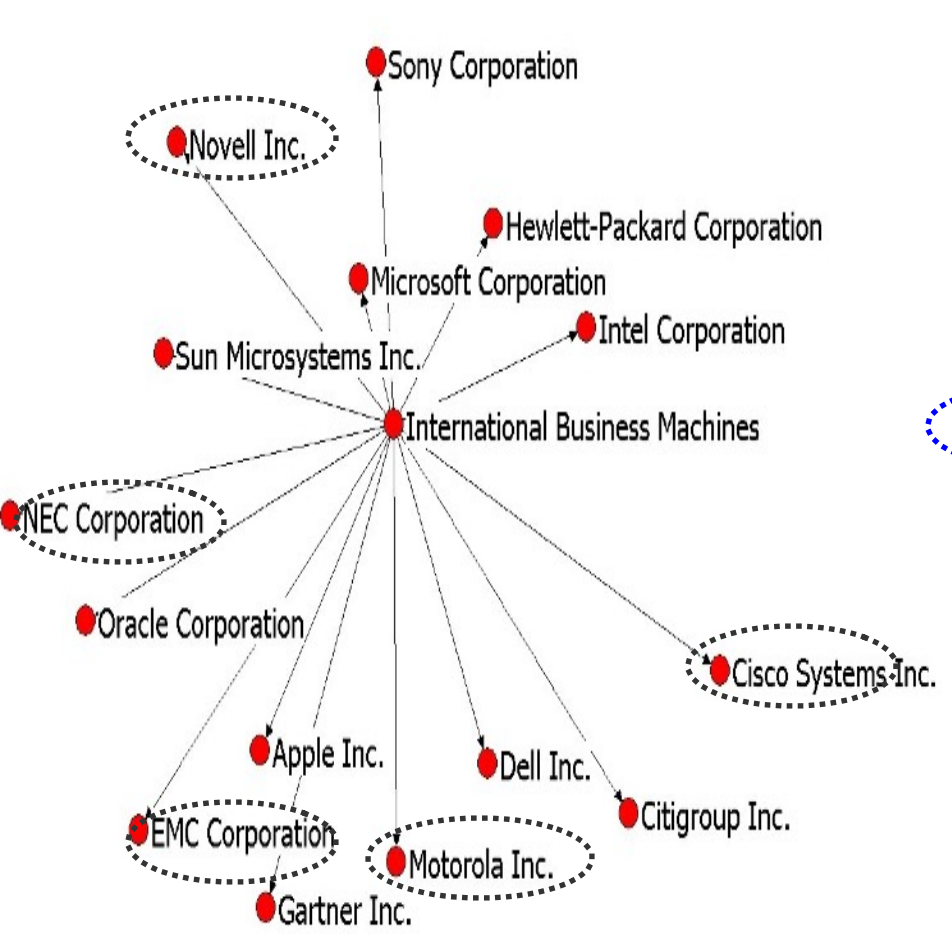
- IBM 1995(new algorithm – doc weights + sentence weights)



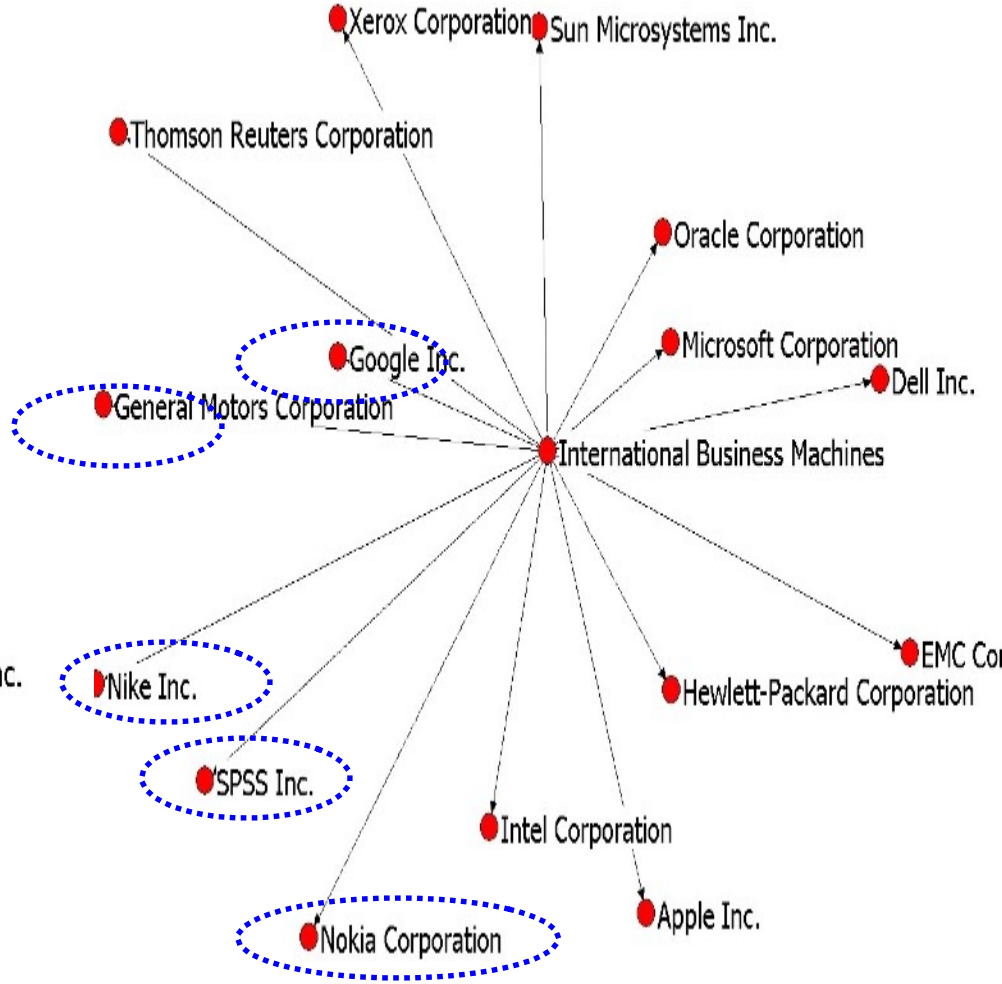
Dominated by big/general companies Better balance between different company sizes

Example of Network Evolution (IBM)

- IBM 2003

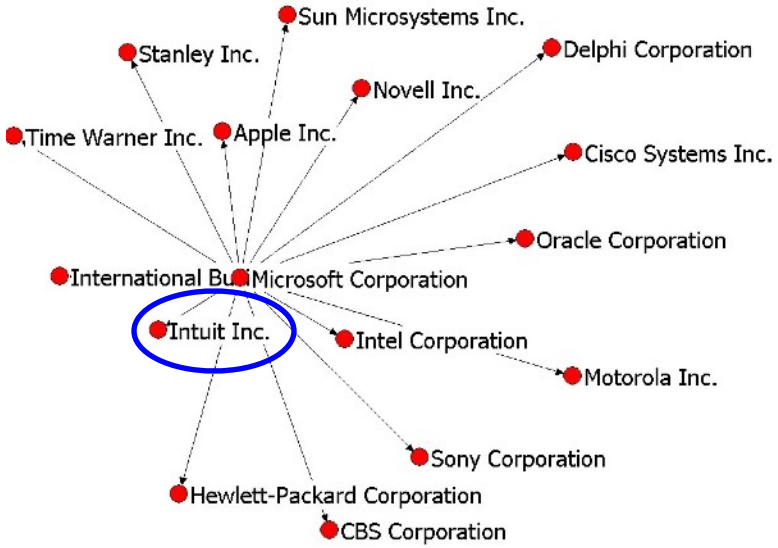


- IBM 2009



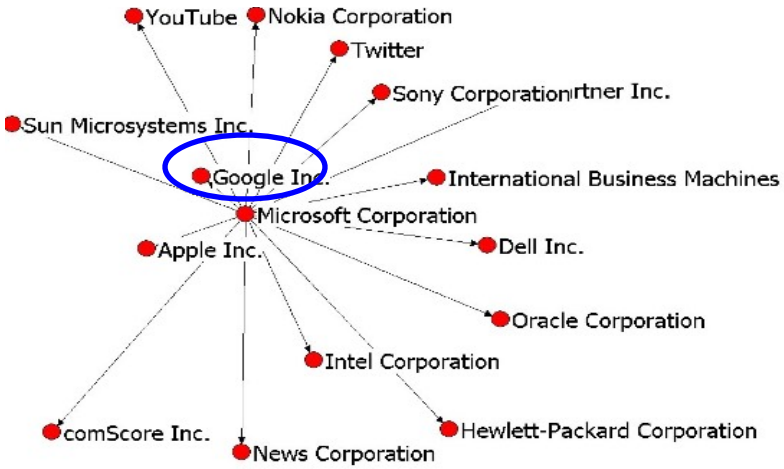
Example of Network Evolution (Microsoft)

- Microsoft 1995



- Microsoft 2003

- Microsoft 2009

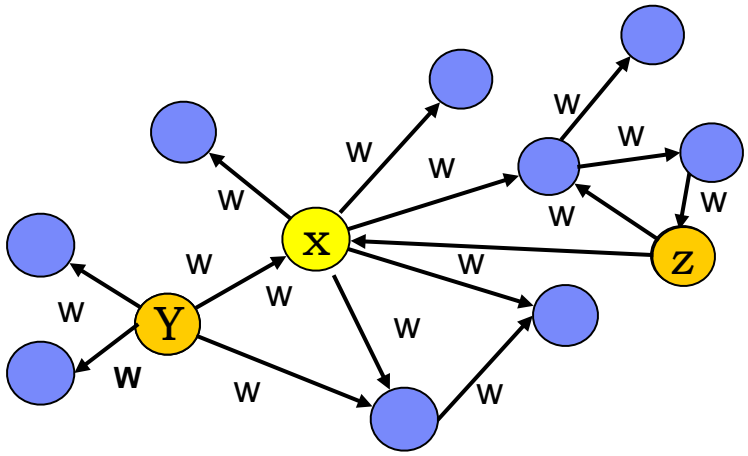


	1995	2003	2009
1	Intuit	I.B.M.	Google
2	I.B.M.	Apple	Apple
3	Intel	Intel	Intel.
4	Apple	Time Warner	Sony
5	Novell	Sony	I.B.M.

Outline

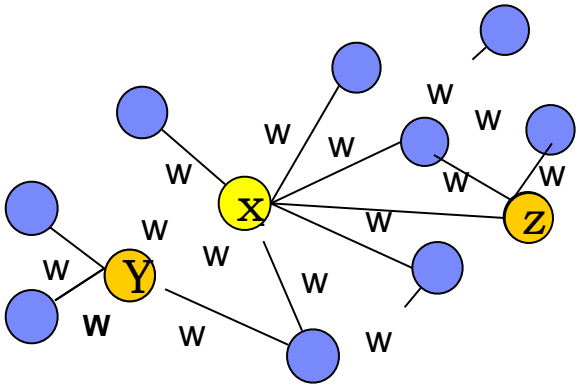
- Background and Study goal
- Infer Company Networks from Public News
- Network Feature Generation & Selection
- Predict Company Value
- Conclusion and Future work

Network Type

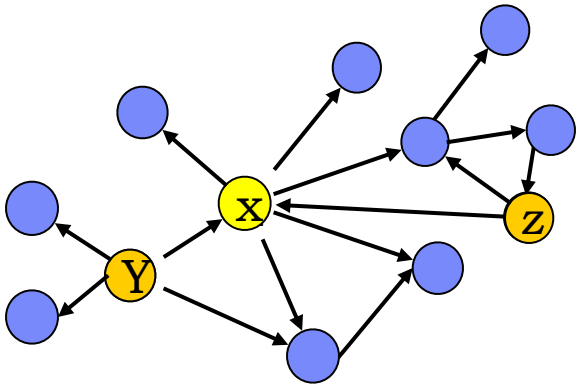


Weighted-Directed Network

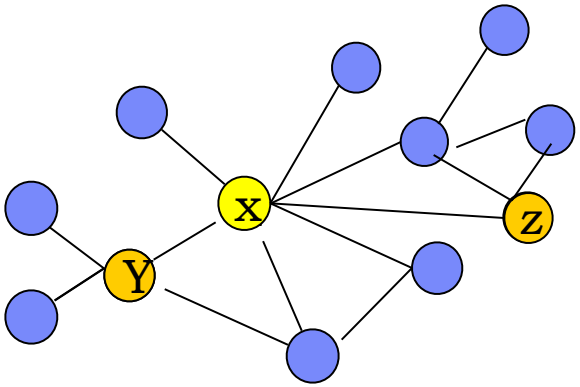
W *i-j* = W *i->j* + W *j->i*



Weighted-Undirected Network



Binary -Directed Network



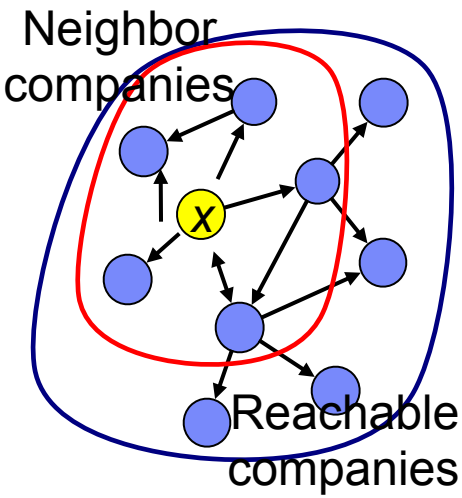
Binary -Undirected Network

Network Feature Generation (1/3)

Who give company *x* impact?
- Neighbor companies on the network
- Reachable companies on the network

Network Features:
- number of neighbors (In-degree, Out-degree)
- number of reachable nodes
- number of connections among neighbors
- number of connections among reachable nodes
- neighbors' degree (In-degree, Out-degree)
- distance of x to all reachable nodes
- distances among neighbors
- ratio of above values between neighbors and reachable nodes …
- etc.

*(Normalize by network size)

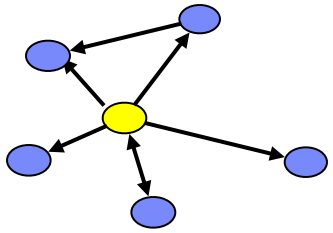


Generate 57 Network features from weighted/binary, directional/undirectional networks

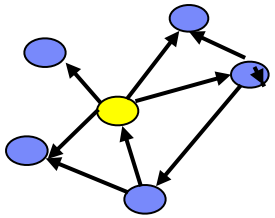
Network Feature Generation (2/3)

Temporal Network Features:

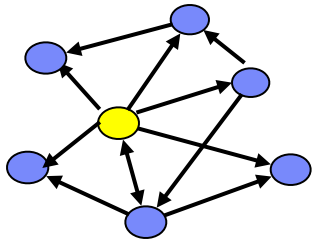
- number of neighbors (In-degree, Out-degree) *last year (or w years ago)*
- number of connections among neighbors *last year (or w years ago)*
- number of connections among reachable nodes *last year (or w years ago)*
- number of neighbors degree *last year (or w years ago)*
- distance of x to all reachable nodes *last year (or w years ago)*
- … etc.



After 5 years



After 10 years



57×Window temporal network features

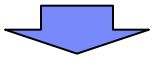
Similar to,

- What's last year's (or w years ago) revenue?
- What's last year's (or w years ago) profit?

Network Feature Generation (3/3)

Delta Change of Network Features:

- *Delta change of* the number of neighbors (In-degree, Out-degree) *from last year (or d years ago)*
- *Delta change of* the number of connections among neighbors *from last year (or d years ago)*
- *Delta change of* the number of connections among reachable nodes *from last year (or d years ago)*
- *Delta change of* the number of neighbors degree *from last year (or d years ago)*
- *Delta change of* the distance of x to all reachable nodes *from last year (or d years ago)*
- … etc.



57×Delta Network features

Network Features

- Network Features for each company
  1. Current Network features: 57
  2. Temporal Network features:57×Window
  3. Delta change of Network features: 57×Delta

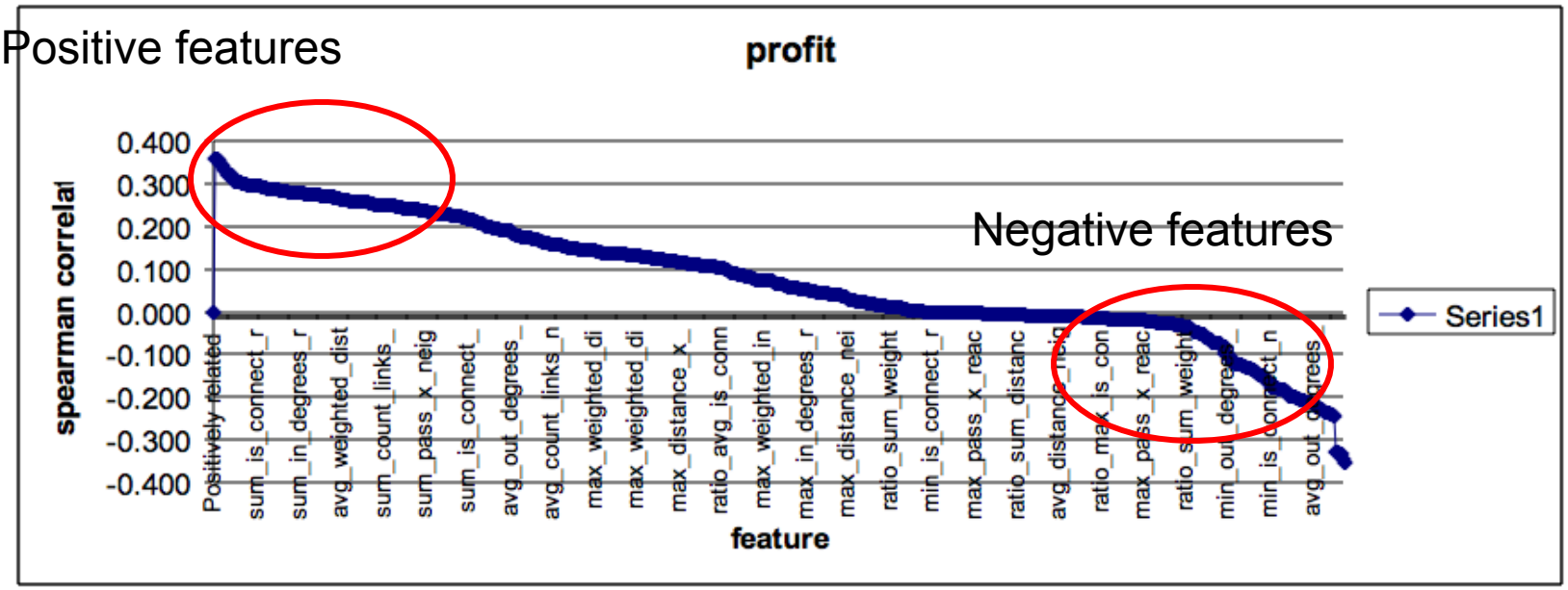
  + Financial statements of companies
    - previous year's profit/ revenue
      - delta-change of profit /revenue
      - … etc.

Steps to Learn for Network Feature Selection

- correlations between ranking of each individual feature and ranking of revenue/profit
- Stability of feature values which should be consistent with different network thresholding
- Selecting Independent Features sets (orthogonal with each other)

Feature Selection

- Feature Selection
  - Filter out some un-useful features from leaning samples.
  - Positive features VS negative features
  - Company-specific selections or General selections



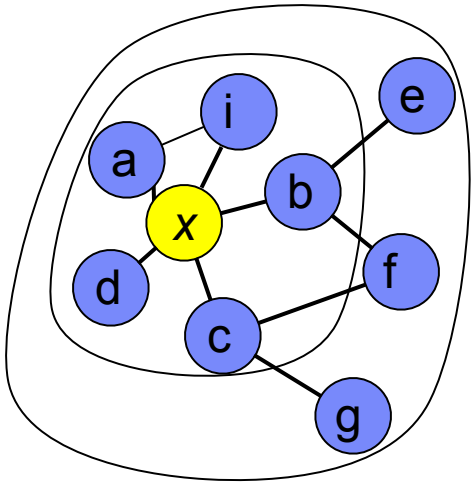
Positive and Negative Features (example)

Correlation	Positively related features
0.421	difference of the ratio of x's neighbors and reachable nodes in binary-undirected network in 3 years
0.421	delta value with 3 years ago of x's degree in binary-undirected network
0.420	2 year ago x's degree in binary-undirected network
0.413	x's degree in binary-undirected network
0.413	ratio number of x's neighbors and reachable nodes in binary-undirected network
0.353	2 year ago x's in-degree in weighted-undirected network
0.344	delta value with 3 years ago of x's out-degree in weighted-directed network

Correlation	Negatively related features
-0.487	previous year's connections among neighbors in binary-undirected network
-0.477	delta value with 2 year's ago of sum of degrees among neighbors in binary-undirected network
-0.462	previous year's connection among neighbors in weighted-undirected network
-0.462	previous year's connection among neighbors in binary-undirected network
-0.381	ratio of connection among neighbors and reachable nodes in weighted-undirected network
-0.379	previous year's ratio of connection among neighbors and reachable nodes in weighted-undirected network

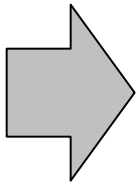
Positive Feature Example

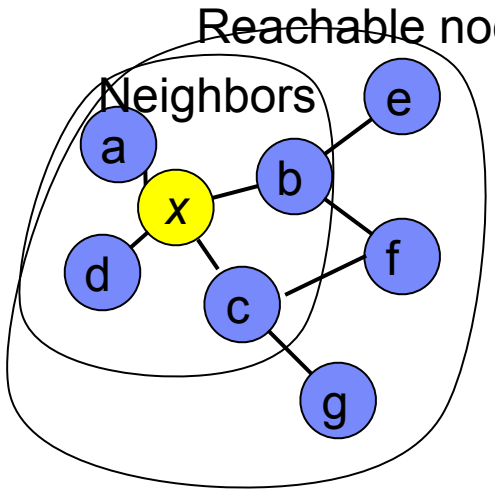
"difference of the ratio of x's neighbors and reachable nodes in binary-undirected network in 3 years"



x's network in 2010

N1= {a,b,c,d,i}
N2={a,b,c,d,e,f,g,h,i}





x's network in 2007

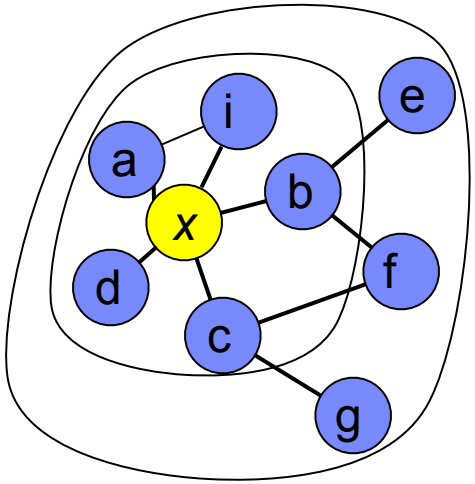
N1= {a,b,c,d}
N2={a,b,c,d,e,f,g}

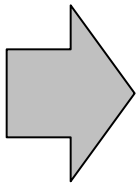
2010: |N1| = 5, |N2|= 8, ratio(|N1|,|N2|) = 5/8 = 0.625
2007: |N1| = 4, |N2|= 7, ratio'(|N1|,|N2|) = 4/7= 0.57

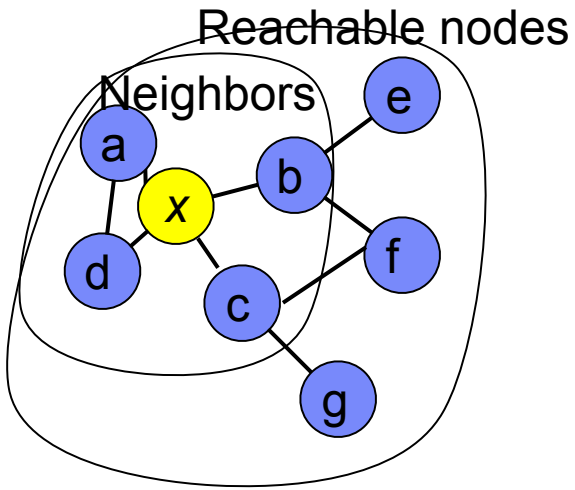
➔ Delta (ratio – ratio') = 5/8 – 4/7 = 0.054

Negative Feature Example

"previous year's connections among neighbors in binary-undirected network"







x's network in 2010

N1= {a,b,c,d,i}
N2={a,b,c,d,e,f,g,h,i}

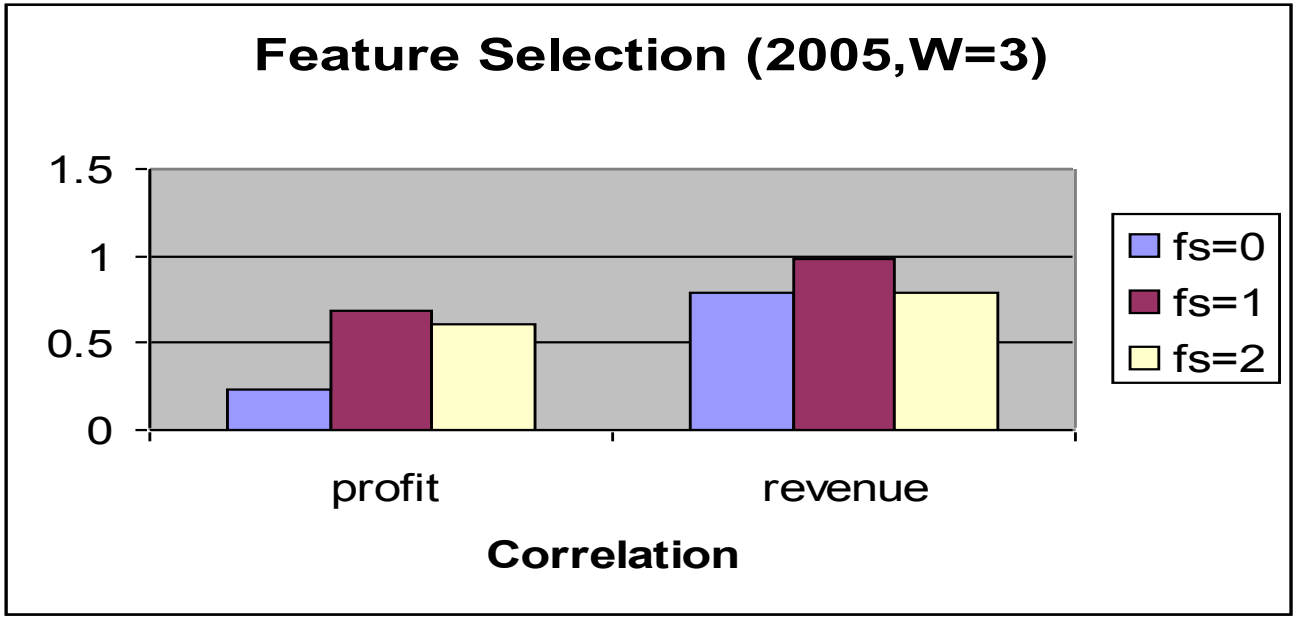
x's network in 2009

N1= {a,b,c,d}
N2={a,b,c,d,e,f,g}

Connection_N1= {b-c, a-d}

➔Connection_t-1= 2

Feature Set Selection



From Leaning samples, move out features which |correlation|<0.2, #sample<50.

fs=0: No feature selection

fs=1: Feature selection (positive features only)

fs=2: Feature selection (positive and negative features)

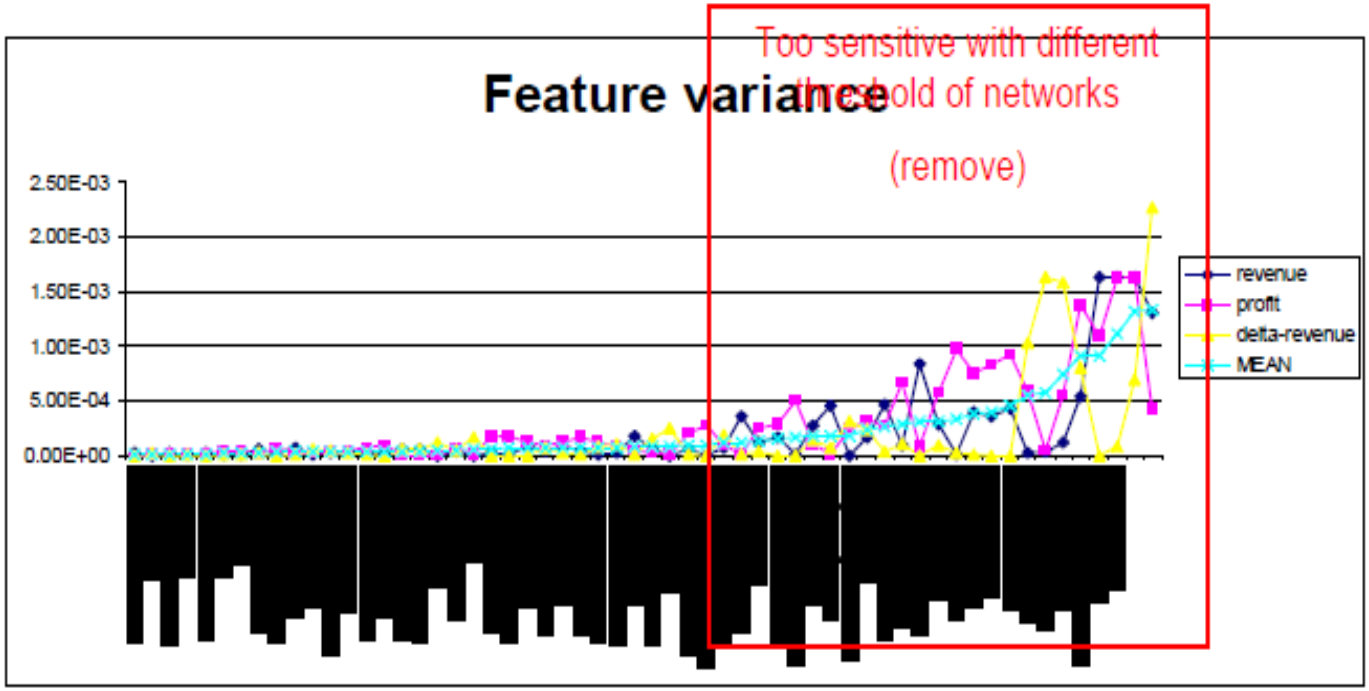
Using positive features are enough for our prediction model.

31

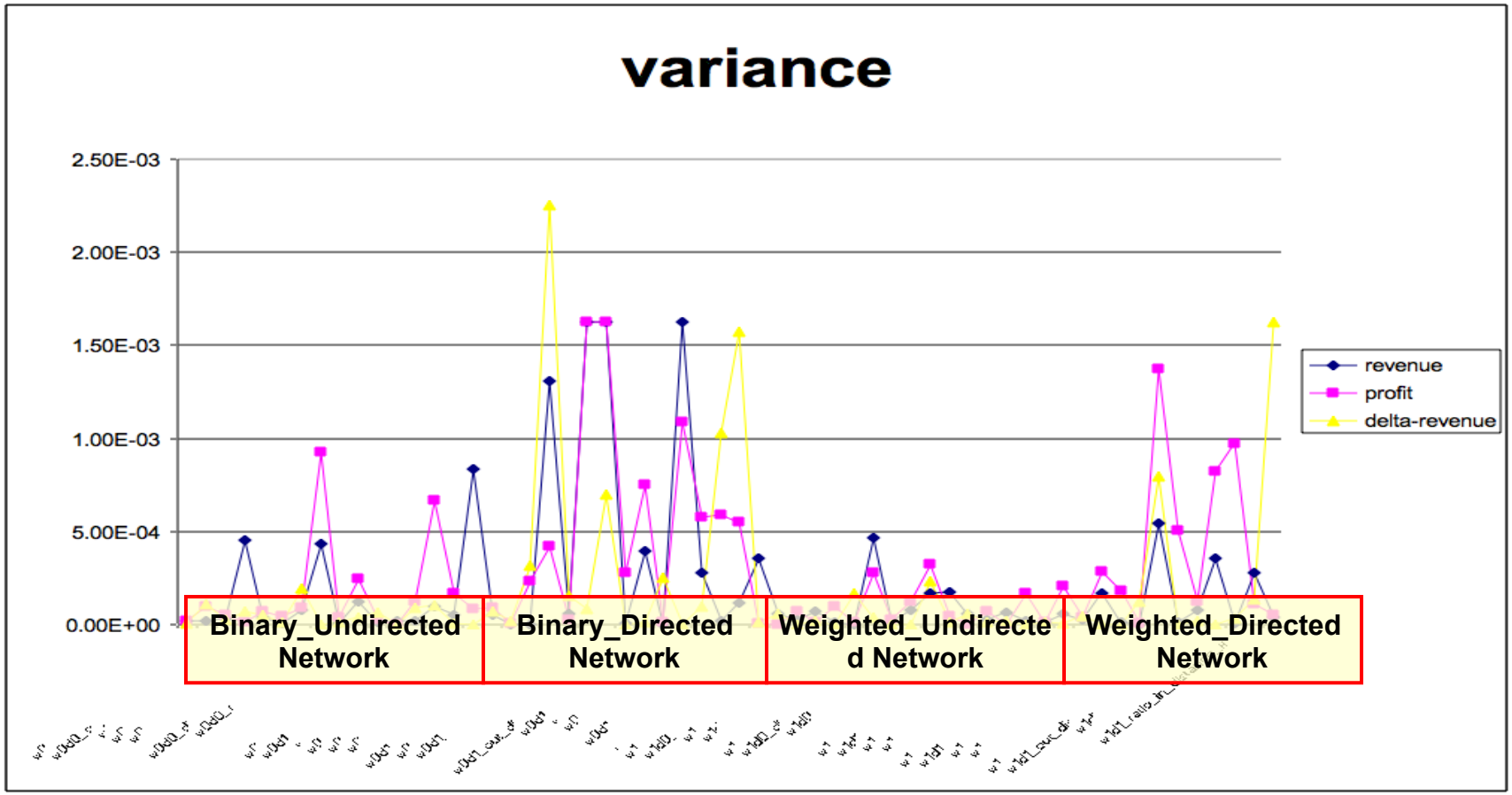
Feature Variances

$$\mathrm{var}_{F_i} = \frac{\sum_{k \in K} (corr_k(F_i, T_i) - \overline{corr})^2}{|K|}$$

k: various networks in different threshold
i: different features



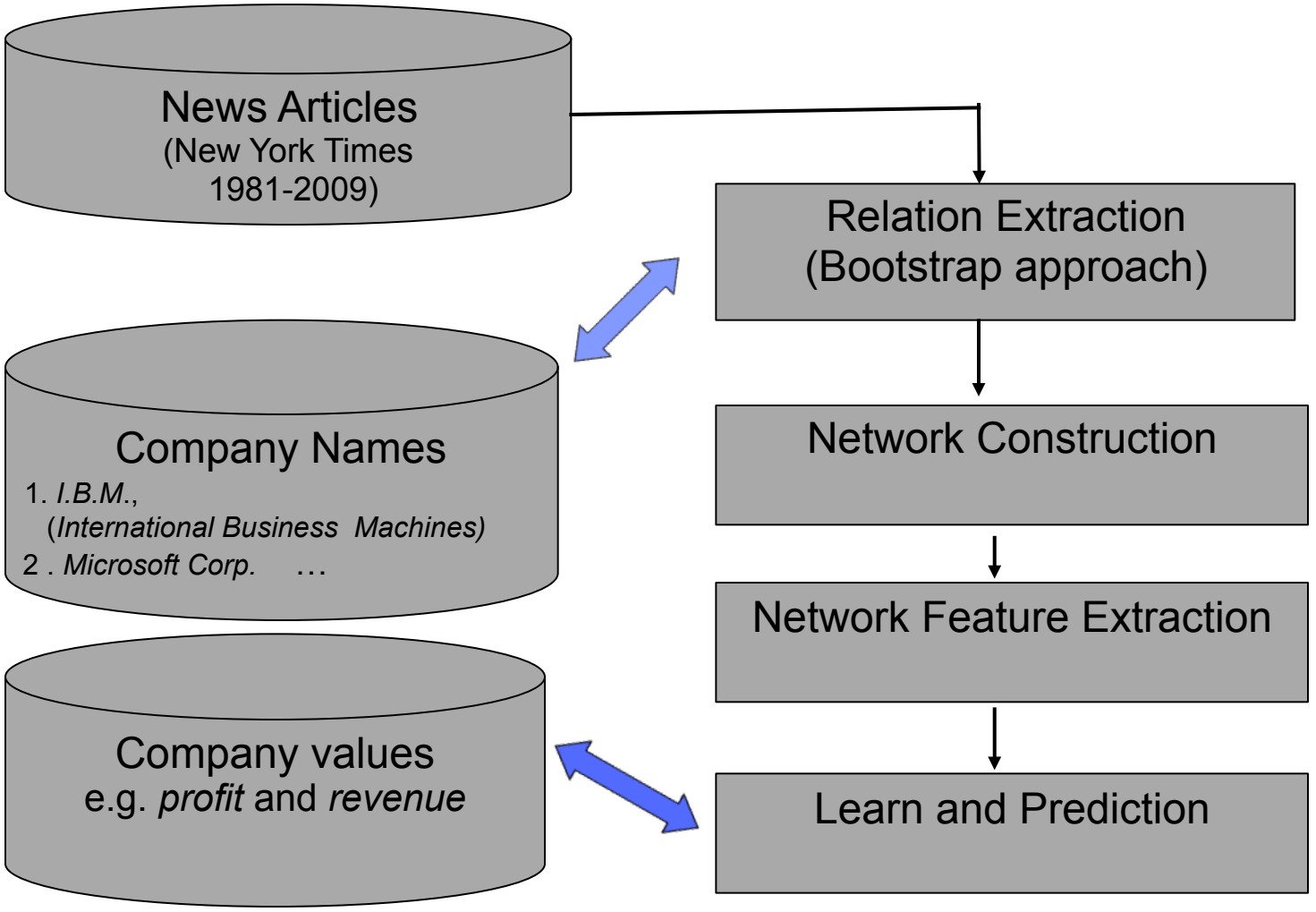
Feature Selection based on Stability of values with different network thresholding



Outline

- Background and Study goal
- Infer Company Networks from Public News
- Network Feature Generation & Selection
- Predict Company Value
- Conclusion and Future work

System Outline



Experiments

- Tasks:
  - } For individual companies, learn from last 10 years, and predict next year's company value
  - } For 20 fortune companies, learn from past 5 years, and predict next year's Companies Value.
    - } Company Value: revenue, profit

- Prediction Model
  - Linear Regression

$$value = a + \sum_i \beta_i feature_i + \varepsilon .$$

  - SVM Regression (using RBF kernel)

$$\min_{\mathbf{w},b,\xi,\xi^*} \frac{1}{2}\mathbf{W}^T\mathbf{W} + C\sum_{i=1}^{l}\xi_i + C\sum_{i=1}^{l}\xi_i^*$$

$$\text{subject to} \quad \mathbf{w}^{\mathrm{T}}\phi(\mathbf{x}_i) + b - z_i \le \varepsilon + \xi_i,$$
$$z_i - \mathbf{w}^{\mathrm{T}}\phi(\mathbf{x}_i) - b \le \varepsilon + \xi_i^*,$$
$$\xi_i \xi_i^* \ge 0, \; i = 1, \ldots, l.$$

$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\gamma \,||\, \mathbf{x}_i - \mathbf{x}_j \,||^2), \gamma > 0.$$

Performance Measures

- **R^2** (squared Correlation Coefficient)

$$R^2 = \frac{(\bar{l}\sum_{i=1}^{\bar{l}} f(\mathbf{x}_i)y_i - \sum_{i=1}^{\bar{l}} f(\mathbf{x}_i)\sum_{i=1}^{\bar{l}} y_i)^2}{(\bar{l}\sum_{i=1}^{\bar{l}} f(\mathbf{x}_i)^2 - (\sum_{i=1}^{\bar{l}} f(\mathbf{x}_i))^2)(\bar{l}\sum_{i=1}^{\bar{l}} y_i^2 - (\sum_{i=1}^{\bar{l}} y_i)^2)}$$

- **MSE** (Mean Squared Error)

$$\text{MSE} = \frac{1}{\bar{l}}\sum_{i=1}^{\bar{l}} (f(\mathbf{x}_i) - y_i)^2$$

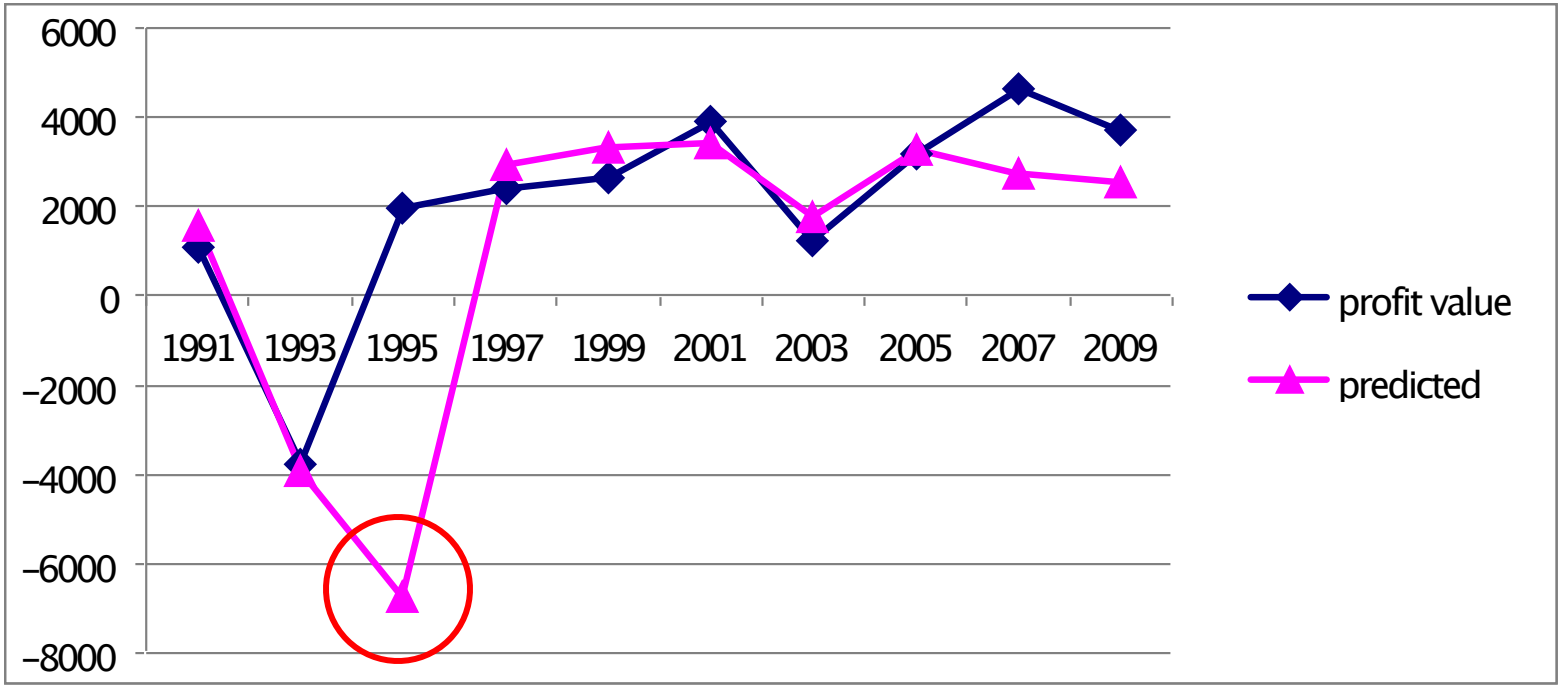
$$\text{Testing data}: \mathbf{x}_1, \ldots, \mathbf{x}_{\bar{l}}$$

$$\text{Target values}: y_i, \ldots, y_{\bar{l}}$$

$$\text{Predicted values}: f(\mathbf{x}_1), \ldots, f(\mathbf{x}_{\bar{l}})$$

Profit Prediction for Fortune Companies

- Predict 20 companies' mean value of profits
  "*I.B.M, Intel, Microsoft, GM, HP, Honda, Nissan, AT&T, Wal-Mart, Yahoo!, Nike, Dell, Starbucks, Chase, PepsiCo, Cisco, FedEx, Gap, AEP, Sun*"

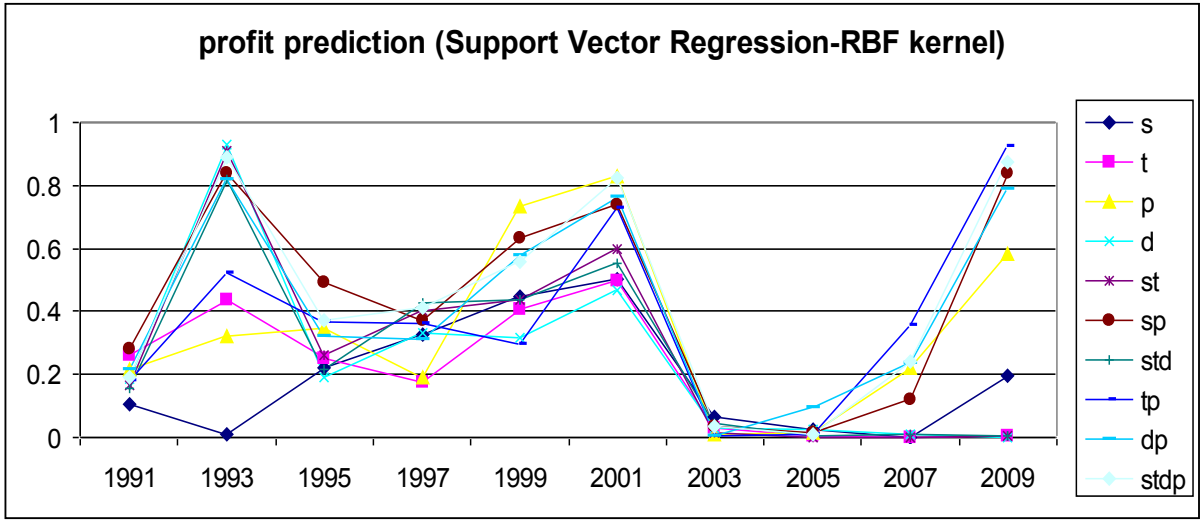


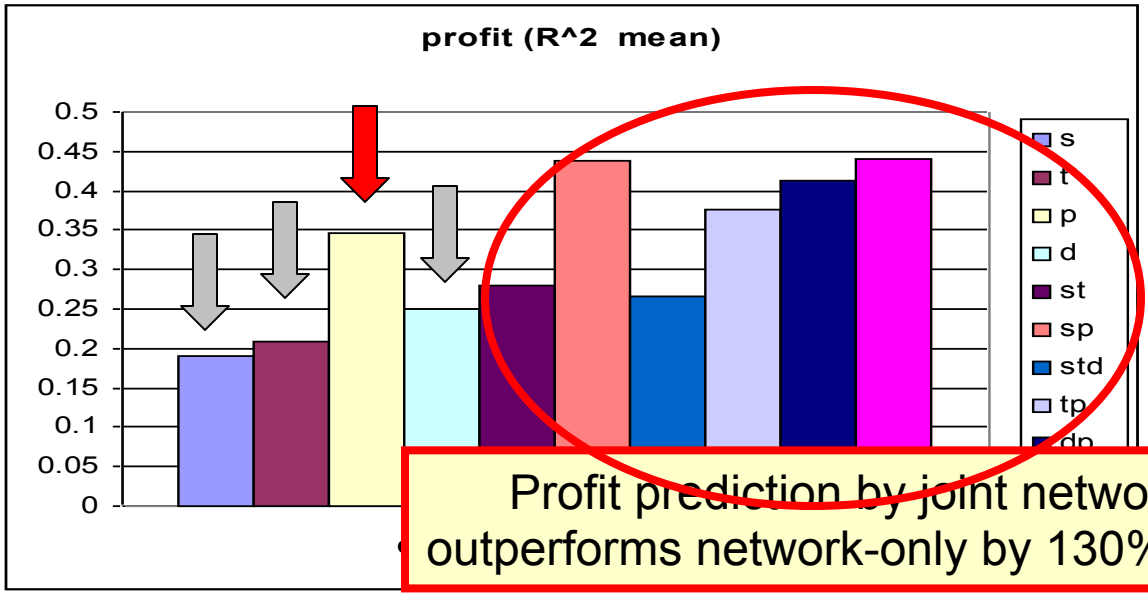
Profit Prediction using different feature sets (SVR)

Targets: 20 Fortune companies' normalized Profits

Goal: Learn from previous 5 years, and predict next year

Model: Support Vector Regression (RBF kernel)





Network feature:
s (current year network feature),
t (temporal network feature),
d (delta value of network feature)
Financial feature:
p (historical profits and revenues)

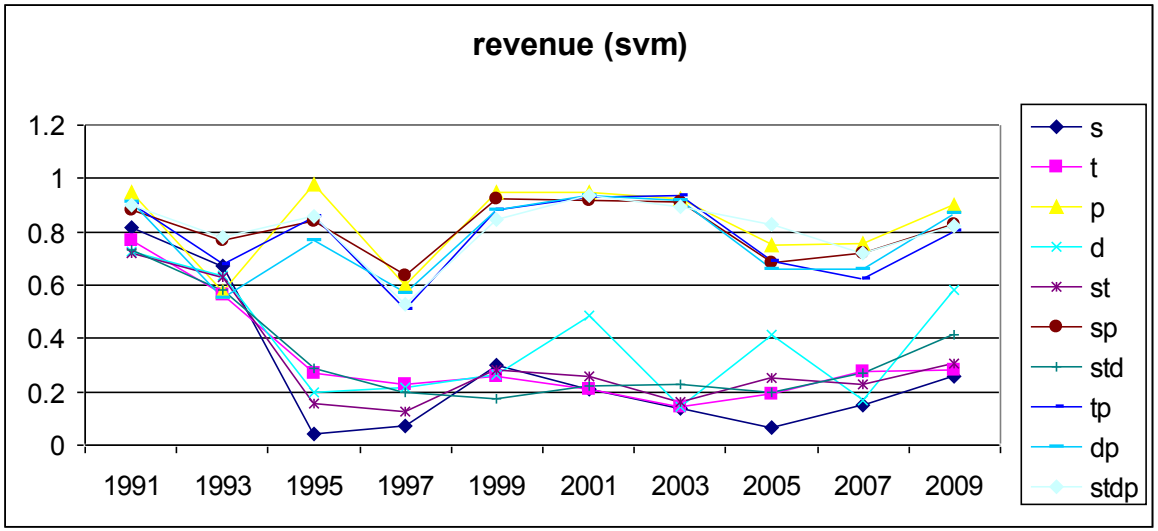
Profit prediction by joint network and financial analysis outperforms network-only by 130% and financial-only by 33%.

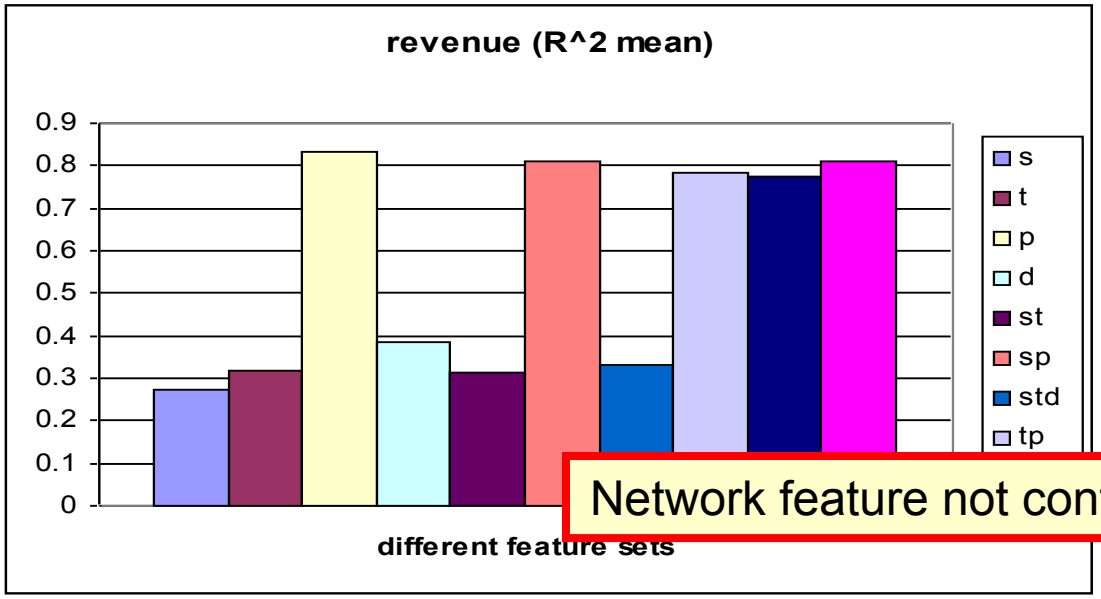
Revenue Prediction using different feature sets (SVR)

Targets: 20 Fortune companies' normalized Profits

Goal: Learn from previous 5 years, and predict next year

Model: Support Vector Regression (RBF kernel)





Network feature:
s (current year network feature),
t (temporal network feature),
d (delta value of network feature)
Financial feature:
p (historical profits and revenues)

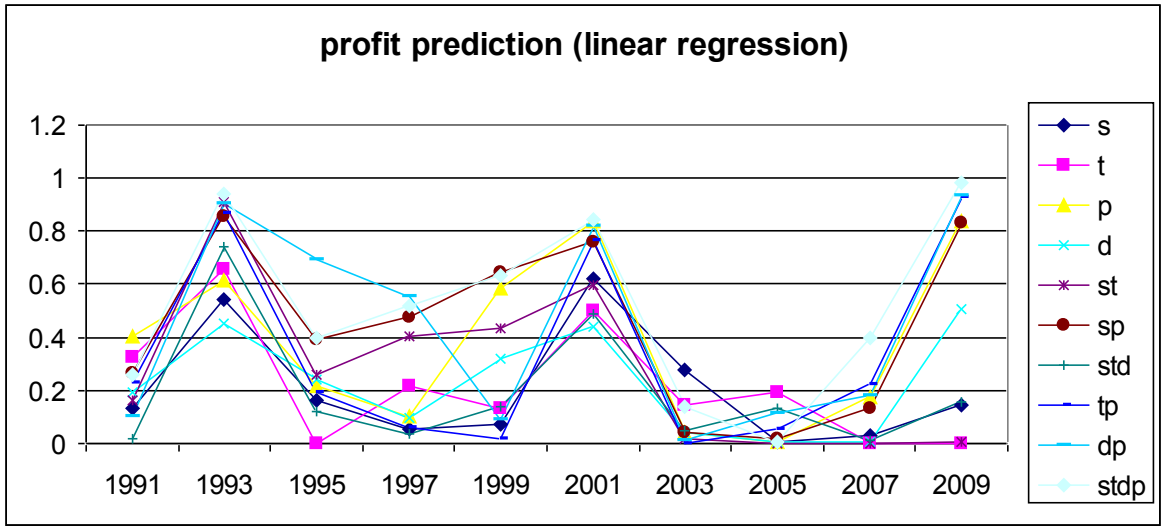
Network feature not contribute to revenue prediction.

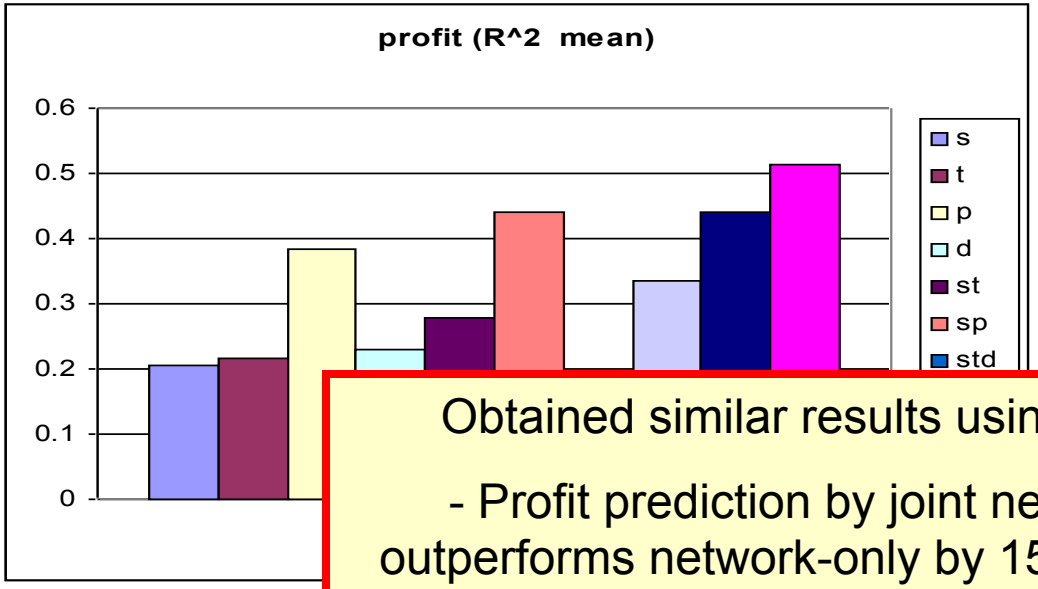
Profit Prediction (Linear Regression)

Targets: 20 Fortune companies' normalized Profits

Goal: Learn from previous 5 years, and predict next year

Model: linear regression





Network feature:
s (current year network feature),
t (temporal network feature),
d (delta value of network feature)
Financial feature:
p (historical profits and revenues)

Obtained similar results using different prediction model.

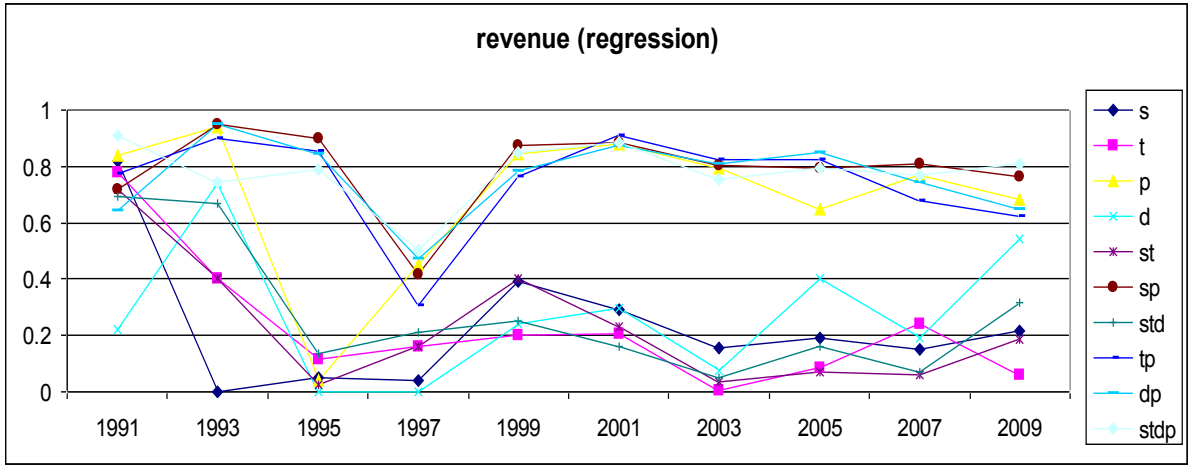
- Profit prediction by joint network and financial analysis outperforms network-only by 150% and financial-only by 37%.

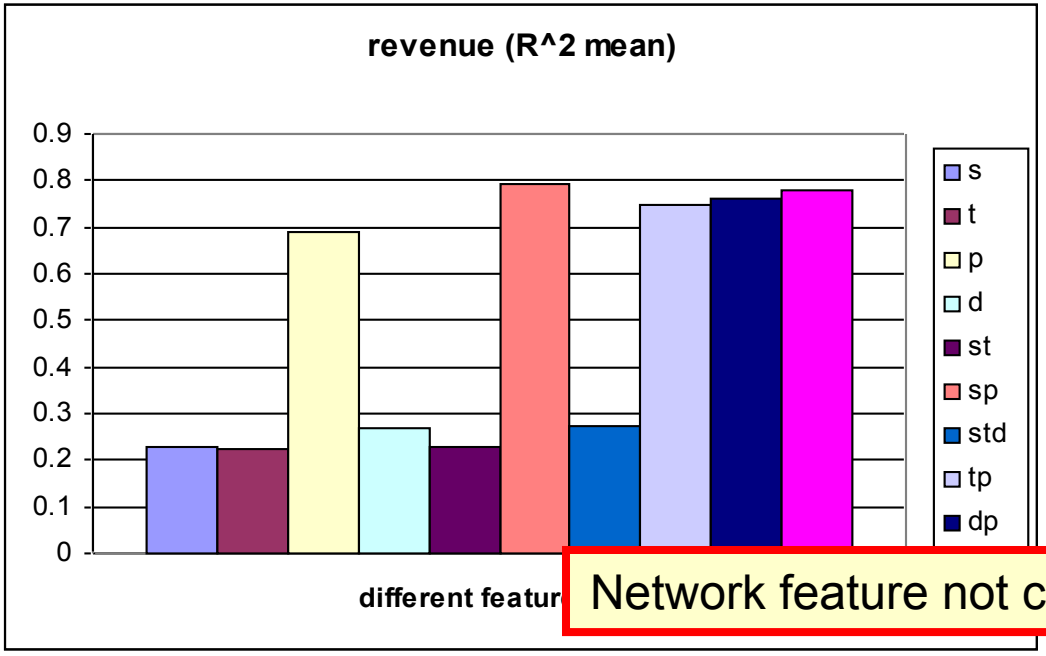
Revenue Prediction (Linear Regression)

Targets: 20 Fortune companies' normalized Profits

Goal: Learn from previous 5 years, and predict next year

Model: linear regression



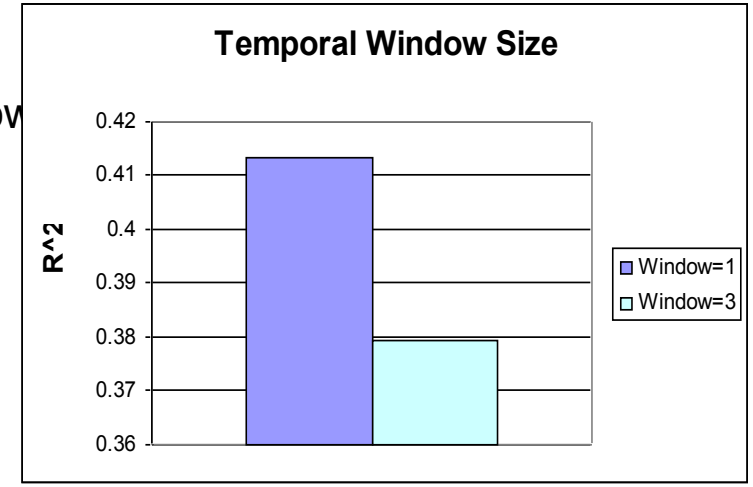


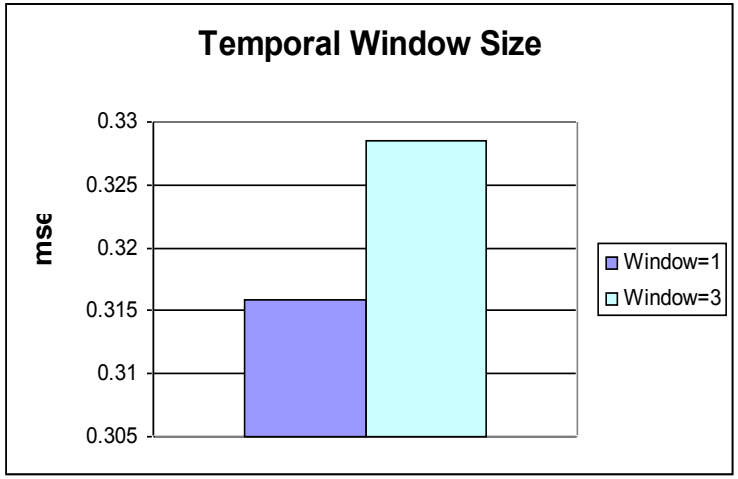
Network feature:
s (current year network feature),
t (temporal network feature),
d (delta value of network feature)
Financial feature:
p (historical profits and revenues)

Network feature not contribute to revenue prediction.

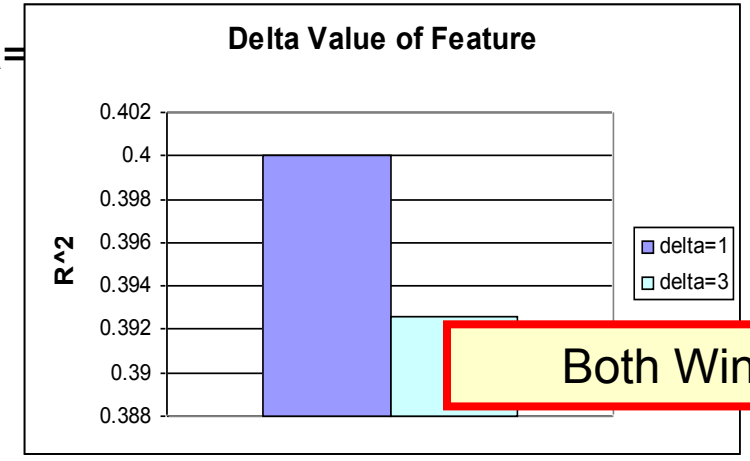
Temporal Window and Delta for Profit Prediction

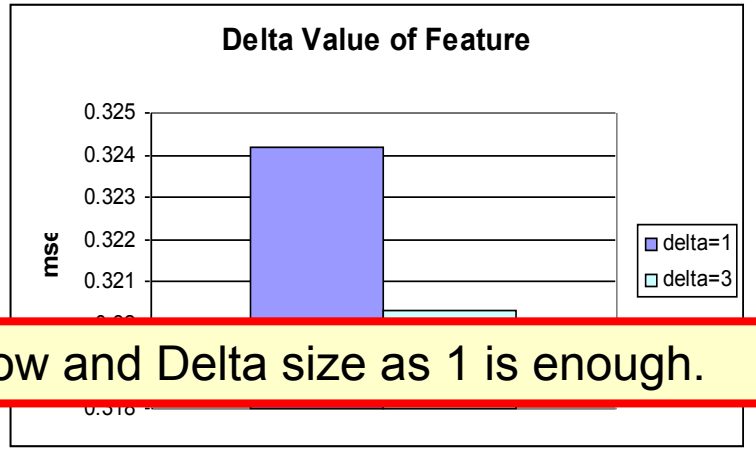
- Windo





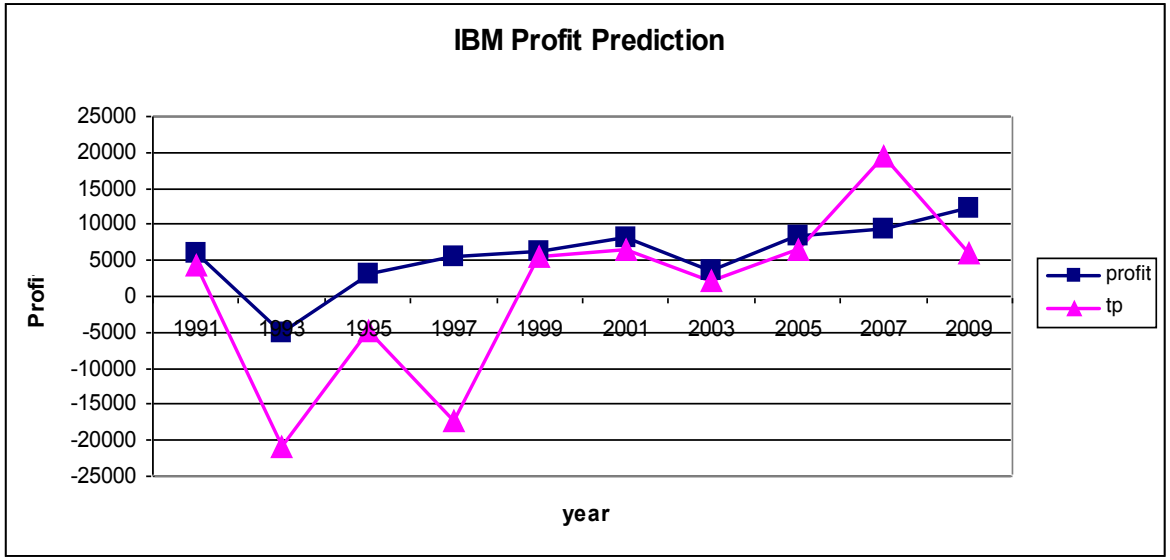
- Delta=



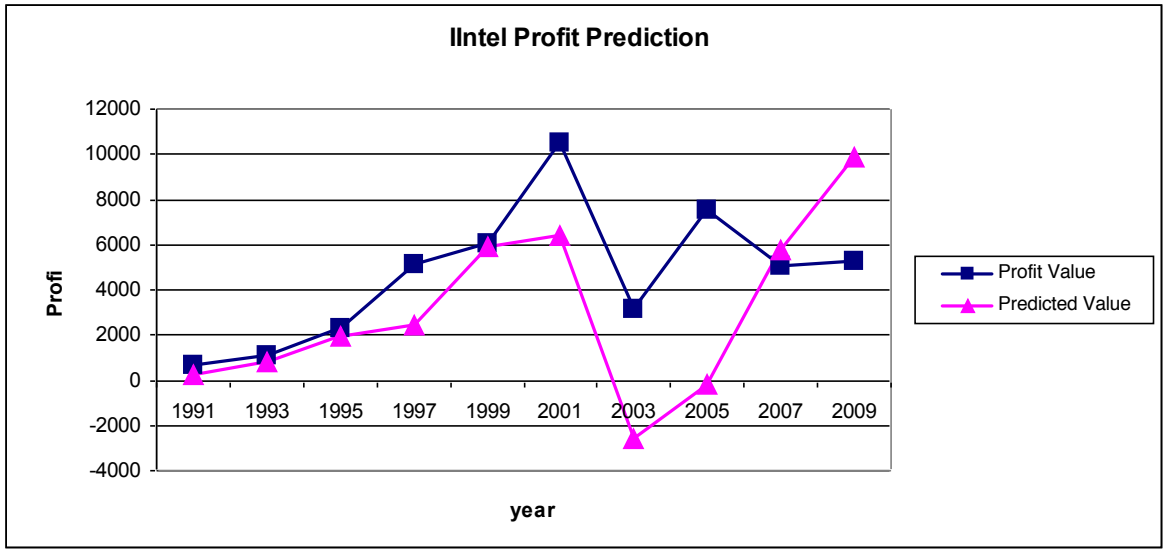


Both Window and Delta size as 1 is enough.

Profit Prediction for IBM and Intel



R^2 = 0.583



R^2 = 0.237

Questions?