

# Chapter 1: Basic Concepts

---

Instructor: Dr. Mehmet S. Aktaş

# Road map

- World Wide Web
- Internet
- Search Engines
- Web Data Mining
  - Why? What? Where?
- Summary

# World Wide Web - WWW

- The Web
  - allows users of one computer to access information stored on another through a world-wide network called the Internet.
- Client-server based implementation
  - Ex: local machine browser, remote machine: Web servers
- Hypertext document based semantic structure
  - Ted Nelson in 1965
  - Hypertext, hyperlinks

# World Wide Web - History

- The Web
  - Invented by Tim Berners-Lee at CERN in 1989 (19 years ago!!!).
  - Distributed hypertext system – basic architecture of the Web.
  - Hypertext Transfer Protocol (HTTP), Hypertext Markup Language (HTML), Universal Resource Locator (URL)
- Browsers
  - Mosaic, 1993, M. Andreesen, Univ. of Illinois
  - Netspace, 1994, M. Andreesen and Jim Clark, Netscape Comm.
  - Internet Explorer, 1995, B. Gates, Microsoft

# Internet - History

- ARPANET - Internet
  - Advanced Research Projects Agency (ARPA) Network, 1969-1972.
    - Connected computers from 40 different locations.
  - TCP/IP (Transmission Control Protocol/Internet Protocol), 1973, Vinton Cerf, Bob Kahn.
  - Internet
    - 1982 TCP/IP was widely adopted.
    - Networks were connected using TCP/IP Protocol and formed inter-net.

# Search Engines

- Excite search system
  - 1993, started by six Stanford Univ. students
- Yahoo
  - 1994, started out as a listing of favorite web sites
  - 2004, provided general search capability with Inktomi
- Many Search Engines
  - 1994 - present, Lycos, Infoseek, Alta Vista, Ask Jeeves etc.
- Google
  - 1998 by Sergey Brin and Larry Page at Stanford Univ.
- MSN
  - 2005 by Microsoft, which used others search engines before

# Characteristics of the Web

- The Web has huge amount of available data
- The Web has all types of data
- The Web has heterogeneous information
- The Web has a hyperlink structure
- The Web has noisy information
- The Web is about services
- The Web is very dynamic
- The Web is a virtual society

# Web Data Mining

## ■ Web Mining

- aims to discover useful information from the Web hyperlink structure, page content and usage data.
- Web structure mining
  - Discovers knowledge from hyperlinks
- Web content mining
  - Mines useful information from Web page contents
- Web usage mining
  - Mines useful information from Web usage logs

# Chapter 2: Association Rules & Sequential Patterns

---

Instructor: Dr. Mehmet S. Aktas

Acknowledgement: Thanks to Dr. Bing Liu for teaching materials.

# Road map

- Basic concepts of Association Rules
- Apriori algorithm
- Different data formats for mining
- Mining class association rules
- Sequential pattern mining
- Summary

# Association rule mining

- Proposed by **Agrawal et al in 1993.**
- It is an important data mining model studied extensively by the database and data mining community.
- Assume all data are categorical.
- No good algorithm for numeric data.
- Initially used for **Market Basket Analysis** to find how items purchased by customers are related.

Bread → Milk [sup = 5%, conf = 100%]

# The model: data

- $I = \{i_1, i_2, \dots, i_m\}$ : a set of *items*.
- Transaction  $t$ :
  - $t$  a set of items, and  $t \subseteq I$ .
- Transaction Database  $T$ : a set of transactions  
 $T = \{t_1, t_2, \dots, t_n\}$ .

# Transaction data: supermarket data

## ■ Market basket transactions:

t1: {bread, cheese, milk}

t2: {apple, eggs, salt, yogurt}

...

...

tn: {biscuit, eggs, milk}

## ■ Concepts:

- An *item*: an item/article in a basket
- $I$ : the set of all items sold in the store
- A *transaction*: items purchased in a basket; it may have TID (transaction ID)
- A *transactional dataset*: A set of transactions

# Transaction data: a set of documents

- **A text document data set. Each document is treated as a “bag” of keywords**

doc1: Student, Teach, School

doc2: Student, School

doc3: Teach, School, City, Game

doc4: Baseball, Basketball

doc5: Basketball, Player, Spectator

doc6: Baseball, Coach, Game, Team

doc7: Basketball, Team, City, Game

# The model: rules

- A transaction  $t$  contains  $X$ , a set of items (**itemset**) in  $I$ , if  $X \subseteq t$ .
- An **association rule** is an implication of the form:  
$$X \rightarrow Y, \text{ where } X, Y \subset I, \text{ and } X \cap Y = \emptyset$$
- An **itemset** is a set of items.
  - E.g.,  $X = \{\text{milk, bread, cereal}\}$  is an itemset.
- A  **$k$ -itemset** is an itemset with  $k$  items.
  - E.g.,  $\{\text{milk, bread, cereal}\}$  is a 3-itemset

# Rule strength measures

- **Support:** The rule holds with **support**  $sup$  in  $T$  (the transaction data set) if  $sup\%$  of transactions contain  $X \cup Y$ .
  - $sup = \Pr(X \cup Y)$ .
- **Confidence:** The rule holds in  $T$  with **confidence**  $conf$  if  $conf\%$  of transactions that contain  $X$  also contain  $Y$ .
  - $conf = \Pr(Y | X)$
- An association rule is a pattern that states when  $X$  occurs,  $Y$  occurs with certain probability.

# Support and Confidence

- **Support count:** The support count of an itemset  $X$ , denoted by  $X.count$ , in a data set  $T$  is the number of transactions in  $T$  that contain  $X$ . Assume  $T$  has  $n$  transactions.
- Then,

$$support = \frac{(X \cup Y).count}{n}$$

$$confidence = \frac{(X \cup Y).count}{X.count}$$

# Goal and key features

- **Goal:** Find all rules that satisfy the user-specified *minimum support* (minsup) and *minimum confidence* (minconf).
- **Key Features**
  - Completeness: find all rules.
  - No target item(s) on the right-hand-side

# An Example:

Transaction-id	Items bought
10	A, B, D
20	A, C, D
30	A, D, E
40	B, E, F
50	B, C, D, E, F

- Itemset  $X = \{x_1, \dots, x_k\}$
- Find all the rules  $X \rightarrow Y$  with minimum support and confidence
  - support,  $s$ , probability that a transaction contains  $X \cup Y$
  - confidence,  $c$ , conditional probability that a transaction having  $X$  also contains  $Y$
- Let  $sup_{min} = 50\%$ ,  $conf_{min} = 50\%$
- Freq. Pat.:  $\{A:3, B:3, D:4, E:3, AD:3\}$
- Association rules:
  - $A \rightarrow D$  (60%, 100%)
  - $D \rightarrow A$  (60%, 75%)

# Another example

- Transaction data
- Assume:

$\text{minsup} = 30\%$

$\text{minconf} = 80\%$

- An example **frequent itemset**:

{Chicken, Clothes, Milk} [sup = 3/7]

- Association rules from the itemset:

Clothes  $\rightarrow$  Milk, Chicken [sup = 3/7, conf = 3/3]

...

...

Clothes, Chicken  $\rightarrow$  Milk, [sup = 3/7, conf = 3/3]

t1:	Beef, Chicken, Milk
t2:	Beef, Cheese
t3:	Cheese, Boots
t4:	Beef, Chicken, Cheese
t5:	Beef, Chicken, Clothes, Cheese, Milk
t6:	Chicken, Clothes, Milk
t7:	Chicken, Milk, Clothes

# Transaction data representation

- A simplistic view of shopping baskets,
- Some important information not considered.  
E.g,
  - the quantity of each item purchased and
  - the price paid.

# Many mining algorithms

- **There are a large number of them!!**
- They use different strategies and data structures.
- Their resulting sets of rules are all the same.
  - Given a transaction data set  $T$ , and a minimum support and a minimum confident, the set of association rules existing in  $T$  is uniquely determined.
- Any algorithm should find the same set of rules although their computational efficiencies and memory requirements may be different.
- We study only one: **the Apriori Algorithm**

# Road map

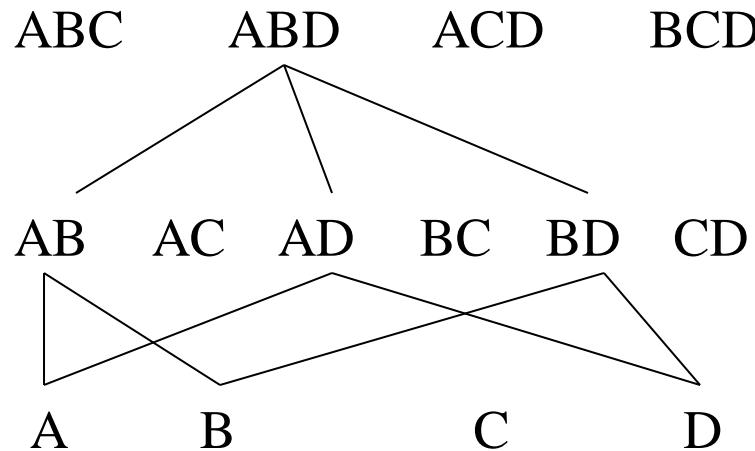
- Basic concepts of Association Rules
- **Apriori algorithm**
- Different data formats for mining
- Mining class association rules
- Sequential pattern mining
- Summary

# The Apriori algorithm

- **The best known algorithm**
- **Two steps:**
  - Find all itemsets that have minimum support (*frequent itemsets*, also called large itemsets).
  - Use frequent itemsets to **generate rules**.
- E.g., a frequent itemset  
    {Chicken, Clothes, Milk}      [sup = 3/7]  
and one rule from the frequent itemset  
    Clothes → Milk, Chicken      [sup = 3/7, conf = 3/3]

# Step 1: Mining all frequent itemsets

- A **frequent itemset** is an itemset whose support is  $\geq \text{minsup}$ .
- **Key idea:** The apriori property (downward closure property): any subsets of a frequent itemset are also frequent itemsets



# The Algorithm

- **Iterative algo.** (also called **level-wise search**):  
Find all 1-item frequent itemsets; then all 2-item frequent itemsets, and so on.
  - In each iteration  $k$ , only consider itemsets that contain some  $k-1$  frequent itemset.

- Find frequent itemsets of size 1:  $F_1$
- From  $k = 2$ 
  - $C_k$  = candidates of size  $k$ : those itemsets of size  $k$  that could be frequent, given  $F_{k-1}$
  - $F_k$  = those itemsets that are actually frequent,  $F_k \subseteq C_k$  (need to scan the database once).

# Example – Finding frequent itemsets

Dataset T  
minsup=0.5

TID	Items
T100	1, 3, 4
T200	2, 3, 5
T300	1, 2, 3, 5
T400	2, 5

itemset:count

1. scan T → C<sub>1</sub>: {1}:2, {2}:3, {3}:3, {4}:1, {5}:3

→ F<sub>1</sub>: {1}:2, {2}:3, {3}:3, {5}:3

→ C<sub>2</sub>: {1,2}, {1,3}, {1,5}, {2,3}, {2,5}, {3,5}

2. scan T → C<sub>2</sub>: {1,2}:1, {1,3}:2, {1,5}:1, {2,3}:2, {2,5}:3, {3,5}:2

→ F<sub>2</sub>: {1,3}:2, {2,3}:2, {2,5}:3, {3,5}:2

→ C<sub>3</sub>: {2, 3, 5}

3. scan T → C<sub>3</sub>: {2, 3, 5}:2 → F<sub>3</sub>: {2, 3, 5}

# Details: the algorithm

## Algorithm Apriori( $T$ )

```
C1 ← init-pass( $T$ );  
F1 ← { $f \mid f \in C_1, f.\text{count}/n \geq \text{minsup}$ }; // n: no. of transactions in T  
for ( $k = 2; F_{k-1} \neq \emptyset; k++$ ) do  
    Ck ← candidate-gen( $F_{k-1}$ );  
    for each transaction  $t \in T$  do  
        for each candidate  $c \in C_k$  do  
            if c is contained in  $t$  then  
                 $c.\text{count}++$ ;  
        end  
    end  
    Fk ← { $c \in C_k \mid c.\text{count}/n \geq \text{minsup}$ }  
end  
return  $F \leftarrow \bigcup_k F_k$ ;
```

# Apriori candidate generation

- The **candidate-gen** function takes  $F_{k-1}$  and returns a **superset** (called the candidates) of the set of all **frequent  $k$ -itemsets**. It has two steps
  - **join step**: Generate all possible candidate itemsets  $C_k$  of length  $k$
  - **prune step**: Remove those candidates in  $C_k$  that cannot be frequent.

# Candidate-gen function

**Function** candidate-gen( $F_{k-1}$ )

```
 $C_k \leftarrow \emptyset;$ 
forall  $f_1, f_2 \in F_{k-1}$ 
  with  $f_1 = \{i_1, \dots, i_{k-2}, i_{k-1}\}$ 
  and  $f_2 = \{i_1, \dots, i_{k-2}, i'_{k-1}\}$ 
  and  $i_{k-1} < i'_{k-1}$  do
     $c \leftarrow \{i_1, \dots, i_{k-1}, i'_{k-1}\};$            // join  $f_1$  and  $f_2$ 
     $C_k \leftarrow C_k \cup \{c\};$ 
    for each ( $k-1$ )-subset  $s$  of  $c$  do
      if ( $s \notin F_{k-1}$ ) then
        delete  $c$  from  $C_k;$                       // prune
    end
  end
  return  $C_k;$ 
```

# An example

- $F_3 = \{\{1, 2, 3\}, \{1, 2, 4\}, \{1, 3, 4\}, \{1, 3, 5\}, \{2, 3, 4\}\}$
- After join
  - $C_4 = \{\{1, 2, 3, 4\}, \{1, 3, 4, 5\}\}$
- After pruning:
  - $C_4 = \{\{1, 2, 3, 4\}\}$   
because  $\{1, 4, 5\}$  is not in  $F_3$  ( $\{1, 3, 4, 5\}$  is removed)

## Step 2: Generating rules from frequent itemsets

- Frequent itemsets  $\neq$  association rules
- One more step is needed to generate association rules
- For each frequent itemset  $X$ ,  
For each proper nonempty subset  $A$  of  $X$ ,
  - Let  $B = X - A$
  - $A \rightarrow B$  is an association rule if
    - $\text{Confidence}(A \rightarrow B) \geq \text{minconf}$ ,
    - $\text{support}(A \rightarrow B) = \text{support}(A \cup B) = \text{support}(X)$
    - $\text{confidence}(A \rightarrow B) = \text{support}(A \cup B) / \text{support}(A)$

# Generating rules: an example

- Suppose  $\{2,3,4\}$  is frequent, with  $\text{sup}=50\%$ 
  - Proper nonempty subsets:  $\{2,3\}$ ,  $\{2,4\}$ ,  $\{3,4\}$ ,  $\{2\}$ ,  $\{3\}$ ,  $\{4\}$ , with  $\text{sup}=50\%$ ,  $50\%$ ,  $75\%$ ,  $75\%$ ,  $75\%$ ,  $75\%$  respectively
  - These generate these association rules:
    - $2,3 \rightarrow 4$ ,      confidence=100%
    - $2,4 \rightarrow 3$ ,      confidence=100%
    - $3,4 \rightarrow 2$ ,      confidence=67%
    - $2 \rightarrow 3,4$ ,      confidence=67%
    - $3 \rightarrow 2,4$ ,      confidence=67%
    - $4 \rightarrow 2,3$ ,      confidence=67%
    - All rules have support = 50%

# Generating rules: summary

- To recap, in order to obtain  $A \rightarrow B$ , we need to have  $\text{support}(A \cup B)$  and  $\text{support}(A)$
- All the required information for confidence computation has already been recorded in itemset generation. No need to see the data  $T$  any more.
- This step is not as time-consuming as frequent itemsets generation.

# On Apriori Algorithm

Seems to be very expensive

- Level-wise search
- $K =$  the size of the largest itemset
- It makes at most  $K$  passes over data
- In practice,  $K$  is bounded (10).
- The algorithm is very fast.
- Scale up to large data sets.

# Road map

- Basic concepts of Association Rules
- Apriori algorithm
- Different data formats for mining
- Mining class association rules
- Sequential pattern mining
- Summary

# Different data formats for mining

- The data can be in transaction form or table form

Transaction form: a, b

a, c, d, e

a, d, f

Table form: Attr1 Attr2 Attr3

a, b, d

b, c, e

- Table data need to be converted to transaction form for association mining

# From a table to a set of transactions

Table form:

Attr1	Attr2	Attr3
a,	b,	d
b,	c,	e

⇒ Transaction form:

(Attr1, a), (Attr2, b), (Attr3, d)

(Attr1, b), (Attr2, c), (Attr3, e)

# Road map

- Basic concepts of Association Rules
- Apriori algorithm
- Different data formats for mining
- Mining class association rules
- Sequential pattern mining
- Summary

# Mining class association rules (CAR)

- Normal association rule mining does not have any target.
- It finds all possible rules that exist in data, i.e., any item can appear as a consequent or a condition of a rule.
- However, in some applications, the user is interested in some targets.
  - E.g, the user has a set of text documents from some known topics. He/she wants to find out what words are associated or correlated with each topic.

# Problem definition

- Let  $T$  be a transaction data set consisting of  $n$  transactions.
- Each transaction is also labeled with a class  $Y$ .
- Let  $I$  be the set of all items in  $T$ ,  $Y$  be the set of all class labels and  $I \cap Y = \emptyset$ .
- A **class association rule (CAR)** is an implication of the form
$$X \rightarrow y, \text{ where } X \subseteq I, \text{ and } y \in Y.$$
- The definitions of **support** and **confidence** are the same as those for normal association rules.

# An example

- A text document data set

doc 1:	Student, Teach, School	: Education
doc 2:	Student, School	: Education
doc 3:	Teach, School, City, Game	: Education
doc 4:	Baseball, Basketball	: Sport
doc 5:	Basketball, Player, Spectator	: Sport
doc 6:	Baseball, Coach, Game, Team	: Sport
doc 7:	Basketball, Team, City, Game	: Sport

- Let  $\text{minsup} = 20\%$  and  $\text{minconf} = 60\%$ . The following are two examples of class association rules:

Student, School → Education [sup= 2/7, conf = 2/2]  
game → Sport [sup= 3/7, conf = 2/3]

# Mining algorithm

- Unlike normal association rules, CARs can be mined directly in one step.
- The key operation is to find all **ruleitems** that have support above *minsup*. A **ruleitem** is of the form:  
 $(condset, y)$   
where **condset** is a set of items from  $I$  (i.e.,  $condset \subseteq I$ ), and  $y \in Y$  is a class label.
- Each ruleitem basically represents a rule:  
 $condset \rightarrow y,$
- The Apriori algorithm can be modified to generate CARs

# Road map

- Basic concepts of Association Rules
- Apriori algorithm
- Different data formats for mining
- Mining class association rules
- Sequential pattern mining
- Summary

# Sequential pattern mining

- Association rule mining does not consider the order of transactions.
- In many applications such orderings are significant. E.g.,
  - in market basket analysis, it is interesting to know whether people buy some items in sequence,
    - e.g., buying bed first and then bed sheets some time later.
  - In Web usage mining, it is useful to find navigational patterns of users in a Web site from sequences of page visits of users

# Basic concepts

- Let  $I = \{i_1, i_2, \dots, i_m\}$  be a set of items.
- **Itemset/element**: A non-empty set of items  $X \subseteq I$ .
- **Sequence**: An ordered list of itemsets. We denote a sequence  $s$  by  $\langle a_1 a_2 \dots a_r \rangle$ , where  $a_i$  is an itemset, which is also called an **element** of  $s$ .
- An element (or an itemset) of a sequence is denoted by  $\{x_1, x_2, \dots, x_k\}$ , where  $x_j \in I$  is an item.
- We assume without loss of generality that items in an element of a sequence are in **lexicographic order**.

# Basic concepts (contd)

- **Size**: The **size** of a sequence is the number of elements (or itemsets) in the sequence.
- **Length**: The **length** of a sequence is the number of items in the sequence.
  - A sequence of length  $k$  is called  **$k$ -sequence**.
- A sequence  $s_1 = \langle a_1 a_2 \dots a_r \rangle$  is a **subsequence** of another sequence  $s_2 = \langle b_1 b_2 \dots b_v \rangle$ , or  $s_2$  is a **supersequence** of  $s_1$ , if there exist integers  $1 \leq j_1 < j_2 < \dots < j_{r-1} < j_r \leq v$  such that  $a_1 \subseteq b_{j_1}$ ,  $a_2 \subseteq b_{j_2}$ , ...,  $a_r \subseteq b_{j_r}$ . We also say that  $s_2$  **contains**  $s_1$ .

# An example

- Let  $I = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$ .
- Sequence  $\langle \{3\}\{4, 5\}\{8\} \rangle$  is **contained** in (or is a **subsequence** of)  $\langle \{6\} \{3, 7\}\{9\}\{4, 5, 8\}\{3, 8\} \rangle$ 
  - because  $\{3\} \subseteq \{3, 7\}$ ,  $\{4, 5\} \subseteq \{4, 5, 8\}$ , and  $\{8\} \subseteq \{3, 8\}$ .
  - The size of the sequence  $\langle \{3\}\{4, 5\}\{8\} \rangle$  is 3, and the length of the sequence is 4.

# Objective

- Given a set  $S$  of **input data sequences** (or sequence database), the problem of mining sequential patterns is to find all the sequences that have **a user-specified minimum support**.
- Each such sequence is called a **frequent sequence**, or a **sequential pattern**.
- The **support** for a sequence is the fraction of total data sequences in  $S$  that contains this sequence.

# Example

Table 1. A set of transactions sorted by customer ID and transaction time

Customer ID	Transaction Time	Transaction (items bought)
1	July 20, 2005	30
1	July 25, 2005	
2	July 9, 2005	10, 20
2	July 14, 2005	
2	July 20, 2005	
3	July 25, 2005	30, 50, 70
4	July 25, 2005	30
4	July 29, 2005	
4	August 2, 2005	
5	July 12, 2005	90

# Example (cond)

Table 2. Data sequences produced from the transaction database in Table 1.

Customer ID	Data Sequence
1	$\langle \{30\} \{90\} \rangle$
2	$\langle \{10, 20\} \{30\} \{40, 60, 70\} \rangle$
3	$\langle \{30, 50, 70\} \rangle$
4	$\langle \{30\} \{40, 70\} \{90\} \rangle$
5	$\langle \{90\} \rangle$

Table 3. The final output sequential patterns

	Sequential Patterns with Support $\geq 25\%$
1-sequences	$\langle \{30\} \rangle, \langle \{40\} \rangle, \langle \{70\} \rangle, \langle \{90\} \rangle$
2-sequences	$\langle \{30\} \{40\} \rangle, \langle \{30\} \{70\} \rangle, \langle \{30\} \{90\} \rangle, \langle \{40, 70\} \rangle$
3-sequences	$\langle \{30\} \{40, 70\} \rangle$

# GSP mining algorithm

- Very similar to the Apriori algorithm

**Algorithm** GSP( $S$ )

```
1    $C_1 \leftarrow \text{init-pass}(S);$                                 // the first pass over  $S$ 
2    $F_1 \leftarrow \{\langle\{f\}\rangle | f \in C_1, f.\text{count}/n \geq \text{minsup}\};$  //  $n$  is the number of sequences in  $S$ 
3   for ( $k = 2; F_{k-1} \neq \emptyset; k++$ ) do                      // subsequent passes over  $S$ 
4      $C_k \leftarrow \text{candidate-gen-SPM}(F_{k-1});$ 
5     for each data sequence  $s \in S$  do                         // scan the data once
6       for each candidate  $c \in C_k$  do
7         if  $c$  is contained in  $s$  then
8            $c.\text{count}++;$                                          // increment the support count
9         end
10      end
11       $F_k \leftarrow \{c \in C_k | c.\text{count}/n \geq \text{minsup}\}$ 
12    end
13  return  $\bigcup_k F_k;$ 
```

**Fig. 12.** The GSP Algorithm for generating sequential patterns

# Road map

- Basic concepts of Association Rules
- Apriori algorithm
- Different data formats for mining
- Mining class association rules
- Sequential pattern mining
- **Summary**

# Summary

- Association rule mining has been extensively studied in the data mining community.
- So is sequential pattern mining
- There are many efficient algorithms and model variations.
- Other related work includes
  - Multi-level or generalized rule mining
  - Constrained rule mining
  - Incremental rule mining
  - Maximal frequent itemset mining
  - Closed itemset mining
  - Rule interestingness and visualization
  - Parallel algorithms
  - ...

---

# Chapter 3: Supervised Learning

---

Dr. Mehmet S. Aktaş

Acknowledgement: Thanks to Dr. Bing Liu for teaching materials.

# Road Map

- **Basic concepts**
- Decision tree induction
- Classification using association rules
- Naïve Bayesian classification
- K-nearest neighbor
- Summary

# An example application

- An emergency room in a hospital measures 17 variables (e.g., blood pressure, age, etc) of newly admitted patients.
- **A decision is needed:** whether to put a new patient in an intensive-care unit.
- Due to the high cost of ICU, those patients who may survive less than a month are given higher priority.
- **Problem:** to predict **high-risk patients** and discriminate them from **low-risk patients**.

# Another application

- A credit card company receives thousands of applications for new cards. Each application contains information about an applicant,
  - age
  - Marital status
  - annual salary
  - outstanding debts
  - credit rating
  - etc.
- **Problem:** to decide whether an application should be approved, or to classify applications into two categories, **approved** and **not approved**.

# Machine learning and our focus

- Like human learning from past experiences.
- A computer does not have “experiences”.
- A computer system learns from data, which represent some “past experiences” of an application domain.
- Our focus: learn a target function that can be used to predict the values of a discrete class attribute, e.g., approve or not-approved, and high-risk or low risk.
- The task is commonly called: Supervised learning, classification, or inductive learning.

# The data and the goal

- **Data:** A set of data records (also called examples, instances or cases) described by
  - *k* attributes:  $A_1, A_2, \dots, A_k$ .
  - a class: Each example is labelled with a pre-defined class.
- **Goal:** To learn a classification model from the data that can be used to predict the classes of new (future, or test) cases/instances.

# An example: data (loan application)

Approved or not

ID	Age	Has_Job	Own_House	Credit_Rating	Class
1	young	false	false	fair	No
2	young	false	false	good	No
3	young	true	false	good	Yes
4	young	true	true	fair	Yes
5	young	false	false	fair	No
6	middle	false	false	fair	No
7	middle	false	false	good	No
8	middle	true	true	good	Yes
9	middle	false	true	excellent	Yes
10	middle	false	true	excellent	Yes
11	old	false	true	excellent	Yes
12	old	false	true	good	Yes
13	old	true	false	good	Yes
14	old	true	false	excellent	Yes
15	old	false	false	fair	No

# An example: the learning task

- Learn a classification model from the data
- Use the model to classify future loan applications into
  - Yes (approved) and
  - No (not approved)
- What is the class for following case/instance?

Age	Has_Job	Own_house	Credit-Rating	Class
young	false	false	good	?

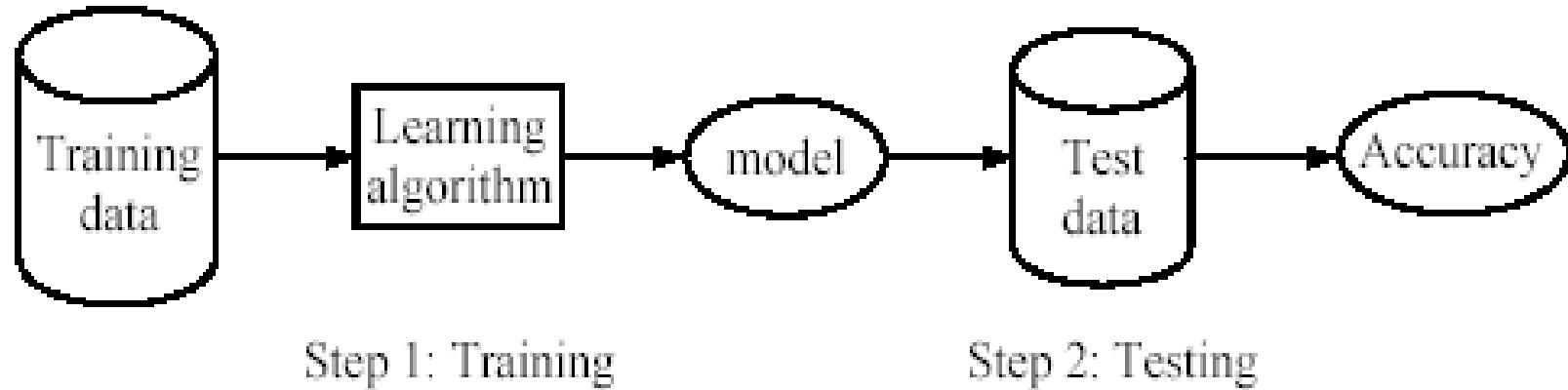
# Supervised vs. unsupervised Learning

- **Supervised learning:** classification is seen as supervised learning from examples.
  - **Supervision:** The data (observations, measurements, etc.) are labeled with pre-defined classes. It is like that a “teacher” gives the classes (**supervision**).
  - Test data are classified into these classes too.
- **Unsupervised learning (clustering)**
  - **Class labels of the data are unknown**
  - Given a set of data, the task is to establish the existence of classes or clusters in the data

# Supervised learning process: two steps

- **Learning (training):** Learn a model using the training data
- **Testing:** Test the model using **unseen test data** to assess the model accuracy

$$Accuracy = \frac{\text{Number of correct classifications}}{\text{Total number of test cases}},$$



# What do we mean by learning?

- Given
  - a data set  $D$ ,
  - a task  $T$ , and
  - a performance measure  $M$ ,
- a computer system is said to **learn** from  $D$  to perform the task  $T$  if after learning the system's performance on  $T$  improves as measured by  $M$ .
- In other words, the learned model helps the system to perform  $T$  better as compared to no learning.

# An example

- **Data:** Loan application data
- **Task:** Predict whether a loan should be approved or not.
- **Performance measure:** accuracy.

No learning: classify all future applications (test data) to the majority class (i.e., Yes):

$$\text{Accuracy} = 9/15 = 60\%.$$

- We can do better than 60% with learning.

# Fundamental assumption of learning

**Assumption:** The distribution of training examples is identical to the distribution of test examples (including future unseen examples).

- In practice, this assumption is often violated to certain degree.
- Strong violations will clearly result in poor classification accuracy.
- To achieve good accuracy on the test data, training examples must be sufficiently representative of the test data.

# Road Map

- Basic concepts
- **Decision tree induction**
- Classification using association rules
- Naïve Bayesian classification
- K-nearest neighbor
- Summary

# Introduction

- Decision tree learning is one of the most widely used techniques for classification.
  - Its classification accuracy is competitive with other methods, and
  - it is very efficient.
- The classification model is a tree, called **decision tree**.
- **C4.5** by Ross Quinlan is perhaps the best known system. It can be downloaded from the Web.

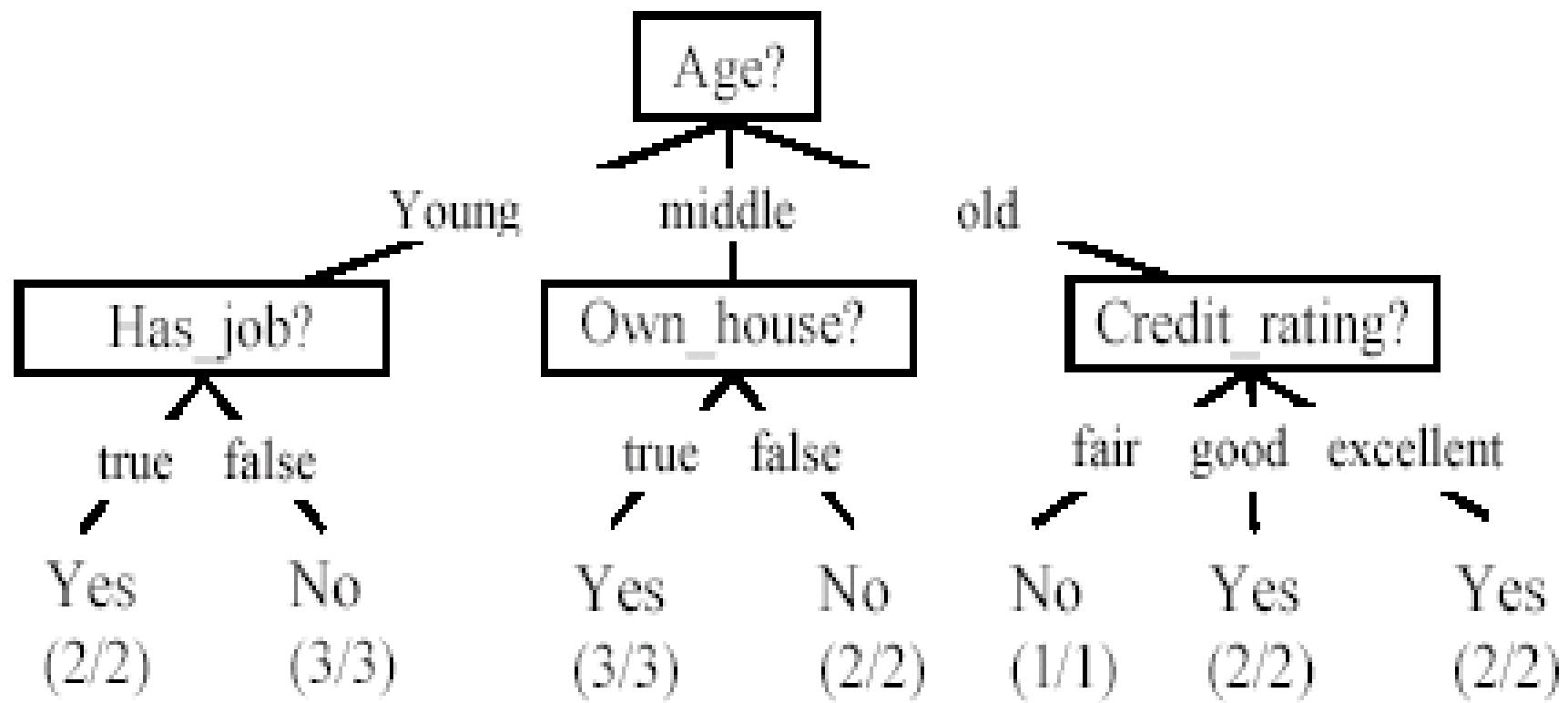
# The loan data (reproduced)

Approved or not

ID	Age	Has_Job	Own_House	Credit_Rating	Class
1	young	false	false	fair	No
2	young	false	false	good	No
3	young	true	false	good	Yes
4	young	true	true	fair	Yes
5	young	false	false	fair	No
6	middle	false	false	fair	No
7	middle	false	false	good	No
8	middle	true	true	good	Yes
9	middle	false	true	excellent	Yes
10	middle	false	true	excellent	Yes
11	old	false	true	excellent	Yes
12	old	false	true	good	Yes
13	old	true	false	good	Yes
14	old	true	false	excellent	Yes
15	old	false	false	fair	No

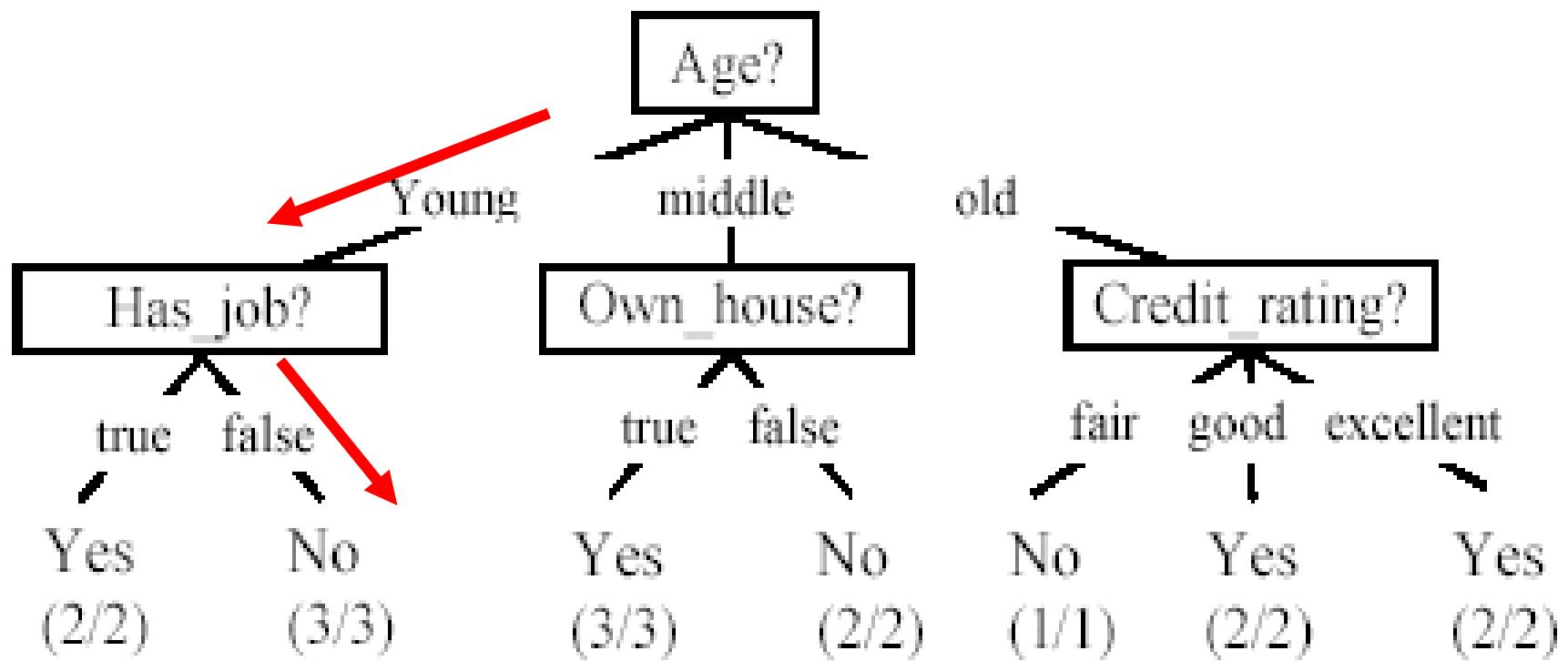
# A decision tree from the loan data

## ■ Decision nodes and leaf nodes (classes)



# Use the decision tree

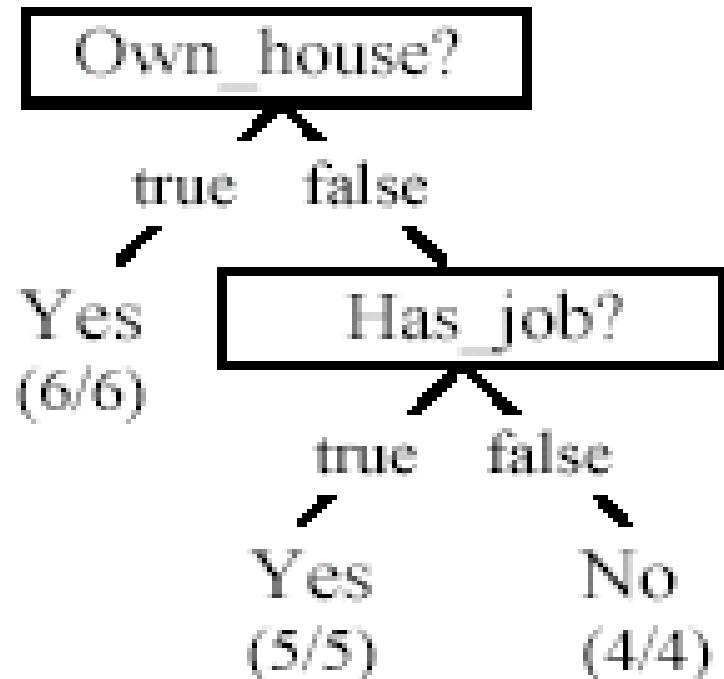
Age	Has_Job	Own_house	Credit-Rating	Class
young	false	false	good	? No



# Is the decision tree unique?

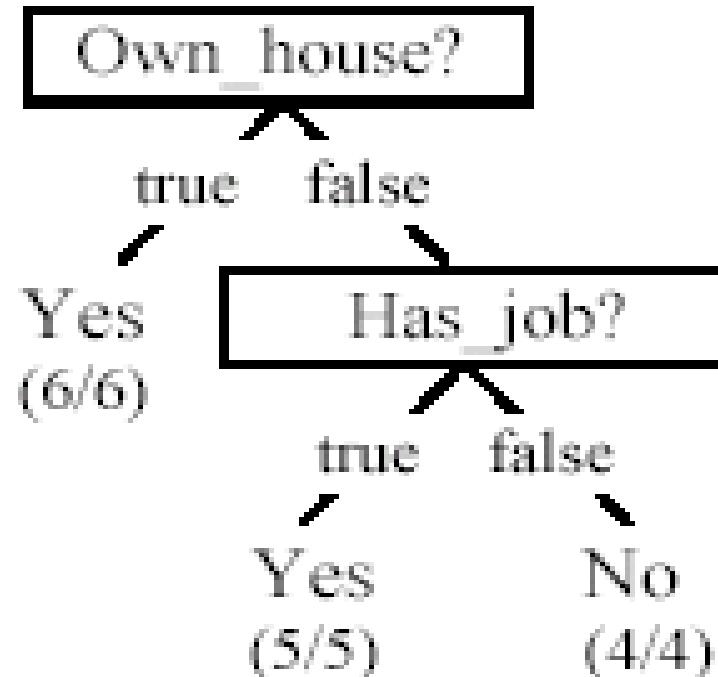
- No. Here is a simpler tree.
- We want smaller tree and accurate tree.
  - Easy to understand and perform better.

- Finding the best tree is NP-hard.
- All current tree building algorithms are heuristic algorithms



# From a decision tree to a set of rules

- A decision tree can be converted to a set of rules
- Each path from the root to a leaf is a rule.



$\text{Own\_house} = \text{true} \rightarrow \text{Class} = \text{Yes}$  [sup=6/15, conf=6/6]

$\text{Own\_house} = \text{false}, \text{Has\_job} = \text{true} \rightarrow \text{Class} = \text{Yes}$  [sup=5/15, conf=5/5]

$\text{Own\_house} = \text{false}, \text{Has\_job} = \text{false} \rightarrow \text{Class} = \text{No}$  [sup=4/15, conf=4/4]

# Algorithm for decision tree learning

- Basic algorithm (a greedy **divide-and-conquer** algorithm)
  - Assume attributes are categorical now (continuous attributes can be handled too)
  - Tree is constructed in a **top-down recursive manner**
  - At start, all the training examples are at the root
  - Examples are partitioned recursively based on selected attributes
  - Attributes are selected on the basis of an impurity function (e.g., **information gain**)
- Conditions for stopping partitioning
  - All examples for a given node belong to the same class
  - There are no remaining attributes for further partitioning – majority class is the leaf
  - There are no examples left

# Decision tree learning algorithm

```
. Algorithm decisionTree( $D, A, T$ )
1   if  $D$  contains only training examples of the same class  $c_j \in C$  then
2       make  $T$  a leaf node labeled with class  $c_j$ ;
3   elseif  $A = \emptyset$  then
4       make  $T$  a leaf node labeled with  $c_j$ , which is the most frequent class in  $D$ 
5   else //  $D$  contains examples belonging to a mixture of classes. We select a single
6       // attribute to partition  $D$  into subsets so that each subset is purer
7        $p_0 = \text{impurityEval-1}(D)$ ;
8       for each attribute  $A_i \in \{A_1, A_2, \dots, A_k\}$  do
9            $p_i = \text{impurityEval-2}(A_i, D)$ 
10      end
11      Select  $A_g \in \{A_1, A_2, \dots, A_k\}$  that gives the biggest impurity reduction,
12          computed using  $p_0 - p_g$ 
13      if  $p_0 - p_g < \text{threshold}$  then //  $A_g$  does not significantly reduce impurity  $p_0$ 
14          make  $T$  a leaf node labeled with  $c_j$ , the most frequent class in  $D$ .
15      else //  $A_g$  is able to reduce impurity  $p_0$ 
16          Make  $T$  a decision node on  $A_g$ ;
17          Let the possible values of  $A_g$  be  $v_1, v_2, \dots, v_m$ . Partition  $D$  into  $m$ 
18          disjoint subsets  $D_1, D_2, \dots, D_m$  based on the  $m$  values of  $A_g$ .
19          for each  $D_j$  in  $\{D_1, D_2, \dots, D_m\}$  do
20              if  $D_j \neq \emptyset$  then
21                  create a branch (edge) node  $T_j$  for  $v_j$  as a child node of  $T$ ;
22                  decisionTree( $D_j, A - \{A_g\}, T_j$ ) //  $A_g$  is removed
23              end
24          end
25      end
26  end
```

# Choose an attribute to partition data

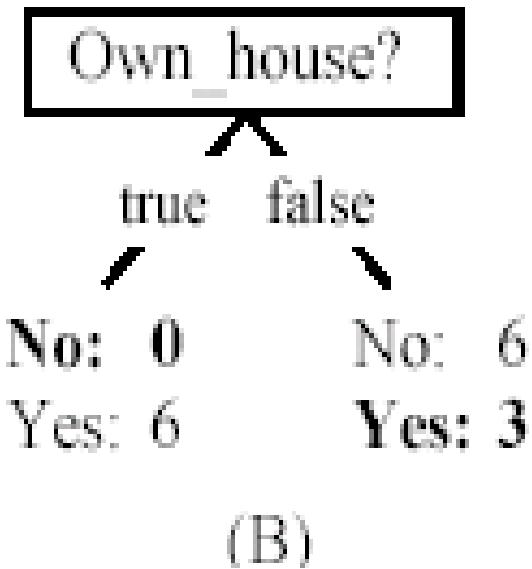
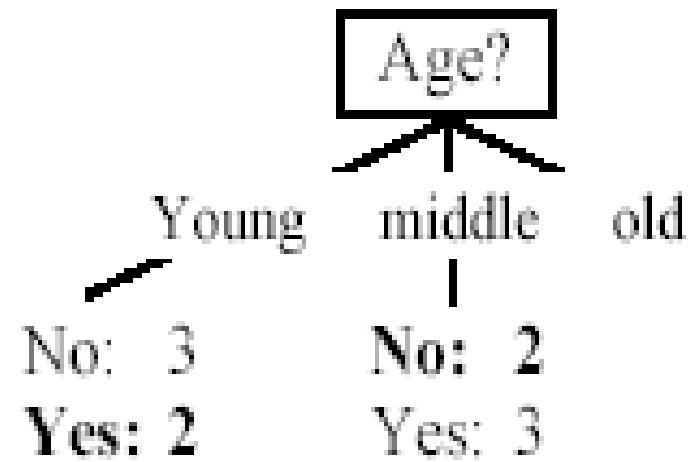
- The *key* to building a decision tree - which attribute to choose in order to branch.
- The objective is to reduce impurity or uncertainty in data as much as possible.
  - A subset of data is **pure** if all instances belong to the same class.
- The *heuristic* in C4.5 is to choose the attribute with the maximum **Information Gain** or **Gain Ratio** based on information theory.

# The loan data (reproduced)

Approved or not

ID	Age	Has_Job	Own_House	Credit_Rating	Class
1	young	false	false	fair	No
2	young	false	false	good	No
3	young	true	false	good	Yes
4	young	true	true	fair	Yes
5	young	false	false	fair	No
6	middle	false	false	fair	No
7	middle	false	false	good	No
8	middle	true	true	good	Yes
9	middle	false	true	excellent	Yes
10	middle	false	true	excellent	Yes
11	old	false	true	excellent	Yes
12	old	false	true	good	Yes
13	old	true	false	good	Yes
14	old	true	false	excellent	Yes
15	old	false	false	fair	No

# Two possible roots, which is better?



- Fig. (B) seems to be better.

# Road Map

- Basic concepts
- Decision tree induction
- **Classification using association rules**
- Naïve Bayesian classification
- K-nearest neighbor
- Summary

# Using Class Association Rules

- **Classification:** mine a small set of rules existing in the data to form a classifier or predictor.
  - It has a target attribute: **Class attribute**
- **Association rules:** have no fixed target, but we can fix a target.
- **Class association rules (CAR):** has a target class attribute. E.g.,
  - Own\_house = true → Class =Yes [sup=6/15, conf=6/6]
  - CARs can obviously be used for classification.

# Decision tree vs. CARs

- The decision tree below generates the following 3 rules.

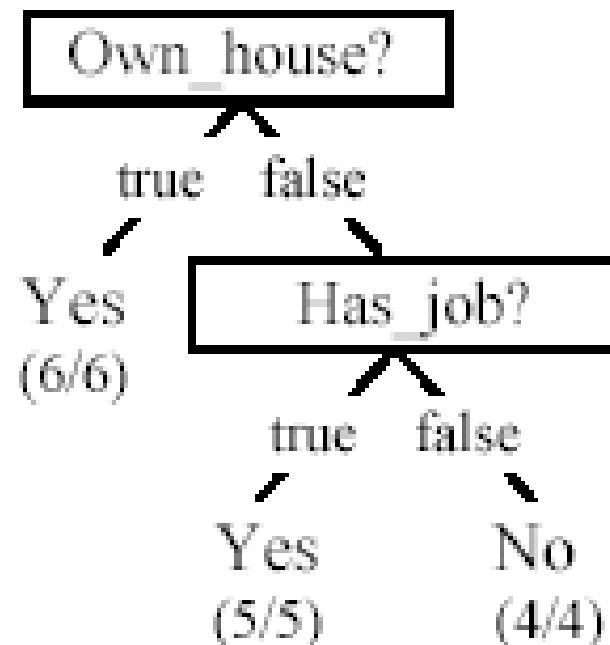
Own\_house = true → Class = Yes

[sup=6/15, conf=6/6]

Own\_house = false, Has\_job = true → Class = Yes [sup=5/15, conf=5/5]

Own\_house = false, Has\_job = false → Class = No [sup=4/15, conf=4/4]

- But there are many other rules that are not found by the decision tree



# There are many more rules

Age = young, Has_job = true → Class=Yes	[sup=2/15, conf=2/2]
Age = young, Has_job = false → Class=No	[sup=3/15, conf=3/3]
Credit_Rating = fair → Class=No	[sup=4/15, conf=4/4]
Credit_Rating = good → Class=Yes	[sup=5/15, conf=5/6]

and many more, if we use  $\text{minsup} = 2/15 = 13.3\%$  and  $\text{minconf} = 80\%$ .

- CAR mining finds all of them.
- In many cases, rules not in the decision tree (or a rule list) may perform classification better.
- Such rules may also be actionable in practice

ID	Age	Has_Job	Own_House	Credit_Rating	Class
1	young	false	false	fair	No
2	young	false	false	excellent	No
3	young	true	false	good	Yes
4	young	true	true	good	Yes
5	young	false	false	fair	No
6	middle	false	false	fair	No
7	middle	false	false	good	No
8	middle	true	true	good	Yes
9	middle	false	true	excellent	Yes
10	middle	false	true	excellent	Yes
11	old	false	true	excellent	Yes
12	old	false	true	good	Yes
13	old	true	false	good	Yes
14	old	true	false	excellent	Yes
15	old	false	false	fair	No

## Decision tree vs. CARs (cont ...)

- Association mining require discrete attributes. Decision tree learning uses both discrete and continuous attributes.
- Decision tree is not constrained by minsup or minconf, and thus is able to find rules with very low support. Of course, such rules may be pruned due to the possible overfitting.

# Building classifiers

- There are many ways to build classifiers using CARs. Several existing systems available.
- Strongest rules: After CARs are mined, do nothing.
  - For each test case, we simply choose the most confident rule that covers the test case to classify it. Microsoft SQL Server has a similar method.
  - Or, using a combination of rules.
- Selecting a subset of Rules
  - used in the CBA system.
  - similar to sequential covering.

## CBA: Rules are sorted first

**Definition:** Given two rules,  $r_i$  and  $r_j$ ,  $r_i \succ r_j$  (also called  $r_i$  precedes  $r_j$  or  $r_i$  has a higher precedence than  $r_j$ ) if

- the confidence of  $r_i$  is greater than that of  $r_j$ , or
- their confidences are the same, but the support of  $r_i$  is greater than that of  $r_j$ , or
- both the confidences and supports of  $r_i$  and  $r_j$  are the same, but  $r_i$  is generated earlier than  $r_j$ .

A CBA classifier  $L$  is of the form:

$$L = \langle r_1, r_2, \dots, r_k, \text{default-class} \rangle$$

# Classifier building using CARs

**Algorithm CBA( $S, D$ )**

```
1  $S = \text{sort}(S)$ ;                                // sorting is done according to the precedence  $\succ$ 
2  $\text{RuleList} = \emptyset$ ;                            // the rule list classifier
3 for each rule  $r \in S$  in sequence do
4     if  $D \neq \emptyset$  AND  $r$  classifies at least one example in  $D$  correctly then
5         delete from  $D$  all training examples covered by  $r$ ;
6         add  $r$  at the end of  $\text{RuleList}$ 
7     end
8 end
9 add the majority class as the default class at the end of  $\text{RuleList}$ 
```

- This algorithm is very inefficient
- CBA has a very efficient algorithm (quite sophisticated) that scans the data at most two times.

# Using normal association rules for classification

- A widely used approach
- Main approach: strongest rules
- Main application
  - Recommendation systems in e-commerce Web site (e.g., amazon.com).
  - Each rule consequent is the recommended item.
- Major advantage: any item can be predicted.
- Main issue:
  - Coverage: rare item rules are not found using classic algo.

# Road Map

- Basic concepts
- Decision tree induction
- Classification using association rules
- **Naïve Bayesian classification**
- K-nearest neighbor
- Summary

# Naïve Bayesian Classifier: Training Dataset

■ Class:

C1:buys\_computer = ‘yes’

C2:buys\_computer = ‘no’

■ Data sample:

X = (age <=30,

Income = medium,

Student = yes,

Credit\_rating = Fair)

age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31...40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31...40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31...40	medium	no	excellent	yes
31...40	high	yes	fair	yes
>40	medium	no	excellent	no

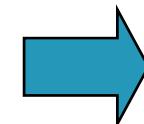
# Bayesian Theorem: Basics

- Let  $\mathbf{X}$  be a data sample (“evidence”): class label is unknown
- Let  $H$  be a *hypothesis* that  $\mathbf{X}$  belongs to class C
- Classification is to determine  $P(H|\mathbf{X})$ , the probability that the hypothesis holds given the observed data sample  $\mathbf{X}$ 
  - $P(H)$  (*prior probability*), the initial probability
    - E.g., **Any given tuple** will buy computer, regardless of age, income, ...
  - $P(\mathbf{X})$ : probability that sample data is observed
  - $P(\mathbf{X}|H)$  (*posteriori probability*), the probability of observing the sample  $\mathbf{X}$ , given that the hypothesis holds
    - E.g., Given that  $\mathbf{X}$  will buy computer, the prob. that  $\mathbf{X}$  is 31..40, medium income

# Bayesian Theorem

- Given training data  $\mathbf{X}$ , *posteriori probability of a hypothesis*  $H$ ,  $P(H|\mathbf{X})$ , follows the Bayes theorem

$$P(H|\mathbf{X}) = \frac{P(\mathbf{X}|H)P(H)}{P(\mathbf{X})}$$



$$P(C_1|\mathbf{X}) = \frac{P(\mathbf{X}|C_1)P(C_1)}{P(\mathbf{X})}$$

- Informally, this can be written as  
posteriori = likelihood x prior/evidence

$$P(C_n|\mathbf{X}) = \frac{P(\mathbf{X}|C_n)P(C_n)}{P(\mathbf{X})}$$

- Predicts  $\mathbf{X}$  belongs to  $C_i$  iff the probability  $P(C_i|\mathbf{X})$  is the highest among all the  $P(C_k|\mathbf{X})$  for all the  $k$  classes
- Practical difficulty: require initial knowledge of many probabilities, significant computational cost

# Towards Naïve Bayesian Classifier

- Let  $D$  be a training set of tuples and their associated class labels, and each tuple is represented by an  $n$ -D attribute vector  $\mathbf{X} = (x_1, x_2, \dots, x_n)$
- Suppose there are  $m$  classes  $C_1, C_2, \dots, C_m$ .
- Classification is to derive the maximum posteriori, i.e., the maximal  $P(C_i|\mathbf{X})$
- This can be derived from Bayes' theorem

$$P(C_i|\mathbf{X}) = \frac{P(\mathbf{X}|C_i)P(C_i)}{P(\mathbf{X})}$$

- Since  $P(\mathbf{X})$  is constant for all classes, only

$$P(C_i|\mathbf{X}) = P(\mathbf{X}|C_i)P(C_i)$$

needs to be maximized

# Derivation of Naïve Bayes Classifier

- A simplified assumption: attributes are conditionally independent (i.e., no dependence relation between attributes):

$$P(\mathbf{X} | C_i) = \prod_{k=1}^n P(x_k | C_i) = P(x_1 | C_i) \times P(x_2 | C_i) \times \dots \times P(x_n | C_i)$$

- This greatly reduces the computation cost: Only counts the class distribution

# Naïve Bayesian Classifier: Training Dataset

■ Class:

C1:buys\_computer = ‘yes’

C2:buys\_computer = ‘no’

■ Data sample

X = (age <=30,

Income = medium,

Student = yes,

Credit\_rating = Fair)

age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31...40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31...40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31...40	medium	no	excellent	yes
31...40	high	yes	fair	yes
>40	medium	no	excellent	no

# Naïve Bayesian Classifier: An Example

- $P(C_i)$ :  $P(\text{buys\_computer} = \text{"yes"}) = 9/14 = 0.643$   
 $P(\text{buys\_computer} = \text{"no"}) = 5/14 = 0.357$

- Compute  $P(X|C_i)$  for each class

$$P(\text{age} = \text{"<=30"} | \text{buys\_computer} = \text{"yes"}) = 2/9 = 0.222$$

$$P(\text{income} = \text{"medium"} | \text{buys\_computer} = \text{"yes"}) = 4/9 = 0.444$$

$$P(\text{student} = \text{"yes"} | \text{buys\_computer} = \text{"yes"}) = 6/9 = 0.667$$

$$P(\text{credit\_rating} = \text{"fair"} | \text{buys\_computer} = \text{"yes"}) = 6/9 = 0.667$$

$$P(\text{age} = \text{"<= 30"} | \text{buys\_computer} = \text{"no"}) = 3/5 = 0.6$$

$$P(\text{income} = \text{"medium"} | \text{buys\_computer} = \text{"no"}) = 2/5 = 0.4$$

$$P(\text{student} = \text{"yes"} | \text{buys\_computer} = \text{"no"}) = 1/5 = 0.2$$

$$P(\text{credit\_rating} = \text{"fair"} | \text{buys\_computer} = \text{"no"}) = 2/5 = 0.4$$

- $X = (\text{age} \leq 30, \text{income} = \text{medium}, \text{student} = \text{yes}, \text{credit\_rating} = \text{fair})$

$$P(X|C_i) : P(X|\text{buys\_computer} = \text{"yes"}) = 0.222 \times 0.444 \times 0.667 \times 0.667 = 0.044$$

$$P(X|\text{buys\_computer} = \text{"no"}) = 0.6 \times 0.4 \times 0.2 \times 0.4 = 0.019$$

$$P(X|C_i) * P(C_i) : P(X|\text{buys\_computer} = \text{"yes"}) * P(\text{buys\_computer} = \text{"yes"}) = 0.028$$

$$P(X|\text{buys\_computer} = \text{"no"}) * P(\text{buys\_computer} = \text{"no"}) = 0.007$$

Therefore,  $X$  belongs to class ("buys\_computer = yes")

# On naïve Bayesian classifier

## ■ Advantages:

- Easy to implement
- Very efficient
- Good results obtained in many applications

## ■ Disadvantages

- Assumption: class conditional independence, therefore loss of accuracy when the assumption is seriously violated (those highly correlated data sets)

# Road Map

- Basic concepts
- Decision tree induction
- Classification using association rules
- Naïve Bayesian classification
- **K-nearest neighbor**
- Summary

# k-Nearest Neighbor Classification (kNN)

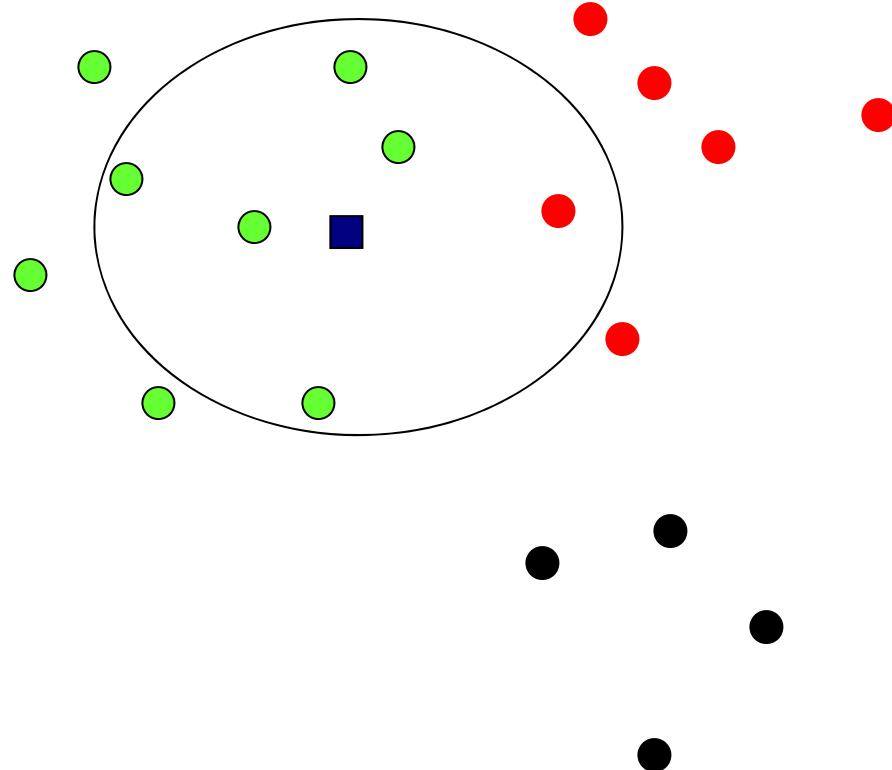
- Unlike all the previous learning methods, kNN does not build model from the training data.
- To classify a test instance  $d$ , define  $k$ -neighborhood  $P$  as  $k$  nearest neighbors of  $d$
- Count number  $n$  of training instances in  $P$  that belong to class  $c_j$
- Estimate  $\Pr(c_j|d)$  as  $n/k$
- No training is needed. Classification time is linear in training set size for each test case.

# kNNAlgorithm

Algorithm  $\text{kNN}(D, d, k)$

- 1 Compute the distance between  $d$  and every example in  $D$ ;
  - 2 Choose the  $k$  examples in  $D$  that are nearest to  $d$ , denote the set by  $P$  ( $\subseteq D$ );
  - 3 Assign  $d$  the class that is the most frequent class in  $P$  (or the majority class);
- 
- $k$  is usually chosen empirically via a validation set or cross-validation by trying a range of  $k$  values.
  - Distance function is crucial, but depends on applications.

## Example: k=6 (6NN)



- Government
- Science
- Arts

A new point  
 $\Pr(\text{science} | \square)$ ?

# Discussions

- kNN can deal with complex and arbitrary decision boundaries.
- Despite its simplicity, researchers have shown that the classification accuracy of kNN can be quite strong and in many cases as accurate as those elaborated methods.
- kNN is slow at the classification time
- kNN does not produce an understandable model

# Road Map

- Basic concepts
- Decision tree induction
- Classification using association rules
- Naïve Bayesian classification
- K-nearest neighbor
- **Summary**

# Summary

- Applications of supervised learning are in almost any field or domain.
- We studied some classification techniques.
- There are still many other methods, e.g.,
  - Bayesian networks
  - Neural networks
  - Genetic algorithms
  - Fuzzy classification

This large number of methods also show the importance of classification and its wide applicability.

- It remains to be an active research area.

# Chapter 4: Unsupervised Learning

---

Dr. Mehmet S. Aktaş

Acknowledgement: Thanks to Dr. Bing Liu for teaching materials.

# Road map

- **Basic concepts**
- K-means algorithm
- Representation of clusters
- Hierarchical clustering
- Distance functions
- Which clustering algorithm to use?
- Cluster evaluation
- Summary

# Supervised learning vs. unsupervised learning

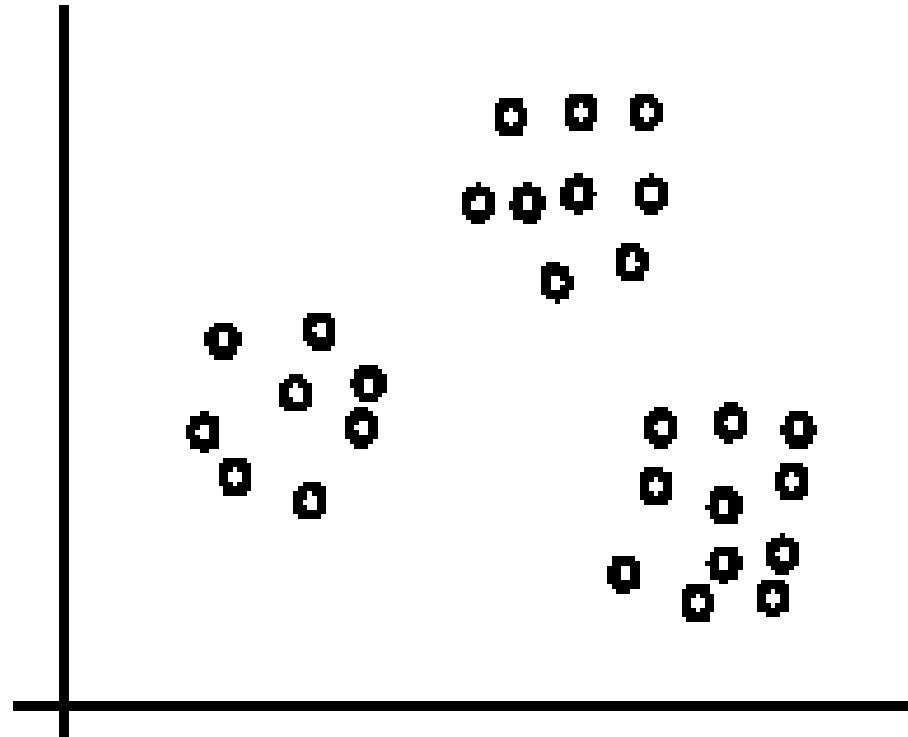
- **Supervised learning:** discover patterns in the data that relate data attributes with a target (class) attribute.
  - These patterns are then utilized to predict the values of the target attribute in future data instances.
- **Unsupervised learning:** The data have no target attribute.
  - We want to explore the data to find some intrinsic structures in them.

# Clustering

- Clustering is a technique for finding **similarity groups** in data, called **clusters**. I.e.,
  - it groups data instances that are similar to (near) each other in one cluster and data instances that are very different (far away) from each other into different clusters.
- Clustering is often called an **unsupervised learning** task as no class values denoting an *a priori* grouping of the data instances are given, which is the case in supervised learning.
- Due to historical reasons, clustering is often considered synonymous with unsupervised learning.
  - In fact, association rule mining is also unsupervised
- This chapter focuses on clustering.

# An illustration

- The data set has three natural groups of data points, i.e., 3 natural clusters.



# What is clustering for?

- Let us see some real-life examples
- Example 1: groups people of similar sizes together to make “small”, “medium” and “large” T-Shirts.
  - Tailor-made for each person: too expensive
  - One-size-fits-all: does not fit all.
- Example 2: In marketing, segment customers according to their similarities
  - To do targeted marketing.

# What is clustering for? (cont...)

- **Example 3:** Given a collection of text documents, we want to organize them according to their content similarities,
  - To produce a topic hierarchy
- **In fact, clustering is one of the most utilized data mining techniques.**
  - It has a long history, and used in almost every field, e.g., medicine, psychology, botany, sociology, biology, archeology, marketing, insurance, libraries, etc.
  - In recent years, due to the rapid increase of online documents, text clustering becomes important.

# Aspects of clustering

- A clustering algorithm
  - Partitional clustering
  - Hierarchical clustering
  - ...
- A distance (similarity, or dissimilarity) function
- Clustering quality
  - Inter-clusters distance  $\Rightarrow$  maximized
  - Intra-clusters distance  $\Rightarrow$  minimized
- The **quality** of a clustering result depends on the algorithm, the distance function, and the application.

# Road map

- Basic concepts
- **K-means algorithm**
- Representation of clusters
- Hierarchical clustering
- Distance functions
- Which clustering algorithm to use?
- Cluster evaluation
- Summary

# K-means clustering

- K-means is a **partitional clustering** algorithm
- Let the set of data points (or instances)  $D$  be
  - $\{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n\}$ ,  
where  $\mathbf{x}_i = (x_{i1}, x_{i2}, \dots, x_{ir})$  is a **vector** in a real-valued space  $X \subseteq R^r$ , and  $r$  is the number of attributes (dimensions) in the data.
- The  $k$ -means algorithm partitions the given data into  $k$  clusters.
  - Each cluster has a cluster **center**, called **centroid**.
  - $k$  is specified by the user

# K-means algorithm

- Given  $k$ , the *k-means* algorithm works as follows:
  - 1) Randomly choose  $k$  data points (**seeds**) to be the initial **centroids**, cluster centers
  - 2) Assign each data point to the closest **centroid**
  - 3) Re-compute the **centroids** using the current cluster memberships.
  - 4) If a convergence criterion is not met, go to 2).

# K-means algorithm – (cont ...)

**Algorithm**  $k$ -means( $k, D$ )

- 1 Choose  $k$  data points as the initial centroids (cluster centers)
- 2 **repeat**
- 3     **for** each data point  $\mathbf{x} \in D$  **do**
- 4         compute the distance from  $\mathbf{x}$  to each centroid;
- 5         assign  $\mathbf{x}$  to the closest centroid           // a centroid represents a cluster
- 6     **endfor**
- 7     re-compute the centroids using the current cluster memberships
- 8 **until** the stopping criterion is met

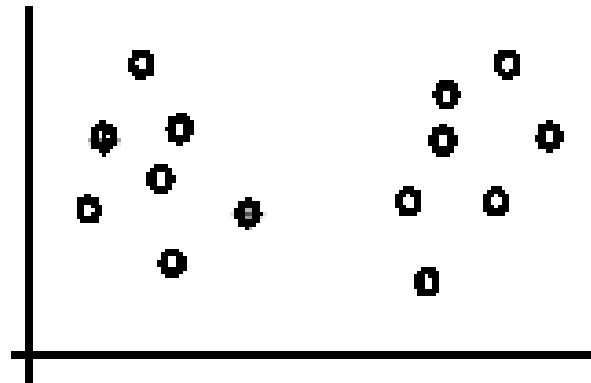
# Stopping/convergence criterion

1. no (or minimum) re-assignments of data points to different clusters,
2. no (or minimum) change of centroids, or
3. minimum decrease in the **sum of squared error** (SSE),

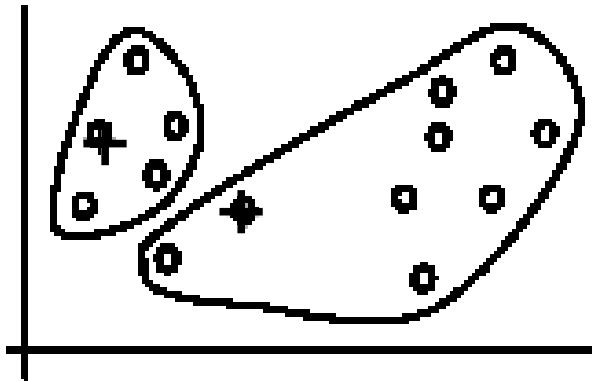
$$SSE = \sum_{j=1}^k \sum_{\mathbf{x} \in C_j} dist(\mathbf{x}, \mathbf{m}_j)^2 \quad (1)$$

- $C_j$  is the  $j$ th cluster,  $\mathbf{m}_j$  is the centroid of cluster  $C_j$  (the mean vector of all the data points in  $C_j$ ), and  $dist(\mathbf{x}, \mathbf{m}_j)$  is the distance between data point  $\mathbf{x}$  and centroid  $\mathbf{m}_j$ .

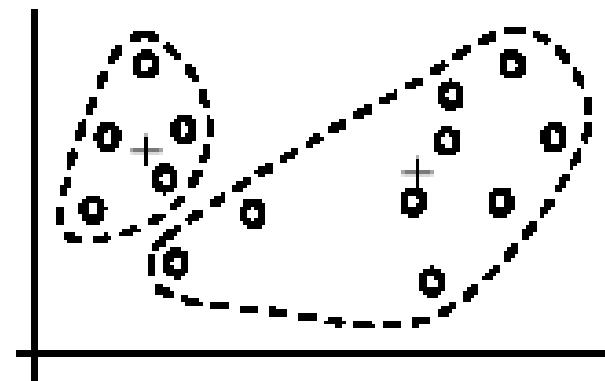
# An example



(A). Random selection of  $k$  centers

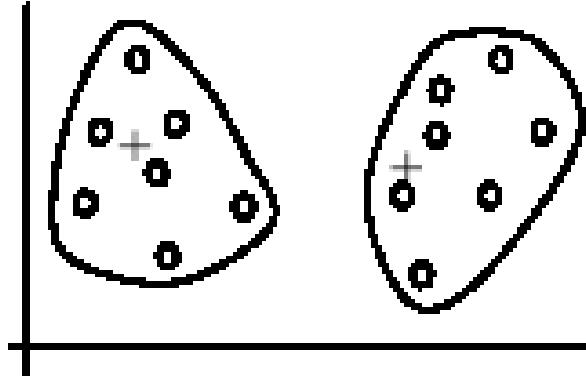


Iteration 1: (B). Cluster assignment

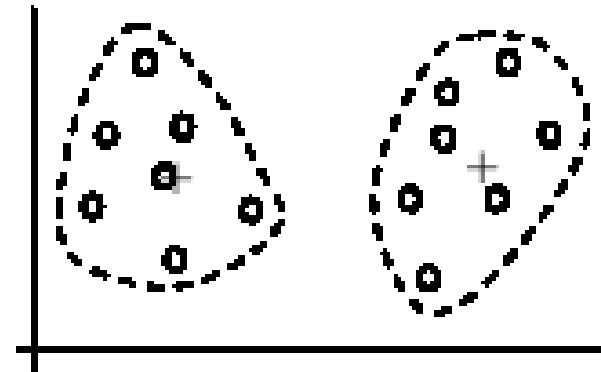


(C). Re-compute centroids

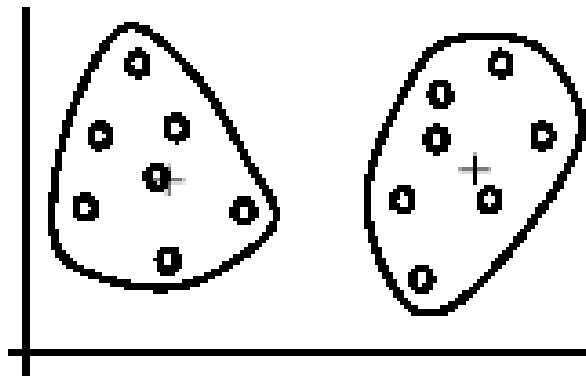
# An example (cont ...)



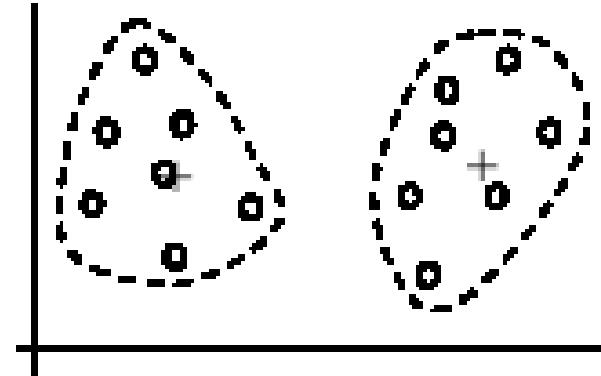
Iteration 2: (D). Cluster assignment



(E). Re-compute centroids



Iteration 3: (F). Cluster assignment



(G). Re-compute centroids

# An example distance function

The  $k$ -means algorithm can be used for any application data set where the mean can be defined and computed. In the Euclidean space, the mean of a cluster is computed with:

$$\mathbf{m}_j = \frac{1}{|C_j|} \sum_{\mathbf{x}_i \in C_j} \mathbf{x}_i \quad (2)$$

where  $|C_j|$  is the number of data points in cluster  $C_j$ . The distance from one data point  $\mathbf{x}_i$  to a mean (centroid)  $\mathbf{m}_j$  is computed with

$$\begin{aligned} dist(\mathbf{x}_i, \mathbf{m}_j) &= \| \mathbf{x}_i - \mathbf{m}_j \| \\ &= \sqrt{(x_{i1} - m_{j1})^2 + (x_{i2} - m_{j2})^2 + \dots + (x_{ir} - m_{jr})^2} \end{aligned} \quad (3)$$

# A disk version of $k$ -means

- K-means can be implemented with data on disk
  - In each iteration, it scans the data once.
  - as the centroids can be computed incrementally
- It can be used to cluster large datasets that do not fit in main memory
- We need to control the number of iterations
  - In practice, a limit is set (< 50).
- Not the best method. There are other scale-up algorithms, e.g., BIRCH.

# A disk version of k-means (cont ...)

**Algorithm** disk- $k$ -means( $k, D$ )

```
1  Choose  $k$  data points as the initial centroids  $\mathbf{m}_j, j = 1, \dots, k$ ,  
2  repeat  
3      initialize  $\mathbf{s}_j = \mathbf{0}, j = 1, \dots, k$ ;           //  $\mathbf{0}$  is a vector with all 0's  
4      initialize  $n_j = 0, j = 1, \dots, k$ ;           //  $n_j$  is the number points in cluster  $j$   
5      for each data point  $\mathbf{x} \in D$  do  
6           $j = \arg \min_j dist(\mathbf{x}, \mathbf{m}_j);$   
7          assign  $\mathbf{x}$  to the cluster  $j$ ;  
8           $\mathbf{s}_j = \mathbf{s}_j + \mathbf{x};$   
9           $n_j = n_j + 1;$   
10     endfor  
11      $\mathbf{m}_i = \mathbf{s}_j/n_j, i = 1, \dots, k$ ,  
12 until the stopping criterion is met
```

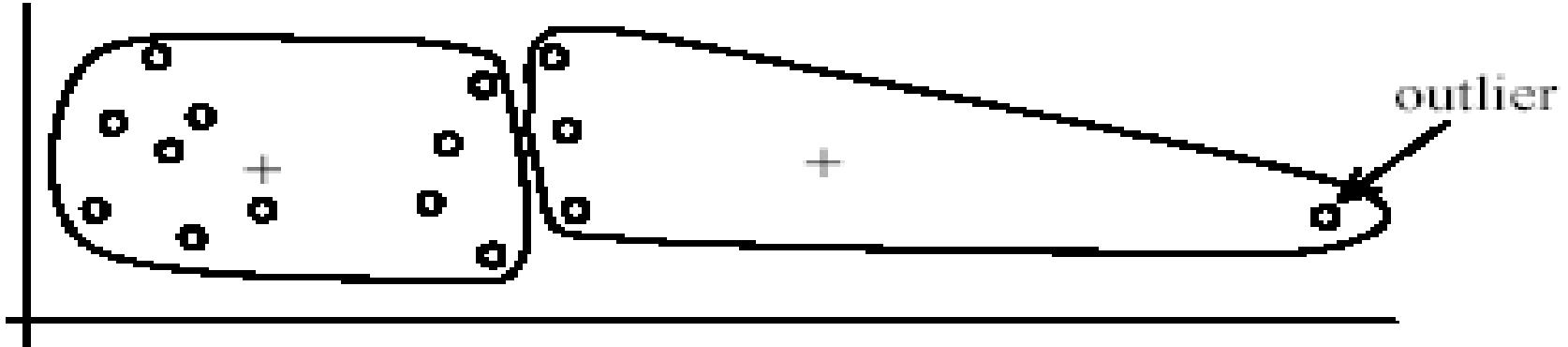
# Strengths of k-means

- Strengths:
  - Simple: easy to understand and to implement
  - Efficient: Time complexity:  $O(tkn)$ ,  
where  $n$  is the number of data points,  
 $k$  is the number of clusters, and  
 $t$  is the number of iterations.
  - Since both  $k$  and  $t$  are small. k-means is considered a linear algorithm.
- K-means is the most popular clustering algorithm.

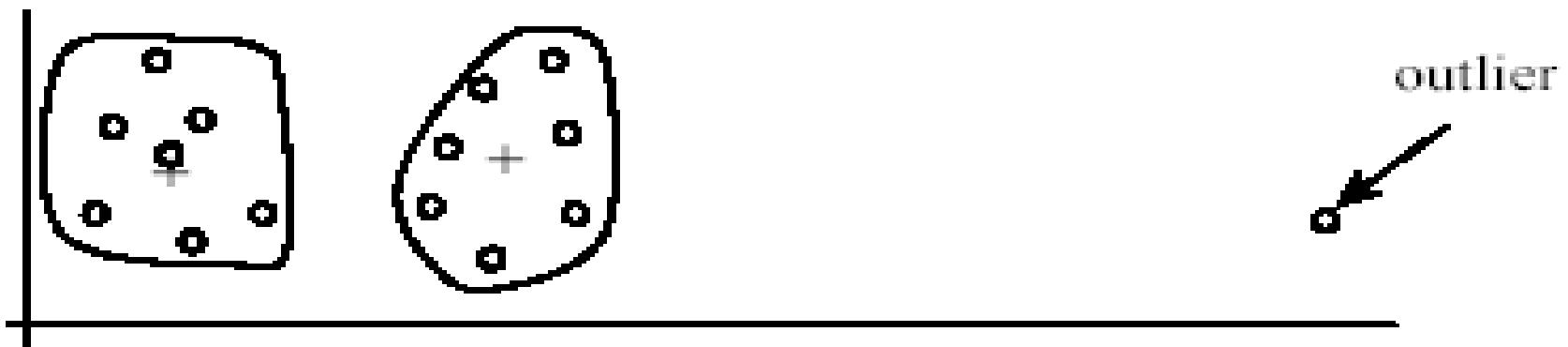
# Weaknesses of k-means

- The algorithm is only applicable if the **mean** is defined.
  - For categorical data, *k*-mode - the centroid is represented by most frequent values.
- The user needs to specify ***k***.
- The algorithm is sensitive to **outliers**
  - Outliers are data points that are very far away from other data points.
  - Outliers could be errors in the data recording or some special data points with very different values.

# Weaknesses of k-means: Problems with outliers



(A): Undesirable clusters



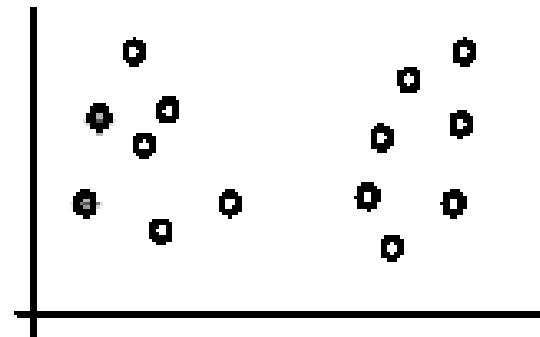
(B): Ideal clusters

# Weaknesses of k-means: To deal with outliers

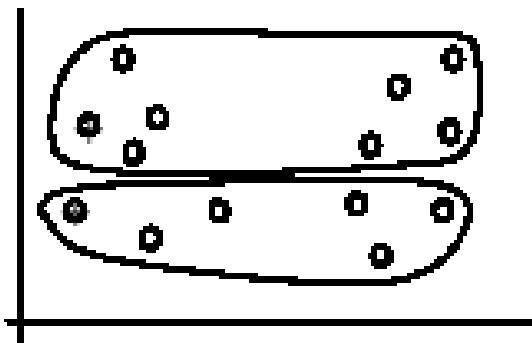
- One method is to remove some data points in the clustering process that are much further away from the centroids than other data points.
  - To be safe, we may want to monitor these possible outliers over a few iterations and then decide to remove them.
- Another method is to perform random sampling. Since in sampling we only choose a small subset of the data points, the chance of selecting an outlier is very small.
  - Assign the rest of the data points to the clusters by distance or similarity comparison, or classification

# Weaknesses of k-means (cont ...)

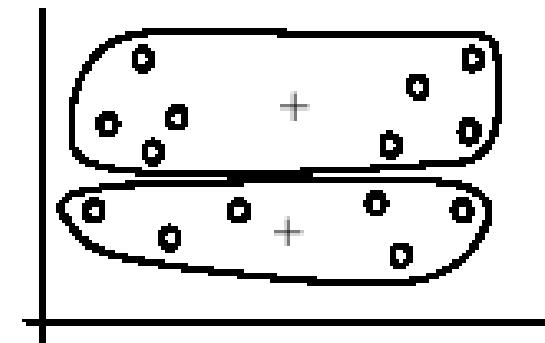
- The algorithm is sensitive to **initial seeds**.



(A). Random selection of seeds (centroids)



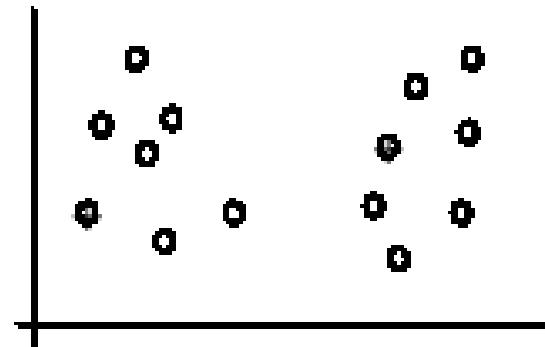
(B). Iteration 1



(C). Iteration 2

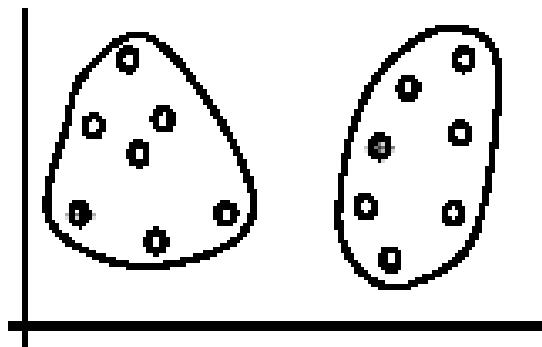
# Weaknesses of k-means (cont ...)

- If we use **different seeds**: good results

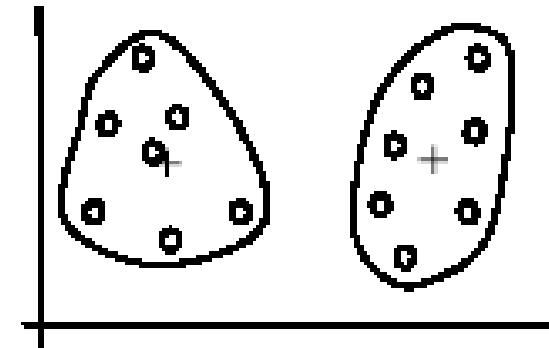


- There are some methods to help choose good seeds

(A). Random selection of  $k$  seeds (centroids)



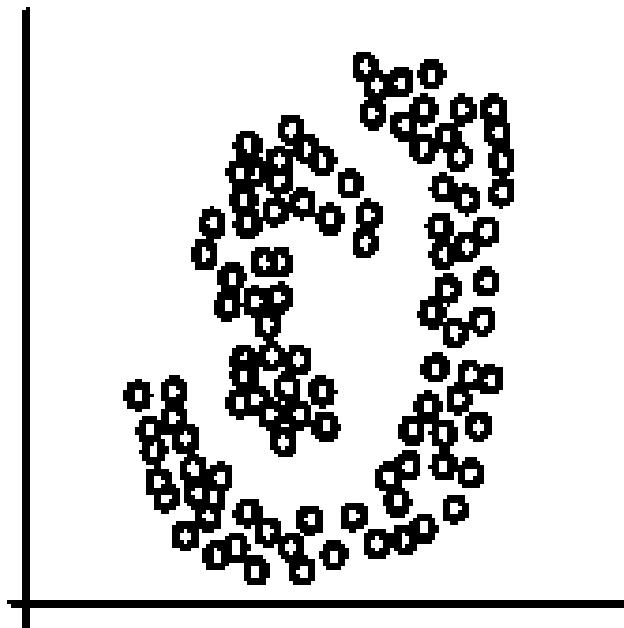
(B). Iteration 1



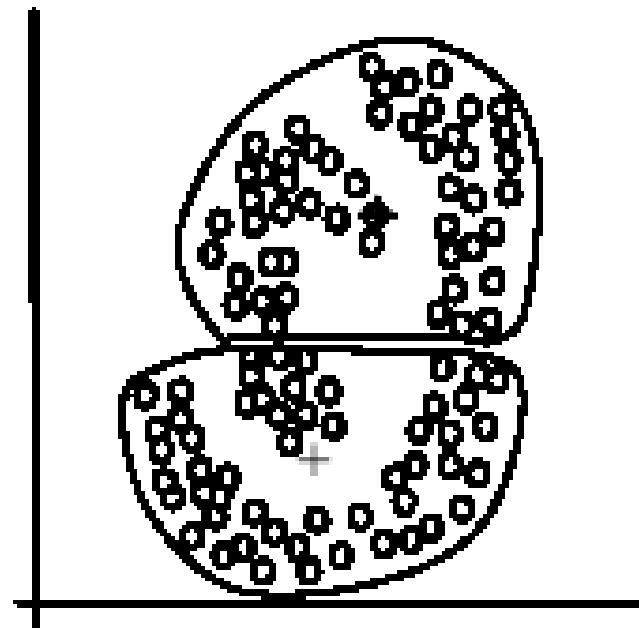
(C). Iteration 2

# Weaknesses of k-means (cont ...)

- The  $k$ -means algorithm is not suitable for discovering clusters that are not hyper-ellipsoids (or hyper-spheres).



(A): Two natural clusters



(B):  $k$ -means clusters

# K-means summary

- Despite weaknesses, *k*-means is still the most popular algorithm due to its simplicity, efficiency and
  - other clustering algorithms have their own lists of weaknesses.
- No clear evidence that any other clustering algorithm performs better in general
  - although they may be more suitable for some specific types of data or applications.

# Road map

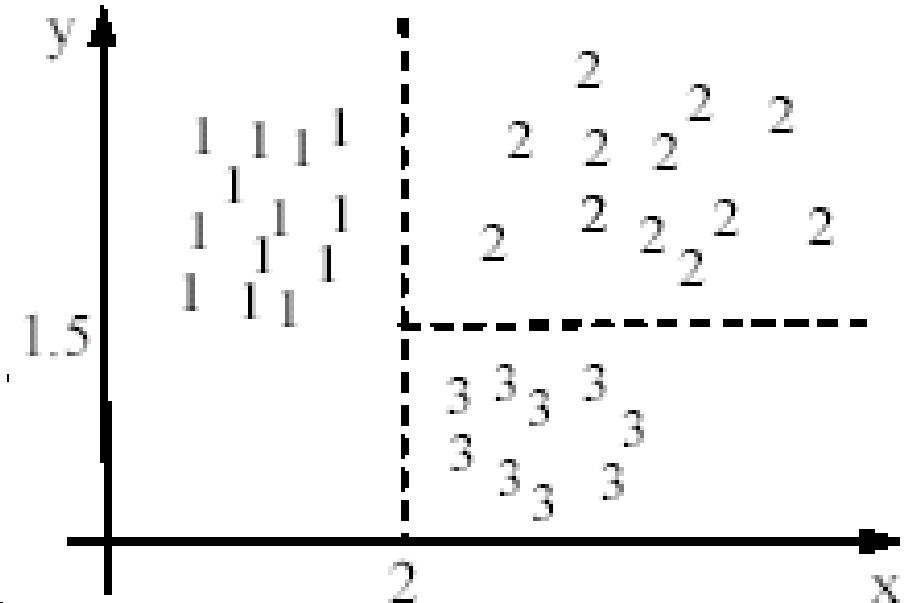
- Basic concepts
- K-means algorithm
- **Representation of clusters**
- Hierarchical clustering
- Distance functions
- Which clustering algorithm to use?
- Cluster evaluation
- Summary

# Common ways to represent clusters

- Use the centroid of each cluster to represent the cluster.
  - compute the radius and
  - standard deviation of the cluster to determine its spread in each dimension
- The centroid representation alone works well if the clusters are of the hyper-spherical shape.
- If clusters are elongated or are of other shapes, centroids are not sufficient

# Using classification model

- All the data points in a cluster are regarded to have the same class label, e.g., the cluster ID.
  - run a supervised learning algorithm on the data to find a classification model.



$x \leq 2 \rightarrow$  cluster 1

$x > 2, y > 1.5 \rightarrow$  cluster 2

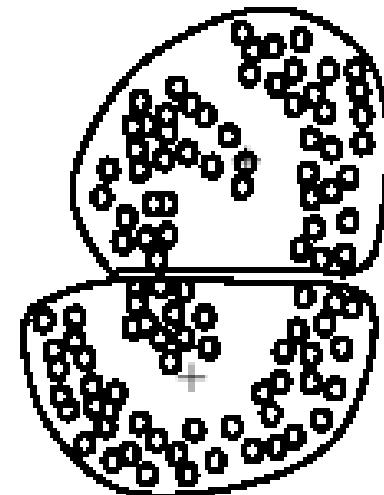
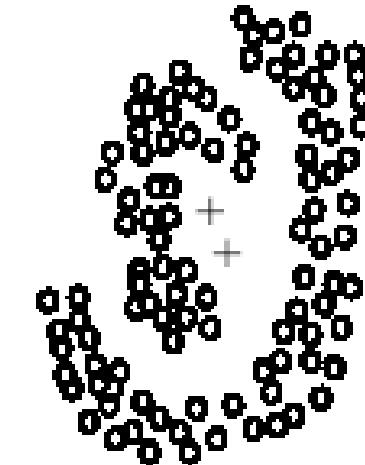
$x > 2, y \leq 1.5 \rightarrow$  cluster 3

# Use frequent values to represent cluster

- This method is mainly for clustering of categorical data (e.g.,  $k$ -modes clustering).
- Main method used in text clustering, where a small set of frequent words in each cluster is selected to represent the cluster.

# Clusters of arbitrary shapes

- Hyper-elliptical and hyper-spherical clusters are usually easy to represent, using their centroid together with spreads.
- **Irregular shape clusters are hard to represent.** They may not be useful in some applications.
  - Using centroids are not suitable (upper figure) in general
  - K-means clusters may be more useful (lower figure), e.g., for making 2 size T-shirts.

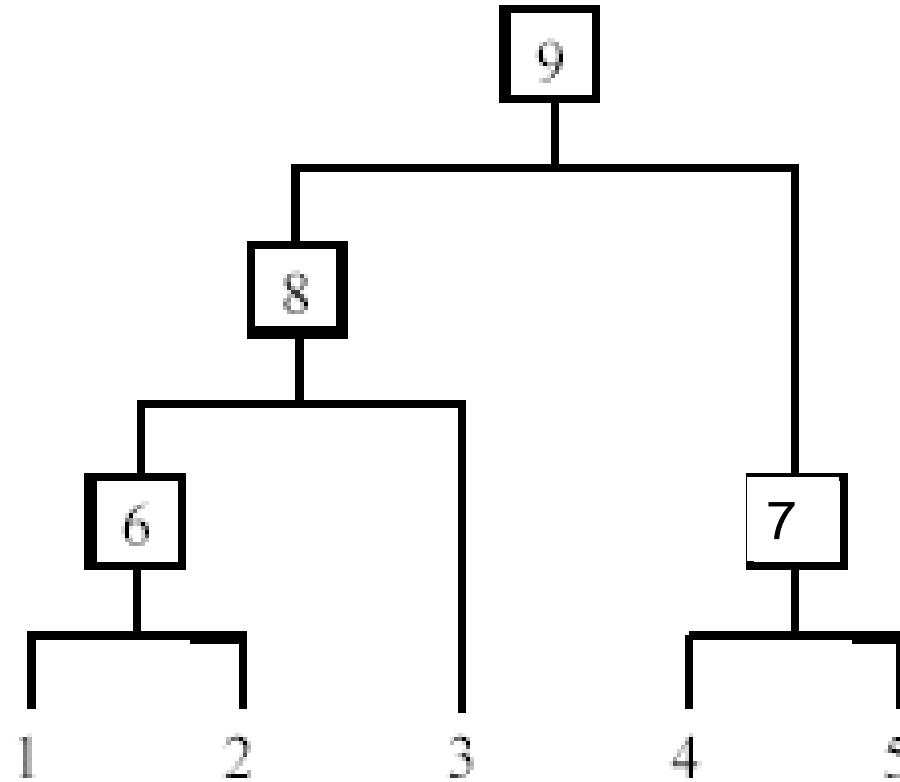


# Road map

- Basic concepts
- K-means algorithm
- Representation of clusters
- **Hierarchical clustering**
- Distance functions
- Which clustering algorithm to use?
- Cluster evaluation
- Summary

# Hierarchical Clustering

- Produce a nested sequence of clusters, a **tree**, also called **Dendrogram**.



# Types of hierarchical clustering

- **Agglomerative (bottom up) clustering:** It builds the dendrogram (tree) from the bottom level, and
  - merges the most similar (or nearest) pair of clusters
  - stops when all the data points are merged into a single cluster (i.e., the root cluster).
- **Divisive (top down) clustering:** It starts with all data points in one cluster, the root.
  - Splits the root into a set of child clusters. Each child cluster is recursively divided further
  - stops when only singleton clusters of individual data points remain, i.e., each cluster with only a single point

# Agglomerative clustering

It is more popular than divisive methods.

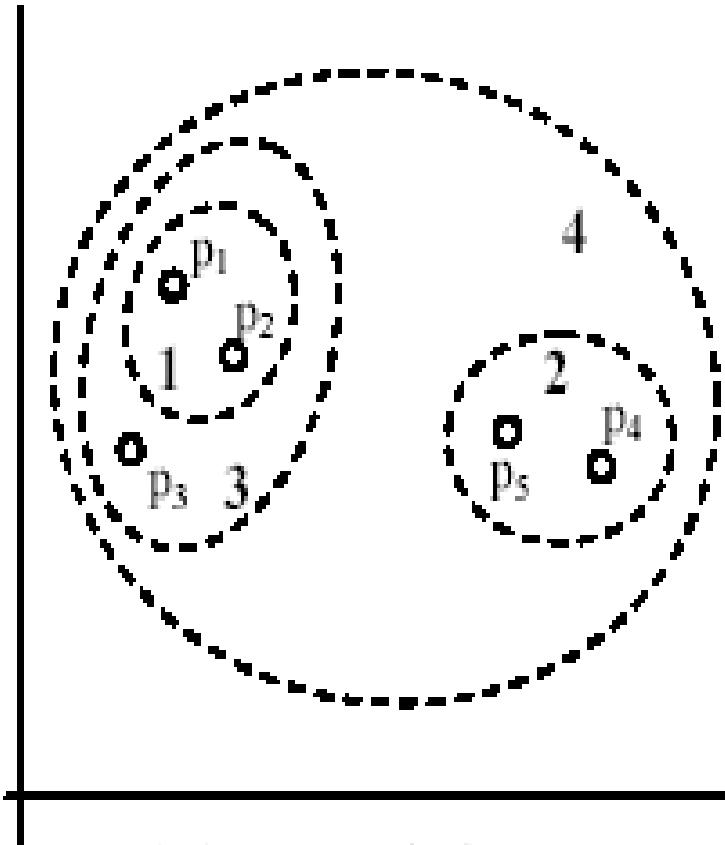
- At the beginning, each data point forms a cluster (also called a node).
- Merge nodes/clusters that have the least distance.
- Go on merging
- Eventually all nodes belong to one cluster

# Agglomerative clustering algorithm

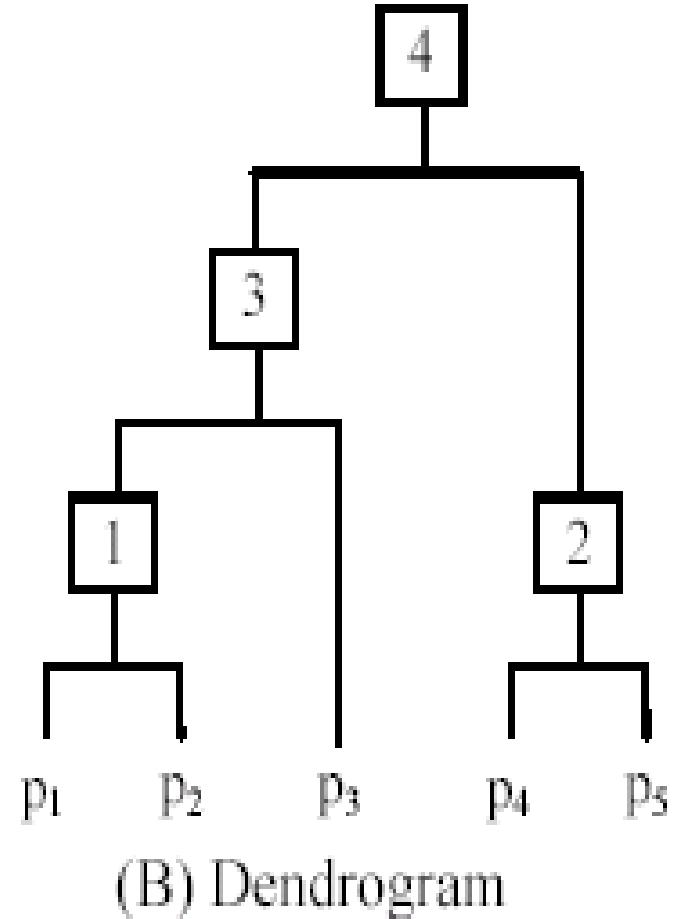
**Algorithm Agglomerative( $D$ )**

- 1 Make each data point in the data set  $D$  a cluster,
- 2 Compute all pair-wise distances of  $x_1, x_2, \dots, x_n \in D$ ;
- 2 repeat
  - 3 find two clusters that are nearest to each other;
  - 4 merge the two clusters form a new cluster  $c$ ;
  - 5 compute the distance from  $c$  to all other clusters;
- 12 until there is only one cluster left

# An example: working of the algorithm



(A). Nested clusters



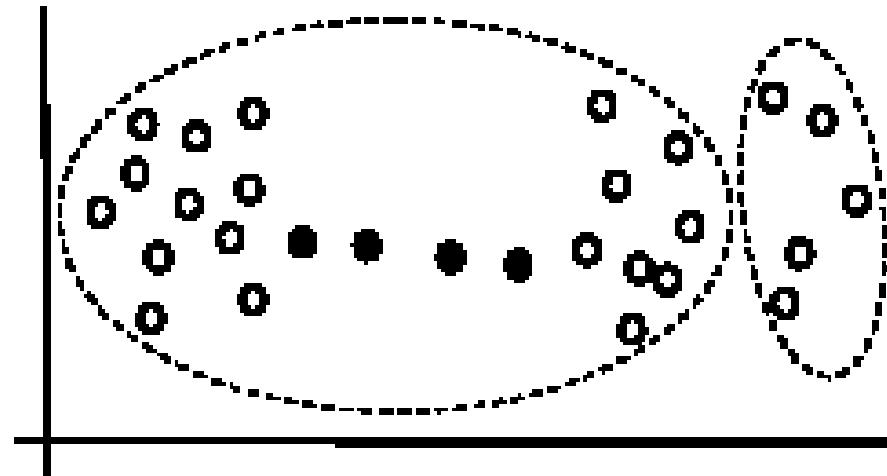
(B) Dendrogram

# Measuring the distance of two clusters

- A few ways to measure distances of two clusters.
- Results in different variations of the algorithm.
  - Single link
  - Complete link
  - Average link
  - Centroids
  - ...

# Single link method

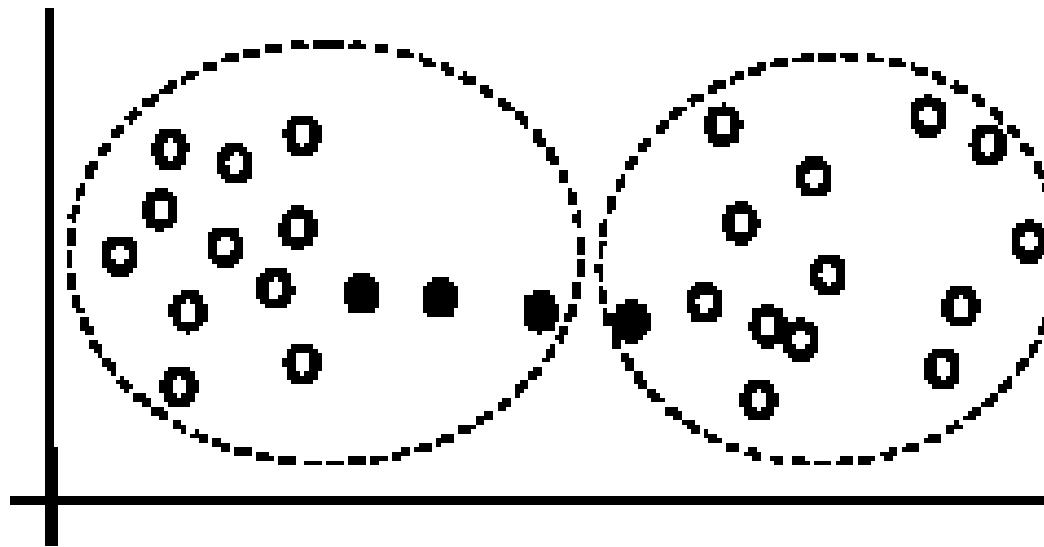
- The distance between two clusters is the distance between two **closest data points** in the two clusters, one data point from each cluster.
- It can find arbitrarily shaped clusters, but
  - It may cause the undesirable “**chain effect**” by noisy points



Two natural clusters are split into two

# Complete link method

- The distance between two clusters is the distance of two **furthest** data points in the two clusters.
- It is sensitive to outliers because they are far away



# Average link and centroid methods

- **Average link:** A compromise between
  - the sensitivity of complete-link clustering to outliers and
  - the tendency of single-link clustering to form long chains that do not correspond to the intuitive notion of clusters as compact, spherical objects.
  - In this method, the distance between two clusters is the average distance of all pair-wise distances between the data points in two clusters.
- **Centroid method:** In this method, the distance between two clusters is the distance between their centroids

# Road map

- Basic concepts
- K-means algorithm
- Representation of clusters
- Hierarchical clustering
- **Distance functions**
- Which clustering algorithm to use?
- Cluster evaluation
- Summary

# Distance functions

- Key to clustering. “**similarity**” and “**dissimilarity**” can also commonly used terms.
- There are numerous distance functions for
  - Different types of data
    - Numeric data
    - Nominal data
  - Different specific applications

# Distance functions for numeric attributes

- Most commonly used functions are
  - Euclidean distance and
  - Manhattan (city block) distance
- We denote distance with:  $dist(\mathbf{x}_i, \mathbf{x}_j)$ , where  $\mathbf{x}_i$  and  $\mathbf{x}_j$  are data points (vectors)
- They are special cases of Minkowski distance.  
 $h$  is positive integer.

$$dist(\mathbf{x}_i, \mathbf{x}_j) = \left( (x_{i1} - x_{j1})^h + (x_{i2} - x_{j2})^h + \dots + (x_{ir} - x_{jr})^h \right)^{\frac{1}{h}}$$

# Euclidean distance and Manhattan distance

- If  $h = 2$ , it is the Euclidean distance

$$dist(\mathbf{x}_i, \mathbf{x}_j) = \sqrt{(x_{i1} - x_{j1})^2 + (x_{i2} - x_{j2})^2 + \dots + (x_{ir} - x_{jr})^2}$$

- If  $h = 1$ , it is the Manhattan distance

$$dist(\mathbf{x}_i, \mathbf{x}_j) = |x_{i1} - x_{j1}| + |x_{i2} - x_{j2}| + \dots + |x_{ir} - x_{jr}|$$

- Weighted Euclidean distance

$$dist(\mathbf{x}_i, \mathbf{x}_j) = \sqrt{w_1(x_{i1} - x_{j1})^2 + w_2(x_{i2} - x_{j2})^2 + \dots + w_r(x_{ir} - x_{jr})^2}$$

# Squared distance and Chebychev distance

- **Squared Euclidean distance:** to place progressively greater weight on data points that are further apart.

$$dist(\mathbf{x}_i, \mathbf{x}_j) = (x_{i1} - x_{j1})^2 + (x_{i2} - x_{j2})^2 + \dots + (x_{ir} - x_{jr})^2$$

- **Chebychev distance:** one wants to define two data points as "different" if they are different on any one of the attributes.

$$dist(\mathbf{x}_i, \mathbf{x}_j) = \max(|x_{i1} - x_{j1}|, |x_{i2} - x_{j2}|, \dots, |x_{ir} - x_{jr}|)$$

# Distance functions for binary and nominal attributes

- **Binary attribute:** has two values or states but no ordering relationships, e.g.,
  - Gender: male and female.
- We use a confusion matrix to introduce the distance functions/measures.
- Let the  $i$ th and  $j$ th data points be  $\mathbf{x}_i$  and  $\mathbf{x}_j$  (vectors)

# Confusion matrix

		Data point $j$		(10)
		1	0	
Data point $i$	1	$a$	$b$	$a+b$
	0	$c$	$d$	$c+d$
		$a+c$	$b+d$	$a+b+c+d$

- a: the number of attributes with the value of 1 for both data points.
- b: the number of attributes for which  $x_{if} = 1$  and  $x_{jf} = 0$ , where  $x_{if}$  ( $x_{jf}$ ) is the value of the  $f$ th attribute of the data point  $\mathbf{x}_i$  ( $\mathbf{x}_j$ ).
- c: the number of attributes for which  $x_{if} = 0$  and  $x_{jf} = 1$ .
- d: the number of attributes with the value of 0 for both data points.

# Symmetric binary attributes

- A binary attribute is **symmetric** if both of its states (0 and 1) have equal importance, and carry the same weights, e.g., male and female of the attribute Gender
- Distance function: **Simple Matching Coefficient**, proportion of mismatches of their values

$$dist(\mathbf{x}_i, \mathbf{x}_j) = \frac{b + c}{a + b + c + d}$$

# Symmetric binary attributes: example

$\mathbf{x}_1$	1	1	1	0	1	0	0
$\mathbf{x}_2$	0	1	1	0	0	1	0

$$dist(\mathbf{x}_i, \mathbf{x}_j) = \frac{2+1}{2+2+1+2} = \frac{3}{7} = 0.429$$

# Asymmetric binary attributes

- **Asymmetric:** if one of the states is more important or more valuable than the other.
  - By convention, state 1 represents the more important state, which is typically the rare or infrequent state.
  - **Jaccard coefficient** is a popular measure

$$dist(\mathbf{x}_i, \mathbf{x}_j) = \frac{b + c}{a + b + c}$$

- We can have some variations, adding weights

# Nominal attributes

- **Nominal attributes:** with more than two states or values.
  - the commonly used distance measure is also based on the **simple matching method**.
  - Given two data points  $\mathbf{x}_i$  and  $\mathbf{x}_j$ , let the number of attributes be  $r$ , and the number of values that match in  $\mathbf{x}_i$  and  $\mathbf{x}_j$  be  $q$ .

$$dist(\mathbf{x}_i, \mathbf{x}_j) = \frac{r - q}{r}$$

# Distance function for text documents

- A text document consists of a sequence of sentences and each sentence consists of a sequence of words.
- To simplify: a document is usually considered a “bag” of words in document clustering.
  - Sequence and position of words are ignored.
- A document is represented with a vector just like a normal data point.
- It is common to use similarity to compare two documents rather than distance.
  - The most commonly used similarity function is the **cosine similarity**. We will study this later.

# Road map

- Basic concepts
- K-means algorithm
- Representation of clusters
- Hierarchical clustering
- Distance functions
- **Which clustering algorithm to use?**
- Cluster evaluation
- Summary

# How to choose a clustering algorithm

- Clustering research has a long history. A vast collection of algorithms are available.
  - We only introduced several main algorithms.
- **Choosing the “best” algorithm is a challenge.**
  - Every algorithm has limitations and works well with certain data distributions.
  - It is very hard, if not impossible, to know what distribution the application data follow. The data may not fully follow any “ideal” structure or distribution required by the algorithms.
  - One also needs to decide how to standardize the data, to choose a suitable distance function and to select other parameter values.

# Choose a clustering algorithm (cont ...)

- Due to these complexities, the common practice is to
  - run several algorithms using different distance functions and parameter settings, and
  - then carefully analyze and compare the results.
- The interpretation of the results must be based on insight into the meaning of the original data together with knowledge of the algorithms used.
- Clustering is highly **application dependent** and to certain extent **subjective** (personal preferences).

# Road map

- Basic concepts
- K-means algorithm
- Representation of clusters
- Hierarchical clustering
- Distance functions
- Which clustering algorithm to use?
- Cluster evaluation
- Summary

# Cluster Evaluation: hard problem

- The quality of a clustering is very hard to evaluate because
  - We do not know the correct clusters
- Some methods are used:
  - User inspection
    - Study centroids, and spreads
    - Rules from a decision tree.
    - For text documents, one can read some documents in clusters.

# Cluster evaluation: ground truth

- We use some labeled data (for classification)
- **Assumption:** Each class is a cluster.
- After clustering, a confusion matrix is constructed. From the matrix, we compute various measurements, entropy, purity, precision, recall and F-score.
  - Let the classes in the data  $D$  be  $C = (c_1, c_2, \dots, c_k)$ . The clustering method produces  $k$  clusters, which divides  $D$  into  $k$  disjoint subsets,  $D_1, D_2, \dots, D_k$ .

# Evaluation measures: Entropy

Entropy: For each cluster, we can measure its entropy as follows:

$$\text{entropy}(D_i) = - \sum_{j=1}^k \Pr_i(c_j) \log_2 \Pr_i(c_j), \quad (29)$$

where  $\Pr_i(c_j)$  is the proportion of class  $c_j$  data points in cluster  $i$  or  $D_i$ . The total entropy of the whole clustering (which considers all clusters) is

$$\text{entropy}_{\text{total}}(D) = \sum_{i=1}^k \frac{|D_i|}{|D|} \times \text{entropy}(D_i) \quad (30)$$

# Evaluation measures: purity

Purity: This again measures the extent that a cluster contains only one class of data. The purity of each cluster is computed with

$$purity(D_i) = \max_j(\Pr_i(c_j)) \quad (31)$$

The total purity of the whole clustering (considering all clusters) is

$$purity_{total}(D) = \sum_{i=1}^k \frac{|D_i|}{|D|} \times purity(D_i) \quad (32)$$

# An example

**Example 14:** Assume we have a text collection  $D$  of 900 documents from three topics (or three classes), Science, Sports, and Politics. Each class has 300 documents. Each document in  $D$  is labeled with one of the topics (classes). We use this collection to perform clustering to find three clusters. Note that class/topic labels are not used in clustering. After clustering, we want to measure the effectiveness of the clustering algorithm.

Cluster	Science	Sports	Politics		Entropy	Purity
1	250	20	10		0.589	0.893
2	20	180	80		1.198	0.643
3	30	100	210		1.257	0.617
Total	300	300	300		1.031	0.711

# A remark about ground truth evaluation

- Commonly used to compare different clustering algorithms.
- A real-life data set for clustering has no class labels.
  - Thus although an algorithm may perform very well on some labeled data sets, no guarantee that it will perform well on the actual application data at hand.
- The fact that it performs well on some label data sets does give us some confidence of the quality of the algorithm.
- This evaluation method is said to be based on **external data** or information.

# Evaluation based on internal information

- **Intra-cluster cohesion** (compactness):
  - Cohesion measures how near the data points in a cluster are to the cluster centroid.
  - Sum of squared error (SSE) is a commonly used measure.
- **Inter-cluster separation** (isolation):
  - Separation means that different cluster centroids should be far away from one another.
- In most applications, expert judgments are still the key.

# Indirect evaluation

- In some applications, clustering is **not the primary task**, but used to help perform another task.
- We can use the performance on the primary task to compare clustering methods.
- For instance, in an application, the primary task is to provide recommendations on book purchasing to online shoppers.
  - If we can cluster books according to their features, we might be able to provide better recommendations.
  - We can evaluate different clustering algorithms based on how well they help with the recommendation task.
  - Here, we assume that the recommendation can be reliably evaluated.

# Road map

- Basic concepts
- K-means algorithm
- Representation of clusters
- Hierarchical clustering
- Distance functions
- Which clustering algorithm to use?
- Cluster evaluation
- Summary

# Summary

- Clustering is has along history and still active
  - There are a huge number of clustering algorithms
  - More are still coming every year.
- We only introduced several main algorithms. There are many others, e.g.,
  - density based algorithm, sub-space clustering, scale-up methods, neural networks based methods, fuzzy clustering, co-clustering, etc.
- Clustering is hard to evaluate, but very useful in practice. This partially explains why there are still a large number of clustering algorithms being devised every year.
- Clustering is highly application dependent and to some extent subjective.

# Chapter 6: Information Retrieval and Web Search

---

Dr. Mehmet S. Aktaş

Acknowledgement: Thanks to Dr. Bing Liu for teaching materials.

# Introduction

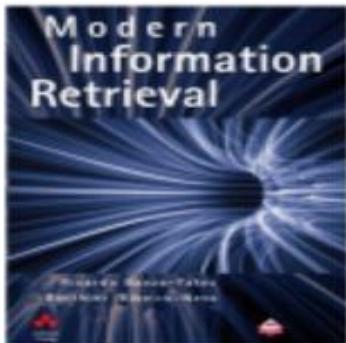
- Text mining refers to data mining using text documents as data.
- Most text mining tasks use **Information Retrieval** (IR) methods to pre-process text documents.
- These methods are quite different from traditional data pre-processing methods used for relational tables.
- Web search also has its root in IR.

# Information Retrieval (IR)

- Conceptually, IR is the study of finding needed information. I.e., IR helps users find information that matches their information needs.
  - Expressed as queries
- Historically, IR is about document retrieval, emphasizing document as the basic unit.
  - Finding documents relevant to user queries
- Technically, IR studies the acquisition, organization, storage, retrieval, and distribution of information.



## What is Information Retrieval (IR)?



IR: Part of computer science which studies the **retrieval of information (not data)** from a collection of **written documents**. The retrieved documents aim at satisfying a **user information need** usually expressed in **natural language**.

- **Documents, unstructured, text, large**
- **Information need**
- **Store, search, find**
- **The World Wide Web?**
- **Relational databases?**

# DIKW

- **Data:** Raw web pages
  - **Information:** Result of query
  - **Knowledge:** Result of processing query result by user
  - **Wisdom:** Synthesis of many such actions by a set of users
- 
- One possible classification of steps in process



## Information Retrieval vs. Databases

Information retrieval	Data retrieval
Retrieve all objects <b>relevant</b> to some <b>information need</b>	Retrieve all objects satisfying some <b>clearly defined conditions</b>
Find all documents about the <b>topic</b> "semantic web"!	<b>SELECT id FROM document WHERE title LIKE "%semantic web%"</b>
<b>Result list</b>	<b>Well-defined result set</b>

Google semanticweb

[Web](#) [Images](#) [News](#) [Videos](#) [Gadgets](#)

**Semantic Web - Wikipedia, the free encyclopedia**  
The Semantic Web is an evolving extension of the World Wide Web in which the semantics of information and services on the Web is defined, making it possible ...  
[en.wikipedia.org/w/index.php?title=Semantic\\_Web&oldid=459522200](#) - 2016 - [Source](#) - [Edit this page](#)

**W3C Semantic Web Activity**  
The Semantic Web provides a common framework that allows data to be shared and reused across applications, enterprises, and community boundaries. ...  
[www.w3.org/2001/sw/Activity/0800](#) - 2016 - [Source](#) - [Edit this page](#)

**Semantic Web Activity Statement**  
The W3C Semantic Web Activity has been established to serve a broadening role, in both the design of specifications and the open, collaborative development. ...  
[www.w3.org/2001/sw/Activity/0800](#) - 2016 - [Source](#) - [Edit this page](#)

```
[selke@tbdb ~]$ db2 "SELECT id FROM document WHERE title LIKE '%semantic web%' FETCH FIRST 3 ROWS ONLY"
```

ID
45489
9635899
98556

3 record(s) selected.



## Web Search

- Very similar to information retrieval
- Main differences:
  - Links between Web pages can be exploited
  - Collecting, storing, and updating documents is more difficult
  - Usually, the number of users is very large
  - Spam is a problem



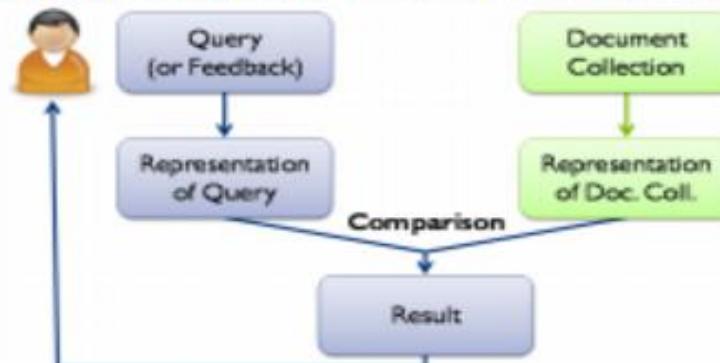
5

<http://www.ifis.cs.tu-bs.de/teaching/ss-11/irws>

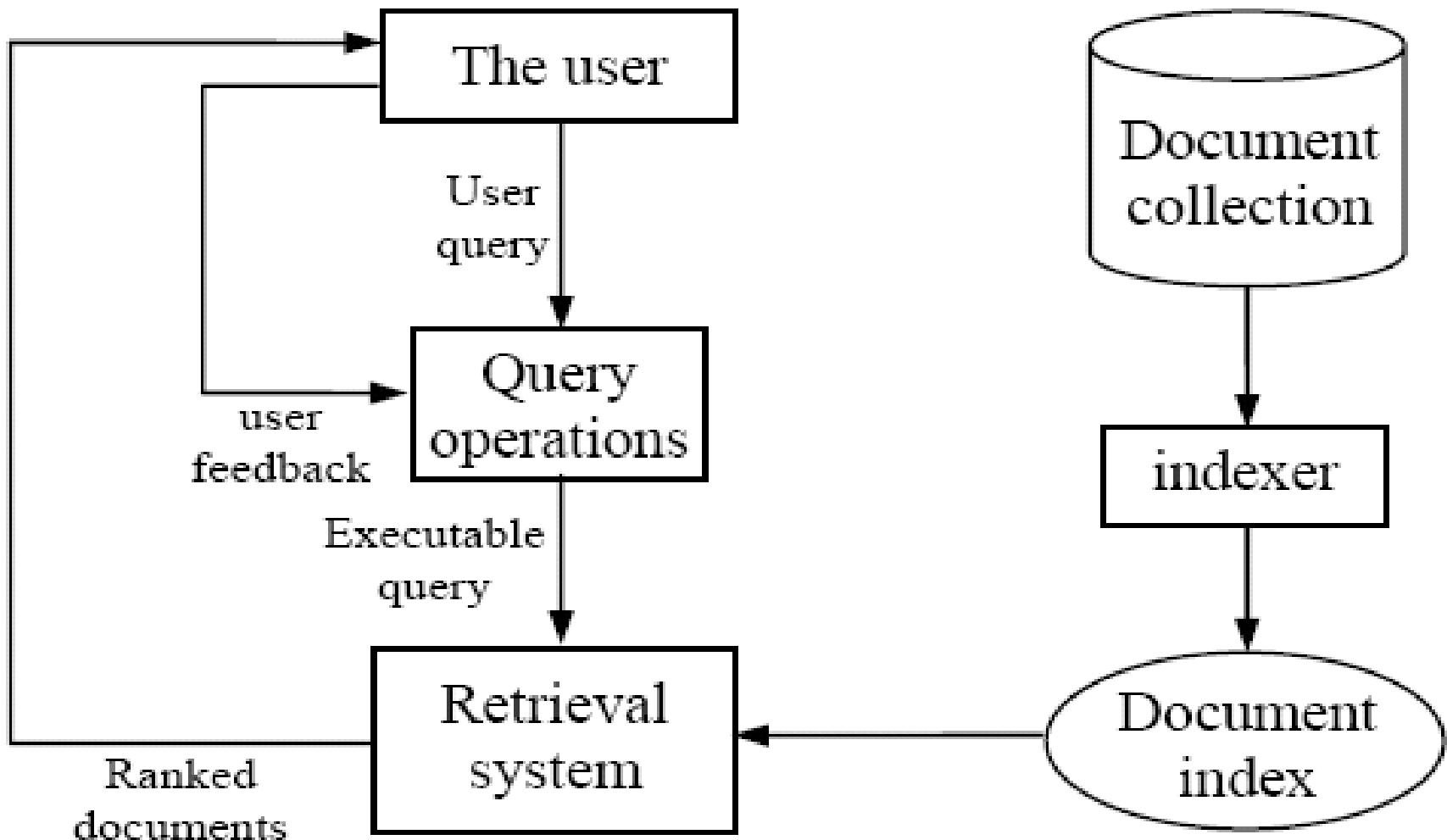


## IR Models

- Any IR system is based on an **IR model**
- The model defines ...
  - ... a **query language**,
  - ... an internal **representation of queries**,
  - ... an internal **representation of documents**,
  - ... a **ranking function** which associates a real number with each query–document pair.
- Optional: A mechanism for **relevance feedback**



# IR architecture



# IR queries

- Keyword queries
- Boolean queries (using AND, OR, NOT)
- Phrase queries
- Proximity queries
- Full document queries
- Natural language questions

# Information retrieval models

- An IR model governs how a document and a query are represented and how the relevance of a document to a user query is defined.
- Main models:
  - Boolean model
  - Vector space model
  - Statistical language model
  - etc

# Boolean model

- Each document or query is treated as a **“bag” of words** or **terms**. Word sequence is not considered.
- Given a collection of documents  $D$ , let  $V = \{t_1, t_2, \dots, t_{|V|}\}$  be the set of distinctive words/terms in the collection.  $V$  is called the **vocabulary**.
- A weight  $w_{ij} > 0$  is associated with each term  $t_i$  of a document  $\mathbf{d}_j \in D$ . For a term that does not appear in document  $\mathbf{d}_j$ ,  $w_{ij} = 0$ .

$$\mathbf{d}_j = (w_{1j}, w_{2j}, \dots, w_{|Vj}),$$

# Boolean model (contd)

- Query terms are combined logically using the Boolean operators **AND**, **OR**, and **NOT**.
  - E.g., ((*data AND mining*) **AND** (**NOT text**))
- Retrieval
  - Given a Boolean query, the system retrieves every document that makes the query logically true.
  - Called **exact match**.
- The retrieval results are usually quite poor because term frequency is not considered.

# Vector space model

- Documents are also treated as a “bag” of words or terms.
- Each document is represented as a vector.
- However, the term weights are no longer 0 or 1. Each term weight is computed based on some variations of **TF** or **TF-IDF** scheme.
- **Term Frequency (TF) Scheme:** The weight of a term  $t_i$  in document  $\mathbf{d}_j$  is the number of times that  $t_i$  appears in  $\mathbf{d}_j$ , denoted by  $f_{ij}$ . Normalization may also be applied.

# TF-IDF term weighting scheme

- The most well known weighting scheme
  - TF: still **term frequency**
  - IDF: **inverse document frequency.**

$N$ : total number of docs

$df_i$ : the number of docs that  $t_i$  appears.

- The final TF-IDF term weight is:

$$tf_{ij} = \frac{f_{ij}}{\max\{f_{1j}, f_{2j}, \dots, f_{|V|j}\}}$$

$$idf_i = \log \frac{N}{df_i}$$

$$w_{ij} = tf_{ij} \times idf_i.$$

# Retrieval in vector space model

- Query  $\mathbf{q}$  is represented in the same way or slightly differently.
- **Relevance of  $\mathbf{d}_j$  to  $\mathbf{q}$ :** Compare the similarity of query  $\mathbf{q}$  and document  $\mathbf{d}_j$ .
- Cosine similarity (the cosine of the angle between the two vectors)

$$\text{cosine}(\mathbf{d}_j, \mathbf{q}) = \frac{\langle \mathbf{d}_j \bullet \mathbf{q} \rangle}{\| \mathbf{d}_j \| \times \| \mathbf{q} \|} = \frac{\sum_{i=1}^{|V|} w_{ij} \times w_{iq}}{\sqrt{\sum_{i=1}^{|V|} w_{ij}^2} \times \sqrt{\sum_{i=1}^{|V|} w_{iq}^2}}$$

- Cosine is also commonly used in text clustering

# An Example

- A document space is defined by three terms:
  - hardware, software, users
  - the vocabulary
- A set of documents are defined as:
  - $A1=(1, 0, 0)$ ,       $A2=(0, 1, 0)$ ,       $A3=(0, 0, 1)$
  - $A4=(1, 1, 0)$ ,       $A5=(1, 0, 1)$ ,       $A6=(0, 1, 1)$
  - $A7=(1, 1, 1)$        $A8=(1, 0, 1)$ .       $A9=(0, 1, 1)$
- If the Query is “hardware and software”
- what documents should be retrieved?

# An Example (cont.)

## ■ In Boolean query matching:

- document A4, A7 will be retrieved (“AND”)
- retrieved: A1, A2, A4, A5, A6, A7, A8, A9 (“OR”)

## ■ In similarity matching (cosine):

- $q=(1, 1, 0)$
- $S(q, A1)=0.71$ ,       $S(q, A2)=0.71$ ,       $S(q, A3)=0$
- $S(q, A4)=1$ ,       $S(q, A5)=0.5$ ,       $S(q, A6)=0.5$
- $S(q, A7)=0.82$ ,       $S(q, A8)=0.5$ ,       $S(q, A9)=0.5$
- Document retrieved set (with ranking)=
  - {A4, A7, A1, A2, A5, A6, A8, A9}

# Okapi relevance method

- Another way to assess the degree of relevance is to directly compute a relevance score for each document to the query.
- The **Okapi** method and its variations are popular techniques in this setting.

The Okapi relevance score of a document  $d_j$  for a query  $q$  is:

$$okapi(d_j, q) = \sum_{t_i \in q, d_j} \ln \frac{N - df_i + 0.5}{df_i + 0.5} \times \frac{(k_1 + 1)f_{ij}}{k_1(1 - b + b \frac{dl_j}{avdl}) + f_{ij}} \times \frac{(k_2 + 1)f_{iq}}{k_2 + f_{iq}},$$

where  $k_1$  (between 1.0-2.0),  $b$  (usually 0.75) and  $k_2$  (between 1-1000)

# Relevance feedback

- Relevance feedback is one of the techniques for improving retrieval effectiveness. The steps:
  - the user first identifies some relevant ( $D_r$ ) and irrelevant documents ( $D_{ir}$ ) in the initial list of retrieved documents
  - the system expands the query  $\mathbf{q}$  by extracting some additional terms from the sample relevant and irrelevant documents to produce  $\mathbf{q}_e$
  - Perform a second round of retrieval.
- **Rocchio method** ( $\alpha$ ,  $\beta$  and  $\gamma$  are parameters)

$$\mathbf{q}_e = \alpha \mathbf{q} + \frac{\beta}{|D_r|} \sum_{\mathbf{d}_r \in D_r} \mathbf{d}_r - \frac{\gamma}{|D_{ir}|} \sum_{\mathbf{d}_{ir} \in D_{ir}} \mathbf{d}_{ir}$$

# Text pre-processing

- Word (term) extraction: easy
- Stopwords removal
- Stemming
- Frequency counts and computing TF-IDF term weights.

# Stopwords removal

- Many of the most frequently used words in English are useless in IR and text mining – these words are called *stop words*.
  - the, of, and, to, ....
  - Typically about 400 to 500 such words
  - For an application, an additional domain specific stopwords list may be constructed
- Why do we need to remove stopwords?
  - Reduce indexing (or data) file size
    - stopwords accounts 20-30% of total word counts.
  - Improve efficiency and effectiveness
    - stopwords are not useful for searching or text mining
    - they may also confuse the retrieval system.

# Stemming

- Techniques used to find out the root/stem of a word. E.g.,
    - user engineering
    - users engineered
    - used engineer
    - using
  - stem: use engineer

# Usefulness:

- improving effectiveness of IR and text mining
    - matching similar words
    - Mainly improve recall
  - reducing indexing size
    - combining words with same roots may reduce indexing size as much as 40-50%.

# Basic stemming methods

Using a set of rules. E.g.,

- remove ending

- if a word ends with a consonant other than s, followed by an s, then delete s.
  - if a word ends in es, drop the s.
  - if a word ends in ing, delete the ing unless the remaining word consists only of one letter or of th.
  - If a word ends with ed, preceded by a consonant, delete the ed unless this leaves only a single letter.
  - .....

- transform words

- if a word ends with “ies” but not “eies” or “aies” then “ies --> y.”

# Frequency counts + TF-IDF

- Counts the number of times a word occurred in a document.
  - Using occurrence frequencies to indicate relative importance of a word in a document.
    - if a word appears often in a document, the document likely “deals with” subjects related to the word.
- Counts the number of documents in the collection that contains each word
- TF-IDF can be computed.

# Evaluation: Precision and Recall

- Given a query:
  - Are all retrieved documents relevant?
  - Have all the relevant documents been retrieved?
- Measures for system performance:
  - The first question is about the **precision** of the search
  - The second is about the completeness (**recall**) of the search.

# “Classic” Information Retrieval

## The Collection

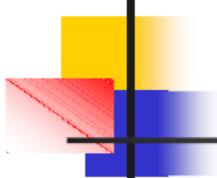
Relevant Documents (A)

Retrieved Documents (B)

- Given a query, the system retrieves a set B of documents
- Every retrieved document is either relevant or irrelevant to the query

Quality metrics:

- Recall :  $(A \cap B) / A$
- Precision :  $(A \cap B) / B$



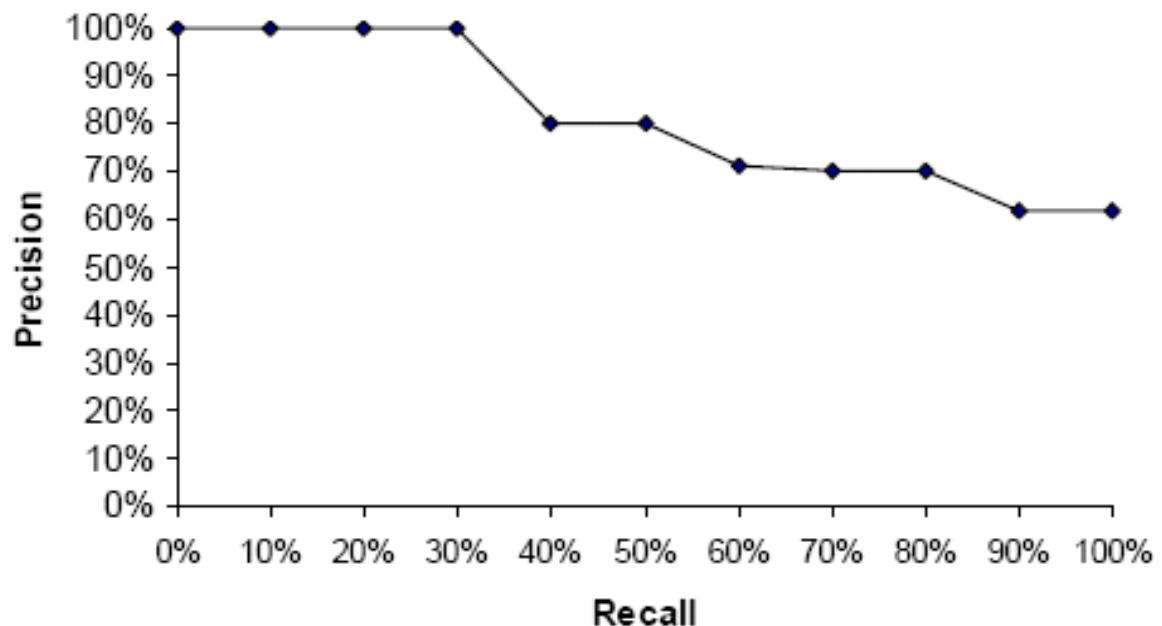
# Recall and Precision on the Web

- Relevance of document to queries is not binary – there are many shades of gray
- Broad-topic queries:
  - abundance problem
  - Precision is the dominating factor: users mostly satisfied with a few good results (a few *authoritative* pages)
- Narrow-topic queries:
  - Find a needle in an enormous haystack
  - Recall demands engines cover significant portions of the Web
- Common measure: precision@10
- Nowadays larger emphasis on *diversity*
  - Positive recall for many aspects of the query

# Precision-recall curve

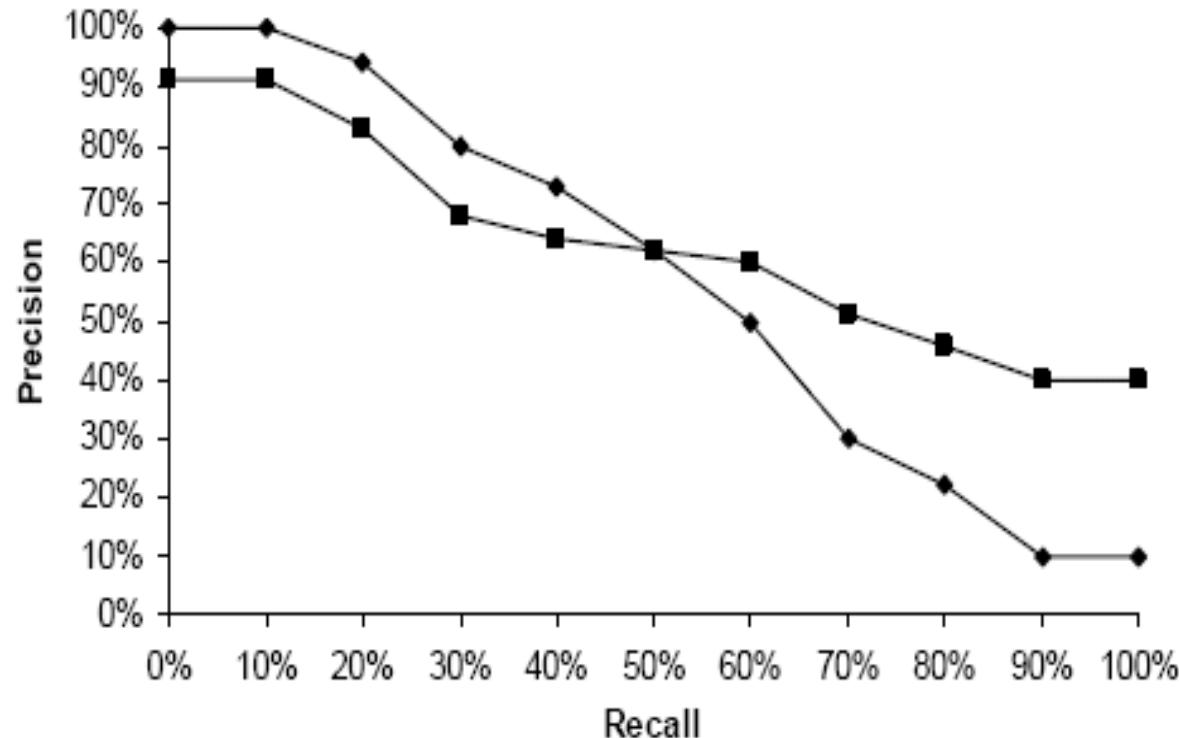
**Example 2:** Following Example 1, we obtain the interpolated precisions at all 11 recall levels in the table of Fig. 6.4. The precision-recall curve is shown on the right.

$i$	$p(r_i)$	$r_i$
0	100%	0%
1	100%	10%
2	100%	20%
3	100%	30%
4	80%	40%
5	80%	50%
6	71%	60%
7	70%	70%
8	70%	80%
9	62%	90%
10	62%	100%



**Fig. 6.4.** The precision-recall curve

# Compare different retrieval algorithms



**Fig. 6.5.** Comparison of two retrieval algorithms based on their precision-recall curves

# Compare with multiple queries

- Compute the average precision at each recall level.

$$\bar{p}(r_i) = \frac{1}{|Q|} \sum_{j=1}^{|Q|} p_j(r_i), \quad (22)$$

where  $Q$  is the set of all queries and  $p_j(r_i)$  is the precision of query  $j$  at the recall level  $r_i$ . Using the average precision at each recall level, we can also draw a precision-recall curve.

- Draw precision recall curves

# Rank precision

- Compute the precision values at some selected rank positions.
- Mainly used in Web search evaluation.
- For a Web search engine, we can compute precisions for the top 5, 10, 15, 20, 25 and 30 returned pages
  - as the user seldom looks at more than 30 pages.
- Recall is not very meaningful in Web search.
  - Why?

# Web Search as a huge IR system

- A Web crawler (robot) crawls the Web to collect all the pages.
- Servers establish a huge inverted indexing database and other indexing databases
- At query (search) time, search engines conduct different types of vector query matching.

# Inverted index

- The inverted index of a document collection is basically a data structure that
  - attaches each distinctive term with a list of all documents that contains the term.
- Thus, in retrieval, it takes constant time to
  - find the documents that contains a query term.
  - multiple query terms are also easy handle as we will see soon.

# An example

**Example 3:** We have three documents of  $id_1$ ,  $id_2$ , and  $id_3$ :

$id_1$ : Web mining is useful.

1      2      3      4

$id_2$ : Usage mining applications.

1      2                3

$id_3$ : Web structure mining studies the Web hyperlink structure.

1      2      3      4      5      6      7      8

Applications:  $id_2$

Hyperlink:  $id_3$

Mining:  $id_1$ ,  $id_2$ ,  $id_3$

Structure:  $id_3$

Studies:  $id_3$

Usage:  $id_2$

Useful:  $id_1$

Web:  $id_1$ ,  $id_3$

Applications:  $\langle id_2, 1, [3] \rangle$

Hyperlink:  $\langle id_3, 1, [7] \rangle$

Mining:  $\langle id_1, 1, [2] \rangle$ ,  $\langle id_2, 1, [2] \rangle$ ,  $\langle id_3, 1, [3] \rangle$

Structure:  $\langle id_3, 2, [2, 8] \rangle$

Studies:  $\langle id_3, 1, [4] \rangle$

Usage:  $\langle id_2, 1, [1] \rangle$

Useful:  $\langle id_1, 1, [4] \rangle$

Web:  $\langle id_1, 1, [1] \rangle$ ,  $\langle id_3, 2, [1, 6] \rangle$

(A)

(B)

**Fig. 6.7.** Two inverted indices: a simple version and a more complex version

# Search using inverted index

Given a query **q**, search has the following steps:

- Step 1 (**vocabulary search**): find each term/word in **q** in the inverted index.
- Step 2 (**results merging**): Merge results to find documents that contain all or some of the words/terms in **q**.
- Step 3 (**Rank score computation**): To rank the resulting documents/pages, using,
  - content-based ranking
  - link-based ranking

# Different search engines

- The real differences among different search engines are
  - their index weighting schemes
    - Including location of terms, e.g., title, body, emphasized words, etc.
  - their query processing methods (e.g., query classification, expansion, etc)
  - **their ranking algorithms**
  - Few of these are published by any of the search engine companies. They are tightly guarded secrets.

# Summary

- We only give a **VERY** brief introduction to IR. There are a large number of other topics, e.g.,
  - Statistical language model
  - Latent semantic indexing (LSI and SVD).
  - (read an IR book or take an IR course)
- Many other interesting topics are not covered, e.g.,
  - Web search
    - Index compression
    - Ranking: combining contents and hyperlinks
  - Web page pre-processing
  - Combining multiple rankings and meta search
  - Web spamming
- Want to know more? Read the textbook

# Chapter 7: Link Analysis-1

---

Instructor: Dr. Mehmet S. Aktas

# Road map

- **Introduction**
- **Social network analysis**
- **Co-citation and bibliographic coupling**

# Introduction

- Early search engines mainly compare content similarity of the query and the indexed pages. I.e.,
  - They use information retrieval methods, cosine, TF-IDF, ...
- From 1996, it became clear that content similarity alone was no longer sufficient.
  - The number of pages grew rapidly in the mid-late 1990's.
    - Try “classification technique”, Google estimates: 10 million relevant pages.
    - How to choose only 30-40 pages and rank them suitably to present to the user?
  - Content similarity is easily spammed.
    - A page owner can repeat some words and add many related words to boost the rankings of his pages and/or to make the pages relevant to a large number of queries.

# Introduction (cont ...)

- Starting around 1996, researchers began to work on the problem. They resort to **hyperlinks**.
  - In Feb, 1997, Yanhong Li (Scotch Plains, NJ) filed a hyperlink based search patent. The method uses words in anchor text of hyperlinks.
- Web pages on the other hand are connected through hyperlinks, which carry important information.
  - Some **hyperlinks**: organize information at the same site.
  - Other **hyperlinks**: point to pages from other Web sites. Such out-going hyperlinks often indicate an **implicit conveyance of authority** to the pages being pointed to.
- Those pages that are pointed to by many other pages are likely to contain authoritative information.

# Introduction (cont ...)

- During 1997-1998, two most influential hyperlink based search algorithms **PageRank** and **HITS** were reported.
- Both algorithms are related to **social networks**. They exploit the hyperlinks of the Web to rank pages according to their levels of “prestige” or “authority”.
  - **HITS**: Jon Kleinberg (Cornel University), at *Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, January 1998
  - **PageRank**: Sergey Brin and Larry Page, PhD students from Stanford University, at *Seventh International World Wide Web Conference (WWW7)* in April, 1998.
- **PageRank powers the Google search engine.**
- **“Stanford University”** **the great!**
  - Google: Sergey Brin and Larry Page (PhD candidates in CS)
  - Yahoo!: Jerry Yang and David Filo (PhD candidates in EE)
  - HP, Sun, Cisco, ...

# Introduction (cont ...)

- Apart from search ranking, hyperlinks are also useful for finding Web communities.
  - A Web community is a cluster of densely linked pages representing a group of people with a special interest.
- Beyond explicit hyperlinks on the Web, links in other contexts are useful too, e.g.,
  - for discovering communities of named entities (e.g., people and organizations) in free text documents, and
  - for analyzing social phenomena in emails..

# Road map

- **Introduction**
- **Social network analysis**
- **Co-citation and bibliographic coupling**

# Social network analysis

- Social network is the study of social entities (people in an organization, called **actors**), and their **interactions and relationships**.
- The interactions and relationships can be represented with **a network or graph**,
  - each vertex (or node) represents an actor and
  - each link represents a relationship.
- From the network, we can study the properties of its structure, and **the role, position** and **prestige** of each social actor.
- We can also find various kinds of sub-graphs, e.g., **communities** formed by groups of actors.

# Social network and the Web

- Social network analysis is useful for the Web because the Web is essentially a virtual society, and thus a virtual social network,
  - Each page: a social actor and
  - each hyperlink: a relationship.
- Many results from social network can be adapted and extended for use in the Web context.
- We study two types of social network analysis, **centrality** and **prestige**, which are closely related to hyperlink analysis and search on the Web.

# Centrality

- **Important or prominent actors** are those that are linked or involved with other actors extensively.
- A person with extensive contacts (links) or communications with many other people in the organization is considered more important than a person with relatively fewer contacts.
- The links can also be called **ties**. A **central actor** is one involved in many ties.

# Degree Centrality

Central actors are the most active actors that have most links or ties with other actors. Let the total number of actors in the network be  $n$ .

**Undirected graph:** In an undirected graph, the **degree centrality** of an actor  $i$  (denoted by  $C_D(i)$ ) is simply the node degree (the number of edges) of the actor node, denoted by  $d(i)$ , normalized with the maximum degree,  $n-1$ .

$$C_D(i) = \frac{d(i)}{n-1} \quad (1)$$

**Directed graph:** In this case, we need to distinguish **in-links** of actor  $i$  (links pointing to  $i$ ), and **out-links** (links pointing out from  $i$ ). The degree centrality is defined based on only the out-degree (the number of out-links or edges),  $d_o(i)$ .

$$C_D(i) = \frac{d_o(i)}{n-1} \quad (2)$$

# Closeness Centrality

This view of centrality is based on the closeness or distance. The basic idea is that an actor  $x_i$  is central if it can easily interact with all other actors. That is, its distance to all other actors is short. Thus, we can use the shortest distance to compute this measure. Let the shortest distance from actor  $i$  to actor  $j$  be  $d(i, j)$ .

**Undirected graph:** The closeness centrality  $C_C(i)$  of actor  $i$  is defined as

$$C_C(i) = \frac{n-1}{\sum_{j=1}^n d(i, j)} \quad (3)$$

The value of this measure also ranges between 0 and 1 as  $n-1$  is the minimum value of the denominator, which is the sum of shortest distances from  $i$  to all other actors. Note that this equation is only meaningful for a connected graph.

**Directed graph:** The same equation can be used for a directed graph. The distance computation needs to consider directions of links or edges.

# Betweenness Centrality

- If two non-adjacent actors  $j$  and  $k$  want to interact and actor  $i$  is on the path between  $j$  and  $k$ , then  $i$  may have some control over the interactions between  $j$  and  $k$ .
- **Betweenness** measures this control of  $i$  over other pairs of actors. Thus,
  - if  $i$  is on the paths of many such interactions, then  $i$  is an important actor.

# Betweenness Centrality (cont ...)

- **Undirected graph:** Let  $p_{jk}$  be the number of shortest paths between actor  $j$  and actor  $k$ .
- The betweenness of an actor  $i$  is defined as the number of shortest paths that pass  $i$  ( $p_{jk}(i)$ ) normalized by the total number of shortest paths.

$$\sum_{j < k} \frac{p_{jk}(i)}{p_{jk}} \quad (4)$$

# Betweenness Centrality (cont ...)

Note that there may be multiple shortest paths between  $j$  and  $k$ . Some passes  $i$  and some do not. If we are to ensure the value range is between 0 and 1, we can normalize it with  $(n-1)(n-2)/2$ , which is the maximum value of the above quantity, i.e., the number of pairs of actors not including  $i$ . The final betweenness of  $i$  is defined as

$$C_B(i) = \frac{2 \sum_{j < k} \frac{p_{jk}(i)}{P_{jk}}}{(n-1)(n-2)} \quad (5)$$

Unlike the closeness measure, the betweenness can be computed even if the graph is not connected.

**Directed graph:** The same equation can be used but must be multiplied by 2 because there are now  $(n-1)(n-2)$  pairs considering a path from  $j$  to  $k$  is different from a path from  $k$  to  $j$ . Likewise,  $p_{jk}$  must consider paths from both directions.

# Prestige

- Prestige is a more refined measure of prominence of an actor than centrality.
  - Distinguish: ties sent ([out-links](#)) and ties received ([in-links](#)).
- A prestigious actor is one who is object of extensive ties as a recipient.
  - To compute the prestige: we use only in-links.
- [Difference between centrality and prestige:](#)
  - centrality focuses on out-links
  - prestige focuses on in-links.
- [We study three prestige measures. Rank prestige](#) forms the basis of most Web page link analysis algorithms, including [PageRank](#) and [HITS](#).

# Degree prestige

Based on the definition of the prestige, it is clear that an actor is prestigious if it receives many in-links or nominations. Thus, the simplest measure of prestige of an actor  $i$  (denoted by  $P_D(i)$ ) is its in-degree.

$$P_D(i) = \frac{d_I(i)}{n-1}, \quad (6)$$

where  $d_I(i)$  is in-degree of  $i$  (the number of in-links of actor  $i$ ) and  $n$  is the total number of actors in the network. As in the degree centrality, dividing  $n - 1$  standardizes the prestige value to the range from 0 and 1. The maximum prestige value is 1 when every other actor links to or chooses actor  $i$ .

# Proximity prestige

- The degree index of prestige of an actor  $i$  only considers the actors that are adjacent to  $i$ .
- The **proximity prestige** generalizes it by considering both the actors directly and indirectly linked to actor  $i$ .
  - We consider every actor  $j$  that can reach  $i$ .
- Let  $I_i$  be the set of actors that can reach actor  $i$ .
- The **proximity** is defined as closeness or distance of other actors to  $i$ .
- Let  $d(j, i)$  denote the distance from actor  $j$  to actor  $i$ .

# Proximity prestige (cont ...)

$$\frac{\sum_{j \in I_i} d(j, i)}{|I_i|}, \quad (7)$$

where  $|I_i|$  is the size of the set  $I_i$ . If we look at the ratio or proportion of actors who can reach  $i$  to the average distance that these actors are from  $i$ , we obtain the following, which has the value range of  $[0, 1]$ :

$$P_P(i) = \frac{|I_i|/(n-1)}{\sum_{j \in I_i} d(j, i) / |I_i|}, \quad (8)$$

where  $|I_i|/(n-1)$  is the proportion of actors that can reach actor  $i$ . In one extreme, every actor can reach actor  $i$ , which gives  $|I_i|/(n-1) = 1$ . The denominator is 1 if every actor is adjacent to  $i$ . Thus,  $P_P(i) = 1$ . On the other extreme, no actor can reach actor  $i$ . Then  $|I_i| = 0$ , and  $P_P(i) = 0$ . Each link has the unit distance.

# Rank prestige

- In the previous two prestige measures, an important factor is considered,
  - the **prominence** of individual actors who do the “voting”
- In the real world, a person *i* chosen by an important person is more prestigious than chosen by a less important person.
  - For example, if a company CEO votes for a person is much more important than a worker votes for the person.
- If one’s circle of influence is full of prestigious actors, then one’s own prestige is also high.
  - Thus one’s prestige is affected by the ranks or statuses of the involved actors.

# Rank prestige (cont ...)

- Based on this intuition, the rank prestige  $P_R(i)$  is defined as a linear combination of links that point to  $i$ :

$$P_R(i) = A_{1i}P_R(1) + A_{2i}P_R(2) + \dots + A_{ni}P_R(n), \quad (9)$$

where  $A_{ji} = 1$  if  $j$  points to  $i$ , and 0 otherwise. This equation says that an actor's rank prestige is a function of the ranks of the actors who vote or choose the actor, which makes perfect sense.

Since we have  $n$  equations for  $n$  actors, mathematically we can write them in the matrix notation. We use  $\mathbf{P}$  to represent the vector that contains all the rank prestige values, i.e.,  $\mathbf{P} = (P_R(1), P_R(2), \dots, P_R(n))^T$  ( $T$  means **matrix transpose**).  $\mathbf{P}$  is represented as a column vector. We use matrix  $\mathbf{A}$  (where  $A_{ij} = 1$  if  $i$  points to  $j$ , and 0 otherwise) to represent the adjacency matrix of the network or graph. As a notational convention, we use bold italic letters to represent matrices. We then have

$$\mathbf{P} = \mathbf{A}^T \mathbf{P} \quad (10)$$

This equation is precisely the characteristic equation used for finding the **eigensystem** of the matrix  $\mathbf{A}^T$ .  $\mathbf{P}$  is an **eigenvector** of  $\mathbf{A}^T$ .

# Road map

- **Introduction**
- **Social network analysis**
- **Co-citation and bibliographic coupling**

# Co-citation and Bibliographic Coupling

- Another area of research concerned with links is **citation analysis** of scholarly publications.
  - A scholarly publication cites related prior work to acknowledge the origins of some ideas and to compare the new proposal with existing work.
- When a paper cites another paper, a relationship is established between the publications.
  - Citation analysis uses these relationships (links) to perform various types of analysis.
- We discuss two types of citation analysis, **co-citation** and **bibliographic coupling**. The HITS algorithm is related to these two types of analysis.

# Co-citation

- If papers  $i$  and  $j$  are both cited by paper  $k$ , then they may be related in some sense to one another.
- The more papers they are cited by, the stronger their relationship is.

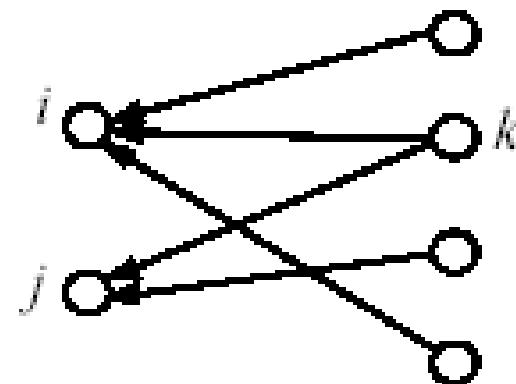


Fig. 2. Paper  $i$  and paper  $j$  are co-cited by paper  $k$

# Co-citation

- Let  $L$  be the citation matrix. Each cell of the matrix is defined as follows:
  - $L_{ij} = 1$  if paper  $i$  cites paper  $j$ , and 0 otherwise.
- **Co-citation** (denoted by  $C_{ij}$ ) is a similarity measure defined as the number of papers that co-cite  $i$  and  $j$ ,

$$C_{ij} = \sum_{k=1}^n L_{ki} L_{kj},$$

- A square matrix  $C$  can be formed with  $C_{ij}$ , and it is called the **co-citation matrix**.

# Bibliographic coupling

- Bibliographic coupling operates on a similar principle.
- Bibliographic coupling links papers that cite the same articles
  - if papers  $i$  and  $j$  both cite paper  $k$ , they may be related.
- The more papers they both cite, the stronger their similarity is.

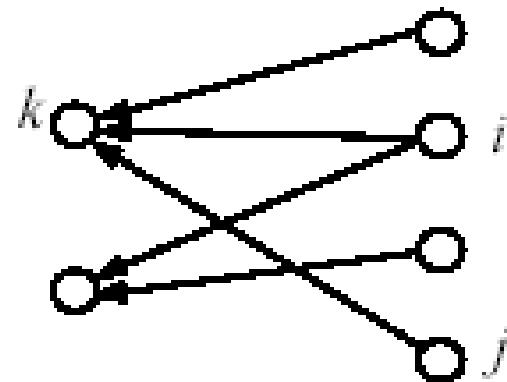


Fig. 3. Both paper  $i$  and paper  $j$  cite paper  $k$

# Bibliographic coupling (cont ...)

We use  $B_{ij}$  to represent the number of papers that are cited by both papers  $i$  and  $j$ .

$$B_{ij} = \sum_{k=1}^n L_{ik} L_{jk}. \quad (12)$$

$B_{ii}$  is naturally the number of references (in the reference list) of paper  $i$ . A square matrix  $B$  can be formed with  $B_{ij}$ , and it is called the **bibliographic coupling matrix**. Bibliographic coupling is also symmetric and is regarded as a similarity measure of two papers in clustering.

# Chapter 7: Link Analysis-2

---

**Instructor: Dr. Mehmet S. Aktaş**

Acknowledgement: Thanks to Dr. Bing Liu for teaching materials.

# Road map

- **PageRank**
- **HITS**
- **Summary**

# PageRank

- The year 1998 was an eventful year for Web link analysis models. Both the **PageRank** and **HITS** algorithms were reported in that year.
- The connections between PageRank and HITS are quite striking.
- Since that eventful year, PageRank has emerged as the dominant link analysis model,
  - due to its query-independence,
  - its ability to combat spamming, and
  - Google's huge business success.

# PageRank: the intuitive idea

- PageRank relies on the democratic nature of the Web by using its vast link structure as an indicator of an individual page's value or quality.
- PageRank interprets a hyperlink from page  $x$  to page  $y$  as a vote, by page  $x$ , for page  $y$ .
- However, PageRank looks at more than the sheer number of votes; it also analyzes the page that casts the vote.
  - Votes casted by “important” pages weigh more heavily and help to make other pages more “important.”
- This is exactly the idea of **rank prestige** in social network.

# More specifically

- A hyperlink from a page to another page is an implicit conveyance of authority to the target page.
  - The more in-links that a page  $i$  receives, the more prestige the page  $i$  has.
- Pages that point to page  $i$  also have their own prestige scores.
  - A page of a higher prestige pointing to  $i$  is more important than a page of a lower prestige pointing to  $i$ .
  - In other words, a page is important if it is pointed to by other important pages.

# PageRank algorithm

- According to **rank prestige**, the importance of page  $i$  ( $i$ 's PageRank score) is the sum of the PageRank scores of all pages that point to  $i$ .
- Since a page may point to many other pages, its prestige score should be shared.
- The Web as a directed graph  $G = (V, E)$ . Let the total number of pages be  $n$ . The PageRank score of the page  $i$  (denoted by  $P(i)$ ) is defined by:

$$P(i) = \sum_{(j,i) \in E} \frac{P(j)}{O_j}, \quad O_j \text{ is the number of out-link of } j$$

# Matrix notation

- We have a system of  $n$  linear equations with  $n$  unknowns. We can use a matrix to represent them.
- Let  $\mathbf{P}$  be a  $n$ -dimensional column vector of PageRank values, i.e.,  $\mathbf{P} = (P(1), P(2), \dots, P(n))^T$ .
- Let  $\mathbf{A}$  be the adjacency matrix of our graph with

$$A_{ij} = \begin{cases} \frac{1}{O_i} & \text{if } (i, j) \in E \\ 0 & \text{otherwise} \end{cases} \quad (14)$$

- We can write the  $n$  equations with (PageRank)

$$\mathbf{P} = \mathbf{A}^T \mathbf{P} \quad (15)$$

# Solve the PageRank equation

$$\mathbf{P} = \mathbf{A}^T \mathbf{P} \quad (15)$$

- This is the characteristic equation of the **eigensystem**, where the solution to  $\mathbf{P}$  is an **eigenvector** with the corresponding **eigenvalue** of 1.
- It turns out that if **some conditions** are satisfied, 1 is the largest **eigenvalue** and the PageRank vector  $\mathbf{P}$  is the **principal eigenvector**.
- A well known mathematical technique called **power iteration** can be used to find  $\mathbf{P}$ .
- **Problem:** the above Equation does not quite suffice because the Web graph does not meet the conditions.

# Using Markov chain

- To introduce these **conditions** and the enhanced equation, let us derive the same Equation (15) based on the **Markov chain**.
  - In the Markov chain, each Web page or node in the Web graph is regarded as a state.
  - A hyperlink is a transition, which leads from one state to another state with a probability.
- This framework models Web surfing as a stochastic process.
- It models **a Web surfer** randomly surfing the Web as state transition.

# Random surfing

- Recall we use  $O_i$  to denote the number of out-links of a node  $i$ .
- Each transition probability is  $1/O_i$ , if we assume the Web surfer will click the hyperlinks in the page  $i$  uniformly at random.
  - The “back” button on the browser is not used and
  - the surfer does not type in an URL.

# Transition probability matrix

- Let  $A$  be the state transition probability matrix,,

$$A = \begin{pmatrix} A_{11} & A_{12} & \cdot & \cdot & \cdot & A_{1n} \\ A_{21} & A_{22} & \cdot & \cdot & \cdot & A_{2n} \\ \cdot & \cdot & & & & \cdot \\ \cdot & \cdot & & & & \cdot \\ \cdot & \cdot & & & & \cdot \\ A_{n1} & A_{n2} & \cdot & \cdot & \cdot & A_{nn} \end{pmatrix}$$

- $A_{ij}$  represents the transition probability that the surfer in state  $i$  (page  $i$ ) will move to state  $j$  (page  $j$ ).  $A_{ij}$  is defined exactly as in Equation (14).

---

$$A_{ij} = \begin{cases} \frac{1}{O_i} & \text{if } (i, j) \in E \\ 0 & \text{otherwise} \end{cases}$$

# Let us start

- Given an **initial probability distribution** vector that a surfer is at each state (or page)
  - $\mathbf{p}_0 = (p_0(1), p_0(2), \dots, p_0(n))^T$  (a column vector) and
  - an  $n \times n$  **transition probability matrix  $A$** ,
- we have

$$\sum_{i=1}^n p_0(i) = 1 \quad (16)$$

$$\sum_{j=1}^n A_{ij} = 1 \quad (17)$$

- If the matrix  $A$  satisfies Equation (17), we say that  $A$  is the **stochastic matrix** of a Markov chain.

# Back to the Markov chain

- In a Markov chain, a question of common interest is:
  - Given  $p_0$  at the beginning, what is the probability that  $m$  steps/transitions later the Markov chain will be at each state  $j$ ?
- We determine the probability that the system (or the random surfer) is in state  $j$  after 1 step (1 transition) by using the following reasoning:

$$p_1(j) = \sum_{i=1}^n A_{ij}(1) p_0(i) \quad (18)$$

# State transition

$$p_1(j) = \sum_{i=1}^n A_{ij}(1)p_0(i), \quad (18)$$

where  $A_{ij}(1)$  is the probability of going from  $i$  to  $j$  after 1 transition, and  $A_{ij}(1) = A_{ij}$ . We can write it with a matrix:

$$p_1 = A^T p_0 \quad (19)$$

In general, the probability distribution after  $k$  steps/transitions is:

$$p_k = A^T p_{k-1} \quad (20)$$

# Stationary probability distribution

- By a Theorem of the Markov chain,
  - a finite Markov chain defined by the **stochastic matrix  $A$**  has a unique **stationary probability distribution** if  $A$  is **irreducible** and **aperiodic**.
- The stationary probability distribution means that after a series of transitions  $p_k$  will converge to a steady-state probability vector  $\pi$  regardless of the choice of the initial probability vector  $p_0$ , i.e.,

$$\lim_{k \rightarrow \infty} p_k = \pi \quad (21)$$

# PageRank again

- When we reach the steady-state, we have  $p_k = p_{k+1} = \pi$ , and thus

$$\pi = A^T \pi.$$

- $\pi$  is the **principal eigenvector** of  $A^T$  with **eigenvalue** of 1.
- In PageRank,  $\pi$  is used as the PageRank vector  $P$ . We again obtain Equation (15), which is re-produced here as Equation (22):

$$P = A^T P \tag{22}$$

# Is $P = \pi$ justified?

- Using the stationary probability distribution  $\pi$  as the PageRank vector is reasonable and quite intuitive because
  - it reflects the long-run probabilities that a random surfer will visit the pages.
  - A page has a high prestige if the probability of visiting it is high.

# Back to the Web graph

- Now let us come back to the real Web context and see whether the above conditions are satisfied, i.e.,
  - whether  $A$  is a **stochastic matrix** and
  - whether it is **irreducible** and **aperiodic**.
- None of them is satisfied.
- Hence, we need to extend the ideal-case Equation (22) to produce the “actual PageRank” model.

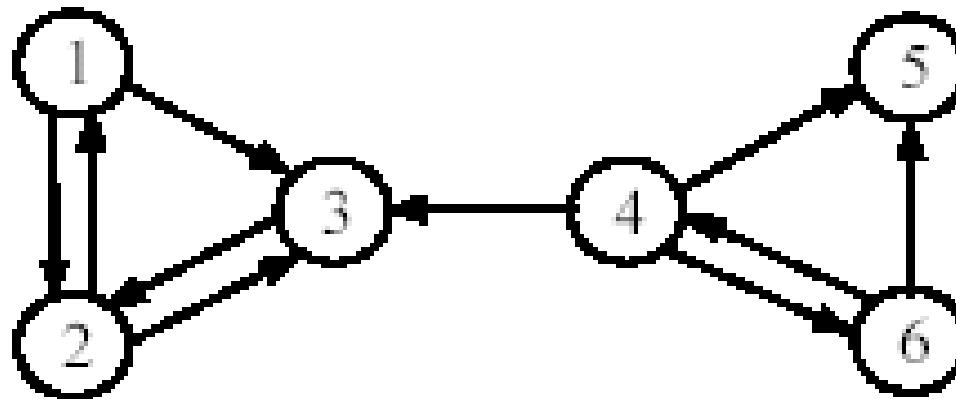
# $A$ is a not stochastic matrix

- $A$  is the transition matrix of the Web graph

$$A_{ij} = \begin{cases} \frac{1}{O_i} & \text{if } (i, j) \in E \\ 0 & \text{otherwise} \end{cases}$$

- It does not satisfy equation (17)  
$$\sum_{j=1}^n A_{ij} = 1$$
- because many Web pages have no out-links, which are reflected in transition matrix  $A$  by some rows of complete 0's.
  - Such pages are called the **dangling pages** (nodes).

# An example Web hyperlink graph



$$A = \begin{pmatrix} 0 & 1/2 & 1/2 & 0 & 0 & 0 \\ 1/2 & 0 & 1/2 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1/3 & 0 & 1/3 & 1/3 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1/2 & 1/2 & 0 \end{pmatrix}$$

# Fix the problem: two possible ways

1. Remove those pages with no out-links during the PageRank computation as these pages do not affect the ranking of any other page directly.
2. Add a complete set of outgoing links from each such page  $i$  to all the pages on the Web.

Let us use the second way

$$\bar{A} = \begin{pmatrix} 0 & 1/2 & 1/2 & 0 & 0 & 0 \\ 1/2 & 0 & 1/2 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1/3 & 0 & 1/3 & 1/3 \\ 1/6 & 1/6 & 1/6 & 1/6 & 1/6 & 1/6 \\ 0 & 0 & 0 & 1/2 & 1/2 & 0 \end{pmatrix}$$

# $A$ is a not irreducible

- Irreducible means that the Web graph  $G$  is **strongly connected**.

**Definition:** A directed graph  $G = (V, E)$  is **strongly connected** if and only if, for each pair of nodes  $u, v \in V$ , there is a path from  $u$  to  $v$ .

- A general Web graph represented by  $A$  is not irreducible because
  - for some pair of nodes  $u$  and  $v$ , there is no path from  $u$  to  $v$ .
  - In our example, there is no directed path from nodes 3 to 4.

# $A$ is a not aperiodic

- A state  $i$  in a Markov chain being **periodic** means that there exists a directed cycle that the chain has to traverse.

**Definition:** A state  $i$  is **periodic** with period  $k > 1$  if  $k$  is the smallest number such that all paths leading from state  $i$  back to state  $i$  have a length that is a multiple of  $k$ .

- If a state is not periodic (i.e.,  $k = 1$ ), it is **aperiodic**.
- A Markov chain is **aperiodic** if all states are aperiodic.

## An example: periodic

- Fig. 5 shows a periodic Markov chain with  $k = 3$ . Eg, if we begin from state 1, to come back to state 1 the only path is 1-2-3-1 for some number of times, say  $h$ . Thus any return to state 1 will take  $3h$  transitions.

$$A = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}$$

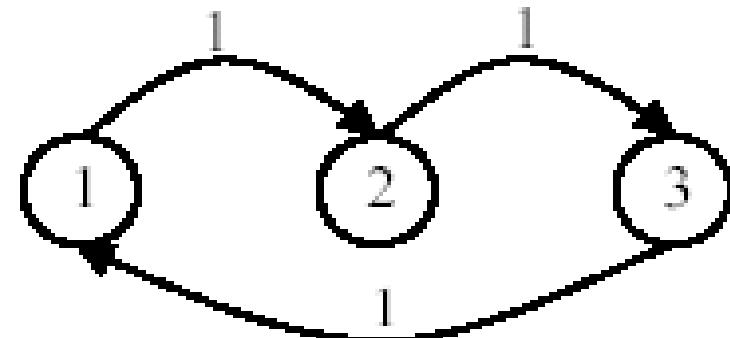


Fig. 5. A Periodic Markov chain with  $k = 3$ .

# Deal with irreducible and aperiodic

- It is easy to deal with the above two problems with a single strategy.
- Add a link from each page to every page and give each link a small transition probability controlled by a parameter  $d$ .
- Obviously, the augmented transition matrix becomes **irreducible** and **aperiodic**

# Improved PageRank

- After this augmentation, at a page, the random surfer has two options
  - With probability  $d$ , he randomly chooses an out-link to follow.
  - With probability  $1-d$ , he jumps to a random page
- Equation (25) gives the improved model,

$$\mathbf{P} = ((1-d) \frac{\mathbf{E}}{n} + d\mathbf{A}^T) \mathbf{P} \quad (25)$$

where  $\mathbf{E}$  is  $\mathbf{e}\mathbf{e}^T$  ( $\mathbf{e}$  is a column vector of all 1's) and thus  $\mathbf{E}$  is a  $n \times n$  square matrix of all 1's.

# Follow our example

$$(1-d)\frac{\mathbf{E}}{n} + d\mathbf{A}^T = \begin{pmatrix} 1/60 & 7/15 & 1/60 & 1/60 & 1/6 & 1/60 \\ 7/15 & 1/60 & 11/12 & 1/60 & 1/6 & 1/60 \\ 7/15 & 7/15 & 1/60 & 19/60 & 1/6 & 1/60 \\ 1/60 & 1/60 & 1/60 & 1/60 & 1/6 & 7/15 \\ 1/60 & 1/60 & 1/60 & 19/60 & 1/6 & 7/15 \\ 1/60 & 1/60 & 1/60 & 19/60 & 1/6 & 1/60 \end{pmatrix}$$

# The final PageRank algorithm

- $(1-d)\mathbf{E}/n + d\mathbf{A}^T$  is a **stochastic matrix** (transposed). It is also **irreducible** and **aperiodic**
- If we scale Equation (25) so that  $\mathbf{e}^T \mathbf{P} = n$ ,

$$\mathbf{P} = (1 - d)\mathbf{e} + d\mathbf{A}^T \mathbf{P} \quad (27)$$

- PageRank for each page  $i$  is

$$P(i) = (1 - d) + d \sum_{j=1}^n A_{ji} P(j) \quad (28)$$

# The final PageRank (cont ...)

- (28) is equivalent to the formula given in the PageRank paper

$$P(i) = (1 - d) + d \sum_{(j,i) \in E} \frac{P(j)}{O_j}$$

- The parameter  $d$  is called the **damping factor** which can be set to between 0 and 1.  $d = 0.85$  was used in the PageRank paper.

# Compute PageRank

- Use the **power iteration** method

PageRank-Iterate( $G$ )

$$P_0 \leftarrow e/n$$

$$k = 1$$

repeat

$$P_{k+1} \leftarrow (1-d)e + dA^T P_k ;$$

$$k = k + 1;$$

until  $||P_{k+1} - P_k||_1 < \varepsilon$

return  $P_{k+1}$

Fig. 6. The power iteration method for PageRank

# Advantages of PageRank

- **Fighting spam.** A page is important if the pages pointing to it are important.
  - Since it is not easy for Web page owner to add in-links into his/her page from other important pages, it is thus not easy to influence PageRank.
- **PageRank is a global measure and is query independent.**
  - PageRank values of all the pages are computed and saved off-line rather than at the query time.
- **Criticism:** Query-independence. It could not distinguish between pages that are authoritative in general and pages that are authoritative on the query topic.

# Road map

- PageRank
- HITS
- Summary

# HITS

- HITS stands for **Hypertext Induced Topic Search.**
- Unlike PageRank which is a static ranking algorithm, HITS is search query dependent.
- When the user issues a search query,
  - HITS first expands the list of relevant pages returned by a search engine and
  - then produces two rankings of the expanded set of pages, **authority ranking** and **hub ranking**.

# Authorities and Hubs

**Authority:** Roughly, a authority is a page with many in-links.

- The idea is that the page may have good or authoritative content on some topic and
- thus many people trust it and link to it.

**Hub:** A hub is a page with many out-links.

- The page serves as an organizer of the information on a particular topic and
- points to many good authority pages on the topic.

# Examples

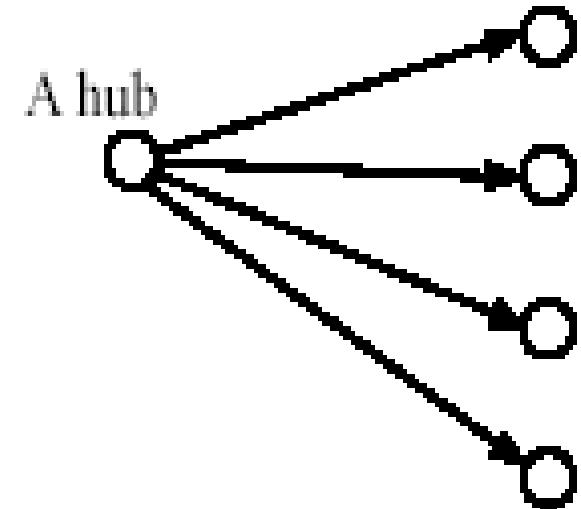
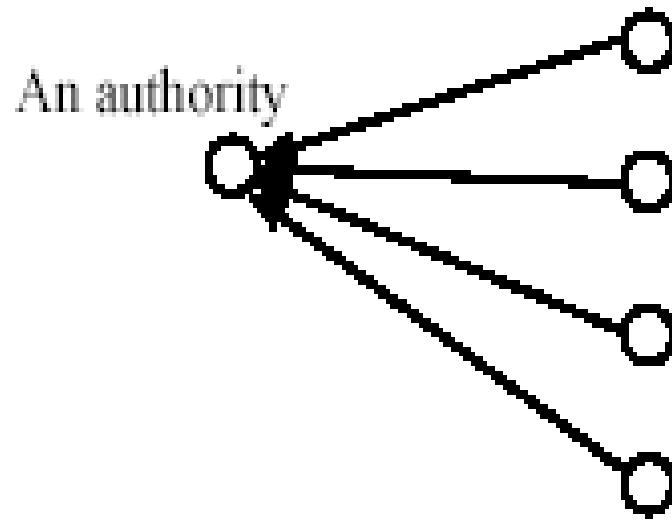


Fig. 7. An authority page and a hub page

# The key idea of HITS

- A good hub points to many good authorities, and
- A good authority is pointed to by many good hubs.
- Authorities and hubs have a **mutual reinforcement relationship**. Fig. 8 shows some densely linked authorities and hubs (a **bipartite sub-graph**).

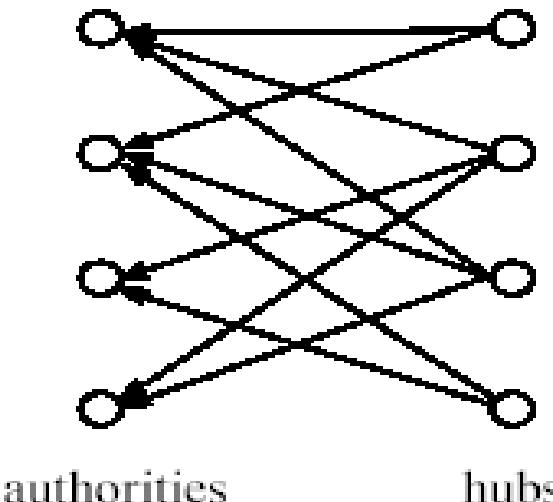


Fig. 8. A densely linked set of authorities and hubs

# The HITS algorithm: Grab pages

- Given a broad search query,  $q$ , HITS collects a set of pages as follows:
  - It sends the query  $q$  to a search engine.
  - It then collects  $t$  ( $t = 200$  is used in the HITS paper) highest ranked pages. This set is called the **root** set  $W$ .
  - It then grows  $W$  by including any page pointed to by a page in  $W$  and any page that points to a page in  $W$ . This gives a larger set  $S$ , **base set**.

# The link graph $G$

- HITS works on the pages in  $S$ , and assigns every page in  $S$  an **authority score** and a **hub score**.
- Let the number of pages in  $S$  be  $n$ .
- We again use  $G = (V, E)$  to denote the hyperlink graph of  $S$ .
- We use  $L$  to denote the adjacency matrix of the graph.

$$L_{ij} = \begin{cases} 1 & \text{if } (i, j) \in E \\ 0 & \text{otherwise} \end{cases}$$

# The HITS algorithm

- Let the authority score of the page  $i$  be  $a(i)$ , and the hub score of page  $i$  be  $h(i)$ .
- The mutual reinforcing relationship of the two scores is represented as follows:

$$a(i) = \sum_{(j,i) \in E} h(j) \quad (31)$$

$$h(i) = \sum_{(i,j) \in E} a(j) \quad (32)$$

# HITS in matrix form

- We use  $\mathbf{a}$  to denote the column vector with all the authority scores,

$$\mathbf{a} = (a(1), a(2), \dots, a(n))^T, \text{ and}$$

- use  $\mathbf{h}$  to denote the column vector with all the hub scores,

$$\mathbf{h} = (h(1), h(2), \dots, h(n))^T,$$

- Then,

$$\mathbf{a} = \mathbf{L}^T \mathbf{h} \tag{33}$$

$$\mathbf{h} = \mathbf{L} \mathbf{a} \tag{34}$$

# Computation of HITS

- The computation of authority scores and hub scores is the same as the computation of the PageRank scores, using **power iteration**.
- If we use  $a_k$  and  $h_k$  to denote authority and hub vectors at the  $k$ th iteration, the iterations for generating the final solutions are

$$a_k = L^T La_{k-1} \quad (35)$$

$$h_k = LL^T h_{k-1} \quad (36)$$

starting with

$$a_0 = h_0 = (1, 1, \dots, 1), \quad (37)$$

# The algorithm

**HITS-Iterate( $G$ )**

$a_0 = h_0 = (1, 1, \dots, 1);$   
 $k = 1$

**Repeat**

$a_k = L^T La_{k-1};$

$h_k = LL^T h_{k-1};$

normalize  $a_k$ ;

normalize  $h_k$ ;

$k = k + 1;$

**until**  $a_k$  and  $h_k$  do not change significantly;  
return  $a_k$  and  $h_k$

, Fig. 9. The HITS algorithm based on power iteration

# Relationships with co-citation and bibliographic coupling

- Recall that co-citation of pages  $i$  and  $j$ , denoted by  $C_{ij}$ , is 
$$C_{ij} = \sum_{k=1}^n L_{ki} L_{kj} = (\mathbf{L}^T \mathbf{L})_{ij}$$
  - the authority matrix  $(\mathbf{L}^T \mathbf{L})$  of HITS is the co-citation matrix  $\mathbf{C}$
- bibliographic coupling of two pages  $i$  and  $j$ , denoted by  $B_{ij}$  is 
$$B_{ij} = \sum_{k=1}^n L_{ik} L_{jk} = (\mathbf{L} \mathbf{L}^T)_{ij},$$
  - the hub matrix  $(\mathbf{L} \mathbf{L}^T)$  of HITS is the bibliographic coupling matrix  $\mathbf{B}$

# Strengths and weaknesses of HITS

- **Strength:** its ability to rank pages according to the query topic, which may be able to provide more relevant authority and hub pages.
- **Weaknesses:**
  - **It is easily spammed.** It is in fact quite easy to influence HITS since adding out-links in one's own page is so easy.
  - **Topic drift.** Many pages in the expanded set may not be on topic.
  - **Inefficiency at query time:** The query time evaluation is slow. Collecting the root set, expanding it and performing eigenvector computation are all expensive operations

# Road map

- PageRank
- HITS
- Summary

# Summary

- In Link Analysis chapter, we introduced
  - Social network analysis, centrality and prestige
  - Co-citation and bibliographic coupling
  - PageRank, which powers Google
  - HITS
- Yahoo! and MSN have their own link-based algorithms as well, but not published.
- **Important to note:** Hyperlink based ranking is not the only algorithm used in search engines. In fact, it is combined with many **content based factors** to produce the final ranking presented to the user.

# Summary

- Links can also be used to find **communities**, which are groups of content-creators or people sharing some common interests.
  - Web communities
  - Email communities
  - Named entity communities
- Focused crawling: combining contents and links to crawl Web pages of a specific topic.
  - Follow links and
  - Use learning/classification to determine whether a page is on topic.

# Chapter 8 - Web Crawling

Web Mining Lecture

Computer Engineering Department, Faculty of Electric and Electronics

Yıldız Technical University

Instructor: Dr. Mehmet Aktaş

Acknowledgement: Thanks to Dr. Fillippo Menczer and Dr. Bing Lui for Teaching Materials

Slides provided By Dr. Filippo Menczer

Indiana University School of Informatics



# Outline

- Motivation and taxonomy of crawlers
- Basic crawlers and implementation issues
- Universal crawlers
- Preferential (focused and topical) crawlers



Google Search: spears

Web Images Groups News Froogle more » spears Search Advanced Search Preferences

Results 1 - 10 of about 9,440,000 for spears [definition]. (0.14 seconds)

**Web**

[News results for spears](#) - View today's top stories

 [Knee Injury Closes Spears' Onyx Hotel](#) - Billboard - 1 hour ago  
[Britney Spears' tour is canceled](#) - San Diego Union Tribune - 7 hours ago  
[As fall approaches, Spears may start to smell Curious](#) - Houston Chronicle - Jun 14, 2004

[Britney Spears :: The Official Web Site](#)  
The Official Web Site of Britney Spears. Your official source for all things Britney. ...  
Remember, proceeds benefit the Britney Spears Foundation. ...  
[www.britneyspears.com/](#) - 41k - Jun 14, 2004 - [Cached](#) - [Similar pages](#)

[Britney Spears - britney.com - Jive Records](#)  
iTunes. Real/Rhapsody. Napster. Under 11.  
[www.britney.com/](#) - 10k - [Cached](#) - [Similar pages](#)

[Britney Spears Portal - pics, lyrics, MP3s and more!](#)  
Britney Spears pics, lyrics, MP3s, news, gossip, fan sites, forums, and much more!  
Britney Spears Portal, ... ); Britney Spears Portal, ...  
[www.britney-spears-portal.com/](#) - 25k - [Cached](#) - [Similar pages](#)

[Britney Spears guide to Semiconductor Physics: semiconductor ...](#)  
Britney Spears lectures on semiconductor physics, radiative and non-radiative transitions, edge emitting lasers and VCSELs, ...  
[britneyspears.ac/lasers.htm](#) - 13k - [Cached](#) - [Similar pages](#)

[BritneySpears.org: Your online guide to Britney!](#)  
A comprehensive Britney Spears fansite which pays tribute to Britney with the most active message board, daily news, many pictures, desktop media and more. ...  
[www.britneyspears.org/](#) - 78k - Jun 14, 2004 - [Cached](#) - [Similar pages](#)

[Britney-Spears.To You! - The Britney Spears Community](#)  
Britney Spears : biography, discography, musics, real, mp3, videos, pictures, clips, guestbook, www board, free page, search engine, links and more. ...  
[www.britney-spears.to/](#) - 9k - [Cached](#) - [Similar pages](#)

[The Mystery of Britney's Breasts](#)  
[www.liquidgeneration.com/poptoons/britneys\\_breasts.asp](#) - 2k - [Cached](#) - [Similar pages](#)

[Britney Spears spelling correction](#)  
The data below shows some of the misspellings detected by our spelling correction system for the query [ britney spears ], and the count of how many different ...  
[www.google.com/jobs/britney.html](#) - 40k - [Cached](#) - [Similar pages](#)

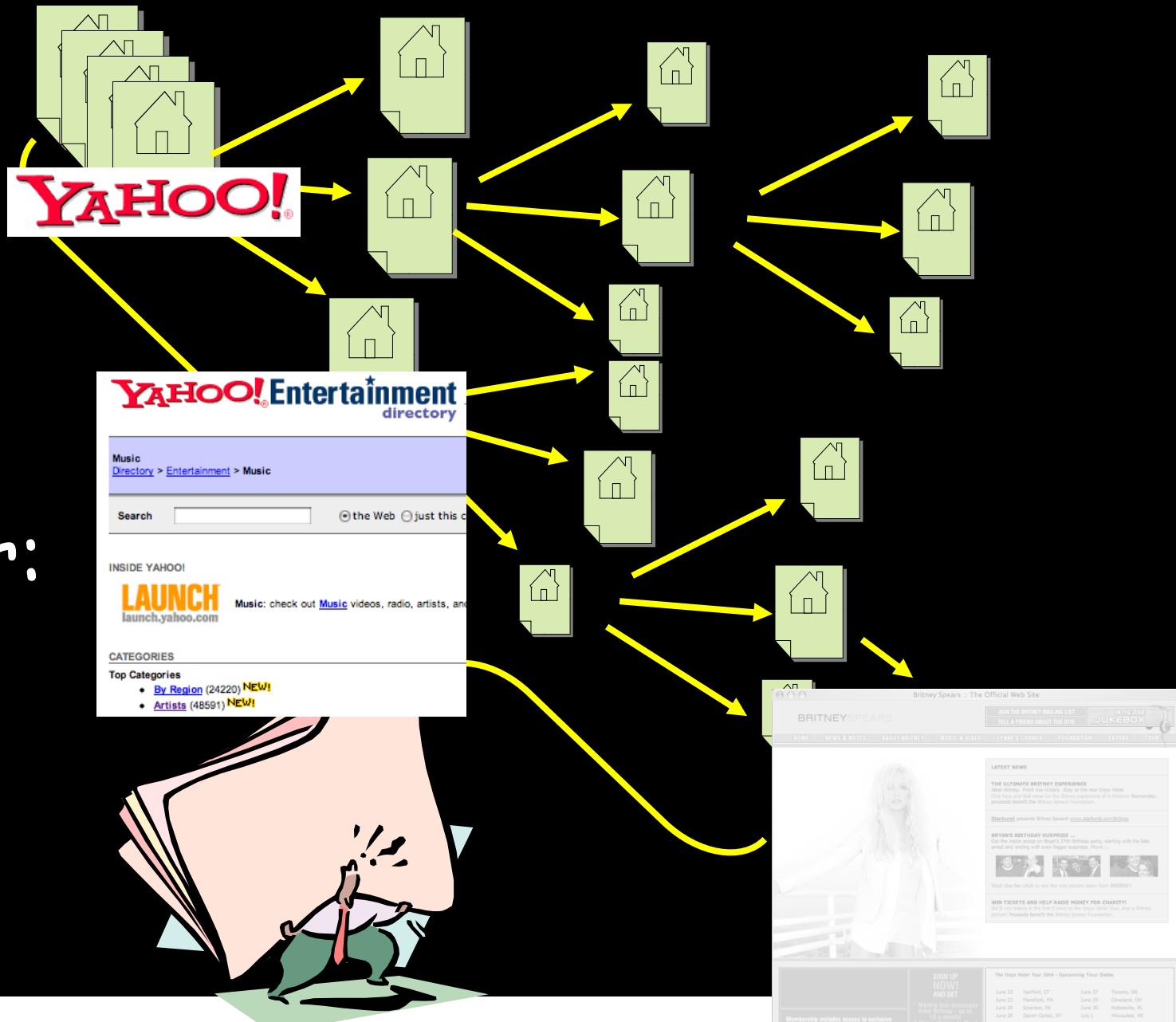
[Britney Spears pictures news music Britney Spears lyrics](#)  
Britney Spears pictures mp3 sites gallery photos images music fun games chat lyrics. ...  
Britney Spears Forum Come see what is inside the Britney Spears forum! ...  
[www.britney-spears.com/](#) - 42k - Jun 14, 2004 - [Cached](#) - [Similar pages](#)

[Britney Spears Zone - Your Guide to Britney Pictures and News](#)  
Britney Spears, Britney Spears, Britney Spears, ... Britney Spears, ...  
[www.britneyzone.com/](#) - 101k - Jun 14, 2004 - [Cached](#) - [Similar pages](#)

Q: How does a search engine know that all these pages contain the query terms?

A: Because all of those pages have been crawled

# Crawler: basic idea



# Many names

- Crawler
- Spider
- Robot (or bot)
- Web agent
- Wanderer, worm, ...
- And famous instances: googlebot, scooter, slurp, msnbot, ...

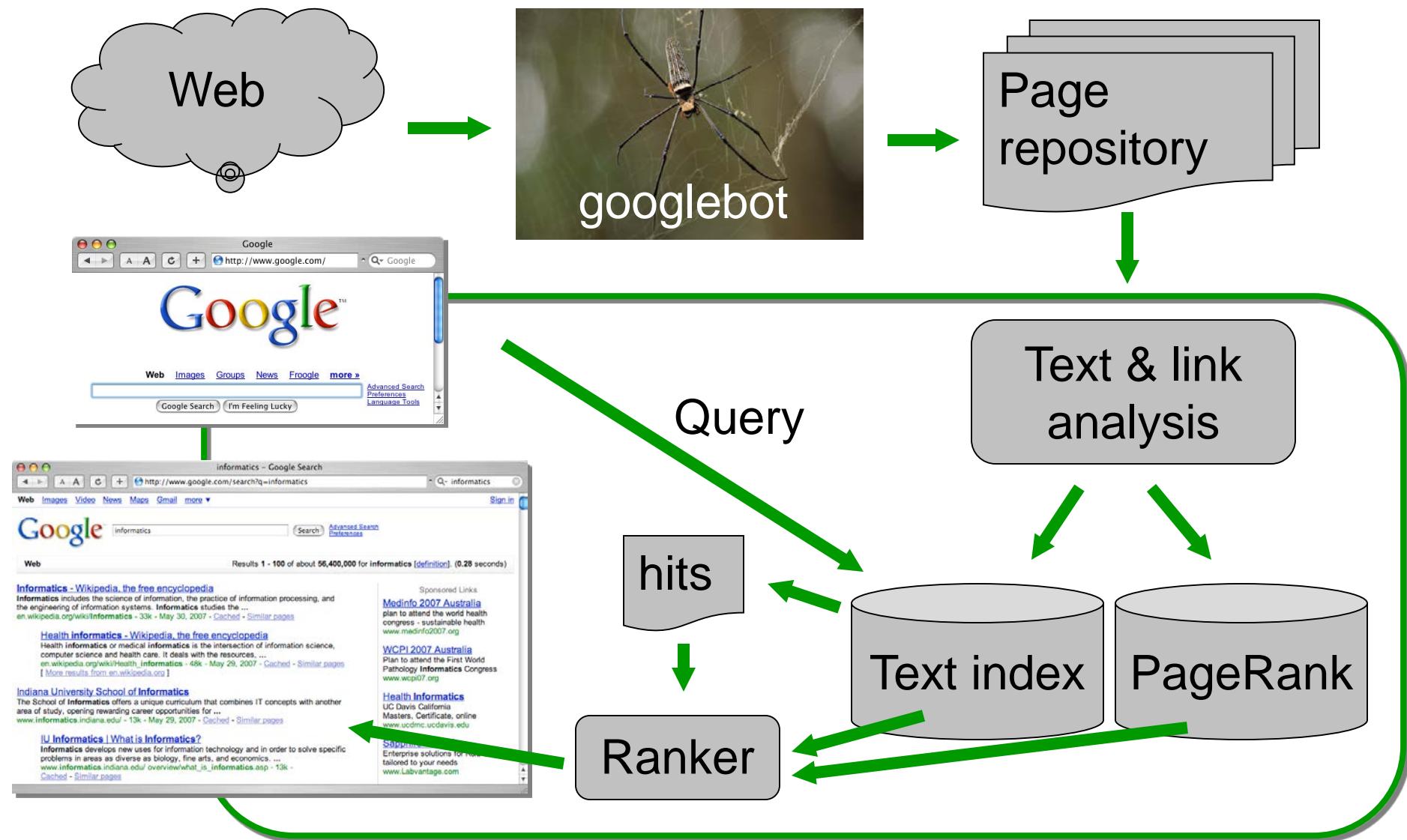


# Motivation for crawlers

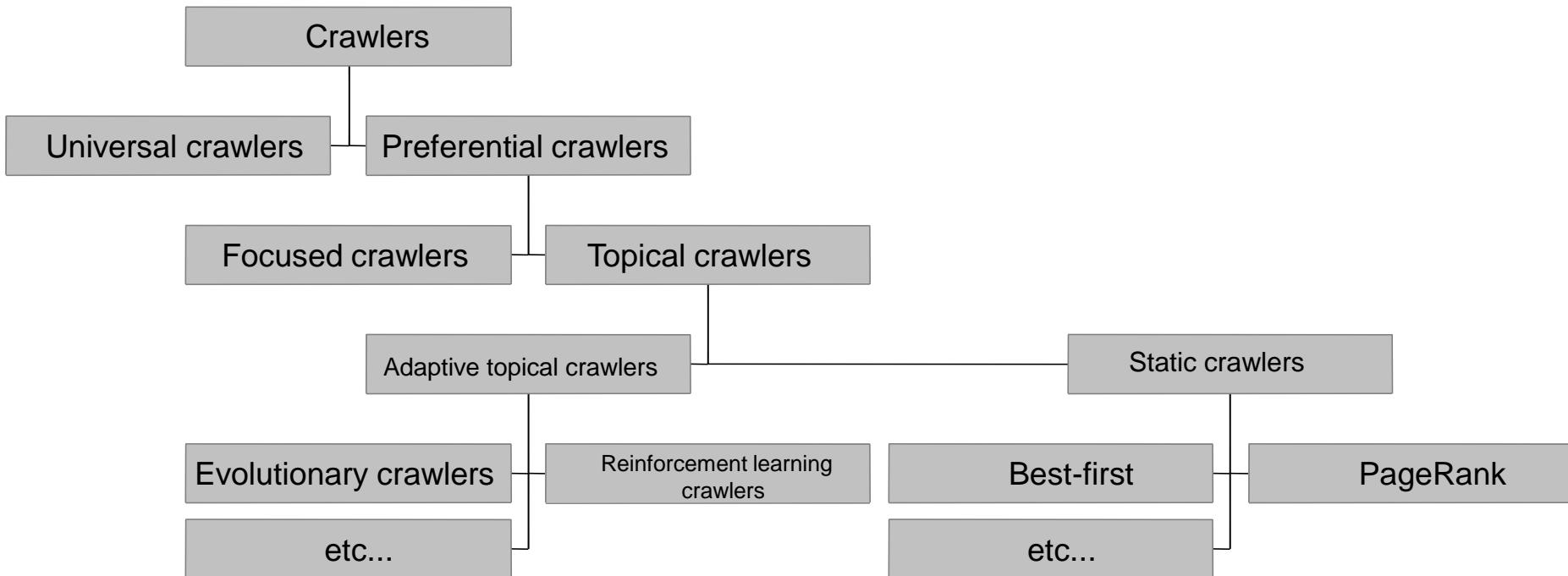
- Support universal search engines (Google, Yahoo, MSN/Windows Live, Ask, etc.)
- Vertical (specialized) search engines, e.g. news, shopping, papers, recipes, reviews, etc.
- Business intelligence: keep track of potential competitors, partners
- Monitor Web sites of interest
- Evil: harvest emails for spamming, phishing...
- ... Can you think of some others?...



# A crawler within a search engine



# One taxonomy of crawlers

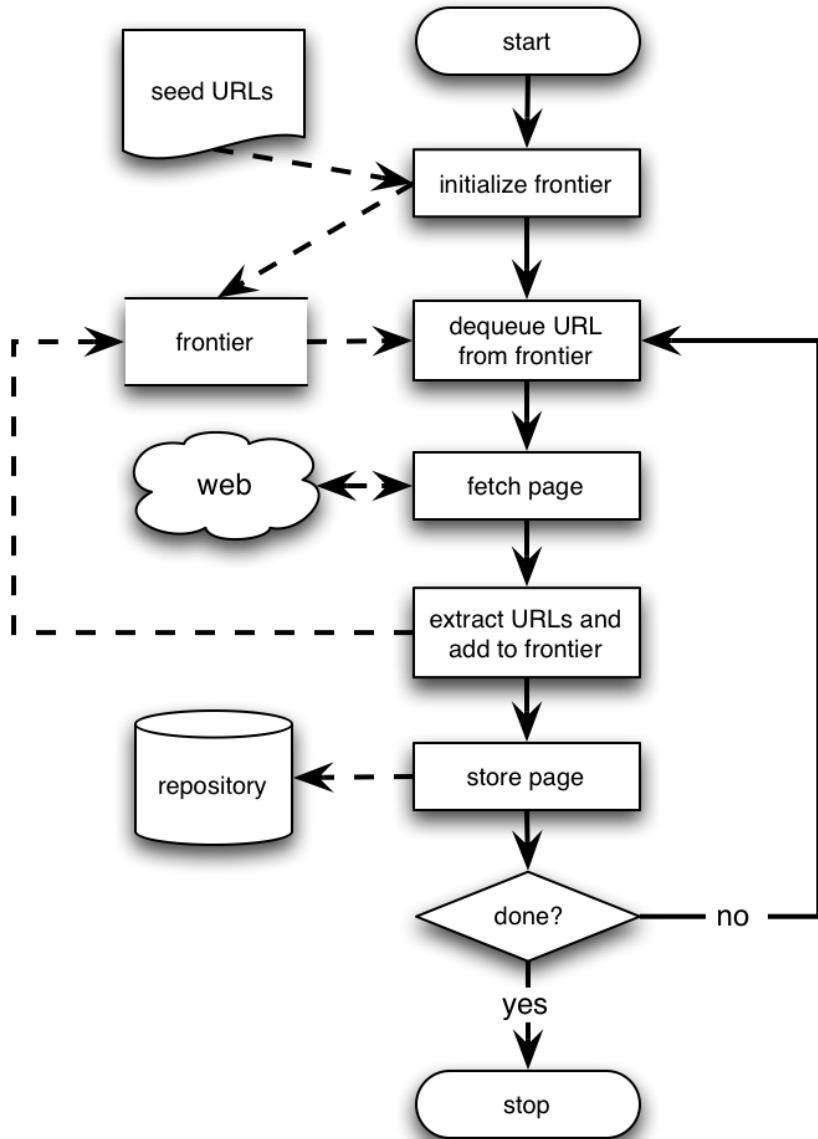


- Many other criteria could be used:
  - Incremental, Interactive, Concurrent, Etc.

# Outline

- Motivation and taxonomy of crawlers
- Basic crawlers and implementation issues
- Universal crawlers
- Preferential (focused and topical) crawlers



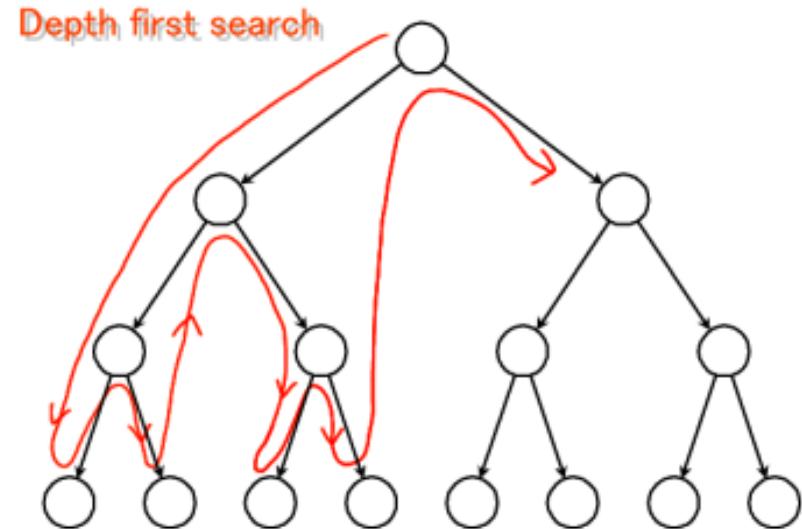
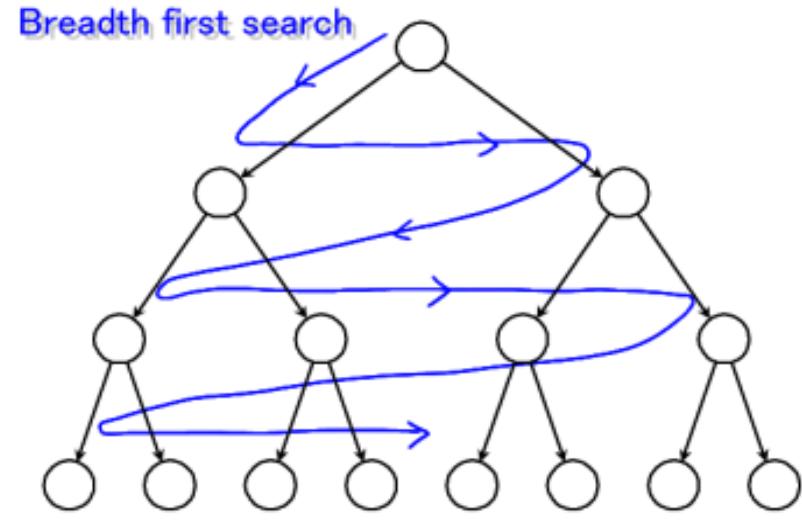


# Basic crawlers

- This is a **sequential** crawler
- **Seeds** can be any list of starting URLs
- Order of page visits is determined by **frontier** data structure
- **Stop** criterion can be anything

# Graph traversal (BFS or DFS?)

- Breadth First Search
  - Implemented with QUEUE (FIFO)
  - Finds pages along shortest paths
  - If we start with “good” pages, this keeps us close; maybe other good stuff...
- Depth First Search
  - Implemented with STACK (LIFO)
  - Wander away (“lost in cyberspace”)



# A basic crawler in Perl

- Queue: a FIFO list (shift and push)

```
my @frontier = read_seeds($file);
while (@frontier && $tot < $max) {
    my $next_link = shift @frontier;
    my $page = fetch($next_link);
    add_to_index($page);
    my @links = extract_links($page, $next_link);
    push @frontier, process(@links);
}
```



# Implementation issues

- Don't want to fetch same page twice!
  - Keep lookup table (hash) of visited pages
  - What if not visited but in frontier already?
- The frontier grows very fast!
  - May need to prioritize for large crawls
- Fetcher must be robust!
  - Don't crash if download fails
  - Timeout mechanism
- Determine file type to skip unwanted files
  - Can try using extensions, but not reliable
  - Can issue 'HEAD' HTTP commands to get Content-Type (MIME) headers, but overhead of extra Internet requests



# More implementation issues

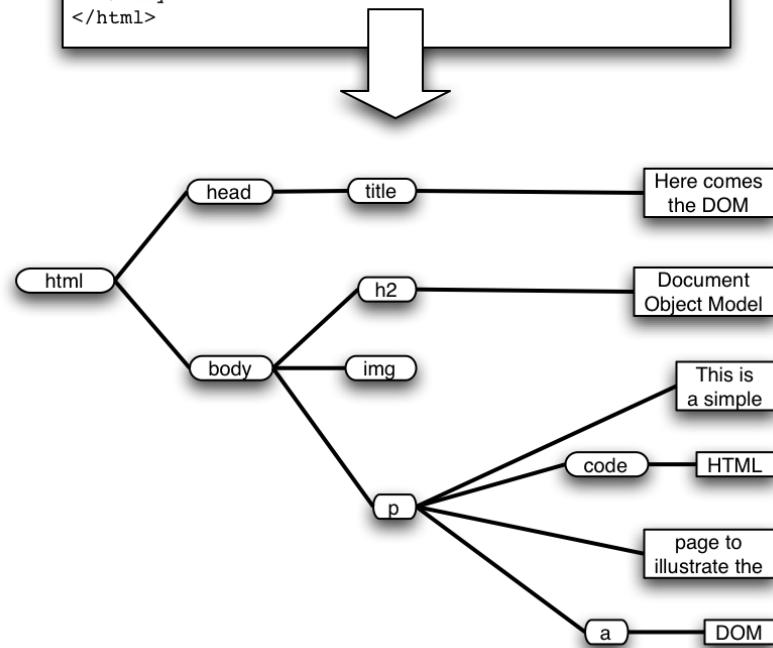
- **Fetching**
  - Get only the first 10-100 KB per page
  - Soft fail for timeout, server not responding, file not found, and other errors



# More implementation issues: Parsing

- HTML has the structure of a DOM (Document Object Model) tree
- Unfortunately actual HTML is often incorrect in a strict syntactic sense
- Crawlers, like browsers, must be robust/forgiving
- Fortunately there are tools that can help
  - E.g. [tidy.sourceforge.net](http://tidy.sourceforge.net)
- Must pay attention to HTML entities and unicode in text
- What to do with a growing number of other formats?
  - Flash, RSS, AJAX...

```
<html>
  <head>
    <title>Here comes the DOM</title>
  </head>
  <body>
    <h2>Document Object Model</h2>
    
    <p>
      This is a simple
      <code>HTML</code>
      page to illustrate the
      <a href="http://www.w3.org/DOM/">DOM</a>
    </p>
  </body>
</html>
```



# More implementation issues

- **Stop words**

- Noise words that do not carry meaning should be eliminated (“stopped”) before they are indexed
- E.g. in English: AND, THE, A, AT, OR, ON, FOR, etc...
- Typically syntactic markers
- Typically the most common terms
- Typically kept in a negative dictionary
  - 10-1,000 elements
  - E.g. [http://ir.dcs.gla.ac.uk/resources/linguistic\\_utils/stop\\_words](http://ir.dcs.gla.ac.uk/resources/linguistic_utils/stop_words)
- Parser can detect these right away and disregard them



# More implementation issues

## Conflation and synonyms

- Idea: improve **recall** by merging words with **same meaning**
- 1. We want to ignore superficial **morphological features**, thus merge semantically similar tokens
  - {student, study, studying, studious} => studi
- 2. We can also conflate **synonyms** into a single form using a thesaurus
  - 30-50% smaller index
  - Doing this in both pages and queries allows to retrieve pages about ‘automobile’ when user asks for ‘car’
  - Thesaurus can be implemented as a hash table



# More implementation issues

- **Stemming**

- Morphological conflation based on rewrite rules
- Language dependent!
- Porter stemmer very popular for English
  - <http://www.tartarus.org/~martin/PorterStemmer/>
  - Context-sensitive grammar rules, eg:
    - “IES” except (“EIES” or “AIES”) --> “Y”
  - Versions in Perl, C, Java, Python, C#, Ruby, PHP, etc.
- Porter has also developed Snowball, a language to create stemming algorithms in any language
  - <http://snowball.tartarus.org/>
  - Ex. Perl modules: [Lingua::Stem](#) and [Lingua::Stem::Snowball](#)



# More implementation issues

- **Static vs. dynamic pages**

- Is it worth trying to eliminate dynamic pages and only index static pages?
- Examples:
  - <http://www.census.gov/cgi-bin/gazetteer>
  - <http://informatics.indiana.edu/research/colloquia.asp>
  - <http://www.amazon.com/exec/obidos/subst/home/home.html/002-8332429-6490452>
  - <http://www.imdb.com/Name?Menczer,+Erico>
  - <http://www.imdb.com/name/nm0578801/>
- Why or why not? How can we tell if a page is dynamic? What about ‘spider traps’?
- What do Google and other search engines do?



# More implementation issues

- **Relative vs. Absolute URLs**

- Crawler must translate relative URLs into absolute URLs
- Need to obtain Base URL from HTTP header, or HTML Meta tag, or else current page path by default
- Examples
  - **Base:** <http://www.cnn.com/linkto/>
  - **Relative URL:** intl.html
  - **Absolute URL:** <http://www.cnn.com/linkto/intl.html>
  - **Relative URL:** /US/
  - **Absolute URL:** <http://www.cnn.com/US/>



# More implementation issues

- URL canonicalization
  - All of these:
    - <http://www.cnn.com/TECH>
    - <http://WWW.CNN.COM/TECH/>
    - <http://www.cnn.com:80/TECH/>
    - <http://www.cnn.com/bogus/./TECH/>
  - Are really equivalent to this canonical form:
    - <http://www.cnn.com/TECH>
  - In order to avoid duplication, the crawler must transform all URLs into canonical form
  - Definition of “canonical” is arbitrary, e.g.:
    - Could always include port
    - Or only include port when not default :80



# More on Canonical URLs

- Some transformation are trivial, for example:

- ✗ <http://informatics.indiana.edu>
- ✓ <http://informatics.indiana.edu/>
- ✗ <http://informatics.indiana.edu/index.html#fragment>
- ✓ <http://informatics.indiana.edu/index.html>
- ✗ <http://informatics.indiana.edu/dir1/../../dir2/>
- ✓ <http://informatics.indiana.edu/dir2/>
- ✗ <http://informatics.indiana.edu/%7Efil/>
- ✓ <http://informatics.indiana.edu/~fil/>
- ✗ <http://INFORMATICS.INDIANA.EDU/fil/>
- ✓ <http://informatics.indiana.edu/fil/>



# More implementation issues

- **Spider traps**
  - Misleading sites: indefinite number of pages dynamically generated by CGI scripts
  - Paths of arbitrary depth created using soft directory links and path rewriting features in HTTP server
  - Only heuristic defensive measures:
    - Check URL length; assume spider trap above some threshold, for example 128 characters
    - Watch for sites with very large number of URLs
    - Eliminate URLs with non-textual data types
    - May disable crawling of dynamic pages, if can detect



# More implementation issues

- **Page repository**
  - Naïve: store each page as a separate file
    - Can map URL to unique filename using a hashing function, e.g. MD5
    - This generates a huge number of files, which is inefficient from the storage perspective
  - Better: combine many pages into a single large file, using some XML markup to separate and identify them
    - Must map URL to {filename, page\_id}
  - Database options
    - Any RDBMS -- large overhead
    - Light-weight, embedded databases such as Berkeley DB

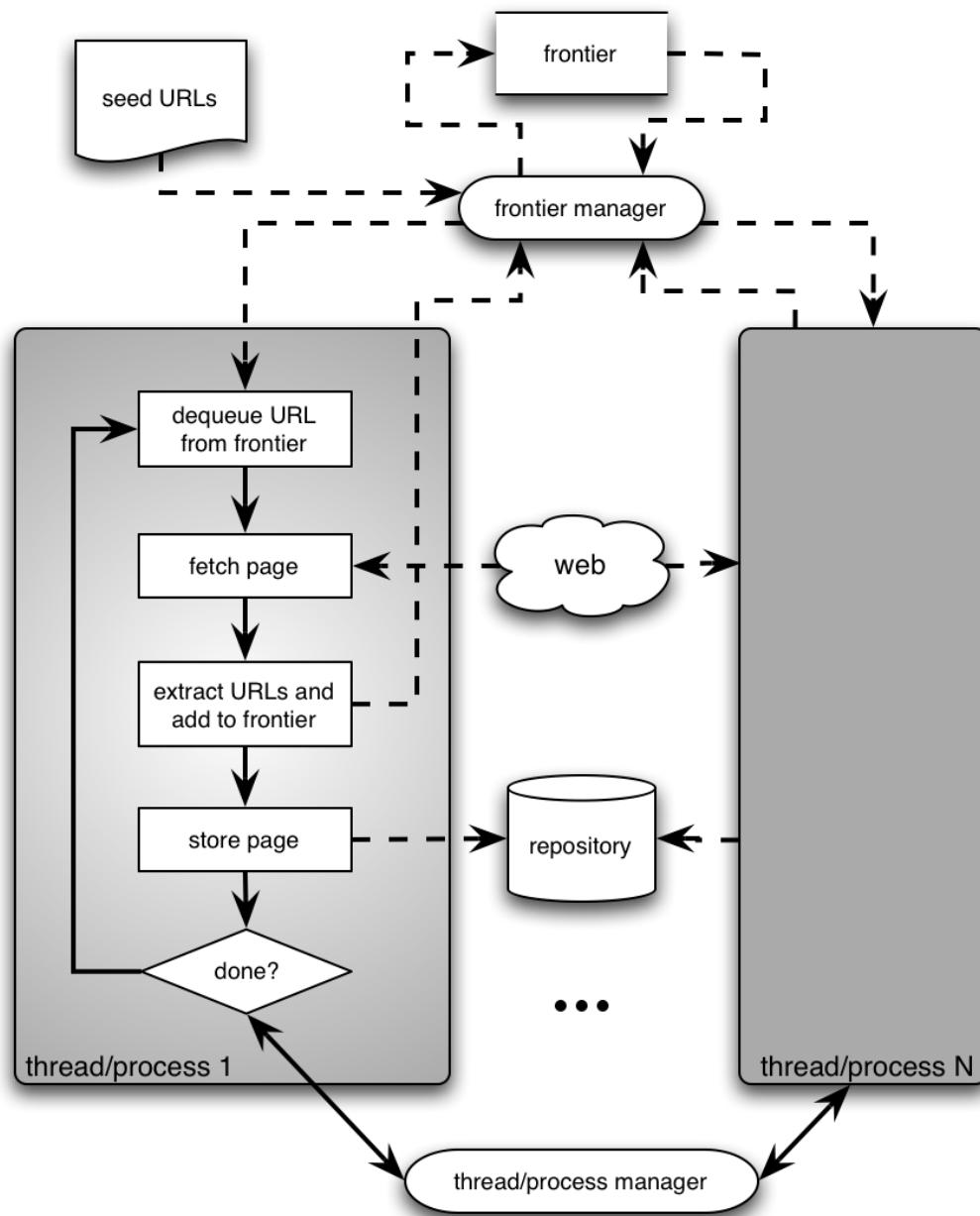


# Concurrency

- A crawler incurs several delays:
  - Resolving the host name in the URL to an IP address using DNS
  - Connecting a socket to the server and sending the request
  - Receiving the requested page in response
- Solution: Overlap the above delays by **fetching many pages concurrently**



# Architecture of a concurrent crawler



# Concurrent crawlers

- Can use multi-processing or multi-threading
- Each process or thread works like a sequential crawler, except they share data structures: frontier and repository
- Shared data structures must be synchronized (locked for concurrent writes)
- Speedup of factor of 5-10 are easy this way



# Outline

- Motivation and taxonomy of crawlers
- Basic crawlers and implementation issues
- Universal crawlers
- Preferential (focused and topical) crawlers

Ψ

Slides © 2007 Filippo Menczer, Indiana University School of Informatics  
Bing Liu: *Web Data Mining*. Springer, 2007  
Ch. 8 *Web Crawling* by Filippo Menczer

Indiana University School of  
**informatics**

# Universal crawlers

- Support universal search engines
- Large-scale
- Huge cost (network bandwidth) of crawl is amortized over many queries from users
- Incremental updates to existing index and other data repositories



# Large-scale universal crawlers

- Two major issues:

## 1. Performance

- Need to scale up to billions of pages

## 2. Policy

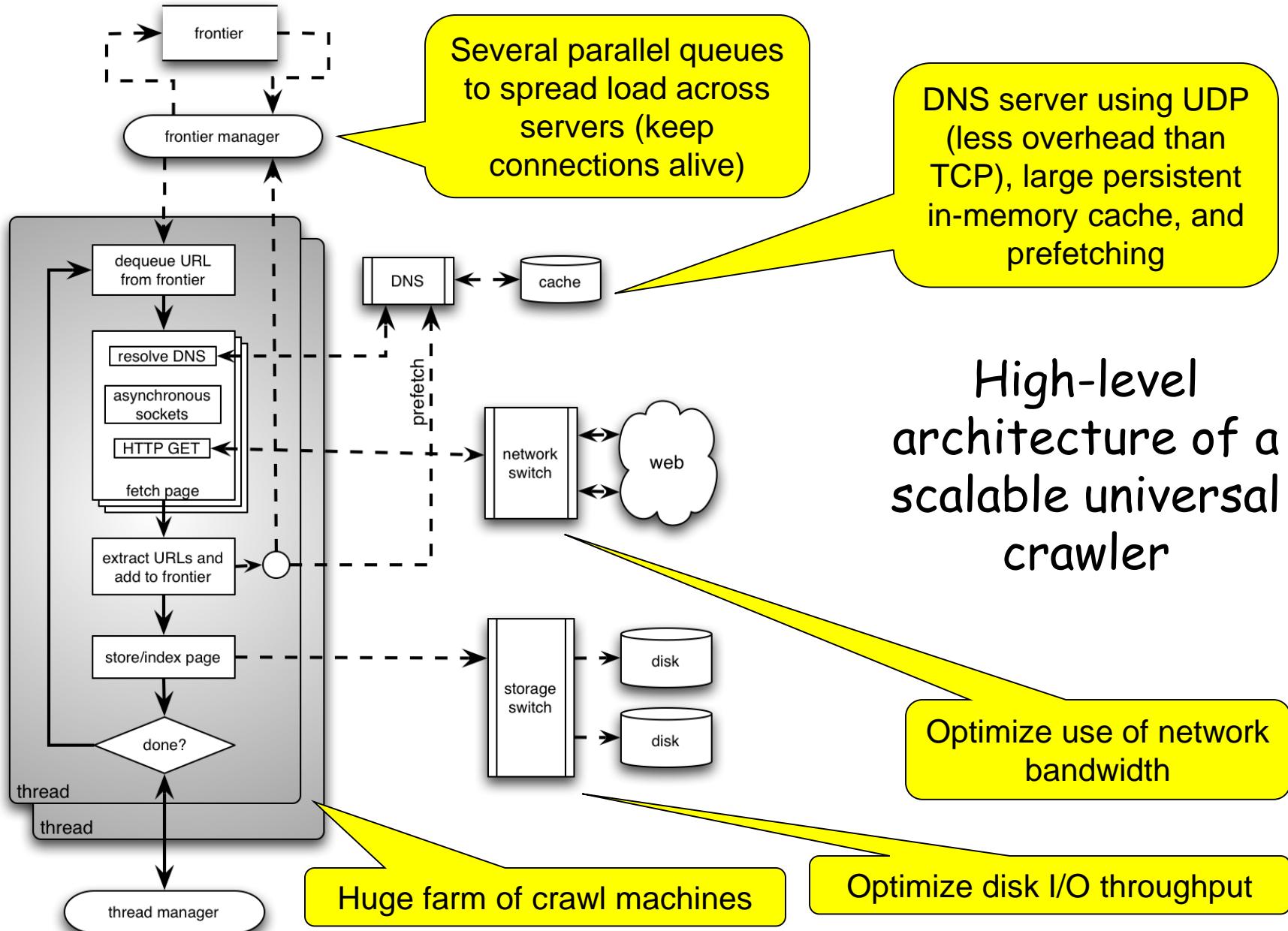
- Need to trade-off coverage, freshness, and bias (e.g. toward “important” pages)



# Large-scale crawlers: scalability

- Need to minimize overhead of DNS lookups
- Need to optimize utilization of network bandwidth and disk throughput (I/O is bottleneck)
- Use asynchronous sockets
  - Multi-processing or multi-threading do not scale up to billions of pages
  - Non-blocking: hundreds of network connections open simultaneously
  - Polling socket to monitor completion of network transfers



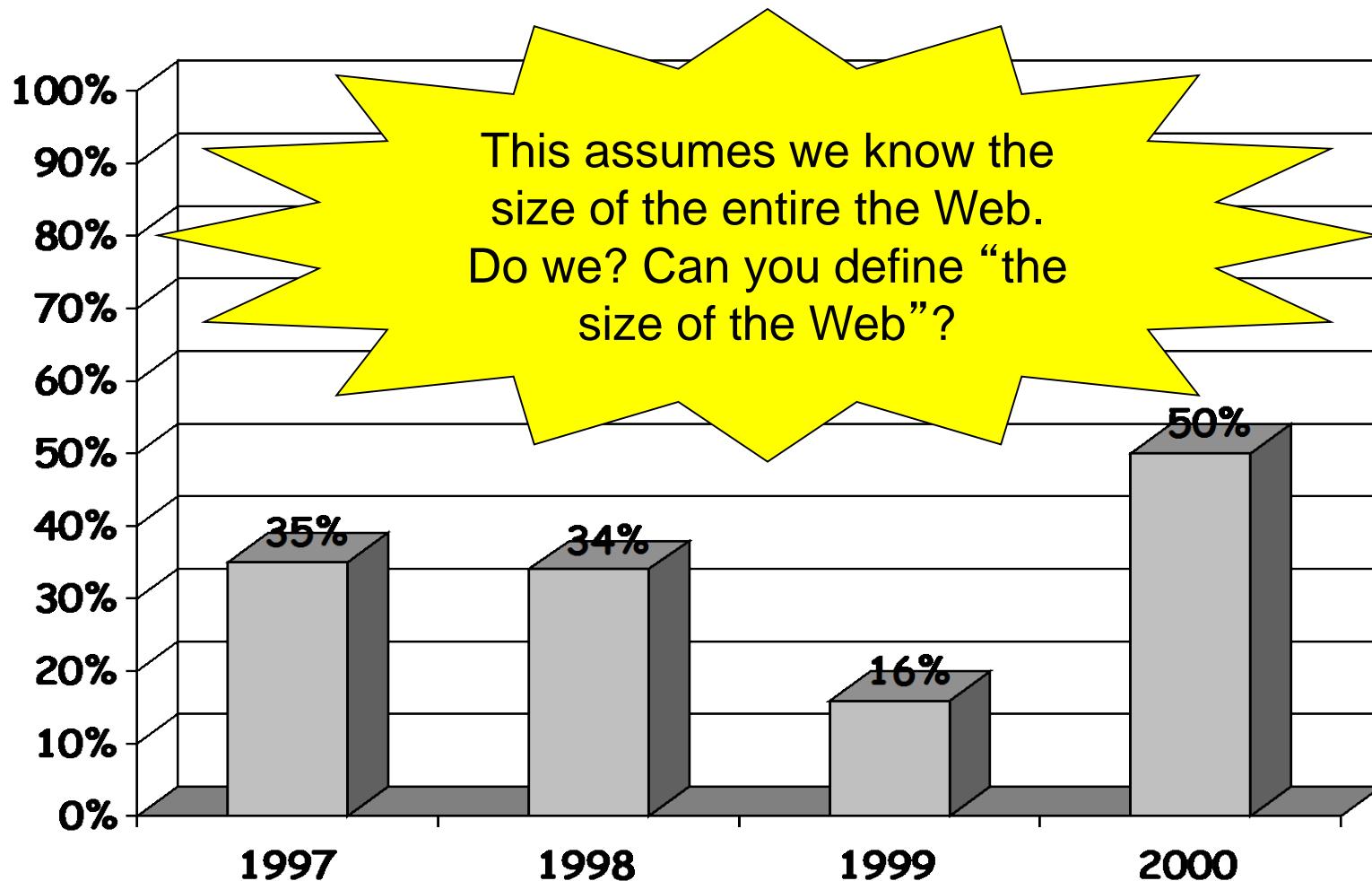


# Universal crawlers: Policy

- Coverage
  - New pages get added all the time
  - Can the crawler find every page?
- Freshness
  - Pages change over time, get removed, etc.
  - How frequently can a crawler revisit ?
- Trade-off!
  - Focus on most “important” pages (crawler bias)?
  - “Importance” is subjective



# Web coverage by search engine crawlers



# Maintaining a “fresh” collection

- Universal crawlers are never “done”
- High variance in rate and amount of page changes
- HTTP headers are notoriously unreliable
  - Last-modified
  - Expires
- Solution
  - Estimate the probability that a previously visited page has changed in the meanwhile
  - Prioritize by this probability estimate



# Estimating page change rates

- Algorithms for maintaining a crawl in which most pages are fresher than a specified epoch
  - Brewington & Cybenko; Cho, Garcia-Molina & Page
- Assumption: recent past predicts the future (Ntoulas, Cho & Olston 2004)
  - Frequency of change not a good predictor
  - Degree of change is a better predictor



# Do we need to crawl the entire Web?

- If we cover too much, it will get stale
- There is an abundance of pages in the Web
- For PageRank, pages with very low prestige are largely useless
- What is the goal?
  - General search engines: pages with high prestige
  - News portals: pages that change often
  - Vertical portals: pages on some topic
- What are appropriate priority measures in these cases? Approximations?



# Outline

- Motivation and taxonomy of crawlers
- Basic crawlers and implementation issues
- Universal crawlers
- Preferential (focused and topical) crawlers



# Preferential crawlers

- Assume we can estimate for each page an importance measure,  $I(p)$
- Want to visit pages in order of decreasing  $I(p)$
- Maintain the frontier as a **priority queue** sorted by  $I(p)$



# Preferential crawlers

- Selective bias toward some pages, eg. most “relevant”/topical, closest to seeds, most popular/largest PageRank, unknown servers, highest rate/amount of change, etc...
- Focused crawlers
  - Supervised learning: **classifier** based on **labeled examples**
- Topical crawlers
  - Best-first search based on **similarity(topic, parent)**
  - Adaptive crawlers
    - Reinforcement learning
    - Evolutionary algorithms/artificial life



# Preferential crawling algorithms: Examples

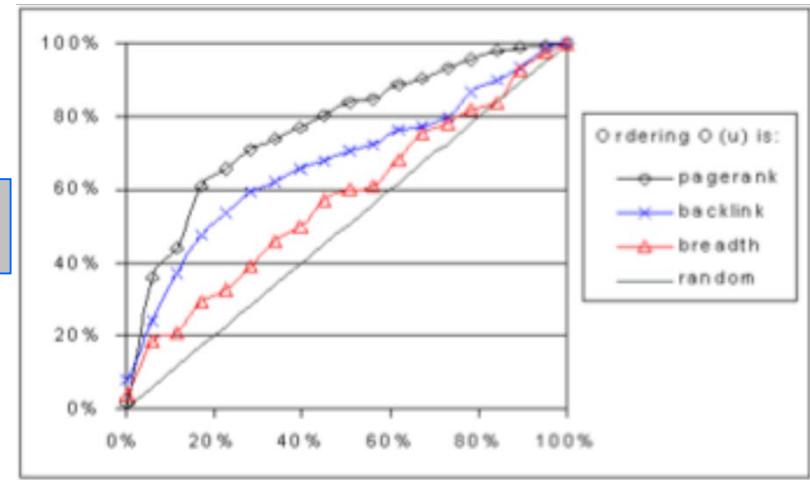
- Breadth-First
  - Exhaustively visit all links in order encountered
- Best-N-First
  - Priority queue sorted by similarity, explore top N at a time
- PageRank
  - Priority queue sorted by keywords, PageRank
- SharkSearch
  - Priority queue sorted by combination of similarity, anchor text, similarity of parent, etc. (powerful cousin of FishSearch)
- InfoSpiders
  - Adaptive distributed algorithm using an evolving population of learning agents



# Preferential crawlers: Examples

- For  $I(p)$  = PageRank (estimated based on pages crawled so far), we can find high-PR pages faster than a breadth-first crawler (Cho, Garcia-Molina & Page 1998)

Recall



Crawl size

# Topical crawlers

- All we have is a topic (query, description, keywords) and a set of seed pages (not necessarily relevant)
- No labeled examples
- Must predict relevance of unvisited links to prioritize
- Original idea: Menczer 1997, Menczer & Belew 1998



Example: [myspiders.informatics.indiana.edu](http://myspiders.informatics.indiana.edu)

Query: java security      Start      Stop

# MySpiders

Crawler Name  
InfoSpiders

Source	URL	Rece...	Score
Spider6	<a href="http://www.rstcorp.com/javasecurity">http://www.rstcorp.com/javasecurity</a>	?	0.63
Seed	<a href="http://www.rstcorp.com/javasecurity/links.html">http://www.rstcorp.com/javasecurity/links.html</a>	?	0.63
Seed	<a href="http://www.rstcorp.com/java-security.html">http://www.rstcorp.com/java-security.html</a>	?	0.63
Spider13	<a href="http://www.digital.com/java.html">http://www.digital.com/java.html</a>	?	0.55
Spider6	<a href="http://www.rstcorp.com/javasecurity/papers.h...">http://www.rstcorp.com/javasecurity/papers.h...</a>	?	0.53
Seed	<a href="http://archives.java.sun.com/archives/java-se...">http://archives.java.sun.com/archives/java-se...</a>	?	0.53
Seed	<a href="http://www.cs.princeton.edu/sip/faq/java-faq...">http://www.cs.princeton.edu/sip/faq/java-faq...</a>	?	0.51
Spider6	<a href="http://www.securingjava.com">http://www.securingjava.com</a>	?	0.46
Seed	<a href="http://www.rstcorp.com/java-security.html">http://www.rstcorp.com/java-security.html</a>	?	0.46
Spider3	<a href="http://www.rstcorp.com/java-security.html">http://www.rstcorp.com/java-security.html</a>	?	0.46
Spider13	<a href="http://www.rstcorp.com/java-security.html">http://www.rstcorp.com/java-security.html</a>	?	0.46
Spider13	<a href="http://www.rstcorp.com/java-security.html">http://www.rstcorp.com/java-security.html</a>	?	0.46

Spider Details

Details

Spider11

- Status
- Energy
- Query

  - Term1
  - Term2
  - Term3
  - hotlist

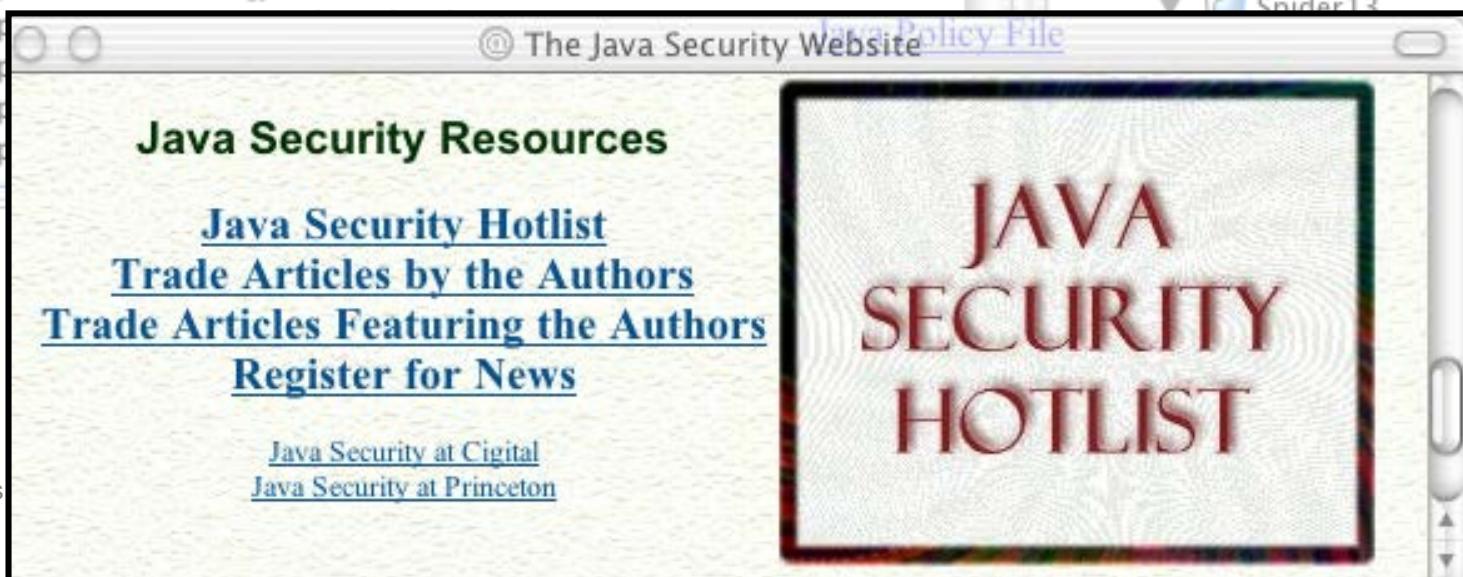
History

Spider6

Spider11

Spider15

Spider13



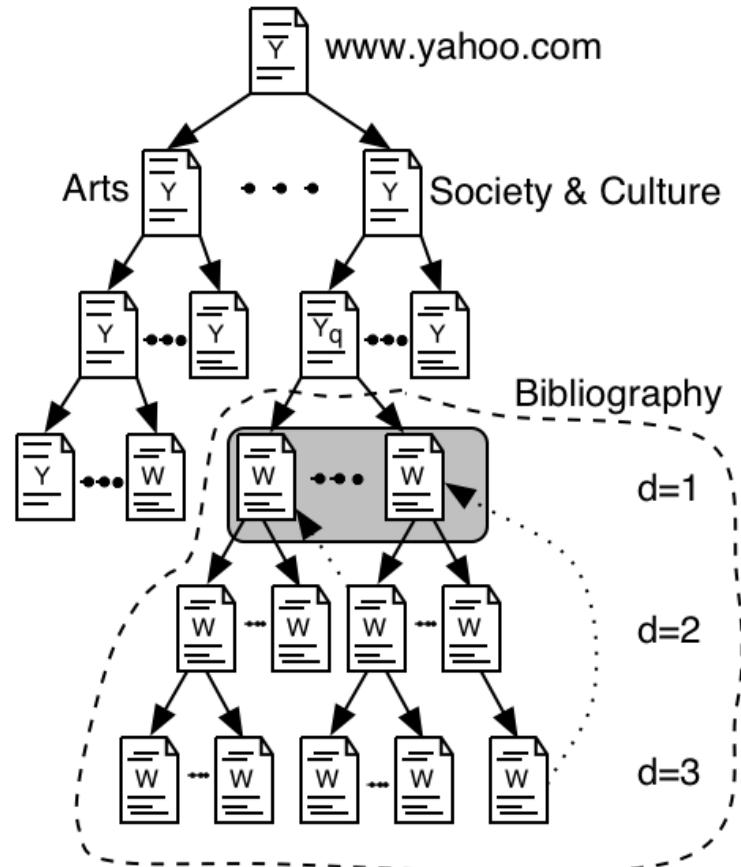
# Topical locality

- Topical locality is a **necessary** condition for a topical crawler to work, and for surfing to be a worthwhile activity for humans
- Links must encode **semantic** information, i.e. say something about neighbor pages, not be random
- It is also a **sufficient** condition if we start from “good” seed pages
- Indeed we know that Web topical locality is strong :
  - Indirectly (crawlers work and people surf the Web)
  - From direct measurements (Davison 2000; Menczer 2004, 2005)

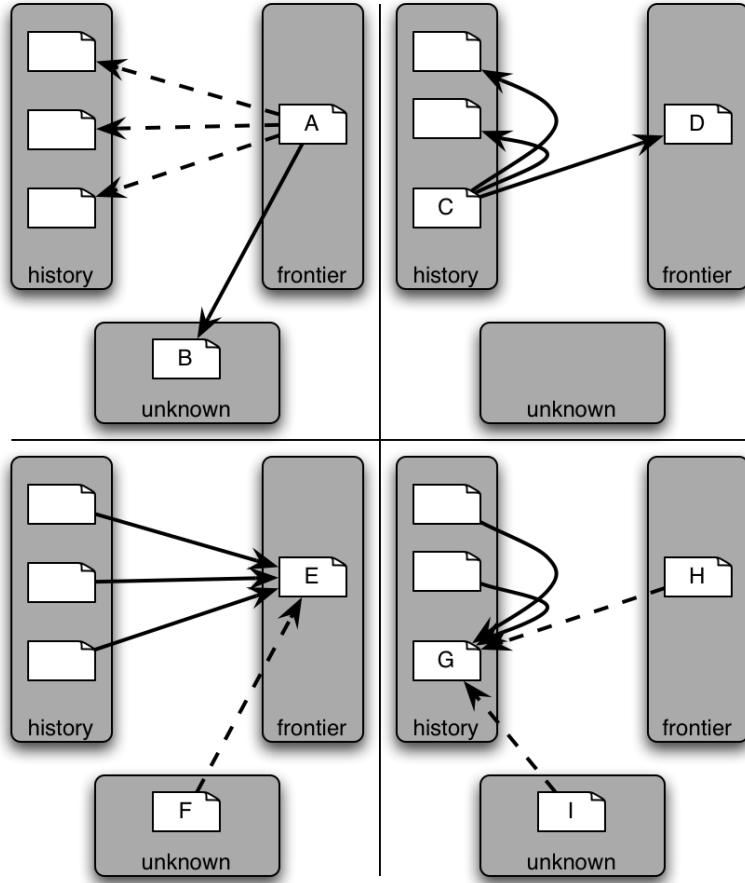


# Quantifying topical locality

- Different ways to pose the question:
  - How quickly does semantic locality decay?
  - How fast is **topic drift**?
  - How quickly does content change as we surf away from a starting page?



# Topical locality-inspired tricks for topical crawlers



- **Co-citation (a.k.a. sibling locality):** A and C are good hubs, thus A and D should be given high priority
- **Co-reference (a.k.a. bibliographic coupling):** E and G are good authorities, thus E and H should be given high priority



# Naïve Best-First

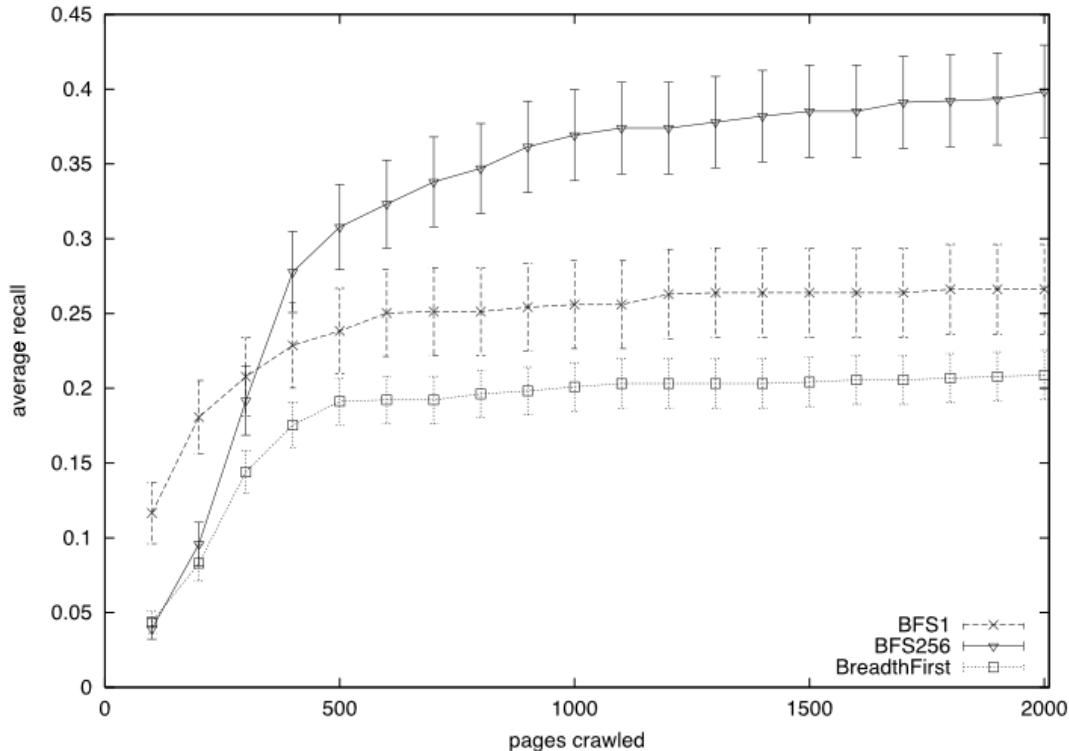
Simplest  
topical crawler:  
Frontier is  
priority queue  
based on text  
similarity  
between topic  
and parent  
page

```
BestFirst(topic, seed_urls) {
    foreach link (seed_urls) {
        enqueue(frontier, link);
    }
    while (#frontier > 0 and visited < MAX_PAGES) {
        link := dequeue_link_with_max_score(frontier);
        doc := fetch_new_document(link);
        score := sim(topic, doc);
        foreach outlink (extract_links(doc)) {
            if (#frontier >= MAX_BUFFER) {
                dequeue_link_with_min_score(frontier);
            }
            enqueue(frontier, outlink, score);
        }
    }
}
```



# Exploration vs Exploitation

- Best-N-First (or BFSN)
- Rather than re-sorting the frontier every time you add links, be lazy and sort only every N pages visited
- Empirically, being less greedy helps crawler performance significantly: escape “local topical traps” by exploring more



Pant et al. 2002



# Outline

- Motivation and taxonomy of crawlers
- Basic crawlers and implementation issues
- Universal crawlers
- Preferential (focused and topical) crawlers

Ψ

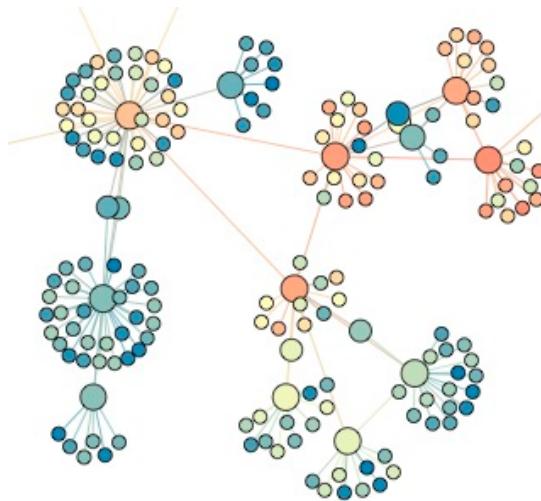
Slides © 2007 Filippo Menczer, Indiana University School of Informatics  
Bing Liu: *Web Data Mining*. Springer, 2007  
Ch. 8 *Web Crawling* by Filippo Menczer

Indiana University School of  
**informatics**

# Need crawling code?

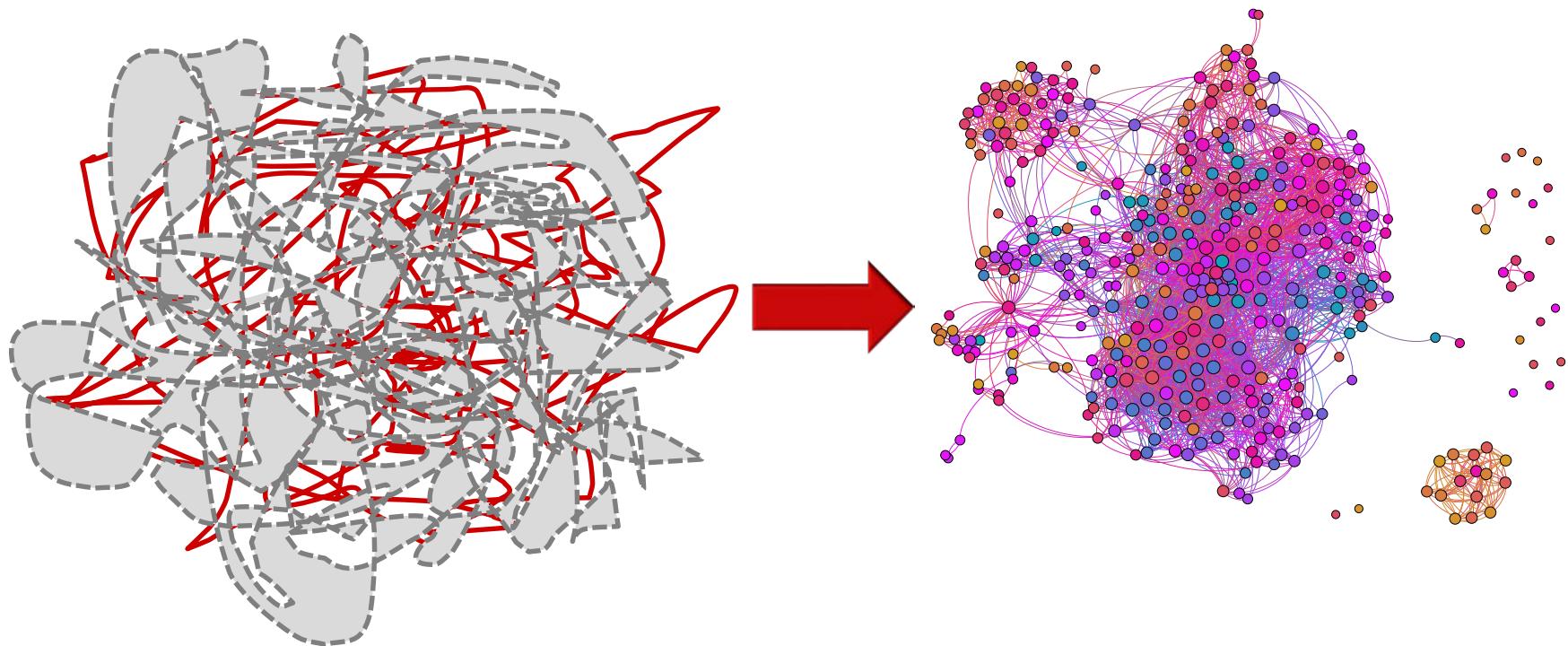
- Reference C implementation of HTTP, HTML parsing, etc
  - w3c-libwww package from World-Wide Web Consortium: [www.w3c.org/Library/](http://www.w3c.org/Library/)
- LWP (Perl)
  - <http://www.oreilly.com/catalog/perlwp/>
  - <http://search.cpan.org/~gaas/libwww-perl-5.804/>
- Open source crawlers/search engines
  - Nutch: <http://www.nutch.org/> (Jakarta Lucene: [jakarta.apache.org/lucene/](http://jakarta.apache.org/lucene/))
  - Herertrix: <http://crawler.archive.org/>
  - WIRE: <http://www.cwr.cl/projects/WIRE/>
  - Terrier: <http://ir.dcs.gla.ac.uk/terrier/>
- Open source topical crawlers, Best-First-N (Java)
  - <http://informatics.indiana.edu/fil/IS/JavaCrawlers/>
- Evaluation framework for topical crawlers (Perl)
  - <http://informatics.indiana.edu/fil/IS/Framework/>





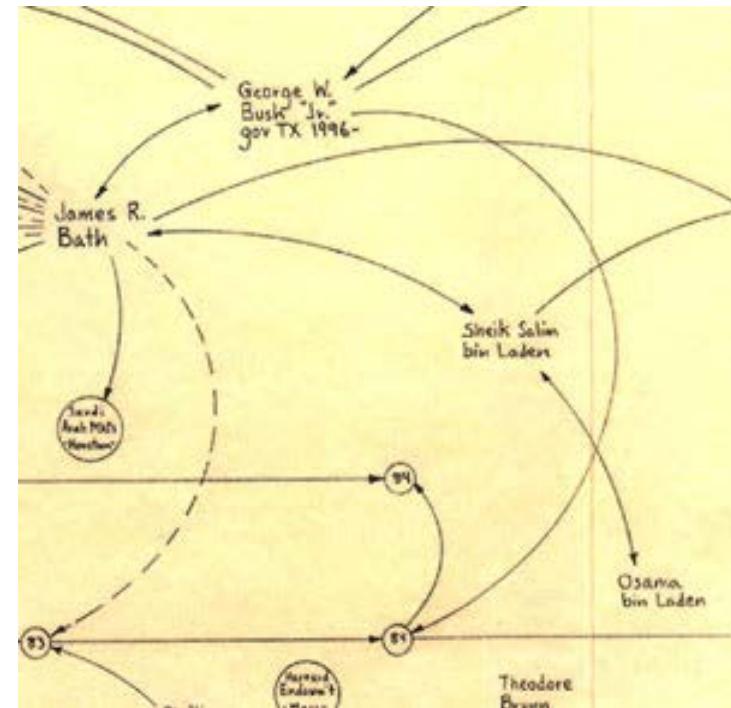
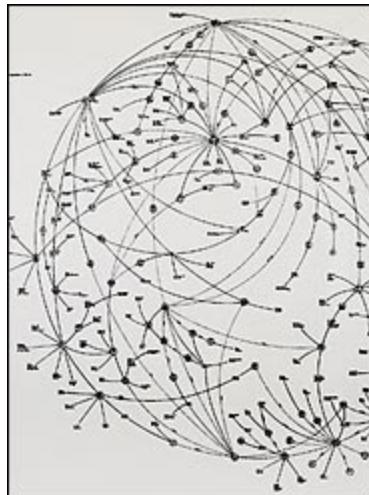
# Social Network Analysis

# What do we get out of studying systems as networks?



# examples: Political/Financial Networks

- Mark Lombardi: tracked and mapped global financial fiascos in the 1980s and 1990s from public sources such as news articles



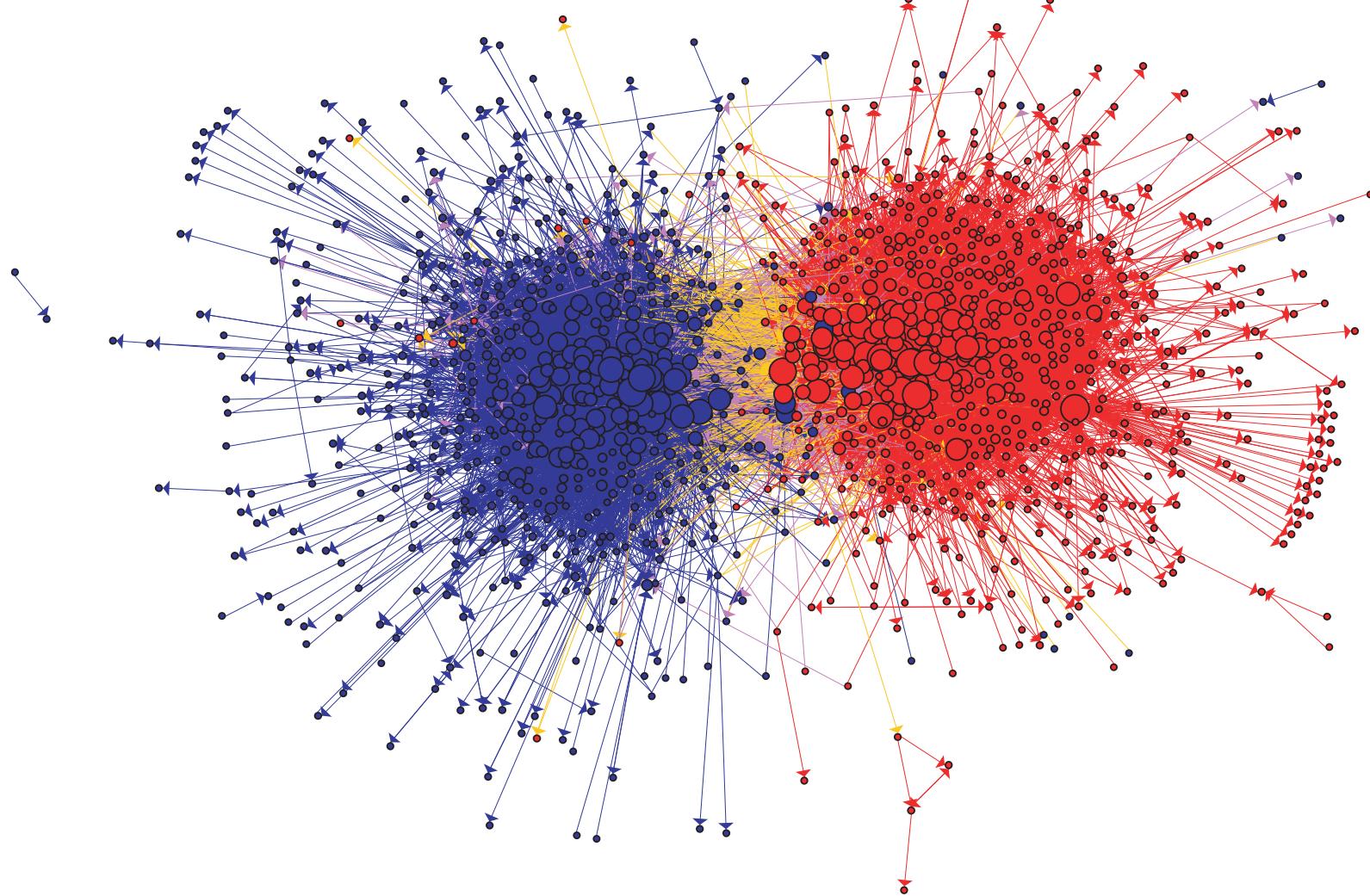
# Understanding through visualization

■ “I happened to be in the Drawing Center when the Lombardi show was being installed and several consultants to the Department of Homeland Security came in to take a look. They said they found the work revelatory, not because the financial and political connections he mapped were new to them, but because Lombardi showed them an elegant way to array disparate information and make sense of things, which they thought might be useful to their security efforts. I didn’t know whether to find that response comforting or alarming, but I saw exactly what they meant.”

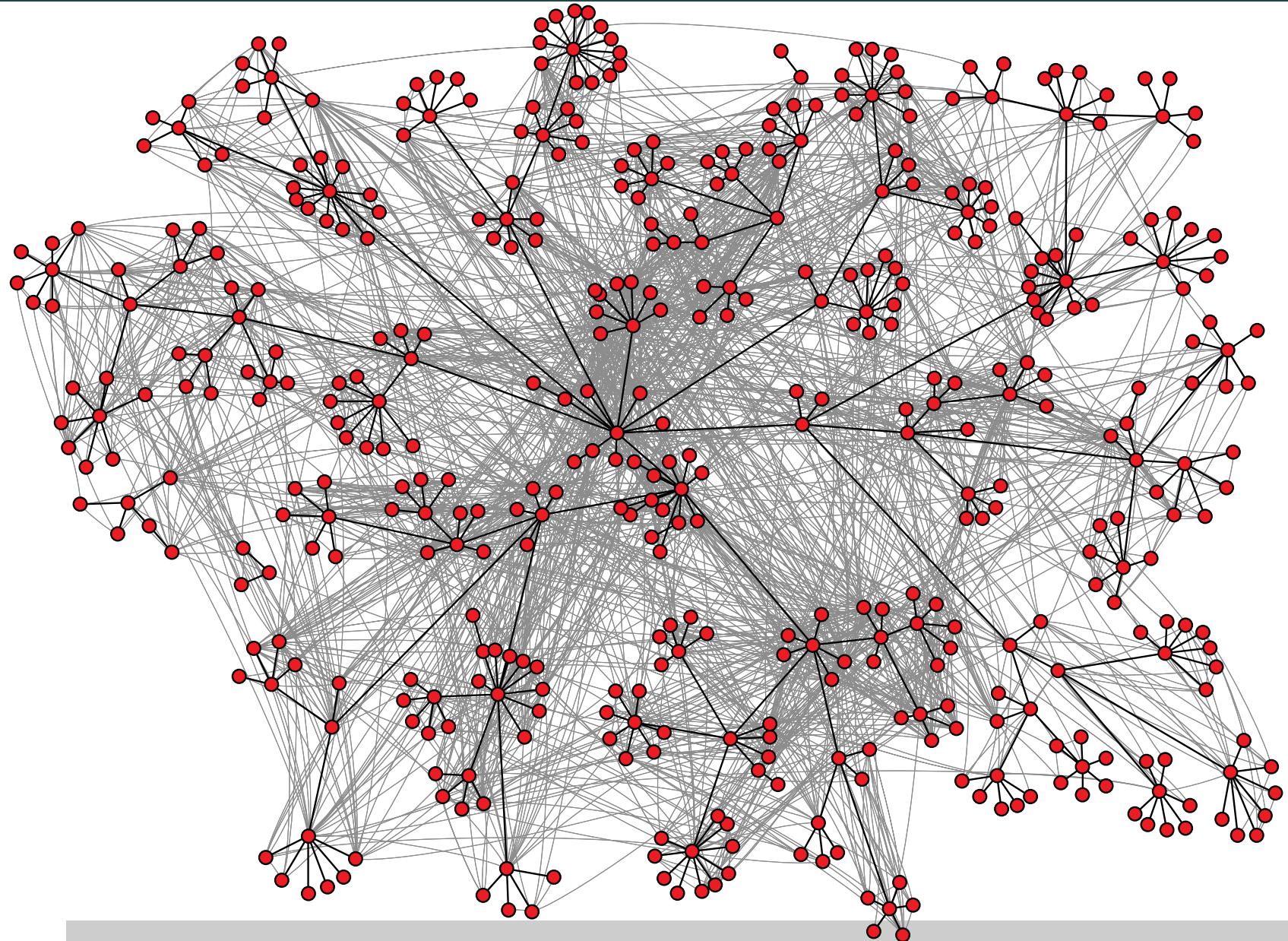
Michael Kimmelman

Webs Connecting the Power Brokers, the Money and the World  
NY Times November 14, 2003

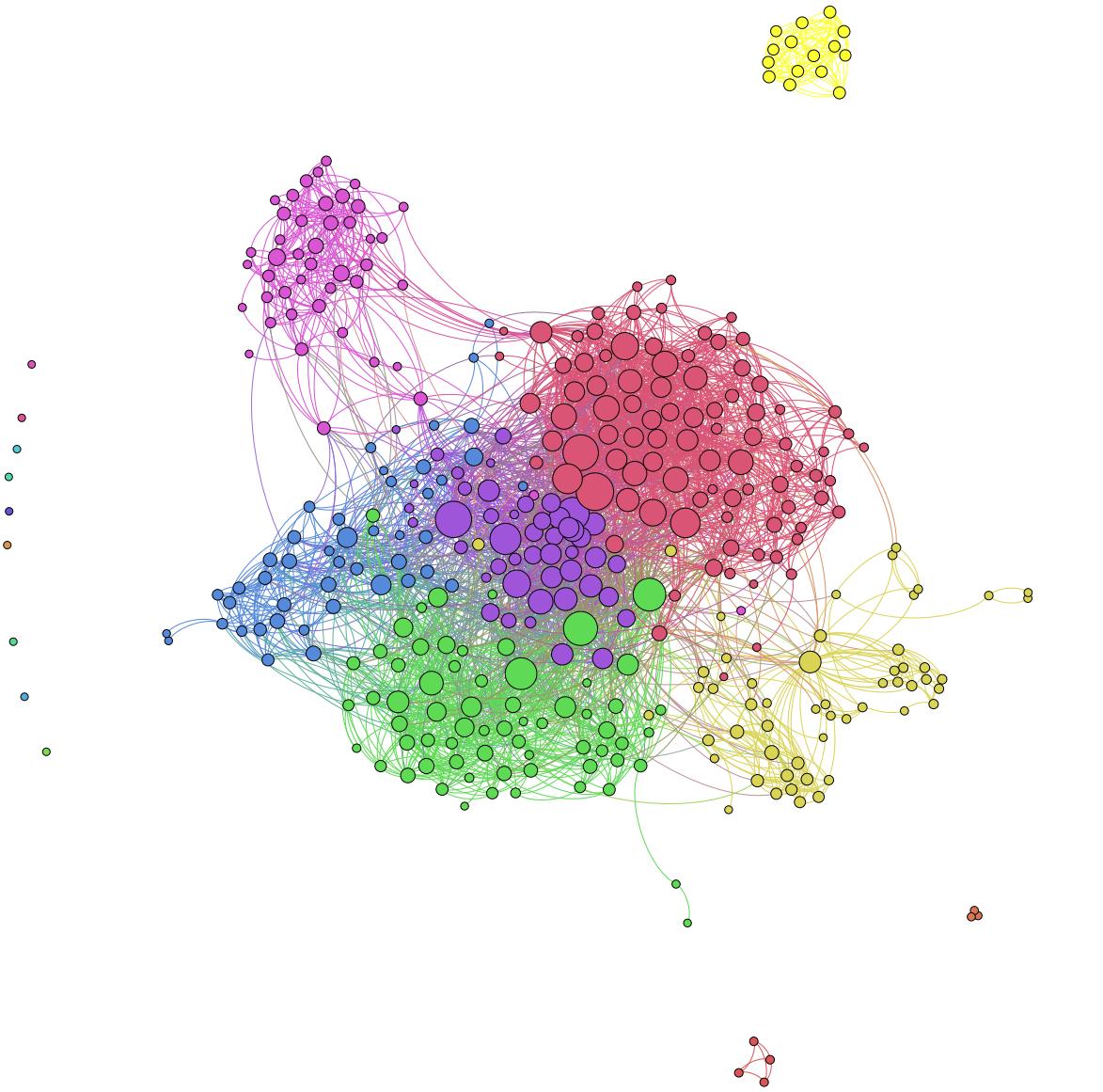
# Political blogs



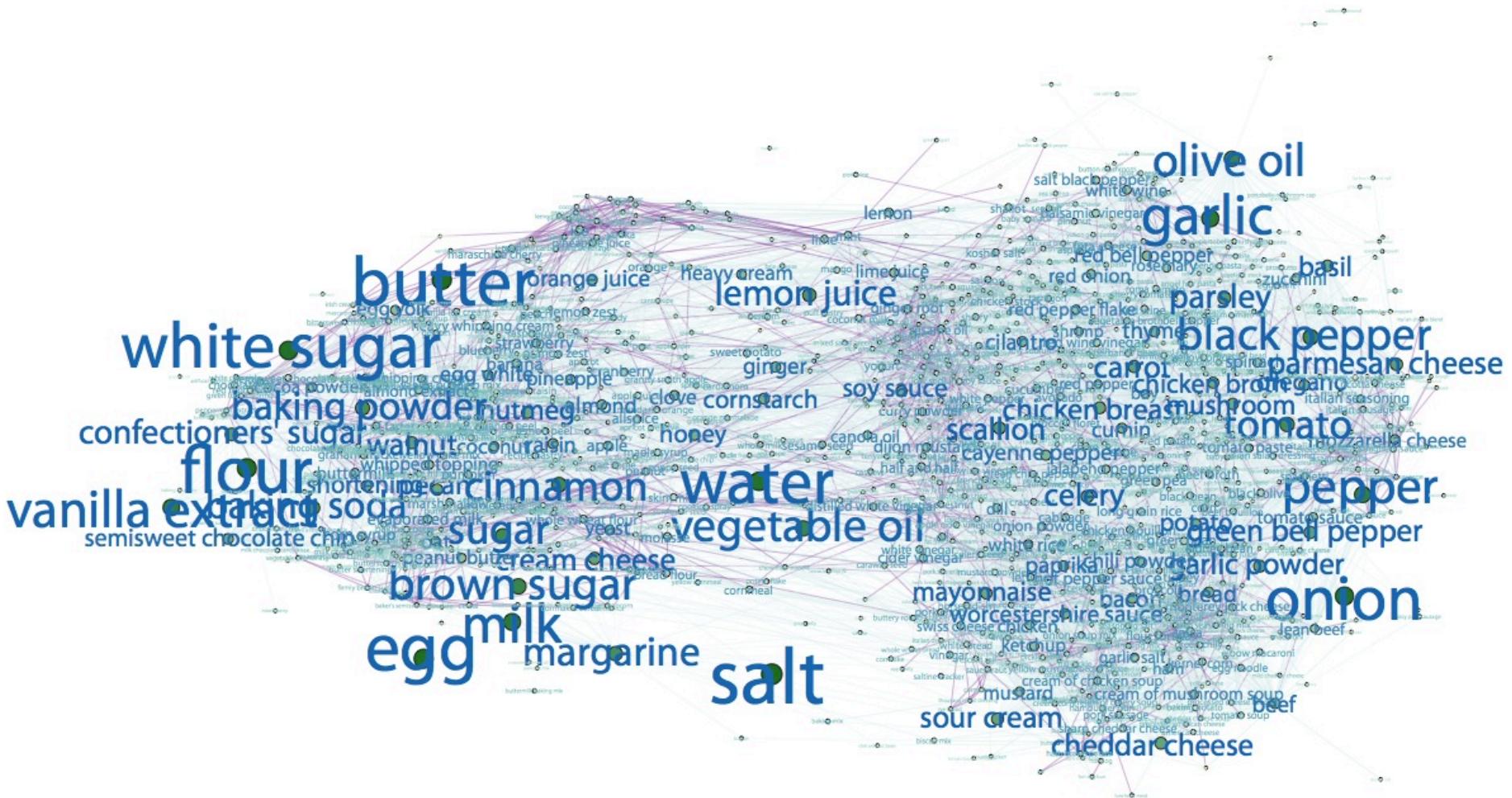
# Organizations



# Facebook networks

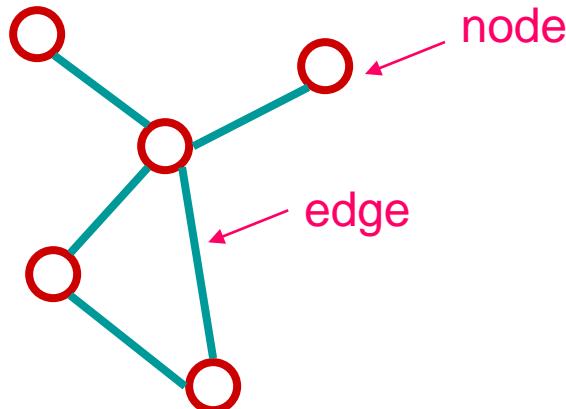


# Ingredient networks



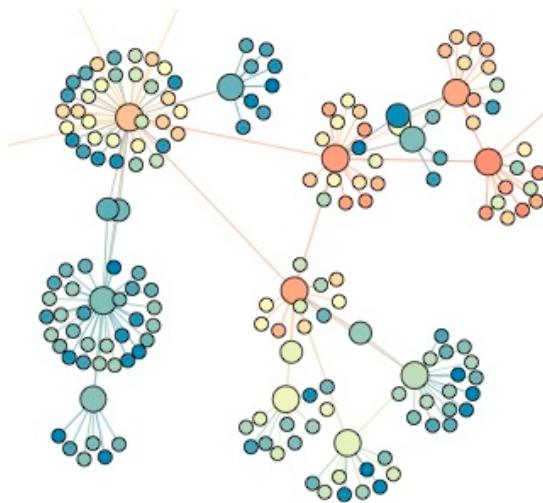
# What are networks?

- Networks are sets of nodes connected by edges.



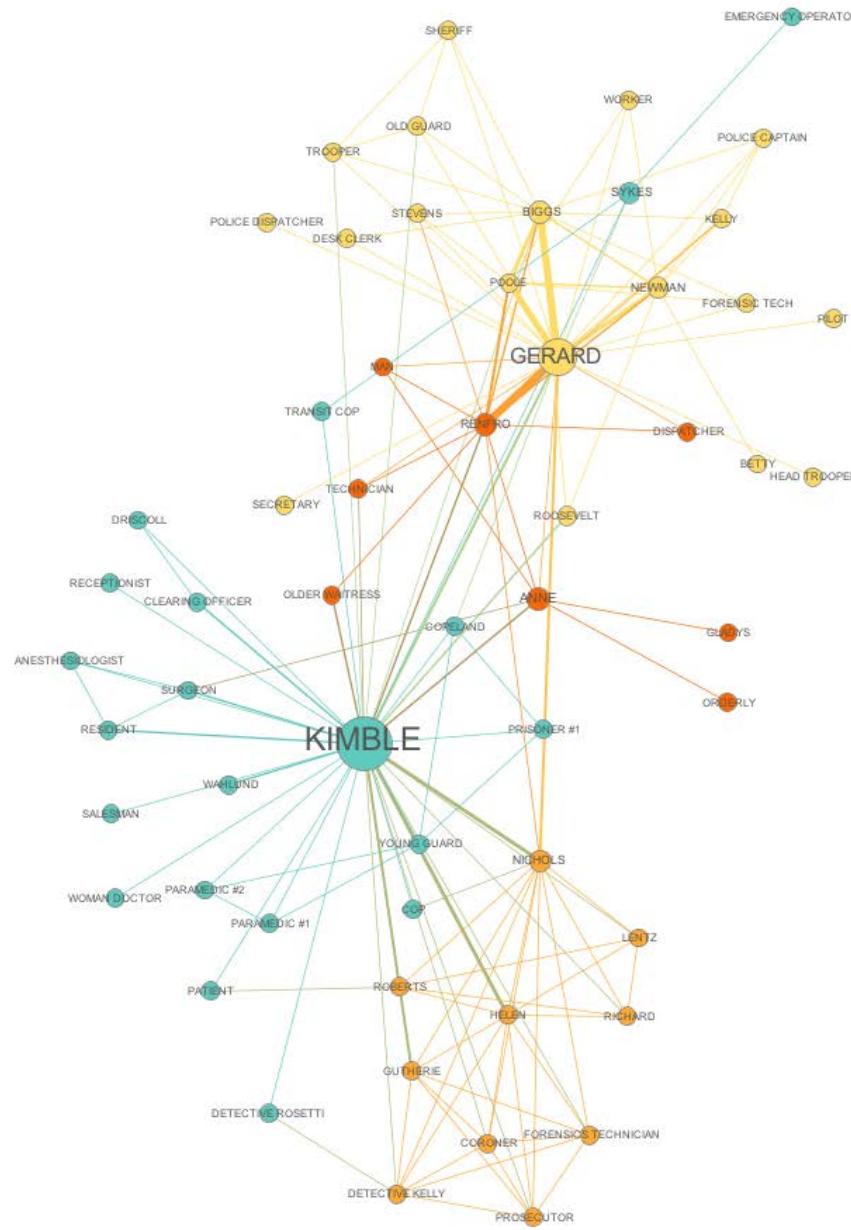
“Network” ≡ “Graph”

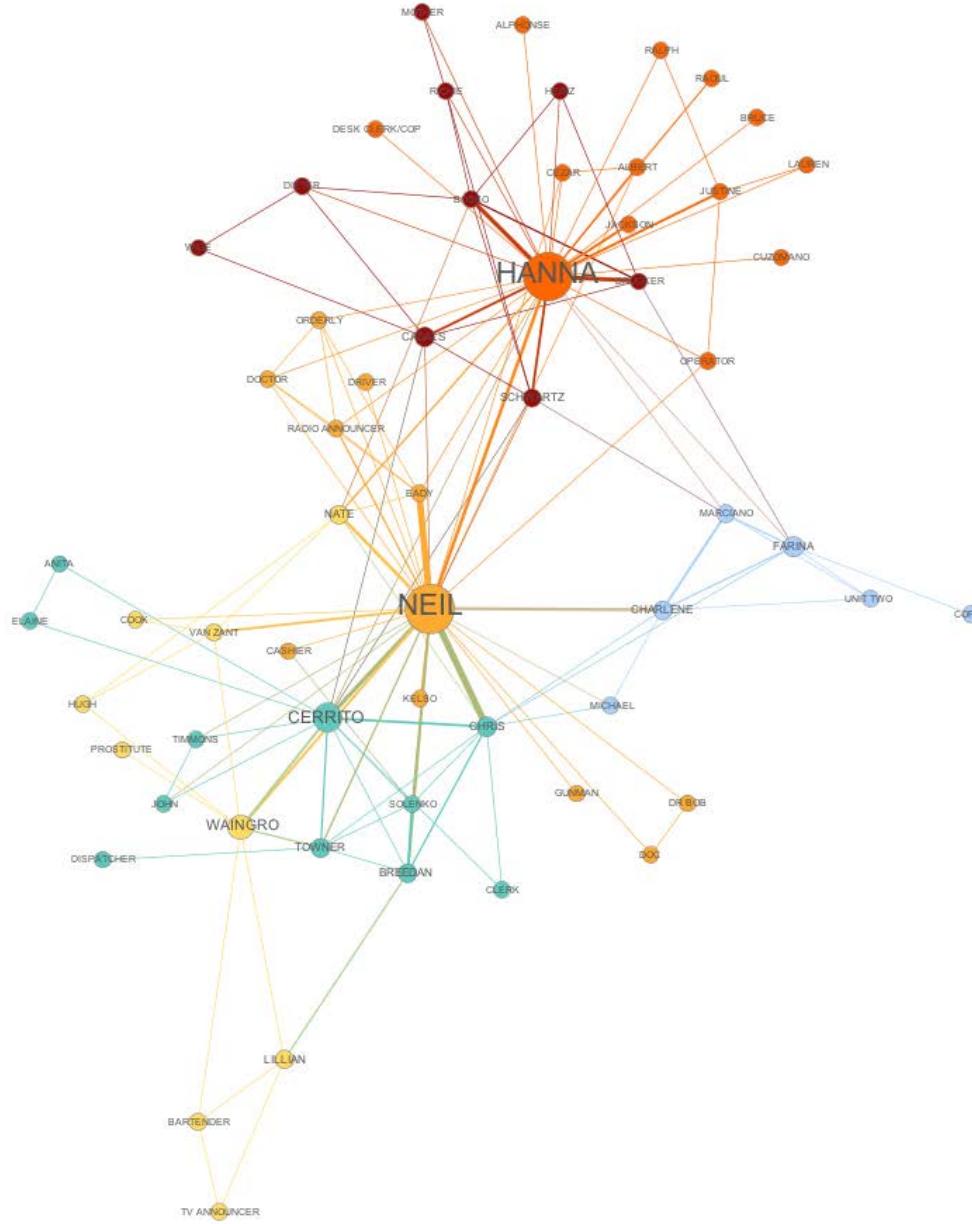
points	lines	
vertices	edges, arcs	math
nodes	links	computer science
sites	bonds	physics
actors	ties, relations	sociology

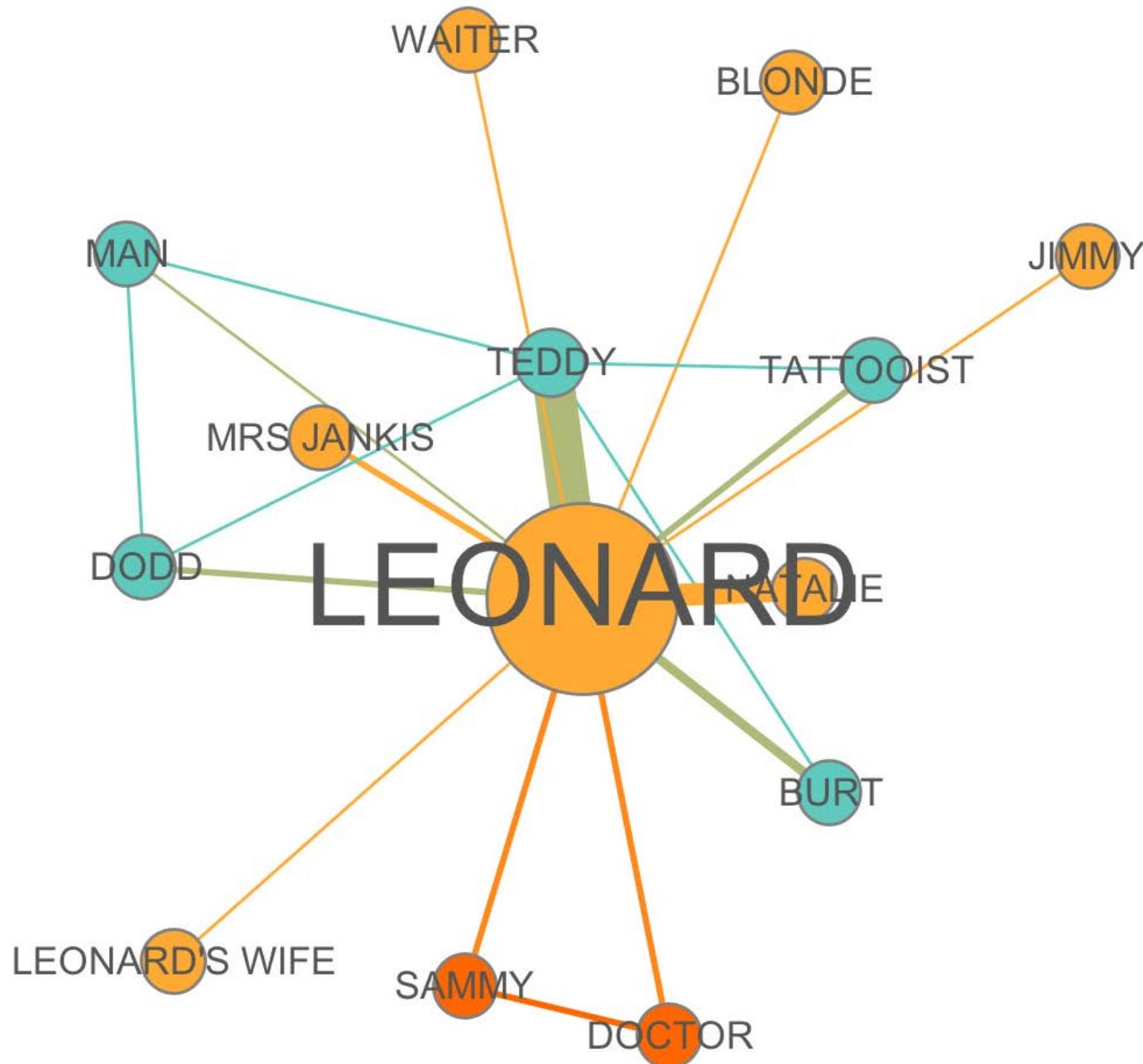


# SNA: Centrality

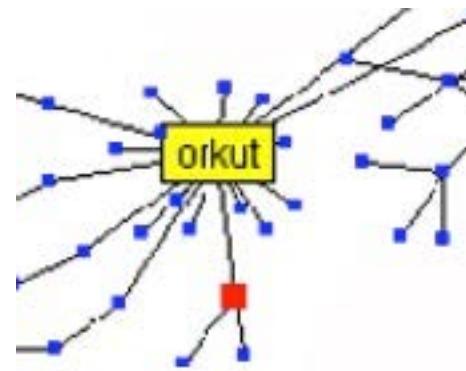
# The Fugitive (1993)



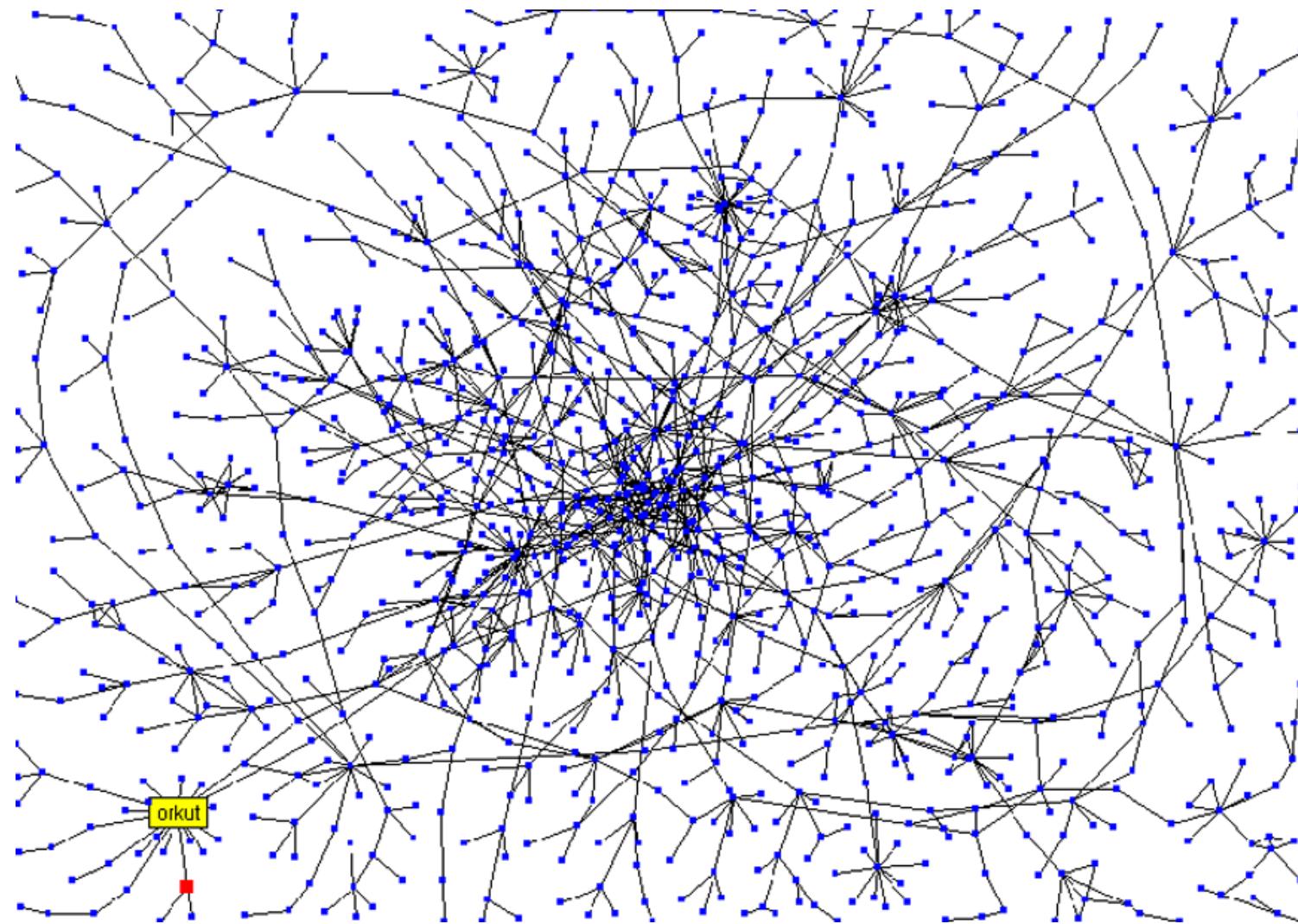




# is counting the edges enough?



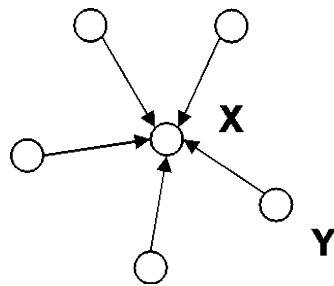
## Stanford Social Web (ca. 1999)



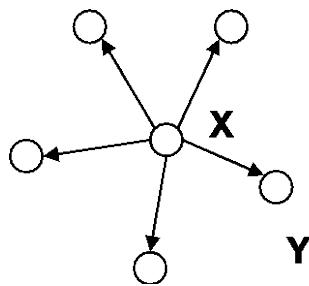
network of personal homepages at Stanford

# different notions of centrality

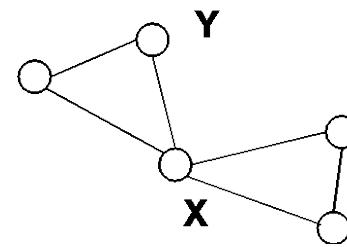
In each of the following networks, X has higher centrality than Y according to a particular measure



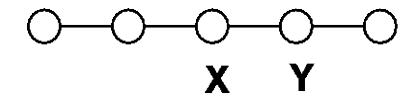
indegree



outdegree

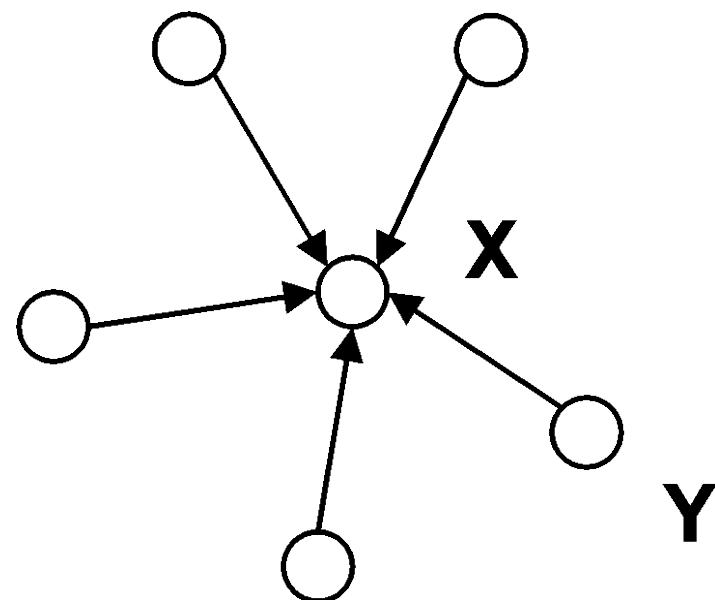


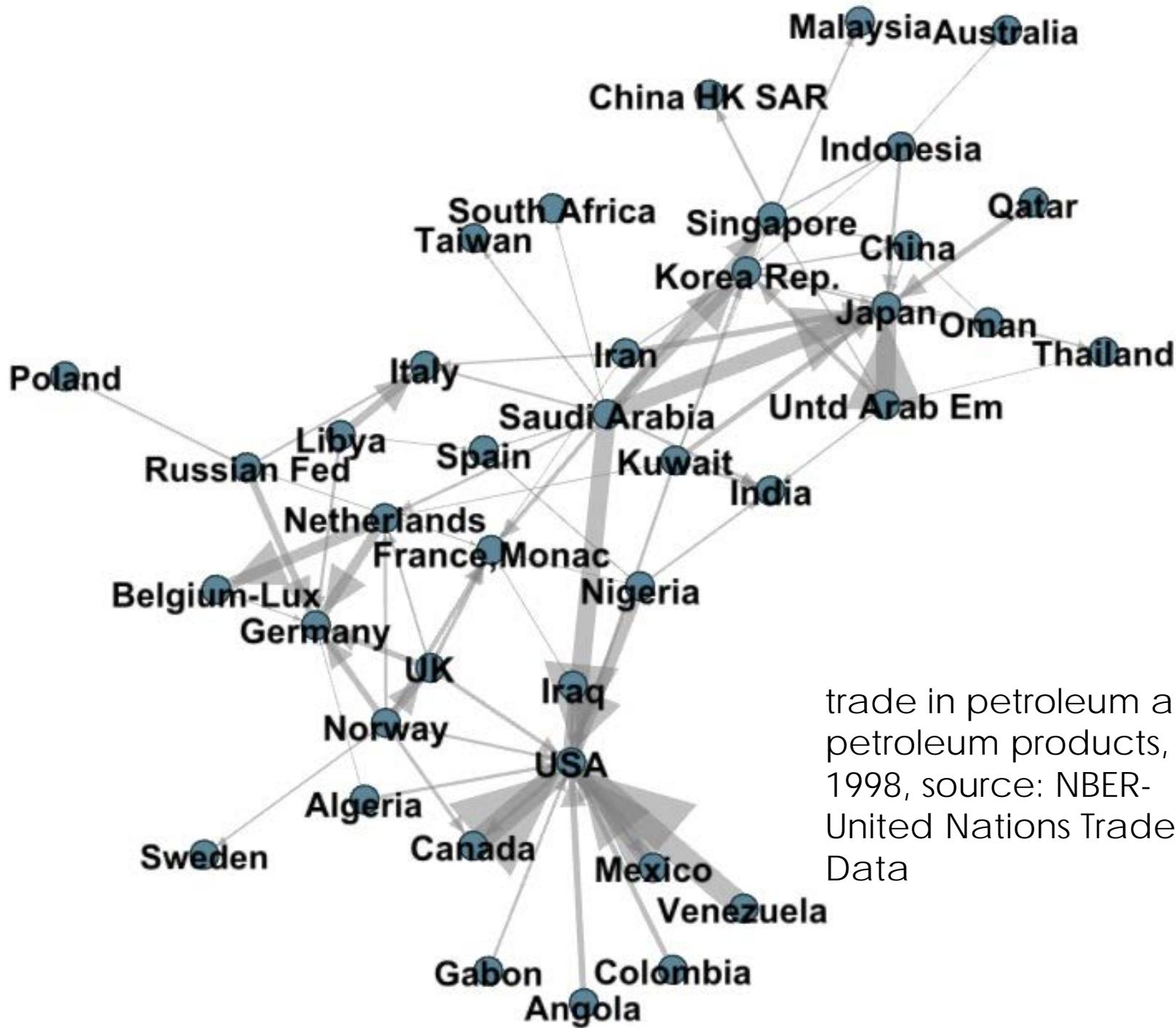
betweenness



closeness

# review: indegree



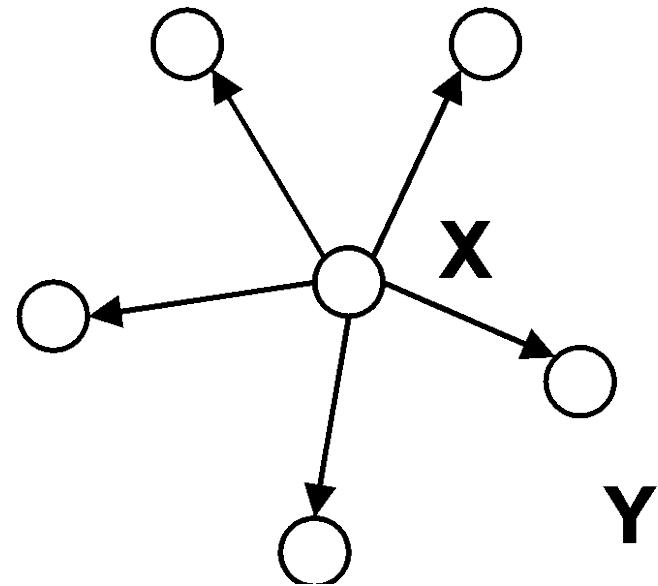


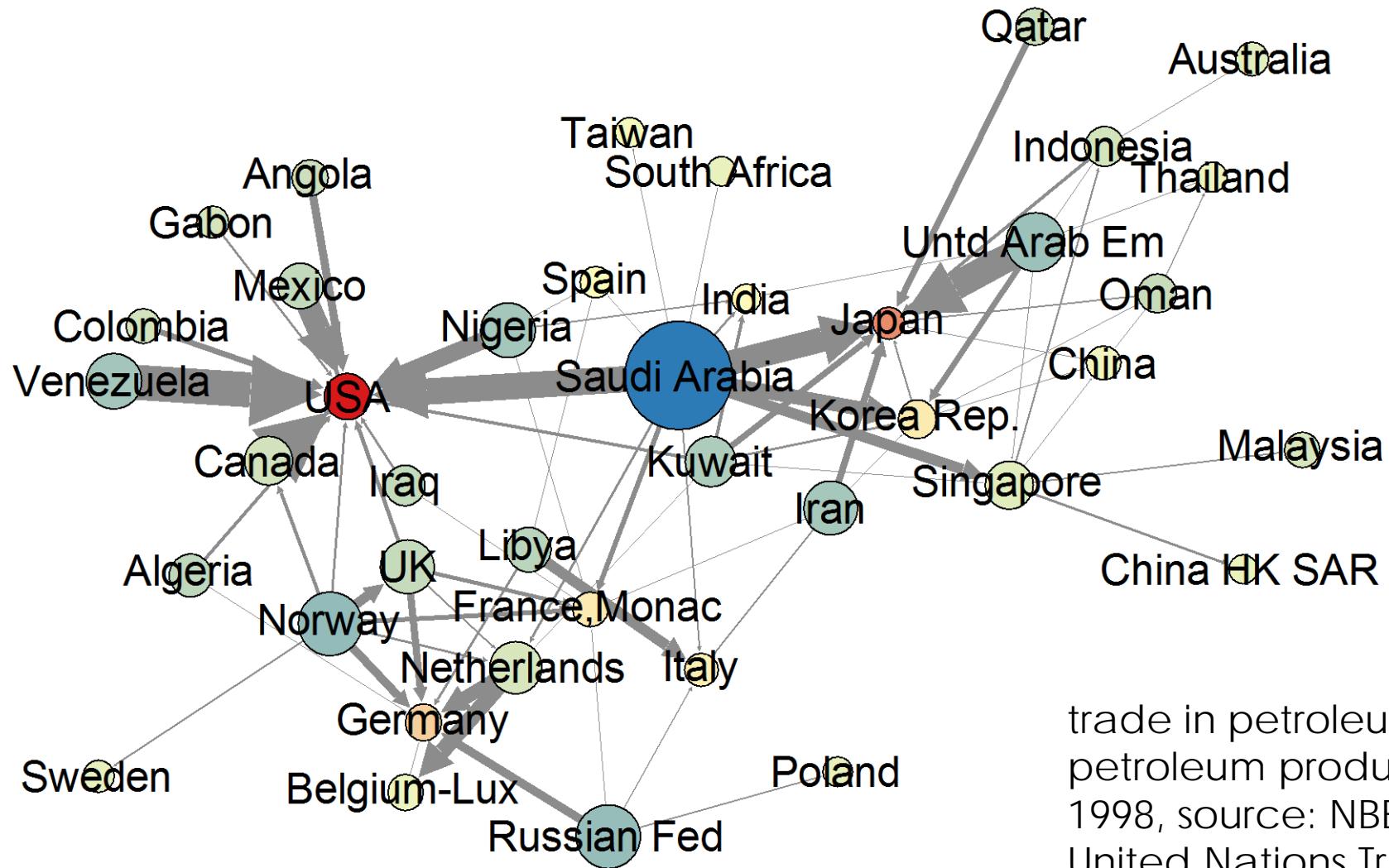
trade in petroleum and  
petroleum products,  
1998, source: NBER-  
United Nations Trade  
Data

## Quiz Q:

- ❑ Which countries have high indegree (import petroleum and petroleum products from many others)
  - ❑ Saudi Arabia
  - ❑ Japan
  - ❑ Iraq
  - ❑ USA
  - ❑ Venezuela

# review: outdegree

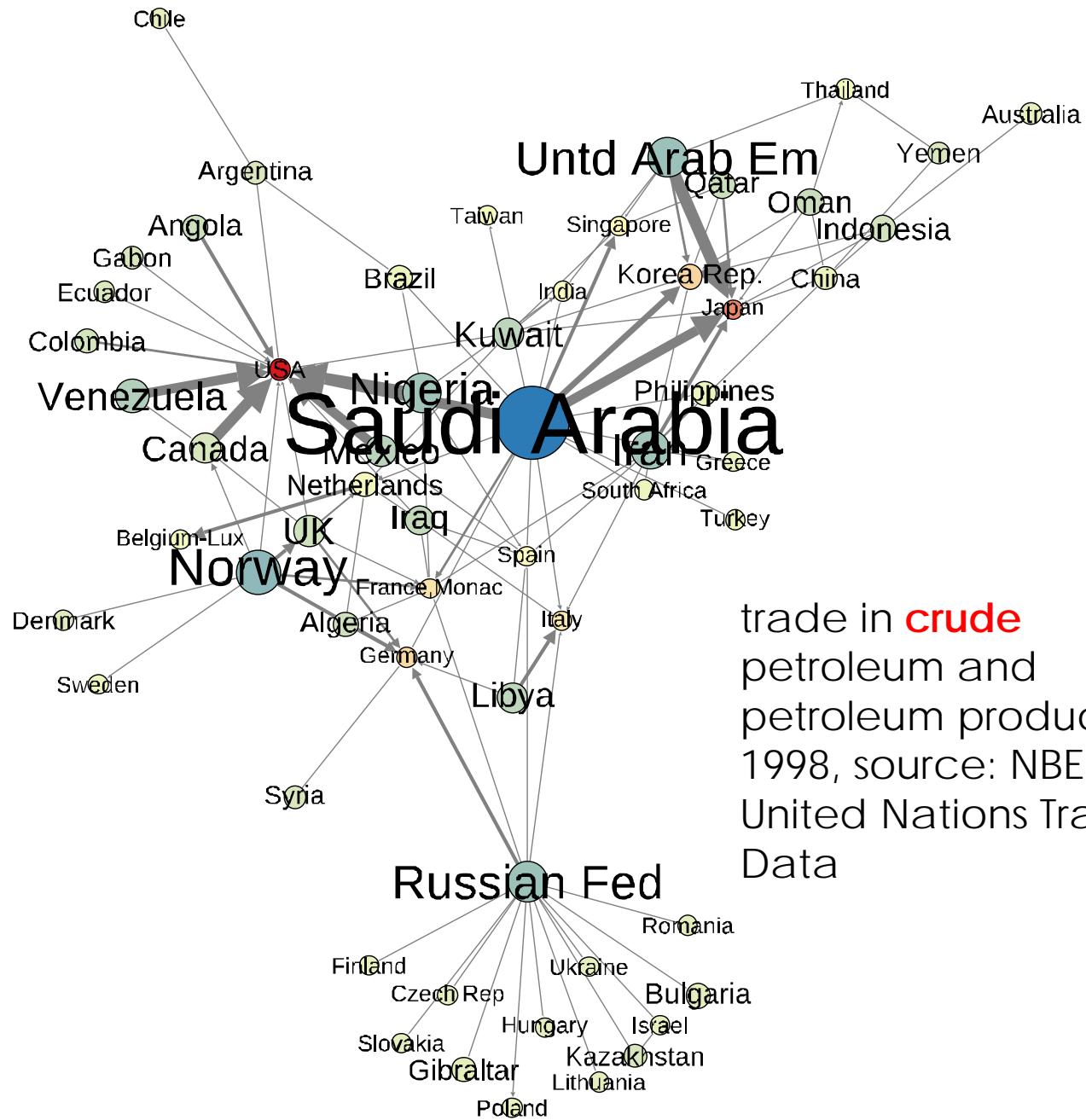




trade in petroleum and  
petroleum products,  
1998, source: NBER-  
United Nations Trade  
Data

## Quiz Q:

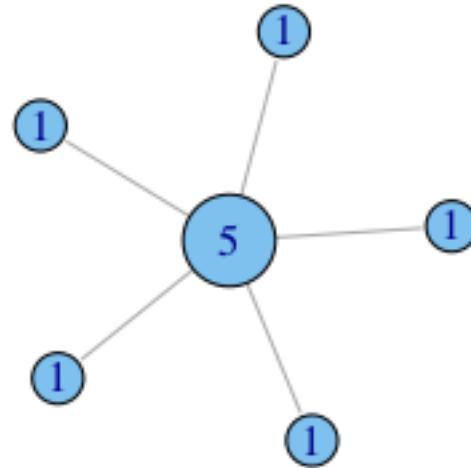
- ❑ Which country has low outdegree but exports a significant quantity (thickness of the edges represents \$\$ value of export) of petroleum products
  - ❑ Saudi Arabia
  - ❑ Japan
  - ❑ Iraq
  - ❑ USA
  - ❑ Venezuela



trade in crude  
petroleum and  
petroleum products,  
1998, source: NBER-  
United Nations Trade  
Data

# putting numbers to it

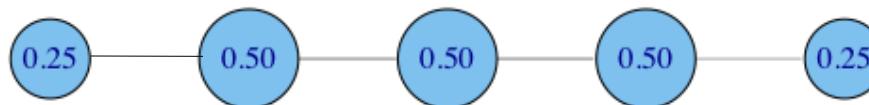
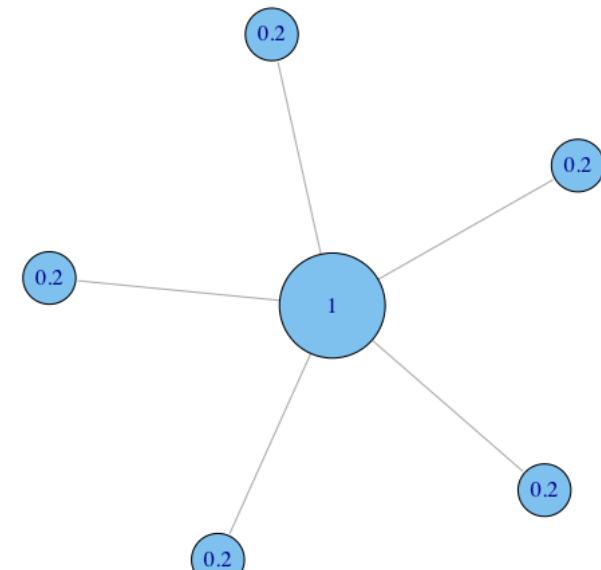
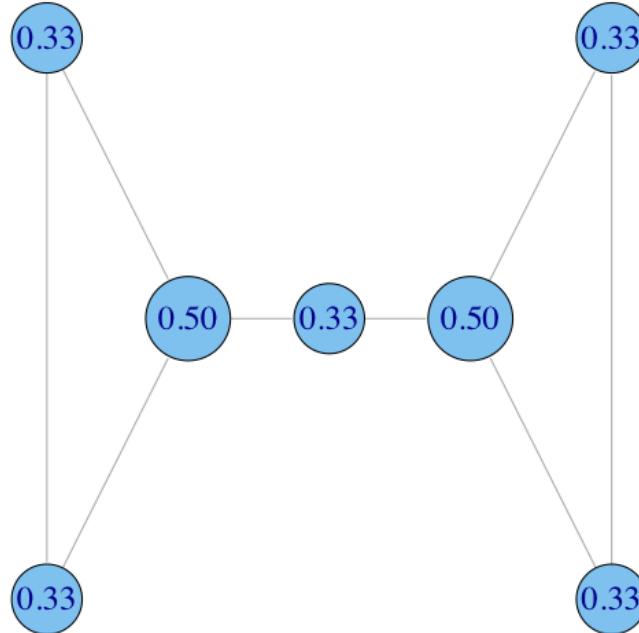
Undirected degree, e.g. nodes with more friends are more central.



Assumption: the connections that your friend has don't matter, it is what they can do directly that does (e.g. go have a beer with you, help you build a deck...)

# normalization

divide degree by the max. possible, i.e.  $(N-1)$



# centralization: skew in distribution

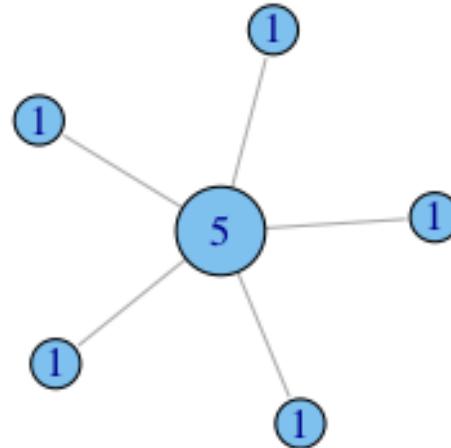
How much variation is there in the centrality scores among the nodes?

Freeman's general formula for centralization (can use other metrics, e.g. gini coefficient or standard deviation):

$$C_D = \frac{\sum_{i=1}^g [C_D(n^*) - C_D(i)]}{[(N-1)(N-2)]}$$

maximum value in the network

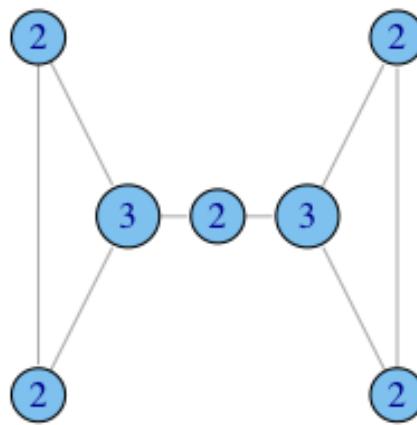
# degree centralization examples



$$C_D = 1.0$$



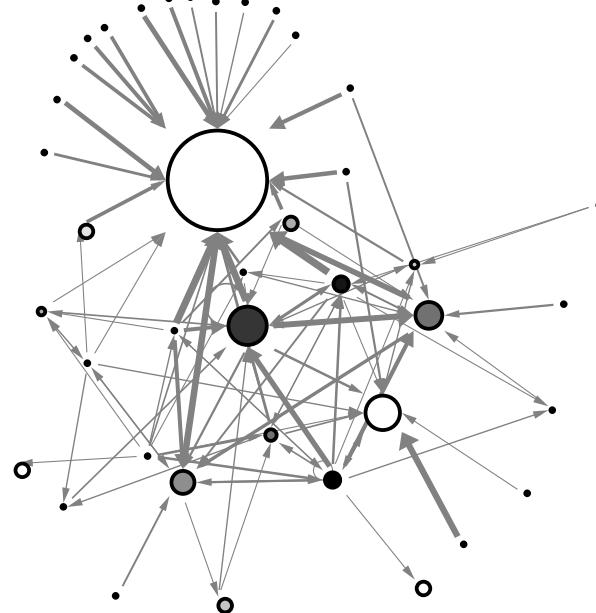
$$C_D = 0.167$$



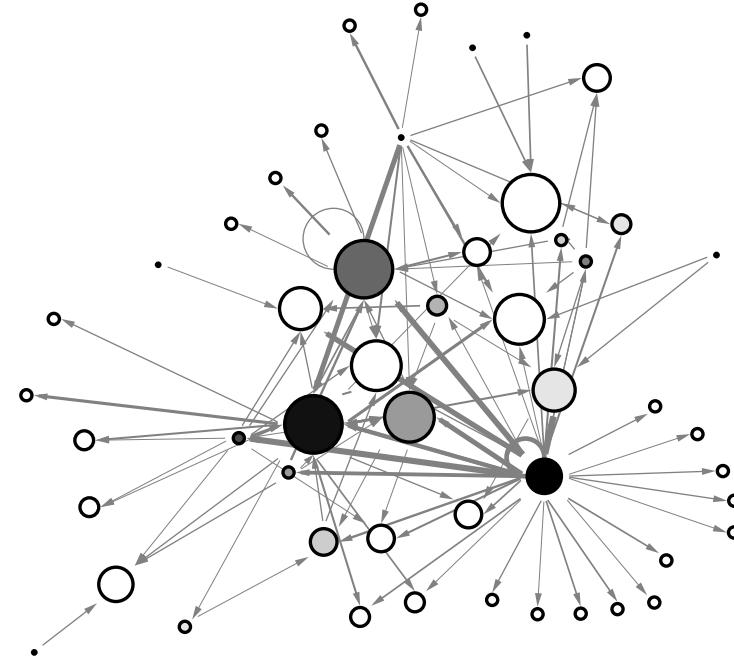
$$C_D = 0.167$$

# real-world examples

## example financial trading networks



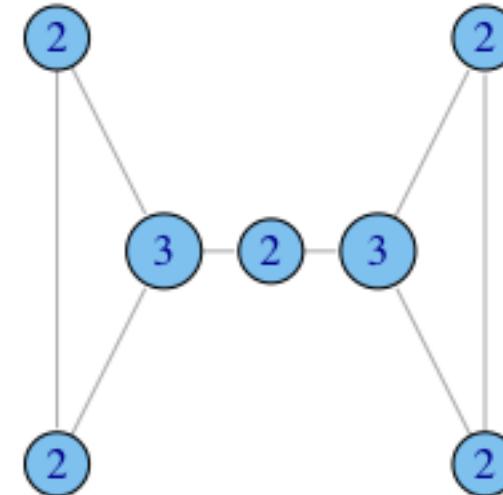
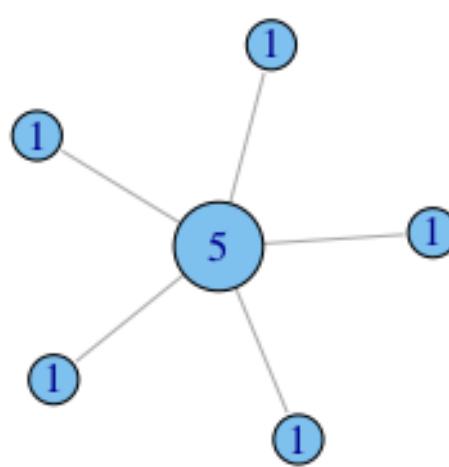
high in-centralization:  
one node buying from  
many others



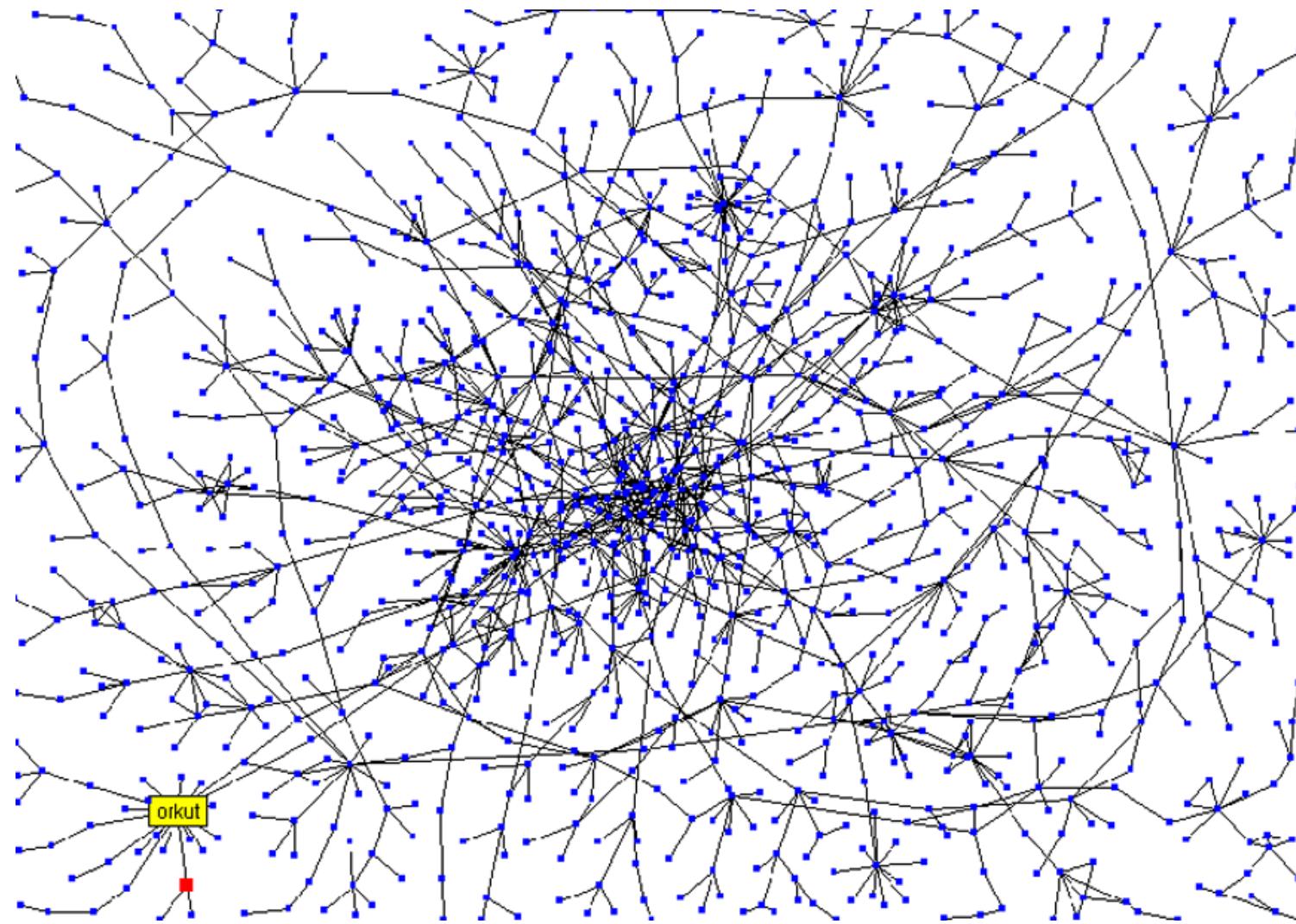
low in-centralization:  
buying is more evenly  
distributed

# what does degree not capture?

In what ways does degree fail to capture centrality in the following graphs?

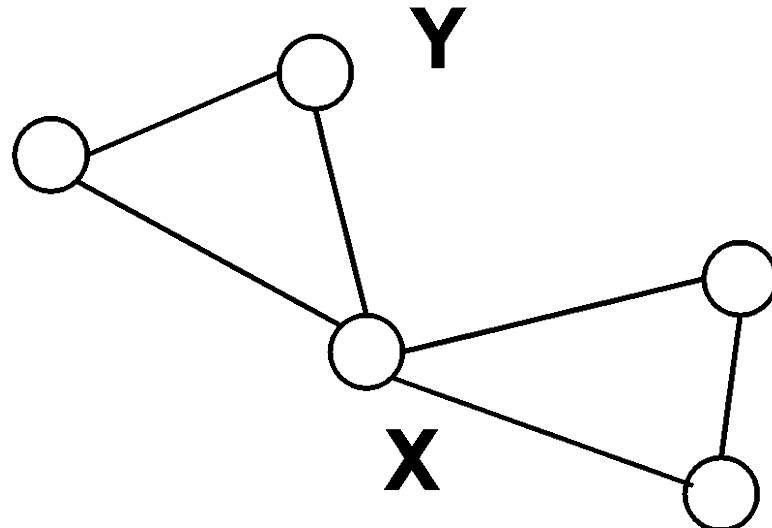


## Stanford Social Web (ca. 1999)

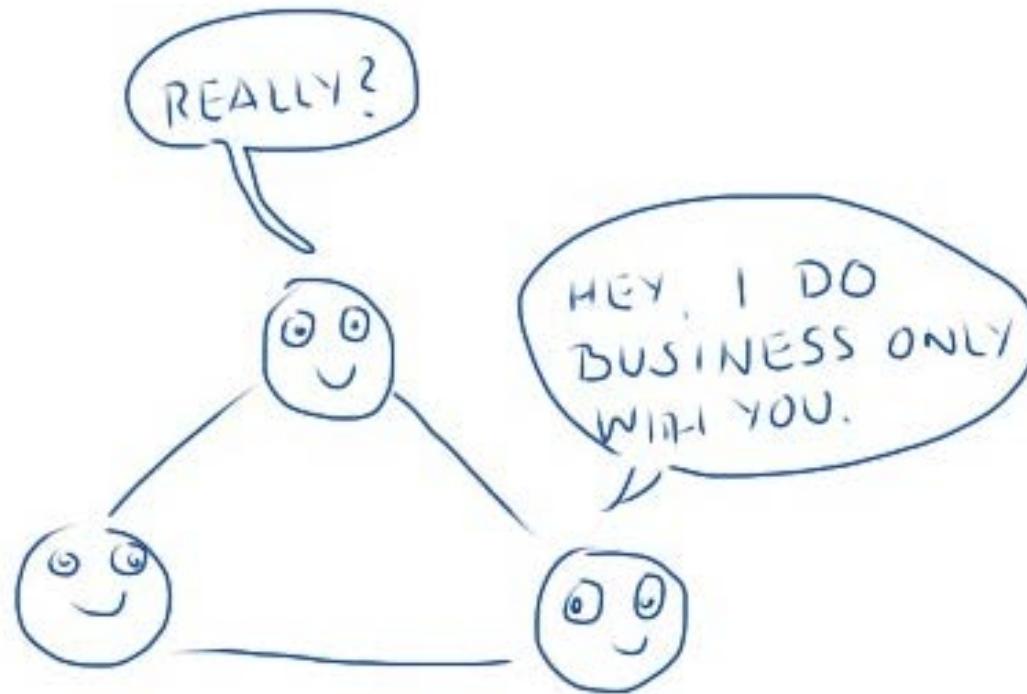


network of personal homepages at Stanford

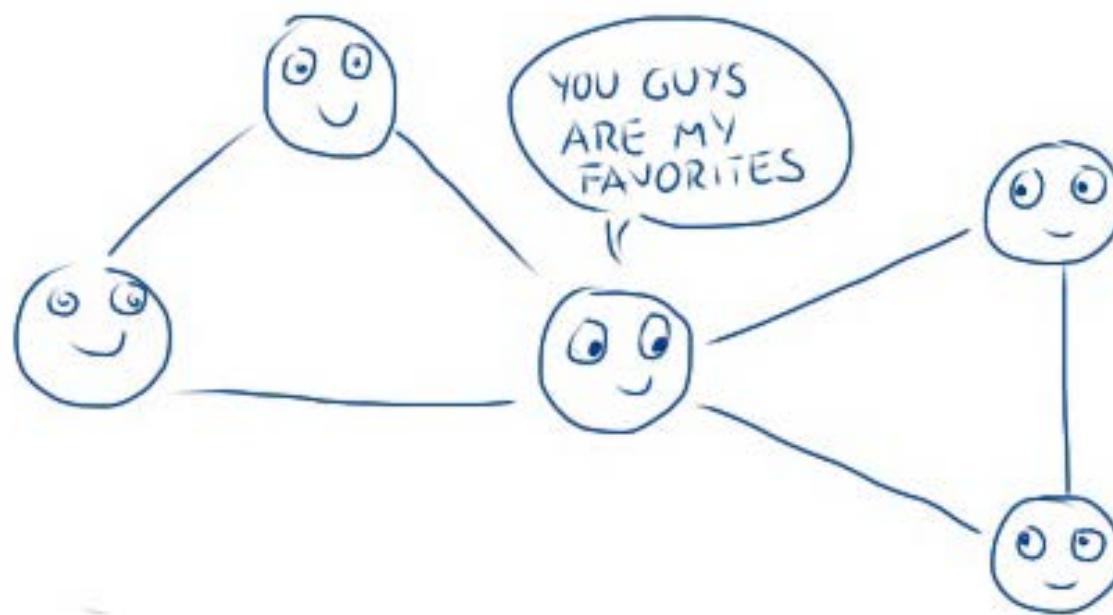
# Brokerage not captured by degree



# constraint

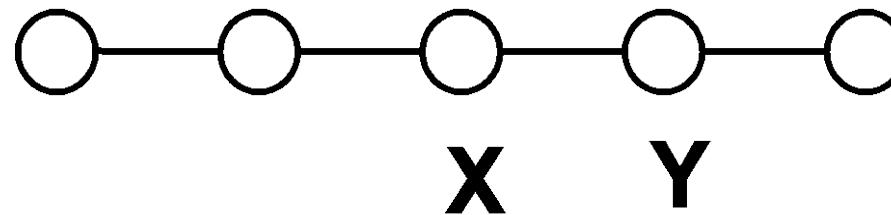


# constraint



# betweenness: capturing brokerage

- ❑ intuition: how many pairs of individuals would have to go through you in order to reach one another in the minimum number of hops?



# betweenness: definition

$$C_B(i) = \sum_{j < k} g_{jk}(i) / g_{jk}$$

Where  $g_{jk}$  = the number of shortest paths connecting  $jk$   
 $g_{jk}(i)$  = the number that actor  $i$  is on.

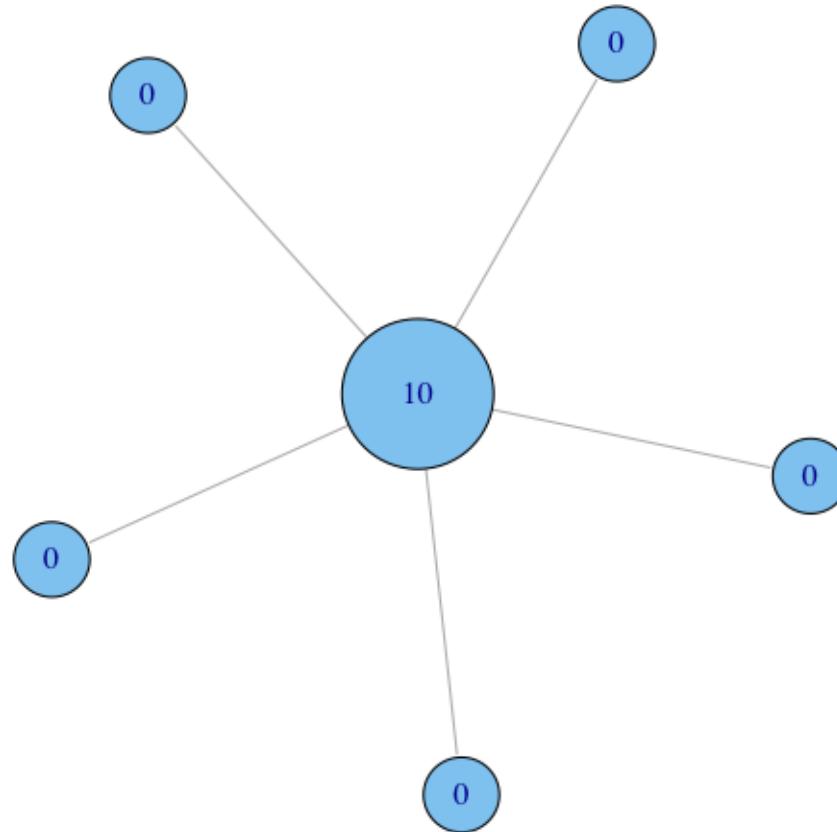
Usually normalized by:

$$C'_B(i) = C_B(i) / [(n - 1)(n - 2)/2]$$

number of pairs of vertices  
excluding the vertex itself

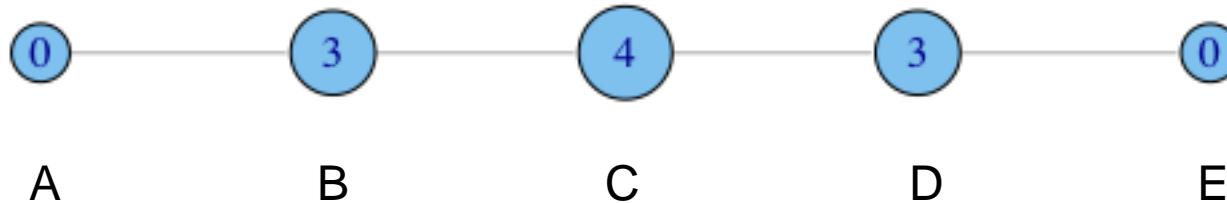
# betweenness on toy networks

□ non-normalized version:



# betweenness on toy networks

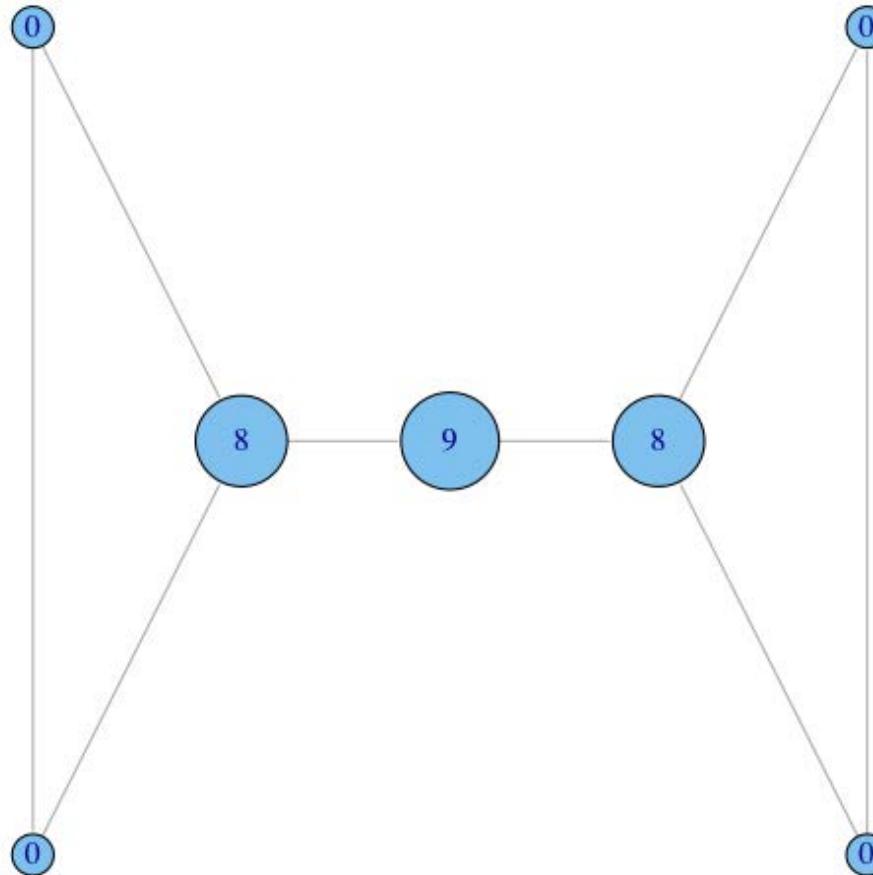
## ■ non-normalized version:



- A lies between no two other vertices
- B lies between A and 3 other vertices: C, D, and E
- C lies between 4 pairs of vertices (A,D),(A,E),(B,D),(B,E)
- note that there are no alternate paths for these pairs to take, so C gets full credit

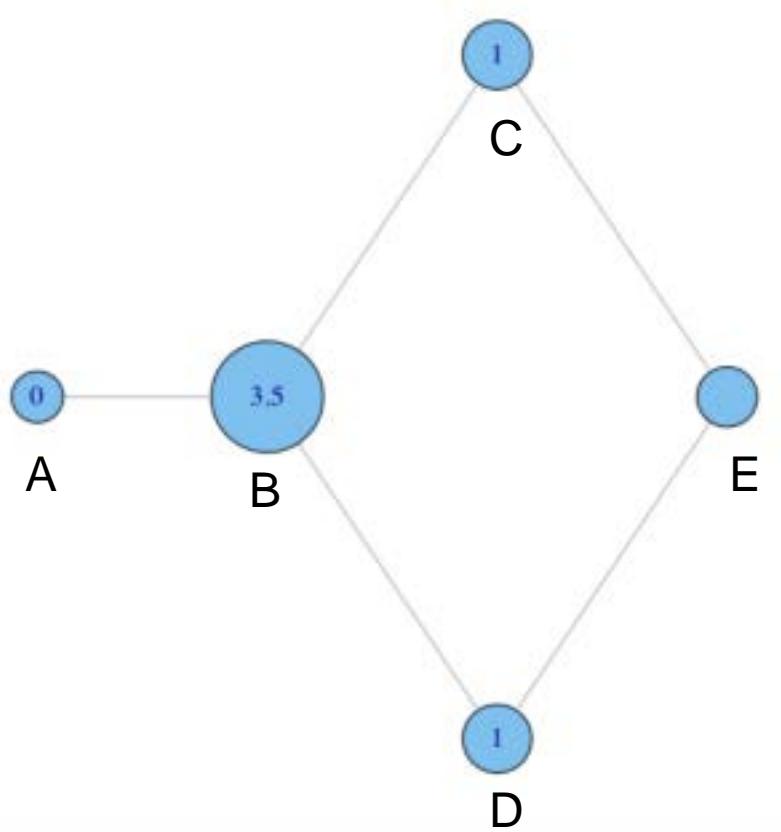
# betweenness on toy networks

□ non-normalized version:



# betweenness on toy networks

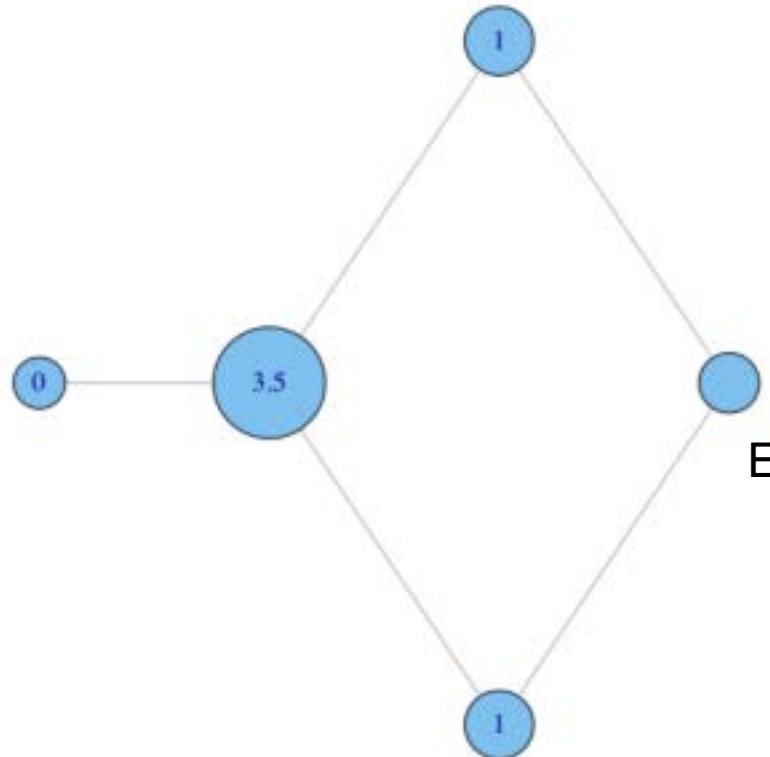
## □ non-normalized version:



- why do C and D each have betweenness 1?
- They are both on shortest paths for pairs (A,E), and (B,E), and so must share credit:
  - $\frac{1}{2} + \frac{1}{2} = 1$

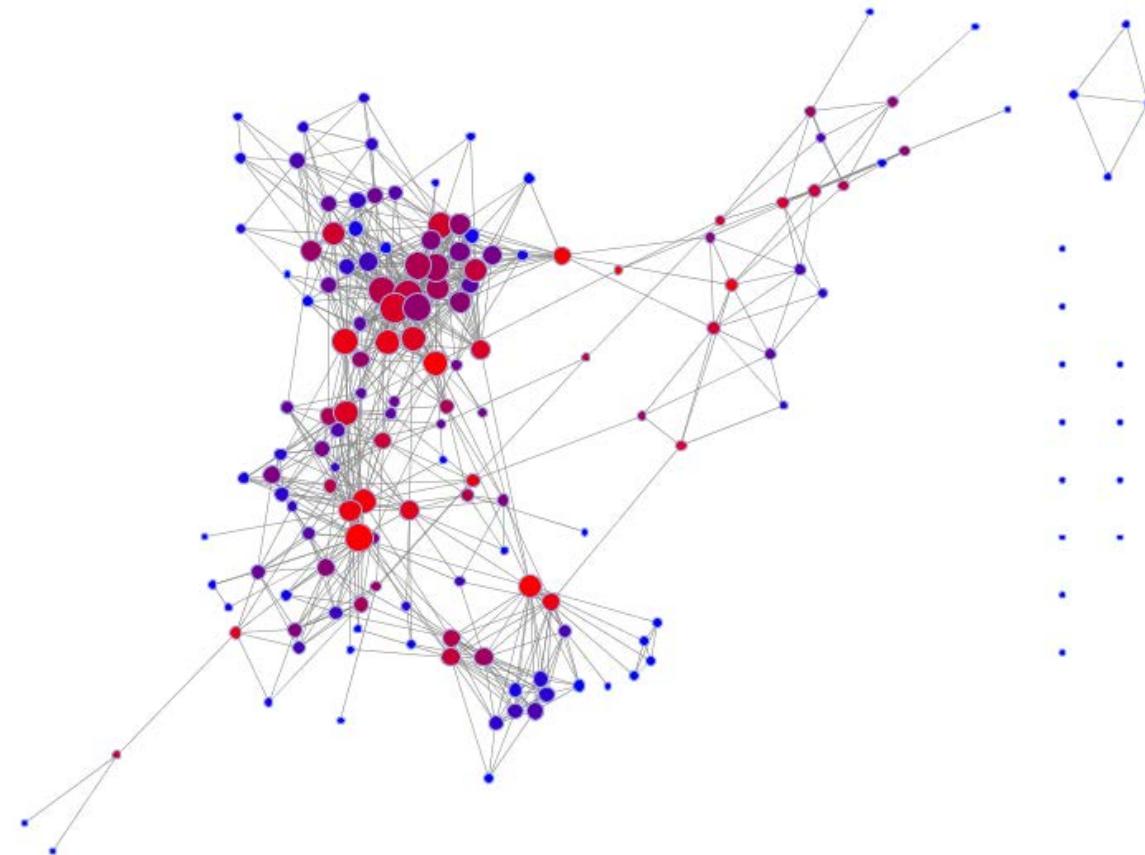
# Quiz Question

■ What is the betweenness of node E?



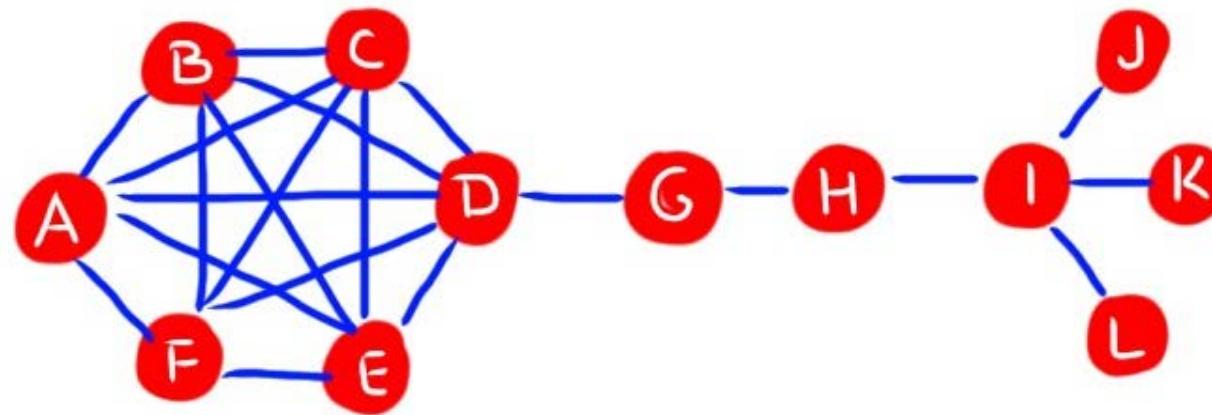
# betweenness: example

Lada's old Facebook network: nodes are sized by degree, and colored by betweenness.



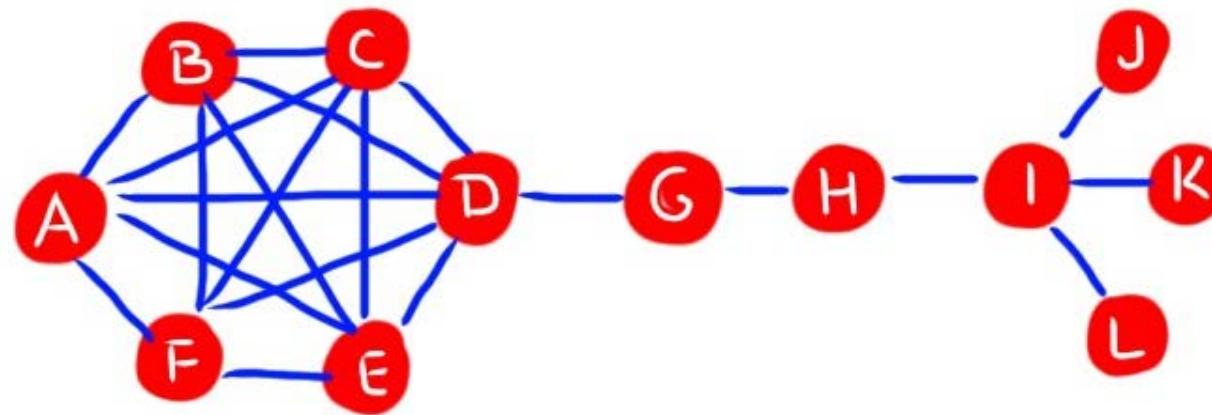
# Quiz Q:

- Find a node that has high betweenness but low degree



# Quiz Q:

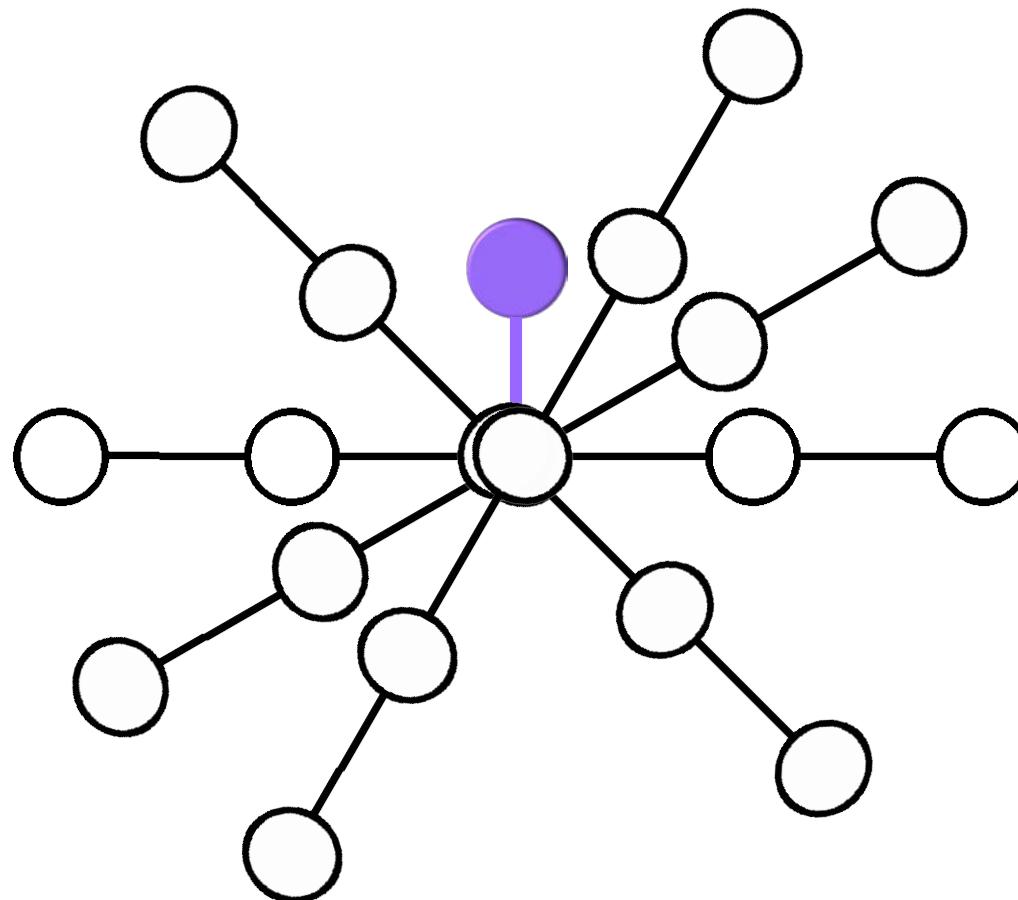
- Find a node that has low betweenness but high degree



# closeness

- ❑ What if it's not so important to have many direct friends?
- ❑ Or be “between” others
- ❑ But one still wants to be in the “middle” of things, not too far from the center

# need not be in a brokerage position



# closeness: definition

Closeness is based on the length of the average shortest path between a node and all other nodes in the network

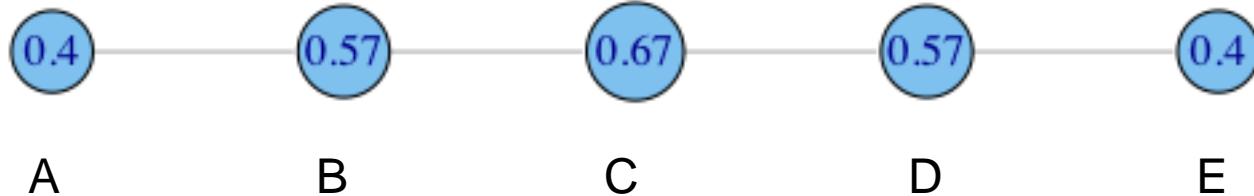
Closeness Centrality:

$$C_c(i) = \left[ \sum_{j=1}^N d(i,j) \right]^{-1}$$

Normalized Closeness Centrality

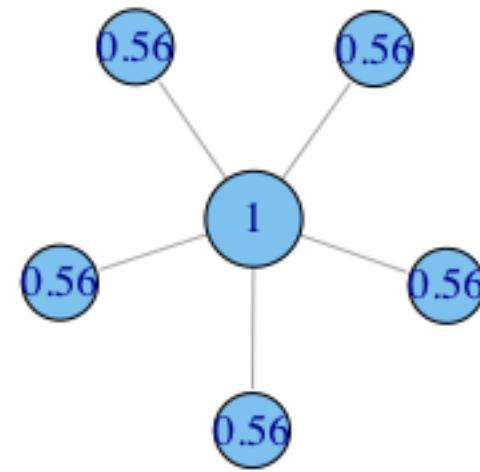
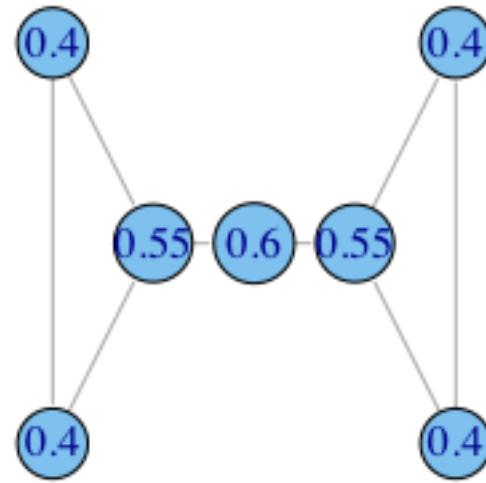
$$C'_c(i) = (C_c(i)) / (N - 1)$$

# closeness: toy example



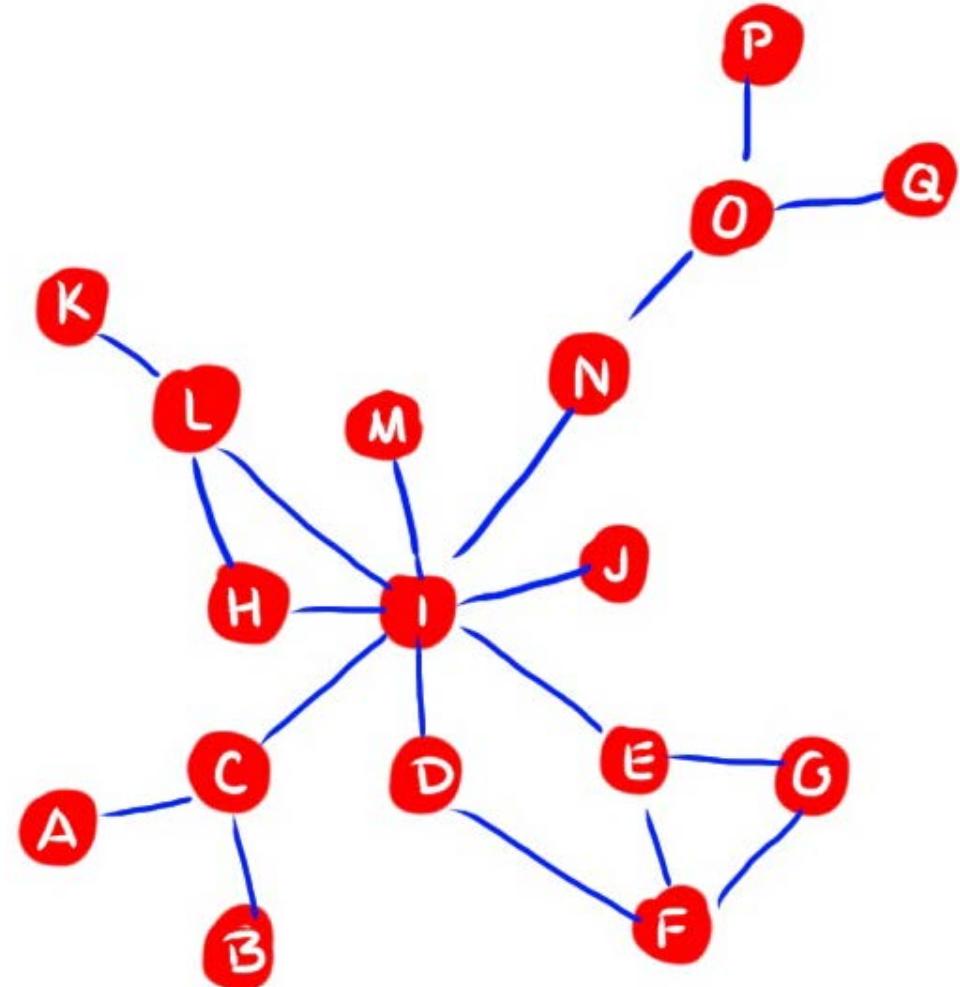
$$C_c(A) = \left[ \frac{\sum_{j=1}^N d(A, j)}{N-1} \right]^{-1} = \left[ \frac{1+2+3+4}{4} \right]^{-1} = \left[ \frac{10}{4} \right]^{-1} = 0.4$$

# closeness: more toy examples



# Quiz Q:

Which node has relatively high degree but low closeness?



# Analysis of Large Graphs: Link Analysis, PageRank

Mining of Massive Datasets  
Leskovec, Rajaraman, and Ullman  
Stanford University



# New Topic: Graph Data!

High dim.  
data

Locality  
sensitive  
hashing

Clustering

Dimensional  
ity  
reduction

Graph  
data

PageRank,  
SimRank

Community  
Detection

Spam  
Detection

Infinite  
data

Filtering  
data  
streams

Web  
advertising

Queries on  
streams

Machine  
learning

SVM

Decision  
Trees

Perceptron,  
kNN

Apps

Recommen  
der systems

Association  
Rules

Duplicate  
document  
detection

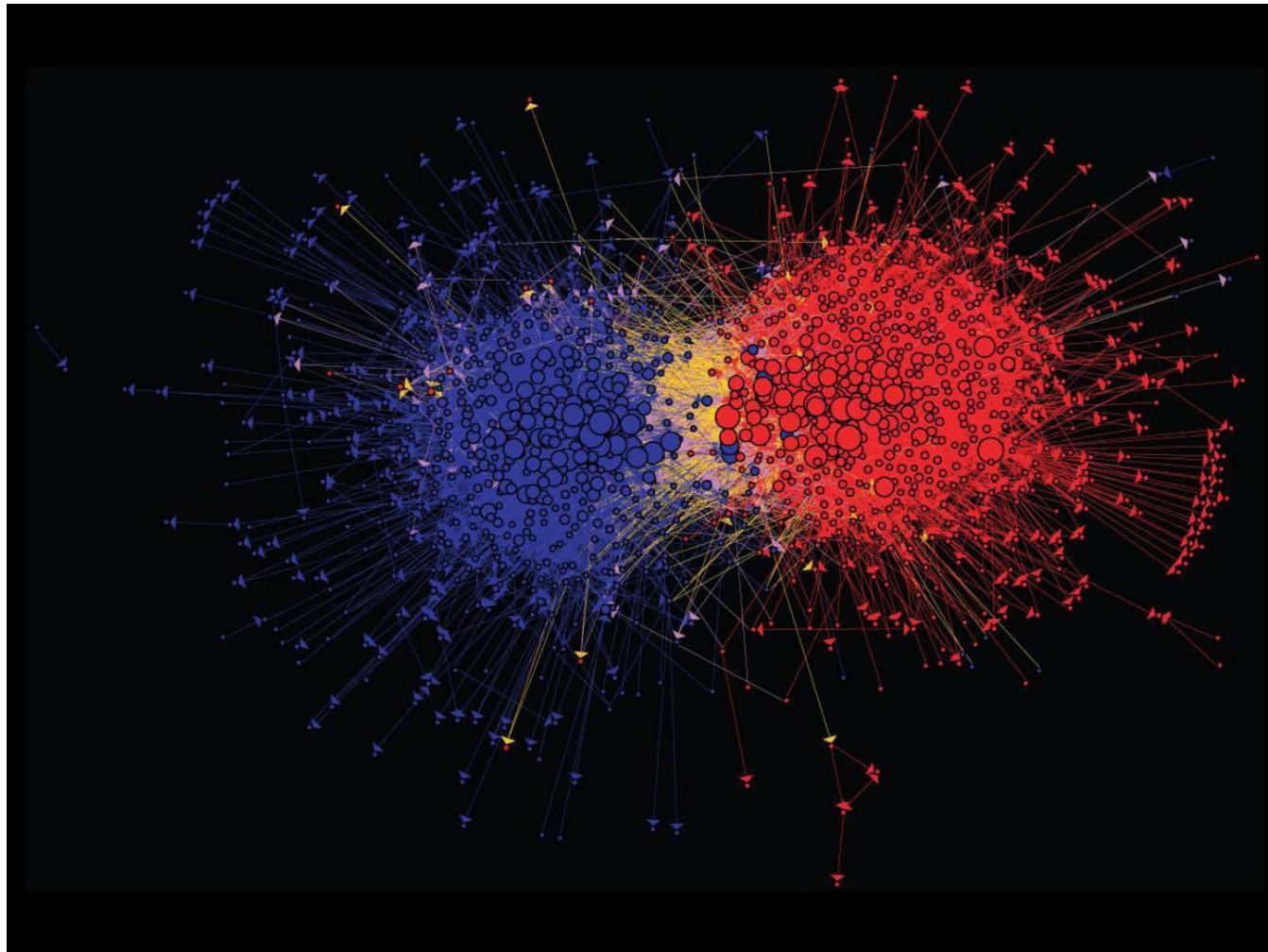
# Graph Data: Social Networks



Facebook social graph

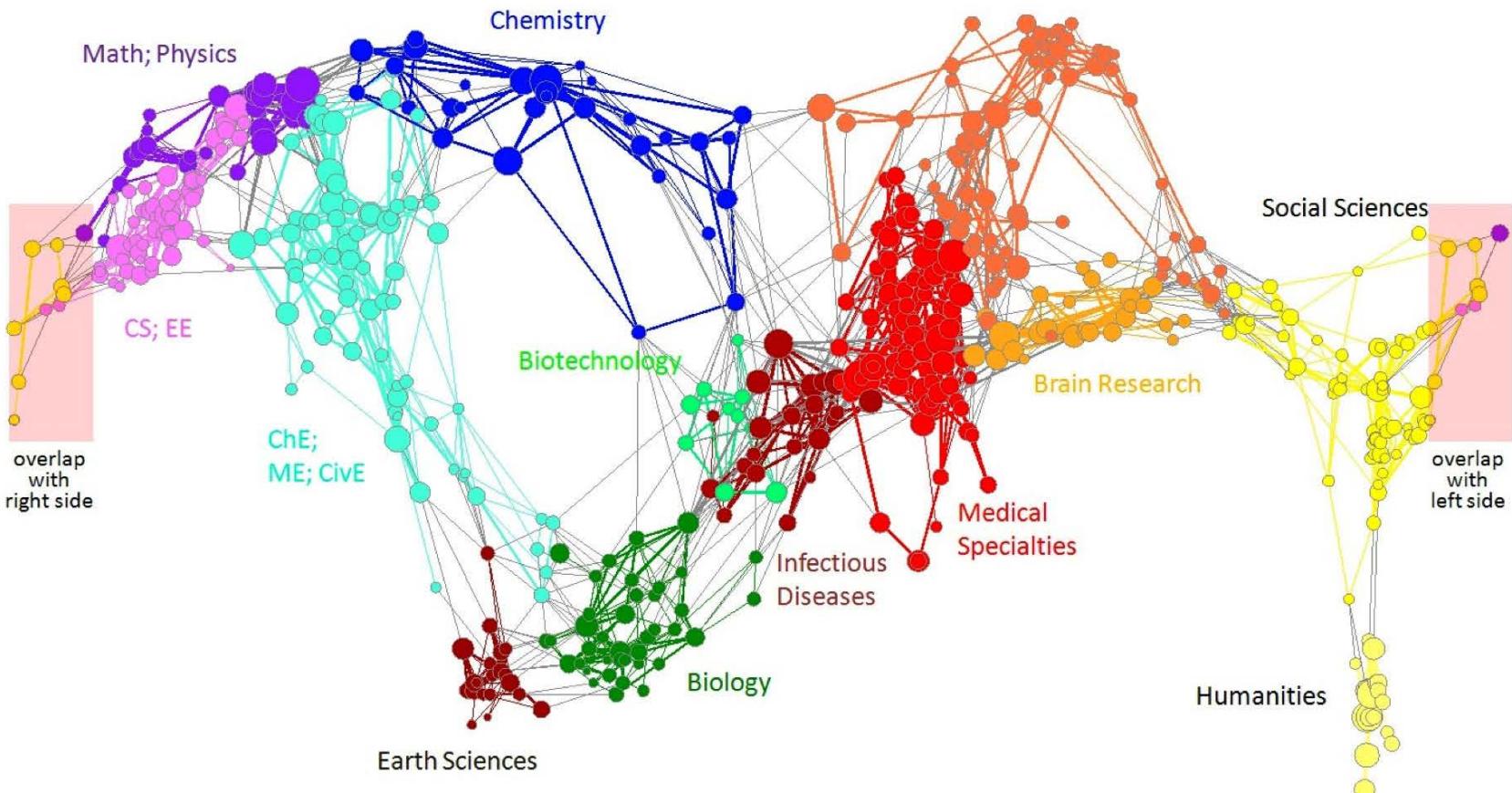
4-degrees of separation [Backstrom-Boldi-Rosa-Ugander-Vigna, 2011]

# Graph Data: Media Networks



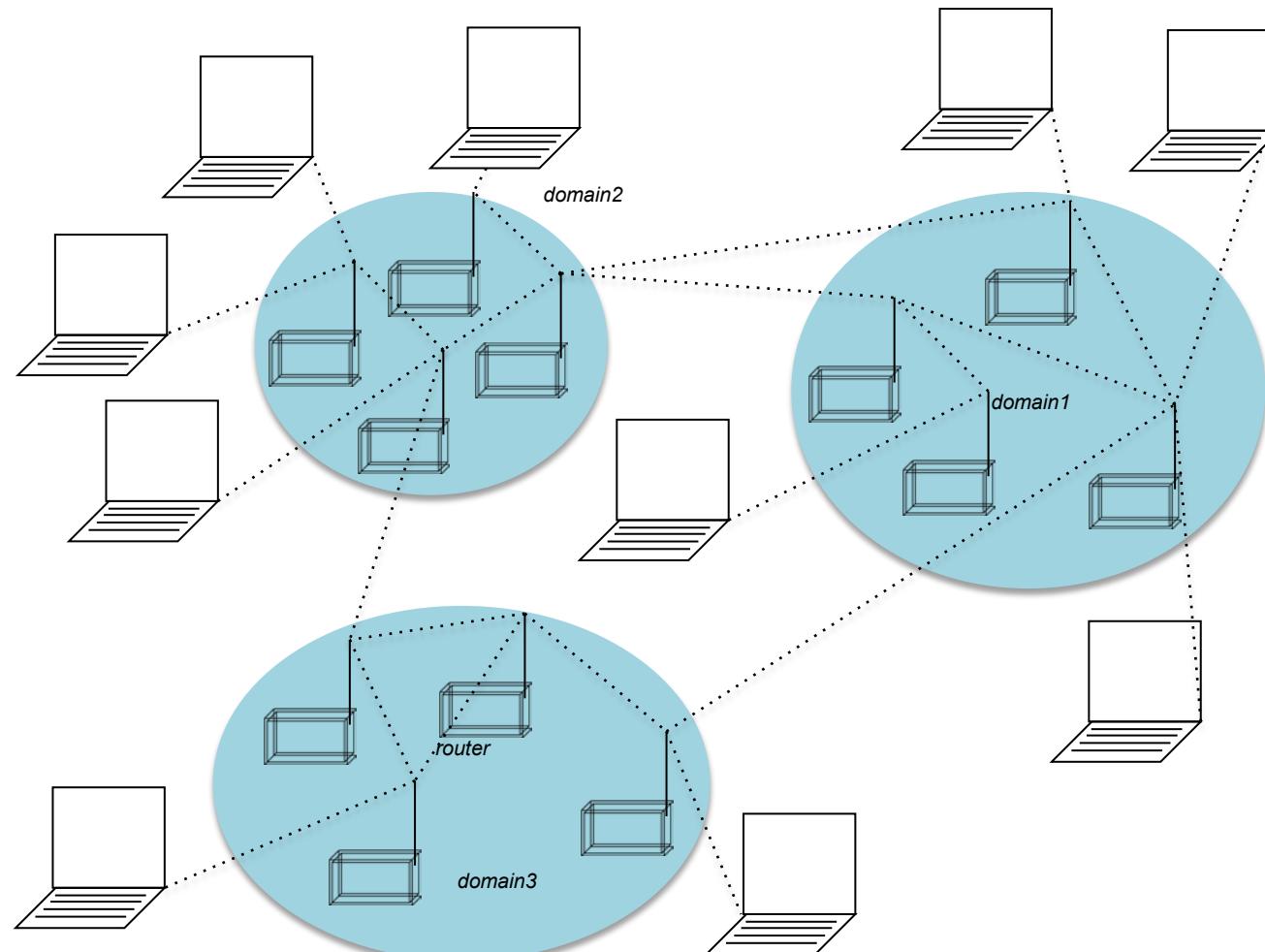
**Connections between political blogs**  
**Polarization of the network [Adamic-Glance, 2005]**

# Graph Data: Information Nets



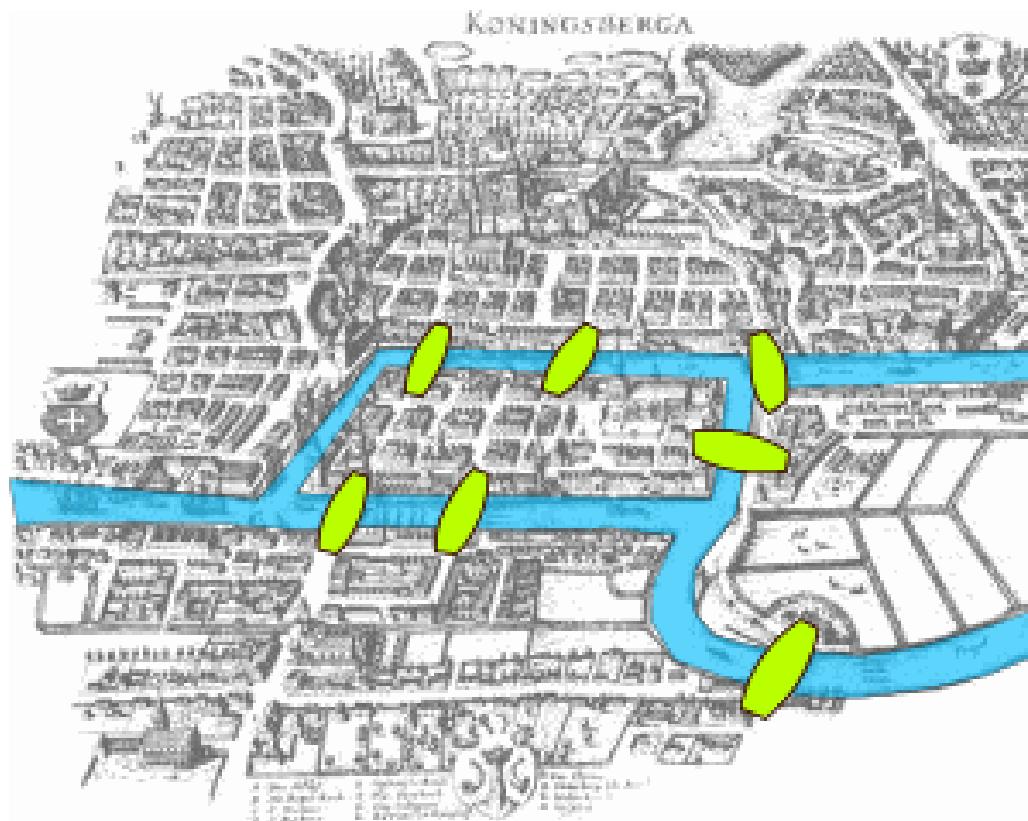
**Citation networks and Maps of science**  
[Börner et al., 2012]

# Graph Data: Communication Nets



# Internet

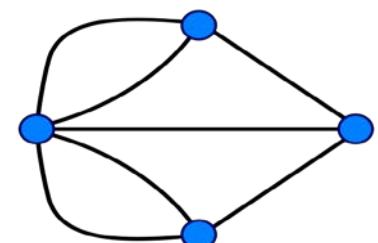
# Graph Data: Technological Networks



## Seven Bridges of Königsberg

[Euler, 1735]

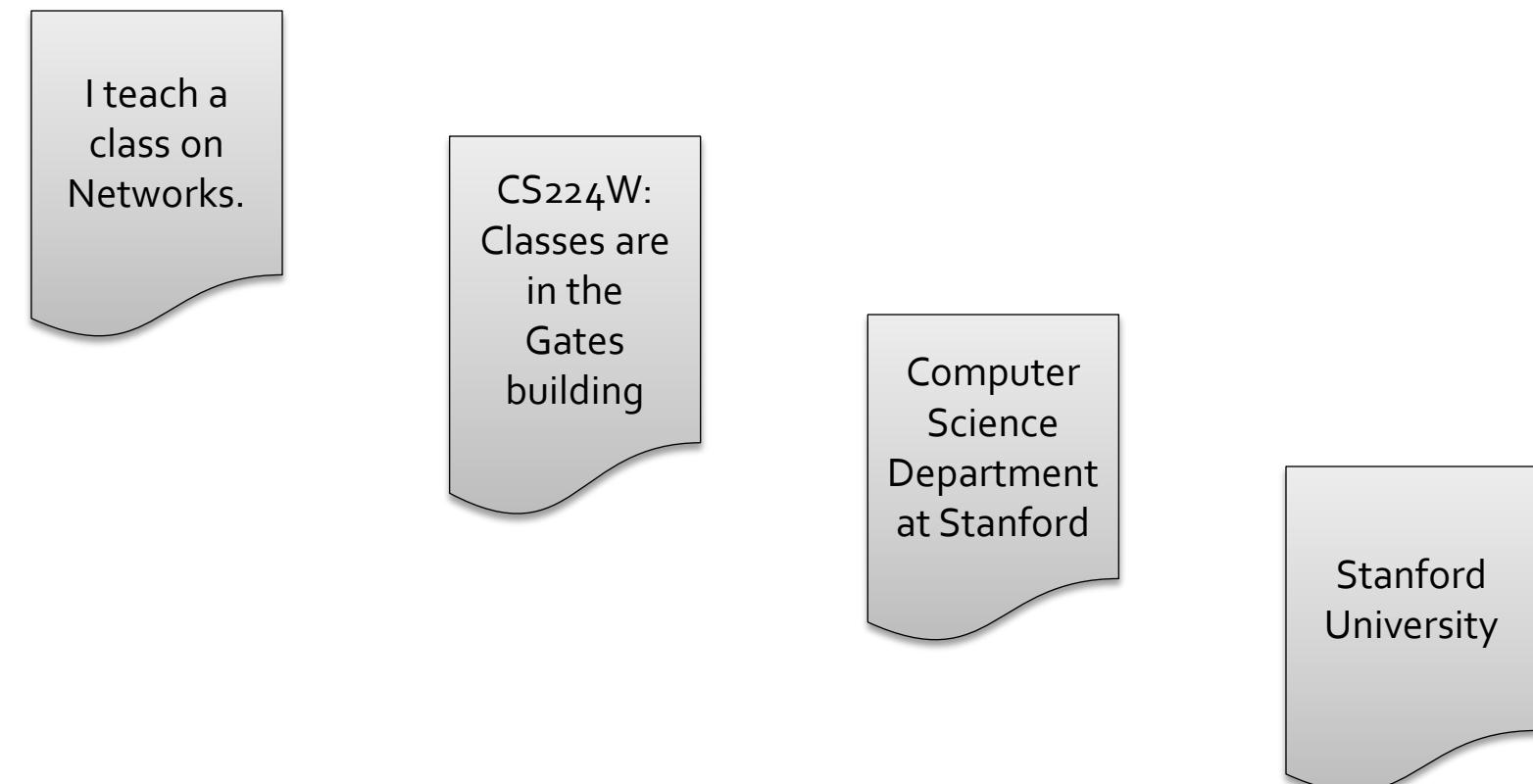
Return to the starting point by traveling each link of the graph once and only once.



# Web as a Graph

- **Web as a directed graph:**

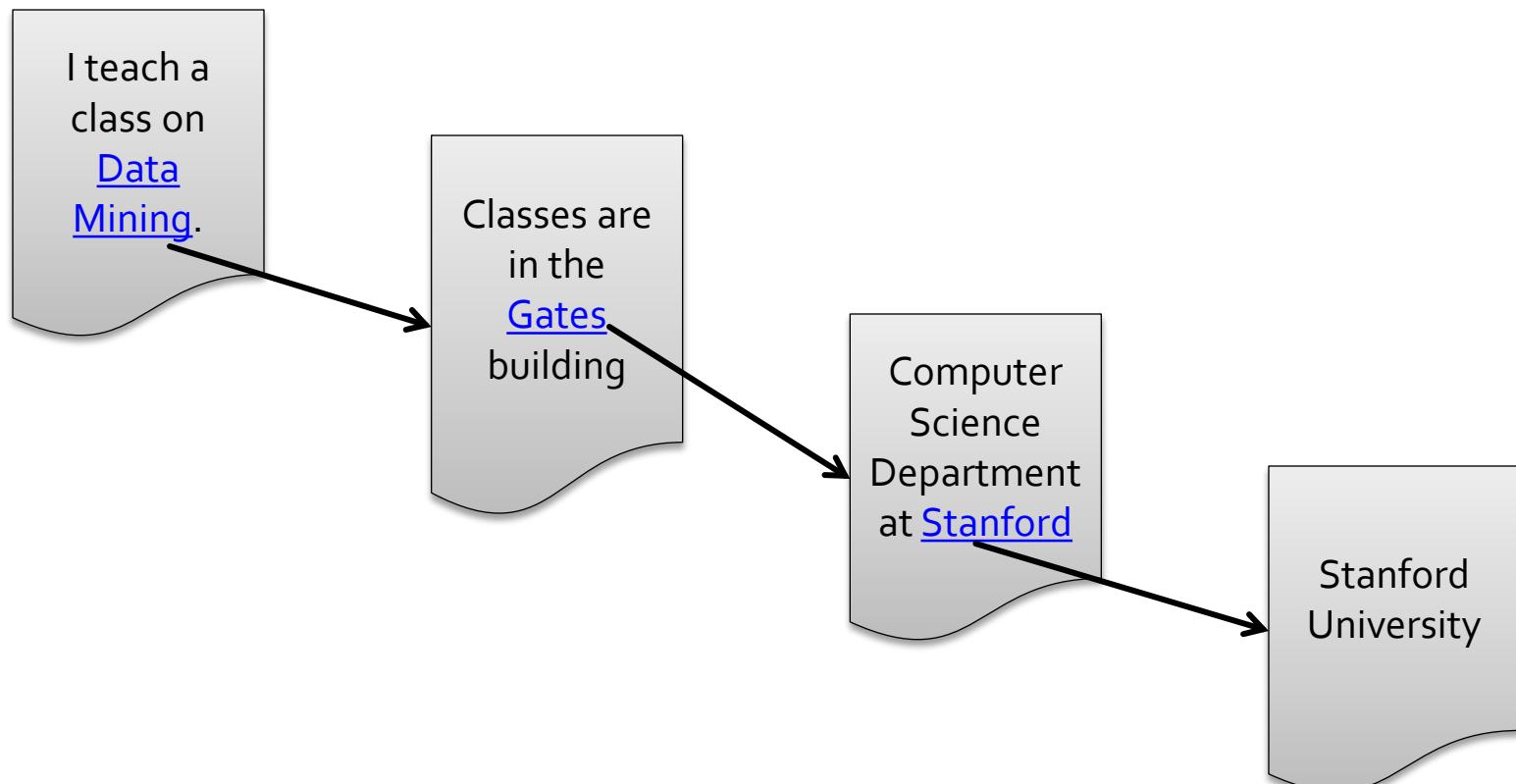
- **Nodes: Webpages**
- **Edges: Hyperlinks**



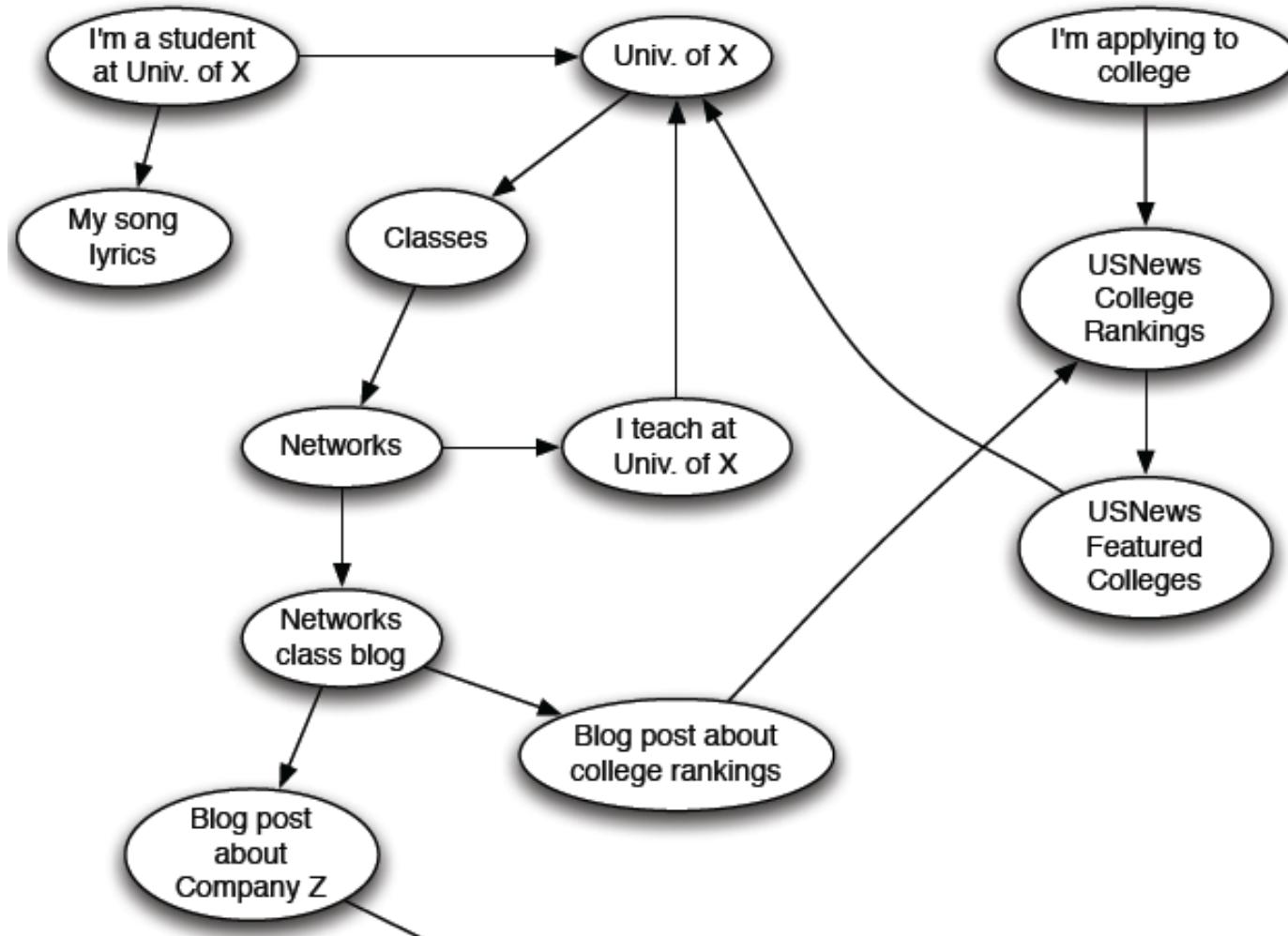
# Web as a Graph

- Web as a directed graph:

- Nodes: Webpages
- Edges: Hyperlinks



# Web as a Directed Graph



# Broad Question

- **How to organize the Web?**
- First try: Human curated  
**Web directories**
  - Yahoo, DMOZ, LookSmart
- Second try: **Web Search**
  - **Information Retrieval** investigates:  
Find relevant docs in a small  
and trusted set
    - Newspaper articles, Patents, etc.
  - **But:** Web is **huge**, full of untrusted documents,  
random things, web spam, etc.



# Web Search: 2 Challenges

## 2 challenges of web search:

- **(1) Web contains many sources of information**  
Who to “trust”?
  - **Trick:** Trustworthy pages may point to each other!
- **(2) What is the “best” answer to query “newspaper”?**
  - No single right answer
  - **Trick:** Pages that actually know about newspapers might all be pointing to many newspapers

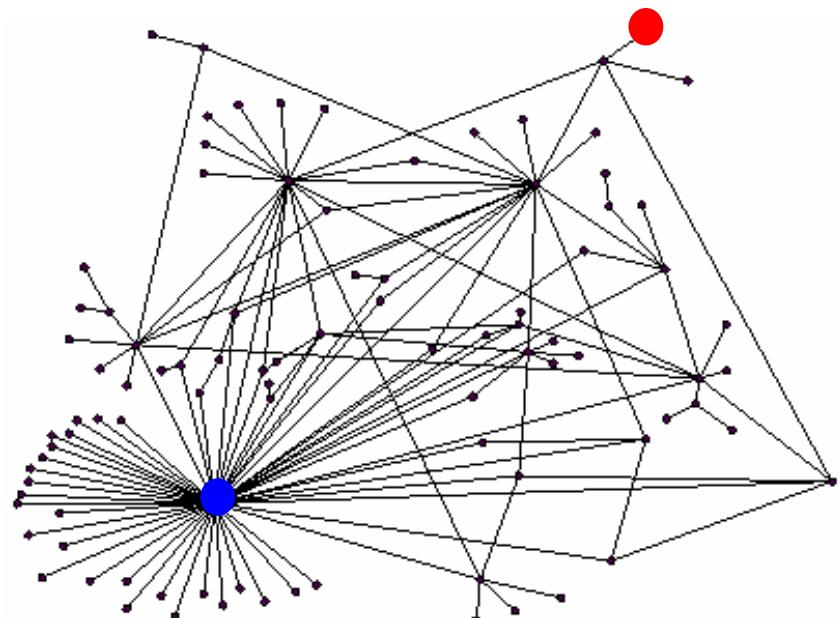
# Ranking Nodes on the Graph

- All web pages are not equally “important”

[www.joe-schmoe.com](http://www.joe-schmoe.com) vs. [www.stanford.edu](http://www.stanford.edu)

- There is large diversity in the web-graph node connectivity.

**Let's rank the pages by the link structure!**



# Link Analysis Algorithms

- We will cover the following **Link Analysis approaches** for computing **importances** of nodes in a graph:
  - Page Rank
  - Hubs and Authorities (HITS)
  - Topic-Specific (Personalized) Page Rank
  - Web Spam Detection Algorithms

# Power Iteration Method

- Given a web graph with  $N$  nodes, where the nodes are pages and edges are hyperlinks
- Power iteration: a simple iterative scheme
  - Suppose there are  $N$  web pages
  - Initialize:  $\mathbf{r}^{(0)} = [1/N, \dots, 1/N]^T$
  - Iterate:  $\mathbf{r}^{(t+1)} = \mathbf{M} \cdot \mathbf{r}^{(t)}$
  - Stop when  $|\mathbf{r}^{(t+1)} - \mathbf{r}^{(t)}|_1 < \varepsilon$

$|\mathbf{x}|_1 = \sum_{1 \leq i \leq N} |x_i|$  is the  $L_1$  norm

Can use any other vector norm, e.g., Euclidean

$$r_j^{(t+1)} = \sum_{i \rightarrow j} \frac{r_i^{(t)}}{d_i}$$

$d_i$  . out-degree of node  $i$

# PageRank: How to solve?

## ■ Power Iteration:

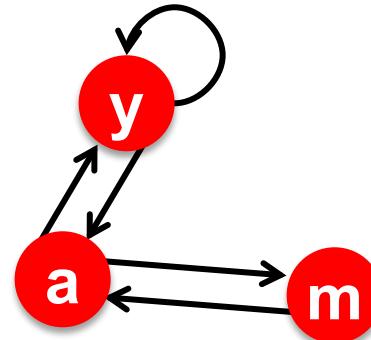
- Set  $r_j = 1/N$
- 1:  $r'_j = \sum_{i \rightarrow j} \frac{r_i}{d_i}$

- 2:  $r = r'$
- If not converged: goto 1

## ■ Example:

$$\begin{bmatrix} r_y \\ r_a \\ r_m \end{bmatrix} = \begin{matrix} 1/3 \\ 1/3 \\ 1/3 \end{matrix}$$

Iteration 0, 1, 2,



	y	a	m
y	1/2	1/2	0
a	1/2	0	1
m	0	1/2	0

$$r_y = r_y/2 + r_a/2$$

$$r_a = r_y/2 + r_m$$

$$r_m = r_a/2$$

# PageRank: How to solve?

## ■ Power Iteration:

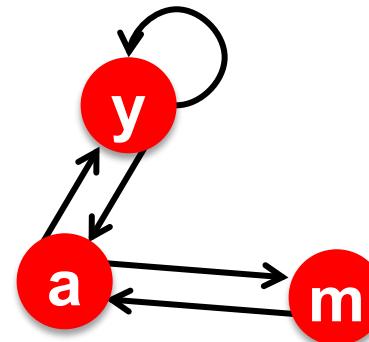
- Set  $r_j = 1/N$
- 1:  $r'_j = \sum_{i \rightarrow j} \frac{r_i}{d_i}$

- 2:  $r = r'$
- If not converged: goto 1

## ■ Example:

$$\begin{bmatrix} r_y \\ r_a \\ r_m \end{bmatrix} = \begin{matrix} 1/3 & 1/3 & 5/12 & 9/24 & 6/15 \\ 1/3 & 3/6 & 1/3 & 11/24 & \dots & 6/15 \\ 1/3 & 1/6 & 3/12 & 1/6 & 3/15 \end{matrix}$$

Iteration 0, 1, 2,



	y	a	m
y	1/2	1/2	0
a	1/2	0	1
m	0	1/2	0

$$r_y = r_y/2 + r_a/2$$

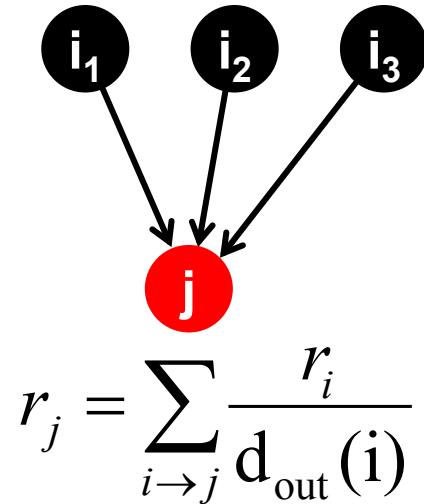
$$r_a = r_y/2 + r_m$$

$$r_m = r_a/2$$

# Random Walk Interpretation

- **Imagine a random web surfer:**

- At any time  $t$ , surfer is on some page  $i$
- At time  $t + 1$ , the surfer follows an out-link from  $i$  uniformly at random
- Ends up on some page  $j$  linked from  $i$
- Process repeats indefinitely



- **Let:**

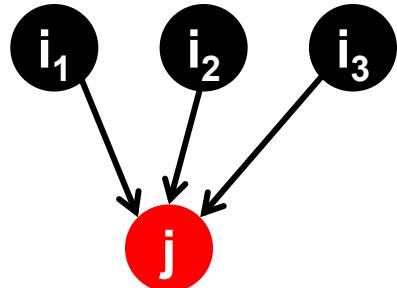
- $p(t)$  ... vector whose  $i^{\text{th}}$  coordinate is the prob. that the surfer is at page  $i$  at time  $t$
- So,  $p(t)$  is a probability distribution over pages

# The Stationary Distribution

## ■ Where is the surfer at time $t+1$ ?

- Follows a link uniformly at random

$$p(t + 1) = M \cdot p(t)$$



$$p(t + 1) = M \cdot p(t)$$

## ■ Suppose the random walk reaches a state

$$p(t + 1) = M \cdot p(t) = p(t)$$

then  $p(t)$  is **stationary distribution** of a random walk

## ■ Our original rank vector $r$ satisfies $r = M \cdot r$

- So,  $r$  is a stationary distribution for the random walk

# Existence and Uniqueness

- A central result from the theory of random walks (a.k.a. Markov processes):

For graphs that satisfy **certain conditions**,  
the **stationary distribution is unique** and  
eventually will be reached no matter what the  
initial probability distribution at time  $t = 0$

# PageRank: The “Flow” Formulation

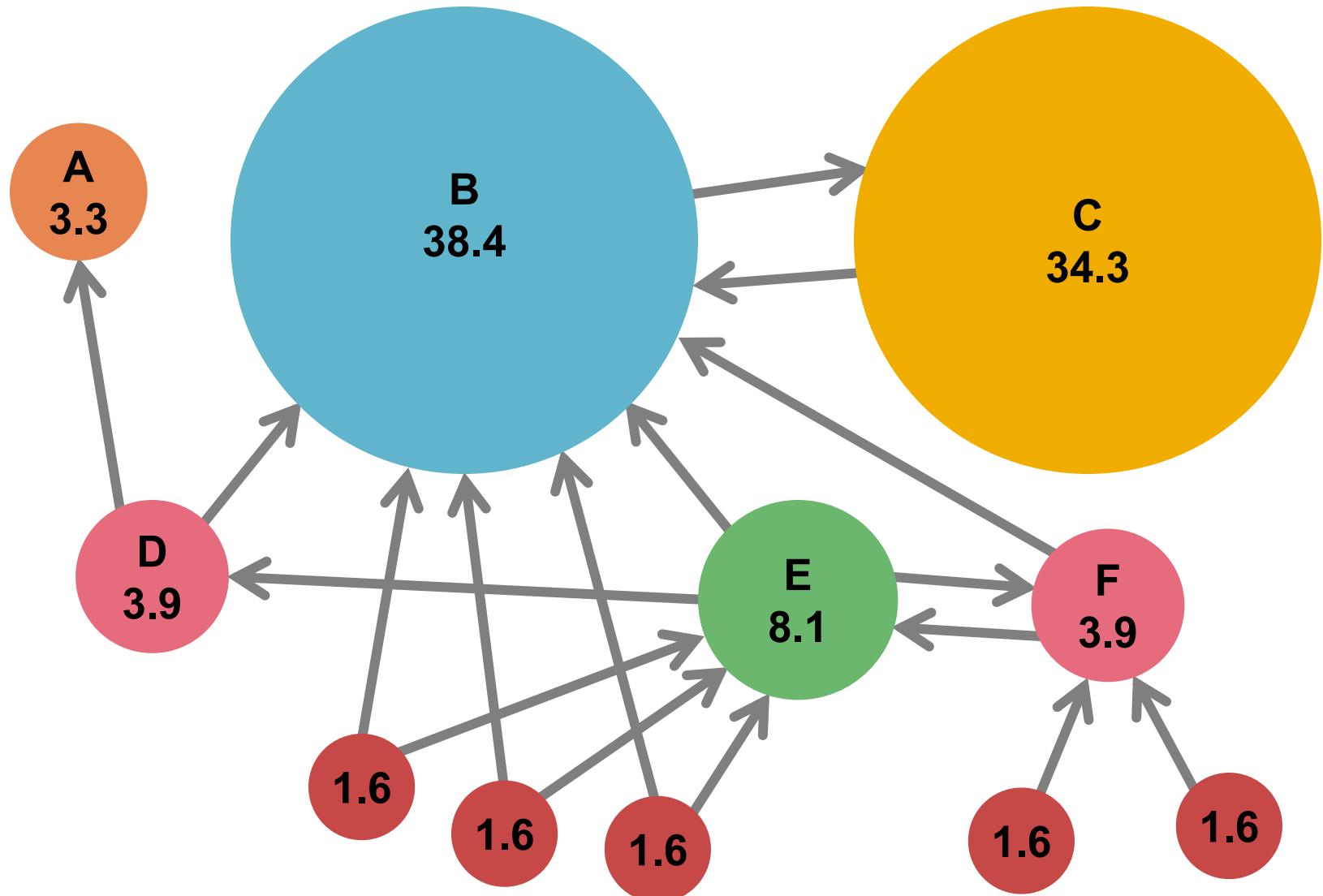
Mining of Massive Datasets  
Leskovec, Rajaraman, and Ullman  
Stanford University



# Links as Votes

- **Idea: Links as votes**
  - Page is more important if it has more links
    - In-coming links? Out-going links?
- **Think of in-links as votes:**
  - [www.stanford.edu](http://www.stanford.edu) has 23,400 in-links
  - [www.joe-schmoe.com](http://www.joe-schmoe.com) has 1 in-link
- **Are all in-links are equal?**
  - Links from important pages count more
  - Recursive question!

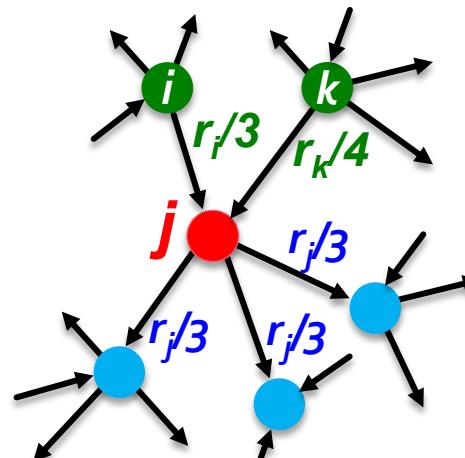
# Example: PageRank Scores



# Simple Recursive Formulation

- Each link's vote is proportional to the **importance** of its source page
- If page  $j$  with importance  $r_j$  has  $n$  out-links, each link gets  $r_j/n$  votes
- Page  $j$ 's own importance is the sum of the votes on its in-links

$$r_j = r_i/3 + r_k/4$$



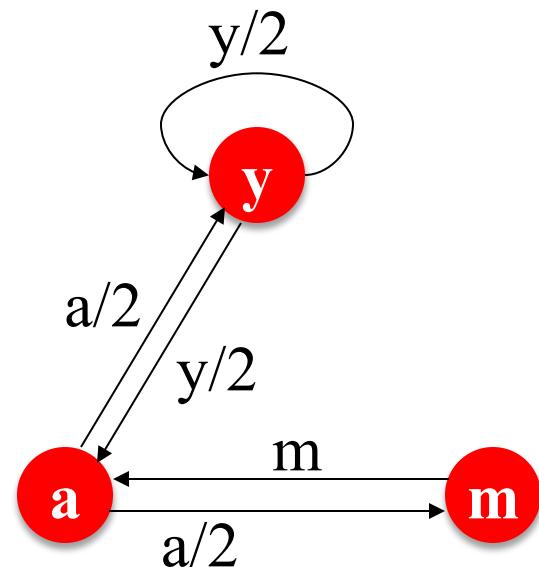
# PageRank: The “Flow” Model

- A “vote” from an important page is worth more
- A page is important if it is pointed to by other important pages
- Define a “rank”  $r_j$  for page  $j$

$$r_j = \sum_{i \rightarrow j} \frac{r_i}{d_i}$$

$d_i$       out-degree of node  $i$

The web in 1839



“Flow” equations:

$$r_y = r_y/2 + r_a/2$$

$$r_a = r_y/2 + r_m$$

$$r_m = r_a/2$$

# Solving the Flow Equations

- **3 equations, 3 unknowns, no constants**
  - No unique solution
  - All solutions equivalent modulo the scale factor
- **Additional constraint forces uniqueness:**
  - $r_y + r_a + r_m = 1$
  - **Solution:**  $r_y = \frac{2}{5}$ ,  $r_a = \frac{2}{5}$ ,  $r_m = \frac{1}{5}$
- **Gaussian elimination method works for small examples, but we need a better method for large web-size graphs**
- **We need a new formulation!**

Flow equations:

$$r_y = r_y/2 + r_a/2$$

$$r_a = r_y/2 + r_m$$

$$r_m = r_a/2$$

# PageRank: The Google Formulation

Mining of Massive Datasets  
Leskovec, Rajaraman, and Ullman  
Stanford University



# PageRank: Three Questions

$$r_j^{(t+1)} = \sum_{i \rightarrow j} \frac{r_i^{(t)}}{d_i}$$

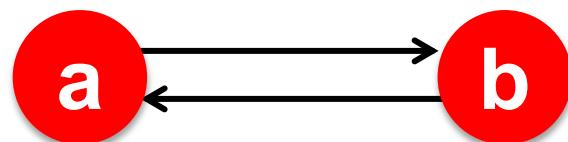
or  
equivalently

$$r = Mr$$

- Does this converge?
- Does it converge to what we want?
- Are results reasonable?

# Does this converge?

- The “Spider trap” problem:



$$r_j^{(t+1)} = \sum_{i \rightarrow j} \frac{r_i^{(t)}}{d_i}$$

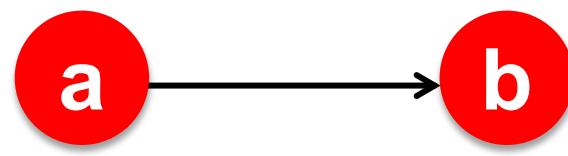
- Example:

$$\begin{array}{c} r_a \\ r_b \end{array} = \begin{array}{cccc} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{array}$$

Iteration 0, 1, 2,

# Does it converge to what we want?

- The “Dead end” problem:



$$r_j^{(t+1)} = \sum_{i \rightarrow j} \frac{r_i^{(t)}}{d_i}$$

- Example:

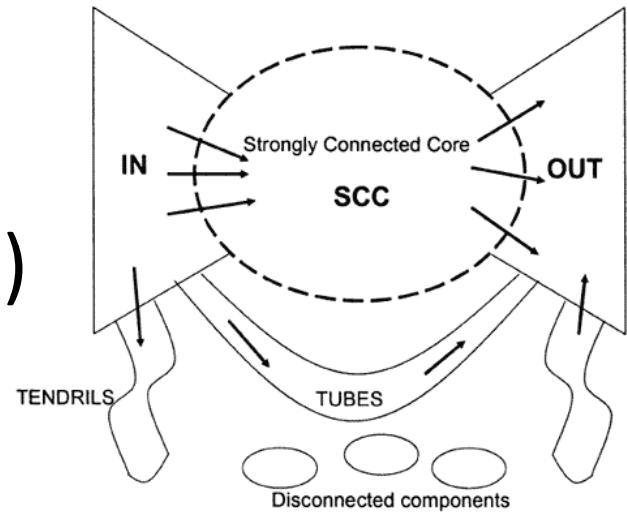
$$\begin{array}{lll} r_a & = & 1 \quad 0 \quad 0 \quad 0 \\ r_b & = & 0 \quad 1 \quad 0 \quad 0 \end{array}$$

Iteration 0, 1, 2,

# RageRank: Problems

## 2 problems:

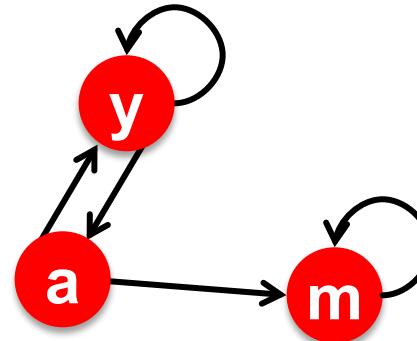
- (1) Some pages are **dead ends** (have no out-links)
  - Such pages cause importance to “leak out”
- (2) **Spider traps**  
(all out-links are within the group)
  - Eventually spider traps absorb all importance



# Problem: Spider Traps

## Power Iteration:

- Set  $r_j = 1$
- $r_j = \sum_{i \rightarrow j} \frac{r_i}{d_i}$
- And iterate



	y	a	m
y	1/2	1/2	0
a	1/2	0	0
m	0	1/2	1

$$r_y = r_y/2 + r_a/2$$

$$r_a = r_y/2$$

$$r_m = r_a/2 + r_m$$

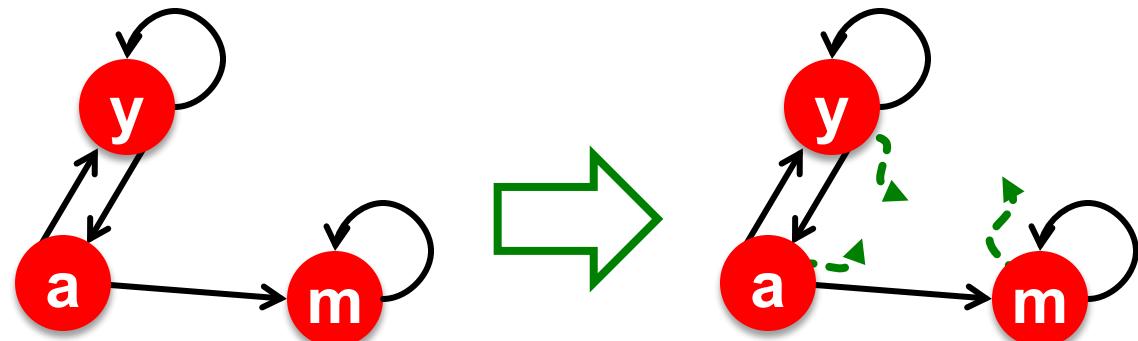
## Example:

$$\begin{bmatrix} r_y \\ r_a \\ r_m \end{bmatrix} = \begin{matrix} 1/3 & 2/6 & 3/12 & 5/24 & 0 \\ 1/3 & 1/6 & 2/12 & 3/24 & \dots & 0 \\ 1/3 & 3/6 & 7/12 & 16/24 & & 1 \end{matrix}$$

Iteration 0, 1, 2,

# Solution: Random Teleports

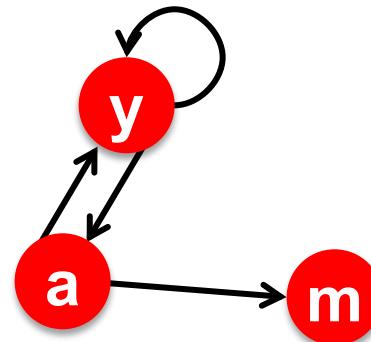
- The Google solution for spider traps: **At each time step, the random surfer has two options**
  - With prob.  $\beta$ , follow a link at random
  - With prob.  $1-\beta$ , jump to some random page
  - Common values for  $\beta$  are in the range 0.8 to 0.9
- **Surfer will teleport out of spider trap within a few time steps**



# Problem: Dead Ends

## Power Iteration:

- Set  $r_j = 1$
- $r_j = \sum_{i \rightarrow j} \frac{r_i}{d_i}$
- And iterate



	y	a	m
y	1/2	1/2	0
a	1/2	0	0
m	0	1/2	0

$$r_y = r_y/2 + r_a/2$$

$$r_a = r_y/2$$

$$r_m = r_a/2$$

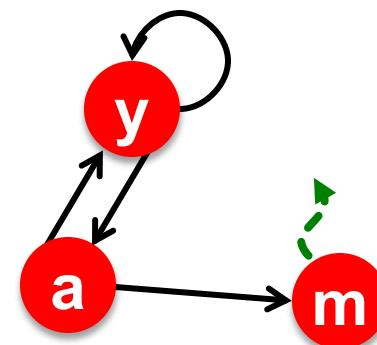
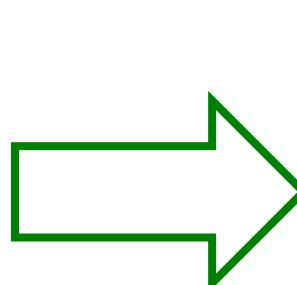
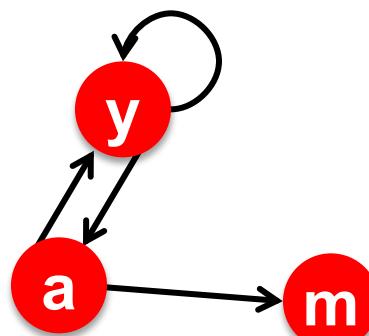
## Example:

$$\begin{bmatrix} r_y \\ r_a \\ r_m \end{bmatrix} = \begin{matrix} 1/3 & 2/6 & 3/12 & 5/24 & 0 \\ 1/3 & 1/6 & 2/12 & 3/24 & \dots & 0 \\ 1/3 & 1/6 & 1/12 & 2/24 & 0 \end{matrix}$$

Iteration 0, 1, 2,

# Solution: Always Teleport

- **Teleports:** Follow random teleport links with probability 1.0 from dead-ends
  - Adjust matrix accordingly



	y	a	m
y	$\frac{1}{2}$	$\frac{1}{2}$	0
a	$\frac{1}{2}$	0	0
m	0	$\frac{1}{2}$	0

	y	a	m
y	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{3}$
a	$\frac{1}{2}$	0	$\frac{1}{3}$
m	0	$\frac{1}{2}$	$\frac{1}{3}$

# PageRank: Matrix Formulation

- **Stochastic adjacency matrix  $M$** 
  - Let page  $i$  has  $d_i$  out-links
  - If  $i \rightarrow j$ , then  $M_{ji} = \frac{1}{d_i}$  else  $M_{ji} = 0$ 
    - $M$  is a **column stochastic matrix**
      - Columns sum to 1
- **Rank vector  $r$ :** vector with an entry per page
  - $r_i$  is the importance score of page  $i$
  - $\sum_i r_i = 1$
- **The flow equations can be written**
$$r = M \cdot r$$

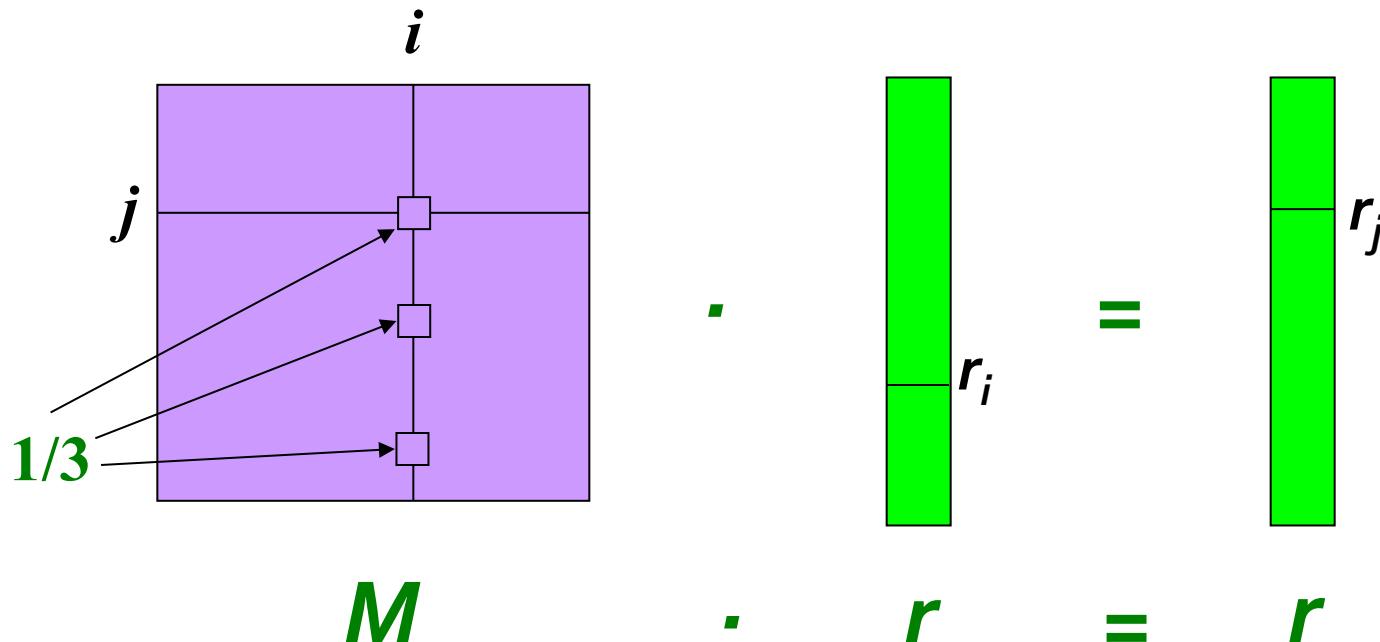
$$r_j = \sum_{i \rightarrow j} \frac{r_i}{d_i}$$

# Example

- Remember the flow equation:  $r_j = \sum_{i \rightarrow j} \frac{r_i}{d_i}$
- Flow equation in the matrix form

$$M \cdot r = r$$

- Suppose page  $i$  links to 3 pages, including  $j$



# Eigenvector Formulation

- The flow equations can be written

$$\mathbf{r} = \mathbf{M} \cdot \mathbf{r}$$

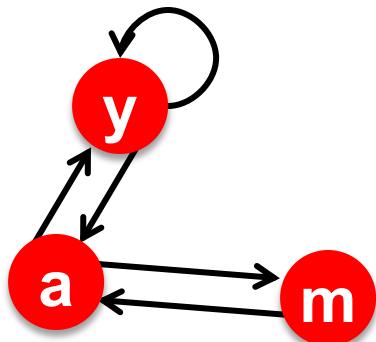
- So the rank vector  $\mathbf{r}$  is an eigenvector of the stochastic web matrix  $\mathbf{M}$

- In fact, its first or principal eigenvector, with corresponding eigenvalue  $1$ 
  - Largest eigenvalue of  $\mathbf{M}$  is  $1$  since  $\mathbf{M}$  is column stochastic
    - *Why? We know  $\mathbf{r}$  is unit length and each column of  $\mathbf{M}$  sums to one, so  $\mathbf{M}\mathbf{r} \leq 1$*

- We can now efficiently solve for  $\mathbf{r}$ !  
The method is called Power iteration

NOTE:  $\mathbf{x}$  is an eigenvector with the corresponding eigenvalue  $\lambda$  if:  
 $A\mathbf{x} = \lambda\mathbf{x}$

# Example: Flow Equations & M



	y	a	m
y	1/2	1/2	0
a	1/2	0	1
m	0	1/2	0

$$r = M \cdot r$$

$$r_y = r_y/2 + r_a/2$$

$$r_a = r_y/2 + r_m$$

$$r_m = r_a/2$$

$$\begin{bmatrix} y \\ a \\ m \end{bmatrix} = \begin{bmatrix} 1/2 & 1/2 & 0 \\ 1/2 & 0 & 1 \\ 0 & 1/2 & 0 \end{bmatrix} \begin{bmatrix} y \\ a \\ m \end{bmatrix}$$

# Recommender Systems

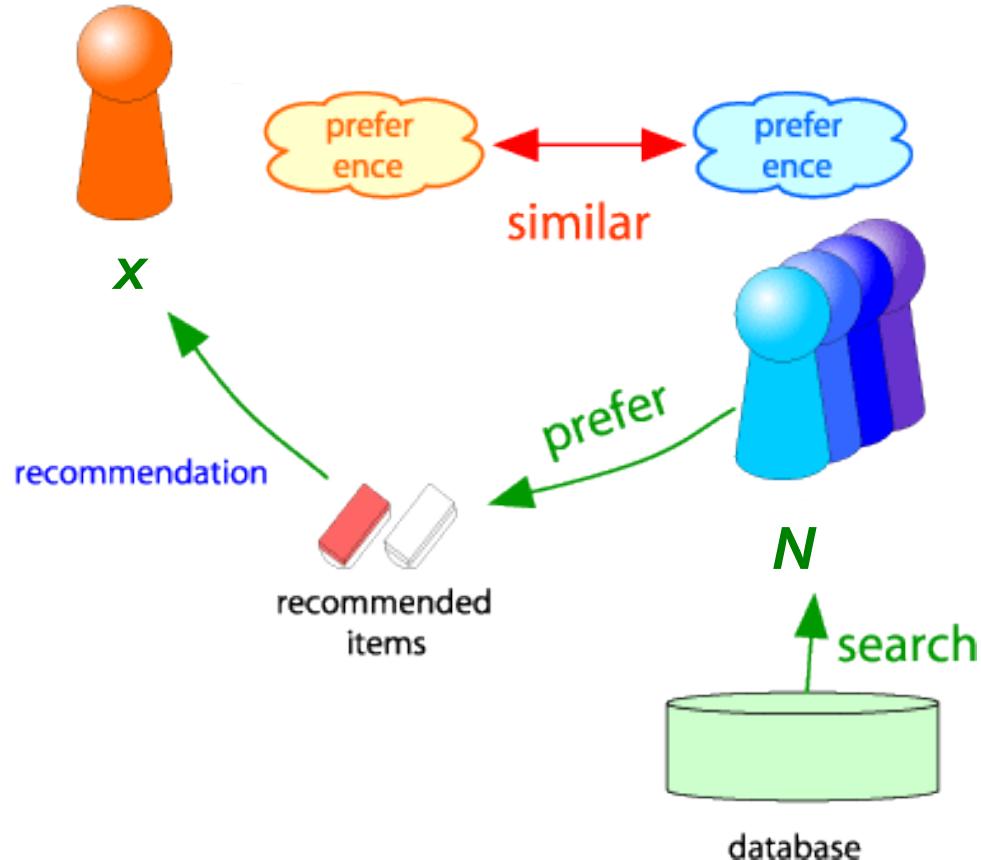
## Collaborative Filtering

Mining of Massive Datasets  
Leskovec, Rajaraman, and Ullman  
Stanford University



# Collaborative Filtering

- Consider user  $x$
- Find set  $N$  of other users whose ratings are “similar” to  $x$ 's ratings
- Estimate  $x$ 's ratings based on ratings of users in  $N$



# Similar Users (1)

	HP1	HP2	HP3	TW	SW1	SW2	SW3
A	4			5	1		
B	5	5	4				
C				2	4	5	
D		3					3

- Consider users  $x$  and  $y$  with rating vectors  $r_x$  and  $r_y$
- We need a similarity metric  $\text{sim}(x, y)$
- Capture intuition that  $\text{sim}(A, B) > \text{sim}(A, C)$

# Option 1: Jaccard Similarity

	HP1	HP2	HP3	TW	SW1	SW2	SW3
A	4			5	1		
B	5	5	4				
C				2	4	5	
D		3					3

- $\text{sim}(A,B) = |r_A \cap r_B| / |r_A \cup r_B|$
- $\text{sim}(A,B) = 1/5; \text{sim}(A,C) = 2/4$ 
  - $\text{sim}(A,B) < \text{sim}(A,C)$
- Problem: Ignores rating values!

# Option 2: Cosine similarity

	HP1	HP2	HP3	TW	SW1	SW2	SW3
A	4			5	1		
B	5	5	4				
C				2	4	5	
D		3					3

- $\text{sim}(A,B) = \cos(r_A, r_B)$
- $\text{sim}(A,B) = 0.38$ ,  $\text{sim}(A,C) = 0.32$ 
  - $\text{sim}(A,B) < \text{sim}(A,C)$ , but not by much
- Problem: treats missing ratings as negative

# Option 3: Centered cosine

- Normalize ratings by subtracting row mean

	HP1	HP2	HP3	TW	SW1	SW2	SW3
A	4			5	1		
B	5	5	4				
C				2	4	5	
D		3					3

	HP1	HP2	HP3	TW	SW1	SW2	SW3
A	2/3			5/3	-7/3		
B	1/3	1/3	-2/3				
C				-5/3	1/3	4/3	
D		0					0

# Centered Cosine similarity (2)

	HP1	HP2	HP3	TW	SW1	SW2	SW3
A	2/3			5/3	-7/3		
B	1/3	1/3	-2/3				
C				-5/3	1/3	4/3	
D		0					0

- $\text{sim}(A,B) = \cos(r_A, r_B) = 0.09$ ;  $\text{sim}(A,C) = -0.56$ 
  - $\text{sim}(A,B) > \text{sim}(A,C)$
- Captures intuition better
  - Missing ratings treated as “average”
  - Handles “tough raters” and “easy raters”
- Also known as Pearson Correlation

# Rating Predictions

- Let  $r_x$  be the vector of user  $x$ 's ratings
- Let  $N$  be the set of  $k$  users most similar to  $x$  who have also rated item  $i$
- Prediction for user  $x$  and item  $i$
- Option 1:  $r_{xi} = 1/k \sum_{y \in N} r_{yi}$
- Option 2:  $r_{xi} = \sum_{y \in N} s_{xy} r_{yi} / \sum_{y \in N} s_{xy}$ 

where  $s_{xy} = \text{sim}(x,y)$

# Item-Item Collaborative Filtering

- So far: User-user collaborative filtering
- Another view: Item-item
  - For item  $i$ , find other similar items
  - Estimate rating for item  $i$  based on ratings for similar items
  - Can use same similarity metrics and prediction functions as in user-user model

$$r_{xi} = \frac{\sum_{j \in N(i;x)} s_{ij} \cdot r_{xj}}{\sum_{j \in N(i;x)} s_{ij}}$$

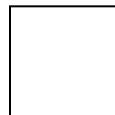
$s_{ij}$ ... similarity of items  $i$  and  $j$

$r_{xj}$ ...rating of user  $x$  on item  $j$

$N(i;x)$ ... set items rated by  $x$  similar to  $i$

# Item-Item CF ( $|N|=2$ )

	users											
	1	2	3	4	5	6	7	8	9	10	11	12
1	1		3			5			5		4	
2			5	4			4			2	1	3
3	2	4		1	2		3		4	3	5	
4		2	4		5			4			2	
5			4	3	4	2					2	5
6	1		3		3			2			4	

 - unknown rating     - rating between 1 to 5

# Item-Item CF ( $|N|=2$ )

	users											
	1	2	3	4	5	6	7	8	9	10	11	12
1	1		3		?	5			5		4	
2			5	4			4			2	1	3
3	2	4		1	2		3		4	3	5	
4		2	4		5			4			2	
5			4	3	4	2					2	5
6	1		3		3			2			4	

 - estimate rating of movie 1 by user 5

# Item-Item CF ( $|N|=2$ )

	users												
	1	2	3	4	5	6	7	8	9	10	11	12	$\text{sim}(1,m)$
movies	1	1		3		?	5			5		4	1.00
	2			5	4			4			2	1	3
	3	2	4		1	2		3		4	3	5	-0.18
	4		2	4		5			4			2	0.41
	5			4	3	4	2				2	5	-0.10
	6	1		3		3			2			4	-0.31
													0.59

## Neighbor selection:

Identify movies similar to movie 1, rated by user 5

Here we use Pearson correlation as similarity:

1) Subtract mean rating  $m_i$  from each movie  $i$

$$m_i = (1+3+5+5+4)/5 = 3.6$$

row 1: [-2.6, 0, -0.6, 0, 0, 1.4, 0, 0, 1.4, 0, 0.4, 0]

2) Compute cosine similarities between rows

# Item-Item CF ( $|N|=2$ )

	users												
	1	2	3	4	5	6	7	8	9	10	11	12	$\text{sim}(1,m)$
1	1		3		?	5			5		4		1.00
2			5	4			4			2	1	3	-0.18
3	2	4		1	2		3		4	3	5		0.41
4		2	4		5			4			2		-0.10
5			4	3	4	2					2	5	-0.31
6	1		3		3			2			4		0.59

Compute similarity weights:

$$s_{13}=0.41, s_{16}=0.59$$

# Item-Item CF ( $|N|=2$ )

	users											
	1	2	3	4	5	6	7	8	9	10	11	12
1	1		3		2.6	5			5		4	
2			5	4			4			2	1	3
3	2	4		1	2		3		4	3	5	
4		2	4		5			4			2	
5			4	3	4	2					2	5
6	1		3		3			2			4	

Predict by taking weighted average:

$$r_{15} = (0.41*2 + 0.59*3) / (0.41+0.59) = 2.6$$

# Item-Item v. User-User

- In theory, user-user and item-item are dual approaches
- In practice, item-item outperforms user-user in many use cases
- Items are “simpler” than users
  - Items belong to a small set of “genres”, users have varied tastes
  - Item Similarity is more meaningful than User Similarity

# Recommender Systems

## Content-based systems

Mining of Massive Datasets  
Leskovec, Rajaraman, and Ullman  
Stanford University



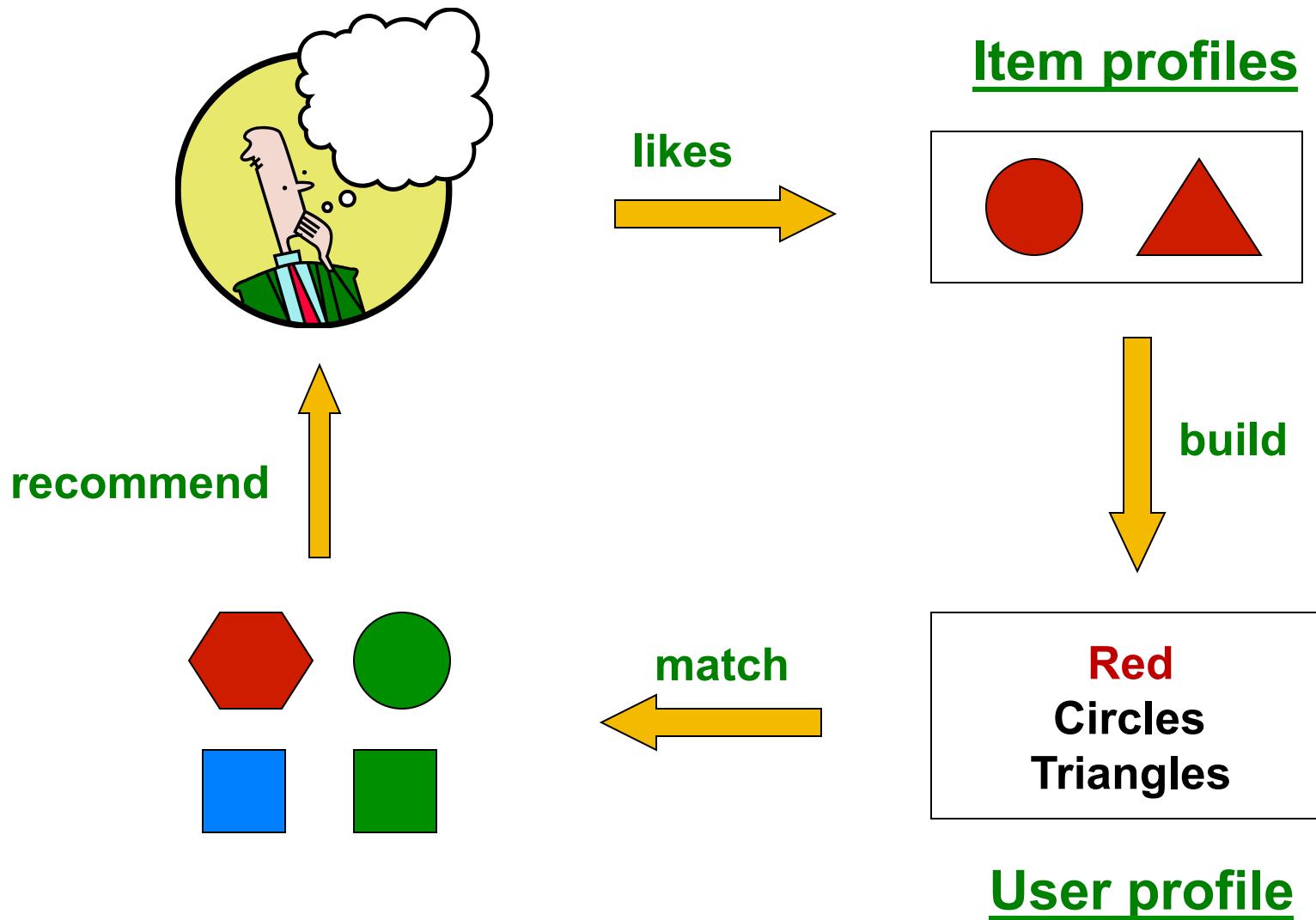
# Content-based Recommendations

**Main idea:** Recommend items to customer  $x$  similar to previous items rated highly by  $x$

## Examples:

- **Movies**
  - Same actor(s), director, genre, ...
- **Websites, blogs, news**
  - Articles with “similar” content
- **People**
  - Recommend people with many common friends

# Plan of Action



# Item Profiles

- For each item, create an **item profile**
- Profile is a set of features
  - **Movies:** author, title, actor, director,...
  - **Images, videos:** metadata and tags
  - **People:** Set of friends
- Convenient to think of the item profile as a vector
  - One entry per feature (e.g., each actor, director,...)
  - Vector might be boolean or real-valued

# Text features

- Profile = set of “important” words in item (document)
- How to pick important words?
  - Usual heuristic from text mining is **TF-IDF** (Term frequency \* Inverse Doc Frequency)

# Sidenote: TF-IDF

$f_{ij}$  = frequency of term (feature)  $i$  in doc (item)  $j$

$$TF_{ij} = \frac{f_{ij}}{\max_k f_{kj}}$$

**Note:** we normalize TF to discount for “longer” documents

$n_i$  = number of docs that mention term  $i$

$N$  = total number of docs

$$IDF_i = \log \frac{N}{n_i}$$

**TF-IDF score:**  $w_{ij} = TF_{ij} \times IDF_i$

**Doc profile** = set of words with highest TF-IDF scores, together with their scores

# User Profiles

- User has rated items with profiles  $i_1, \dots, i_n$
- Simple: (weighted) average of rated item profiles
- Variant: Normalize weights using average rating of user
- More sophisticated aggregations possible

# Example 1: Boolean Utility Matrix

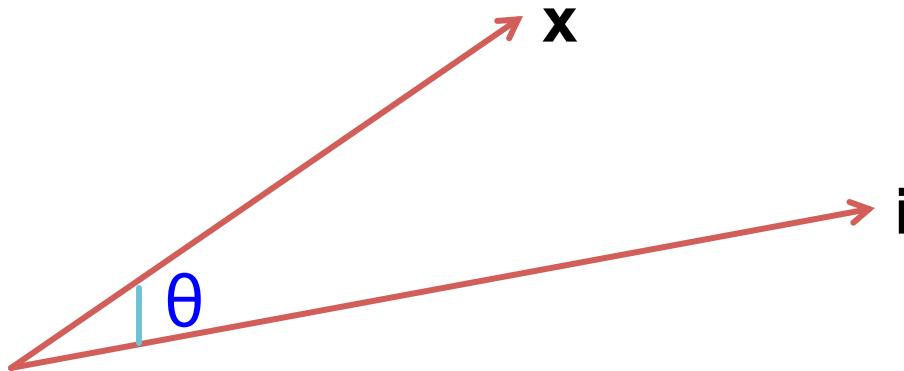
- Items are movies, only feature is “Actor”
  - Item profile: vector with 0 or 1 for each Actor
- Suppose user x has watched 5 movies
  - 2 movies featuring actor A
  - 3 movies featuring actor B
- User profile = mean of item profiles
  - Feature A’s weight =  $2/5 = 0.4$
  - Feature B’s weight =  $3/5 = 0.6$

# Example 2: Star Ratings

- Same example, 1-5 star ratings
  - Actor A's movies rated 3 and 5
  - Actor B's movies rated 1, 2 and 4
- Useful step: Normalize ratings by subtracting user's mean rating (3)
  - Actor A's normalized ratings = 0, +2
    - Profile weight =  $(0 + 2)/2 = 1$
  - Actor B's normalized ratings = -2, -1, +1
    - Profile weight =  $-2/3$

# Making predictions

- User profile  $x$ , Item profile  $i$
- Estimate  $U(x,i) = \cos(\theta) = (x \cdot i) / (|x| |i|)$



Technically, the cosine distance is actually the angle  $\theta$   
And the cosine similarity is the angle  $180-\theta$

For convenience, we use  $\cos(\theta)$  as our similarity measure  
and call it the “cosine similarity” in this context.

# Pros: Content-based Approach

- No need for data on other users
- Able to recommend to users with unique tastes
- Able to recommend new & unpopular items
  - No first-rater problem
- Explanations for recommended items
  - Content features that caused an item to be recommended

# Cons: Content-based Approach

- Finding the appropriate features is hard
  - E.g., images, movies, music
- Overspecialization
  - Never recommends items outside user's content profile
  - People might have multiple interests
  - Unable to exploit quality judgments of other users
- Cold-start problem for new users
  - How to build a user profile?

# Recommender Systems

Overview

Content-based systems

Collaborative Filtering

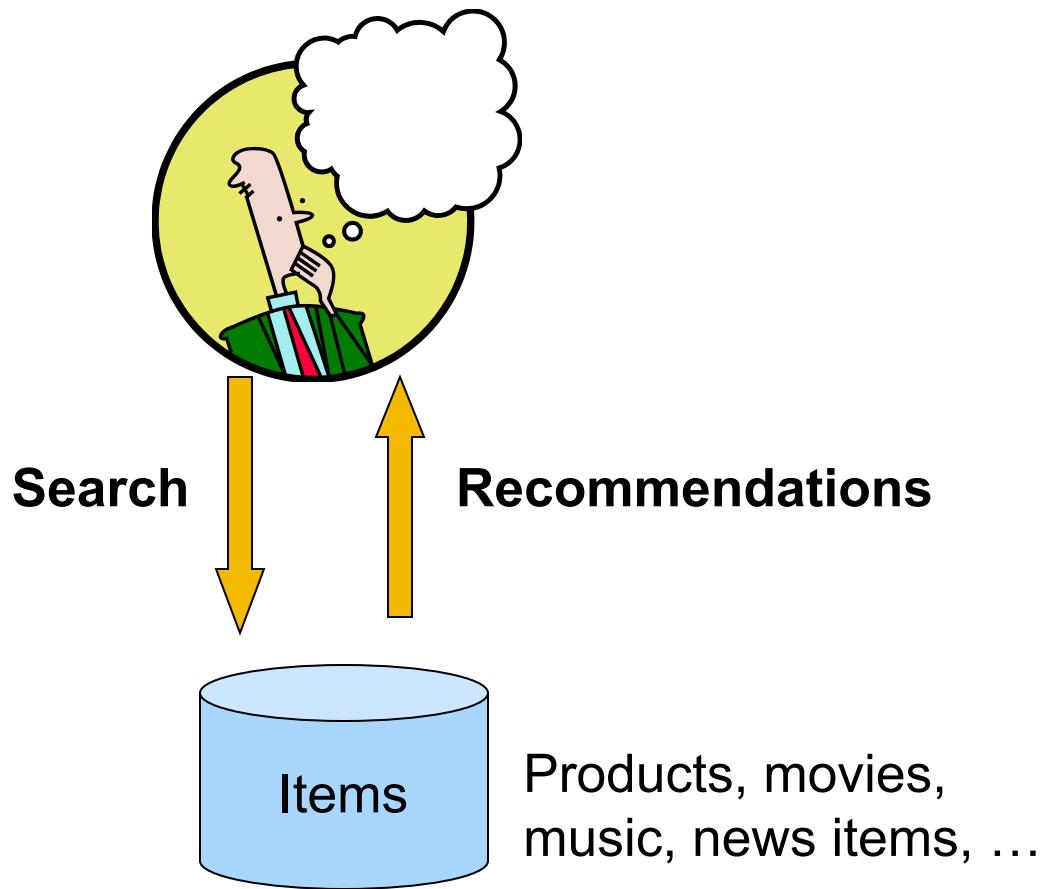
Evaluating recommender systems

Mining of Massive Datasets

Leskovec, Rajaraman, and Ullman  
Stanford University



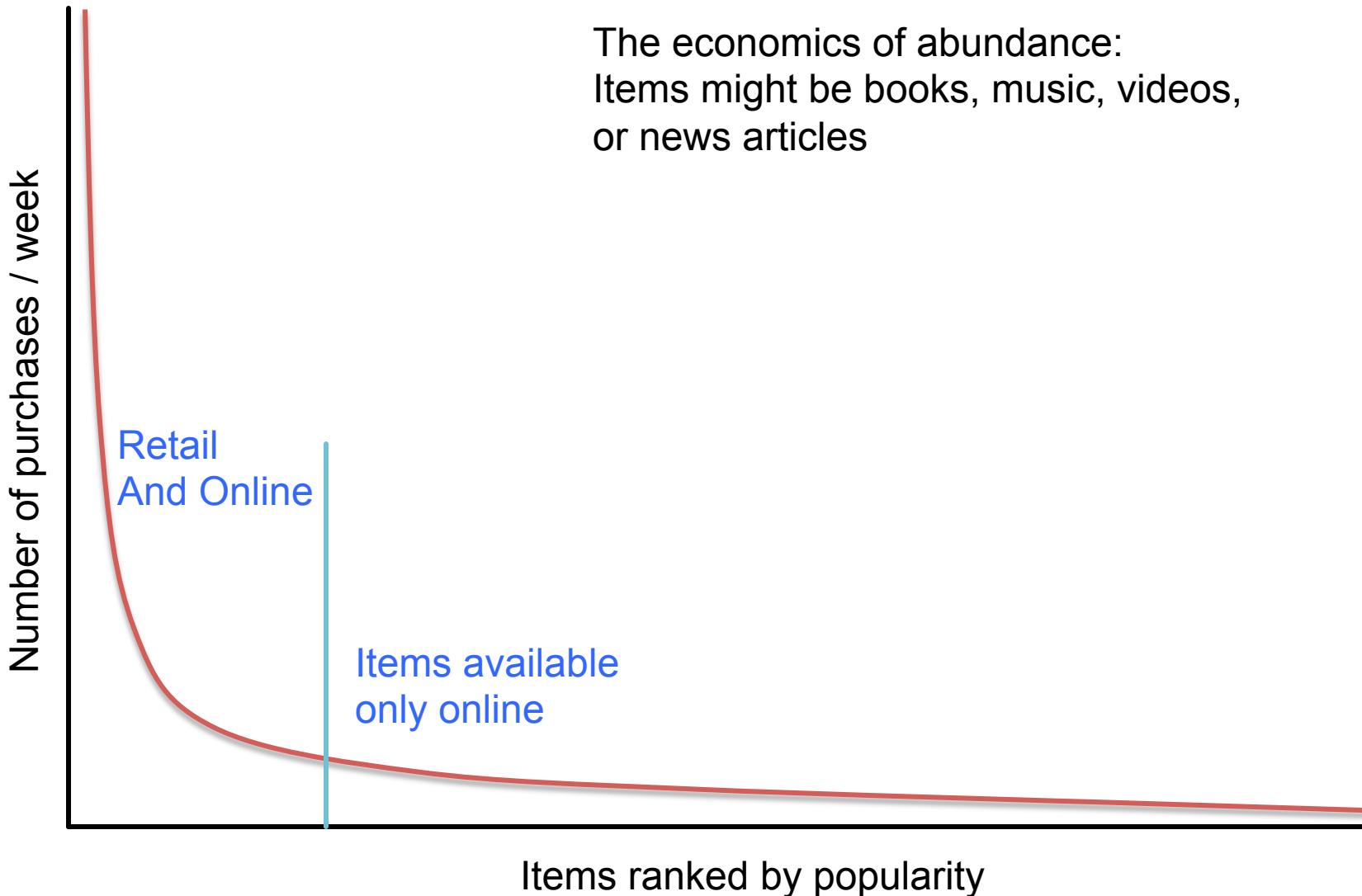
# Recommendations



# From Scarcity to Abundance

- Shelf space is a scarce commodity for traditional retailers
  - Also: TV networks, movie theaters,...
- The web enables near-zero-cost dissemination of information about products
  - From scarcity to abundance
  - Gives rise to the “Long Tail” phenomenon

# The Long Tail (1)



# The Long Tail (2)

- More choice necessitates better filters
  - Recommendation engines
  - How **Into Thin Air** made **Touching the Void** a bestseller (<http://www.wired.com/wired/archive/12.10/tail.html>)
- Examples
  - Books, movies, music, news articles
  - People (friend recommendations on Facebook, LinkedIn, and Twitter)

# Types of Recommendations

- Editorial and hand curated
  - List of favorites
  - Lists of “essential” items
- Simple aggregates
  - Top 10, Most Popular, Recent Uploads
- Tailored to individual users
  - Amazon, Netflix, Pandora ...
  - Our focus here

# Formal Model

- $C$  = set of **Customers**
- $S$  = set of **Items**
- **Utility function**  $u: C \times S \rightarrow R$ 
  - $R$  = set of ratings
  - $R$  is a totally ordered set
  - e.g., 0-5 stars, real number in  $[0,1]$

# Utility Matrix

	Avatar	LOTR	Matrix	Pirates
Alice	1		0.2	
Bob		0.5		0.3
Carol	0.2		1	
David				0.4

# Key Problems

## (1) Gathering “known” ratings for matrix

- How to collect the data in the utility matrix

## (2) Extrapolate unknown ratings from the known ones

- Mainly interested in high unknown ratings
  - We are not interested in knowing what you don't like but what you like

## (3) Evaluating extrapolation methods

- How to measure success/performance of recommendation methods

# (1) Gathering Ratings

- **Explicit**
  - Ask people to rate items
  - Doesn't scale: only a small fraction of users leave ratings and reviews
- **Implicit**
  - Learn ratings from user actions
    - E.g., purchase implies high rating
  - What about low ratings?

# (2) Extrapolating Utilities

- Key problem: matrix  $U$  is sparse
  - Most people have not rated most items
  - Cold start:
    - New items have no ratings
    - New users have no history
- Three approaches to recommender systems
  - 1) Content-based
  - 2) Collaborative
  - 3) Latent factor based

# Sentiment Analysis and Opinion Mining

*New book:*

Bing Liu. Sentiment Analysis and Opinion Mining  
Morgan & Claypool Publishers, May 2012.

Bing Liu  
Department of Computer Science  
University Of Illinois at Chicago  
liub@cs.uic.edu

# Introduction

- Opinion mining or sentiment analysis
  - Computational study of opinions, sentiments, subjectivity, evaluations, attitudes, appraisal, affects, views, emotions, etc., expressed in text.
    - Reviews, blogs, discussions, news, comments, feedback, or any other documents
- Terminology:
  - Sentiment analysis is more widely used in industry.
  - Both are widely used in academia
- But they can be used interchangeably.

# Why are opinions important?

- “Opinions” are key influencers of our behaviors.
- Our beliefs and perceptions of reality are conditioned on how others see the world.
- Whenever we need to make a decision, we often seek out the opinions of others. In the past,
  - Individuals: seek opinions from friends and family
  - Organizations: use surveys, focus groups, opinion polls, consultants.

# Introduction – social media + beyond

- **Word-of-mouth on the Web**
  - Personal experiences and opinions about anything in reviews, forums, blogs, Twitter, micro-blogs, etc
  - Comments about articles, issues, topics, reviews, etc.
  - Postings at social networking sites, e.g., facebook.
- **Global scale:** No longer – one's circle of friends
- **Organization internal data**
  - Customer feedback from emails, call centers, etc.
- **News and reports**
  - Opinions in news articles and commentaries

# Introduction – applications

- **Businesses and organizations**
  - Benchmark products and services; market intelligence.
    - Businesses spend a huge amount of money to find consumer opinions using consultants, surveys and focus groups, etc
- **Individuals**
  - Make decisions to buy products or to use services
  - Find public opinions about political candidates and issues
- **Ads placements:** Place ads in the social media content
  - Place an ad if one praises a product.
  - Place an ad from a competitor if one criticizes a product.
- **Opinion retrieval:** provide general search for opinions.

# A fascinating problem!

- Intellectually challenging & many applications.
  - A popular research topic in NLP, text mining, and Web mining in recent years (Shanahan, Qu, and Wiebe, 2006 (edited book); Surveys - Pang and Lee 2008; Liu, 2006 and 2011; 2010)
  - It has spread from computer science to management science (Hu, Pavlou, Zhang, 2006; Archak, Ghose, Ipeirotis, 2007; Liu Y, et al 2007; Park, Lee, Han, 2007; Dellarocas, Zhang, Awad, 2007; Chen & Xie 2007).
  - 40-60 companies in USA alone
- It touches every aspect of NLP and yet is confined.
  - Little research in NLP/Linguistics in the past.
- Potentially a major technology from NLP.
  - But it is hard.

# A large research area

- Many names and tasks with somewhat different objectives and models
  - Sentiment analysis
  - Opinion mining
  - Sentiment mining
  - Subjectivity analysis
  - Affect analysis
  - Emotion detection
  - Opinion spam detection
  - *Etc.*

# About this tutorial

- Like a traditional tutorial, I will introduce the research in the field.
  - Key topics, main ideas and approaches
  - Since there are a large number of papers, it is not possible to introduce them all, but a comprehensive reference list will be provided.
- Unlike many traditional tutorials, this tutorial is also based on my experience in working with clients in a startup, and in my consulting
  - I focus more on practically important tasks (IMHO)

# Roadmap

- 
- **Opinion Mining Problem**
    - Document sentiment classification
    - Sentence subjectivity & sentiment classification
    - Aspect-based sentiment analysis
    - Aspect-based opinion summarization
    - Opinion lexicon generation
    - Mining comparative opinions
    - Some other problems
    - Opinion spam detection
    - Utility or helpfulness of reviews
    - Summary

# Structure the unstructured (Hu and Liu 2004)

- **Structure the unstructured:** Natural language text is often regarded as **unstructured data**.
- The problem definition should provide a structure to the unstructured problem.
  - **Key tasks:** Identify key tasks and their inter-relationships.
  - **Common framework:** Provide a common framework to unify different research directions.
  - **Understanding:** help us understand the problem better.

# Problem statement

- It consists of two aspects of abstraction

- (1) Opinion definition. What is an opinion?

- Can we provide a structured definition?
    - If we cannot structure a problem, we probably do not understand the problem.

- (2) Opinion summarization. why?

- Opinions are subjective. An opinion from a single person (unless a VIP) is often not sufficient for action.
  - We need opinions from many people, and thus opinion summarization.

# Abstraction (1): what is an opinion?

- Id: Abc123 on 5-1-2008 “*I bought an iPhone a few days ago. It is such a nice phone. The touch screen is really cool. The voice quality is clear too. It is much better than my old Blackberry, which was a terrible phone and so difficult to type with its tiny keys. However, my mother was mad with me as I did not tell her before I bought the phone. She also thought the phone was too expensive, ...*”
- One can look at this review/blog at the
  - document level, i.e., is this review + or -?
  - sentence level, i.e., is each sentence + or -?
  - entity and feature/aspect level

# Entity and aspect/feature level

- **Id:** Abc123 on 5-1-2008 “*I bought an iPhone a few days ago. It is such a nice phone. The touch screen is really cool. The voice quality is clear too. It is much better than my old Blackberry, which was a terrible phone and so difficult to type with its tiny keys. However, my mother was mad with me as I did not tell her before I bought the phone. She also thought the phone was too expensive, ...*”
- **What do we see?**
  - **Opinion targets:** entities and their features/aspects
  - **Sentiments:** positive and negative
  - **Opinion holders:** persons who hold the opinions
  - **Time:** when opinions are expressed

# Two main types of opinions

(Jindal and Liu 2006; Liu, 2010)

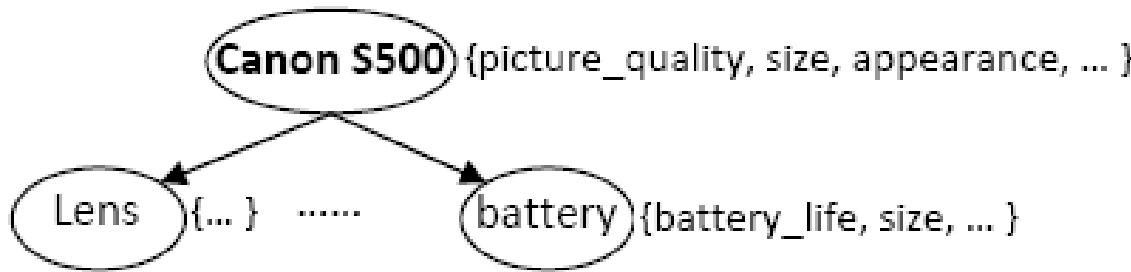
- **Regular opinions:** Sentiment/opinion expressions on some target entities
  - Direct opinions:
    - “The touch screen is really cool.”
  - Indirect opinions:
    - “After taking the drug, my pain has gone.”
- **Comparative opinions:** Comparisons of more than one entity.
  - E.g., “iPhone is better than Blackberry.”
- We focus on regular opinions first, and just call them opinions.

# A (regular) opinion

- Opinion (a restricted definition)
  - An opinion (or regular opinion) is simply a **positive or negative** sentiment, view, attitude, emotion, or appraisal about **an entity** or **an aspect of the entity** (Hu and Liu 2004; Liu 2006) from **an opinion holder** (Bethard et al 2004; Kim and Hovy 2004; Wiebe et al 2005).
- Sentiment orientation of an opinion
  - Positive, negative, or neutral (no opinion)
    - Also called *opinion orientation*, *semantic orientation*, *sentiment polarity*.

# Entity and aspect (Hu and Liu, 2004; Liu, 2006)

- **Definition (entity):** An *entity e* is a product, person, event, organization, or topic.  $e$  is represented as
  - a hierarchy of components, sub-components, and so on.
  - Each node represents a component and is associated with a set of attributes of the component.



- An opinion can be expressed on any node or attribute of the node.
- For simplicity, we use the term **aspects (features)** to represent both components and attributes.

# Opinion definition (Liu, Ch. in NLP handbook, 2010)

## ■ An *opinion* is a quintuple

$$(e_j, a_{jk}, so_{ijkl}, h_i, t_i),$$

where

- $e_j$  is a target entity.
- $a_{jk}$  is an aspect/feature of the entity  $e_j$ .
- $so_{ijkl}$  is the sentiment value of the opinion from the opinion holder  $h_i$  on feature  $a_{jk}$  of entity  $e_j$  at time  $t_i$ .  
 $so_{ijkl}$  is +ve, -ve, or neu, or more granular ratings.
- $h_i$  is an opinion holder.
- $t_i$  is the time when the opinion is expressed.

# Some remarks about the definition

- Although introduced using a product review, the definition is generic
  - Applicable to other domains,
  - E.g., politics, social events, services, topics, etc.
- $(e_j, a_{jk})$  is also called the opinion target
  - Opinion without knowing the target is of limited use.
- The five components in  $(e_j, a_{jk}, so_{ijkl}, h_i, t_l)$  must correspond to one another. Very hard to achieve
- The five components are essential. Without any of them, it can be problematic in general.

# Some remarks (contd)

- Of course, one can add any number of other components to the tuple for more analysis. E.g.,
  - Gender, age, Web site, post-id, etc.
- The original definition of an entity is a hierarchy of parts, sub-parts, and so on.
  - The simplification can result in information loss.
    - E.g., “The **seat** of this car is really **ugly**.”
    - “**seat**” is a part of the car and “**appearance**” (implied by ugly) is an aspect of “**seat**” (not the car).
  - But it is usually sufficient for practical applications.
    - It is too hard without the simplification.

# “Confusing” terminologies

- **Entity** is also called **object**.
- **Aspect** is also called **feature**, **attribute**, **facet**, etc
- **Opinion holder** is also called **opinion source**
- Some researchers also use **topic** to mean **entity** and/or **aspect**.
  - Separating entity and aspect is preferable
- In specific applications, some specialized terms are also commonly used, e.g.,
  - Product features, political issues

# Reader's standing point

- See this sentence
  - “I am so happy that Google price shot up today.”
- Although the sentence gives an explicit sentiment, different readers may feel very differently.
  - If a reader sold his Google shares yesterday, he will not be that happy.
  - If a reader bought a lot of Google shares yesterday, he will be very happy.
- Current research either implicitly assumes a standing point, or ignores the issue.

# Our example blog in quintuples

- **Id:** Abc123 **on** 5-1-2008 “*I bought an iPhone a few days ago. It is such a nice phone. The touch screen is really cool. The voice quality is clear too. It is much better than my old Blackberry, which was a terrible phone and so difficult to type with its tiny keys. However, my mother was mad with me as I did not tell her before I bought the phone. She also thought the phone was too expensive, ...”*
- **In quintuples**

(iPhone, GENERAL, +, Abc123, 5-1-2008)

(iPhone, touch\_screen, +, Abc123, 5-1-2008)

....

- We will discuss comparative opinions later.

# Structure the unstructured

- **Goal:** Given an opinionated document,
  - Discover all quintuples  $(e_j, f_{jk}, so_{ijkl}, h_i, t_l)$ ,
  - Or, solve some simpler forms of the problem
    - E.g., sentiment classification at the document or sentence level.
- **With the quintuples,**
  - **Unstructured Text → Structured Data**
    - Traditional data and visualization tools can be used to slice, dice and visualize the results.
    - Enable qualitative and quantitative analysis.

# Two closely related concepts

- **Subjectivity** and **emotion**.
- **Sentence subjectivity**: An *objective sentence* presents some factual information, while a *subjective sentence* expresses some personal feelings, views, emotions, or beliefs.
- **Emotion**: Emotions are people's subjective feelings and thoughts.

# Subjectivity

- Subjective expressions come in many forms, e.g., opinions, allegations, desires, beliefs, suspicions, speculations (Wiebe 2000; Wiebe et al 2004; Riloff et al 2006).
  - A subjective sentence may contain a positive or negative opinion
- Most opinionated sentences are subjective, but objective sentences can imply opinions too (Liu, 2010)
  - “The machine stopped working in the second day”
  - “We brought the mattress yesterday, and a body impression has formed.”
  - “After taking the drug, there is no more pain”

# Emotion

- No agreed set of basic emotions of people among researchers.
- Based on (Parrott, 2001), people have six main emotions,
  - love, joy, surprise, anger, sadness, and fear.
- Strengths of opinions/sentiments are related to certain emotions, e.g., joy, anger.
  - However, the concepts of emotions and opinions are not equivalent.

# Rational and emotional evaluations

- **Rational evaluation:** Many evaluation/opinion sentences express no emotion
  - e.g., “The voice of this phone is clear”
- **Emotional evaluation**
  - e.g., “I love this phone”
  - “The voice of this phone is crystal clear” (?)
- Some emotion sentences express no (positive or negative) opinion/sentiment
  - e.g., “I am so surprised to see you”.

# Sentiment, subjectivity, and emotion

- Although they are clearly related, these concepts are not the same
  - Sentiment  $\neq$  subjective  $\neq$  emotion
- Sentiment is not a subset of subjectivity (without implied sentiments by facts, it should be)
  - sentiment  $\not\subset$  subjectivity
- The following should hold
  - emotion  $\subset$  subjectivity
  - sentiment  $\not\subset$  emotion, ...

# Abstraction (2): opinion summary

- **With a lot of opinions, a summary is necessary.**
  - A multi-document summarization task
- For factual texts, summarization is to select the most important facts and present them in a sensible order while avoiding repetition
  - 1 fact = any number of the same fact
- But for opinion documents, it is different because opinions have a quantitative side & have targets
  - 1 opinion ≠ a number of opinions
  - **Aspect-based summary** is more suitable
    - Quintuples form the basis for opinion summarization

# Aspect-based opinion summary<sup>1</sup>

(Hu & Liu, 2004)

“I bought an *iPhone* a few days ago. It is such a nice *phone*. The *touch screen* is really cool. The *voice quality* is clear too. It is much better than my old *Blackberry*, which was a terrible *phone* and so *difficult to type* with its *tiny keys*. However, my *mother* was mad with me as I did not tell her before I bought the *phone*. She also thought the *phone* was too *expensive*, ...”

1. Originally called *feature-based opinion mining and summarization*

## Feature Based Summary of iPhone:

### Feature1: Touch screen

Positive: 212

- The *touch screen* was really cool.
- The *touch screen* was so easy to use and can do amazing things.

...

Negative: 6

- The *screen* is easily scratched.
- I have a lot of difficulty in removing finger marks from the *touch screen*.

...

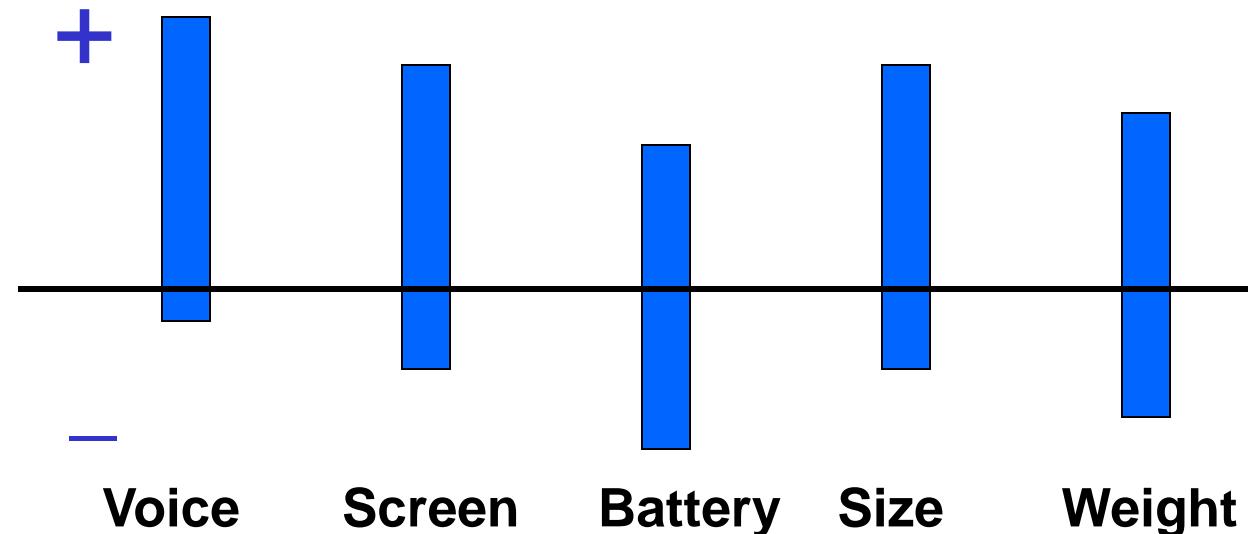
### Feature2: voice quality

...

Note: We omit opinion holders

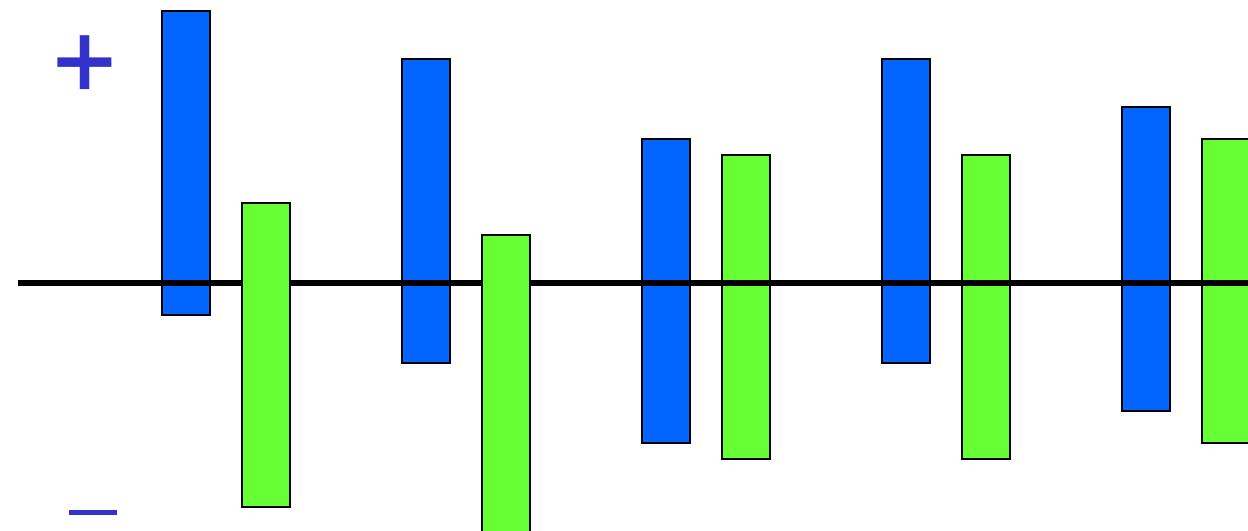
# Opinion Observer (Liu et al. 2005)

- Summary of reviews of
- Cell Phone 1



- Comparison of reviews of

- Cell Phone 1
- Cell Phone 2



# Aspect-based opinion summary

Bing™

HP printer

ALL RESULTS

Shopping

POPULAR FEATURES

- all
- Affordability
- Speed
- Print Quality
- Reliability
- Ease Of Use
- Brand
- Installation
- Size
- Compatibility

SHOPPING

HP LaserJet 1020 - printer - B/W - laser, 15ppm, USB



from \$179 (2 stores)  Bing cashback · 3%

★★★★☆ user reviews (177)

The HP LaserJet 1020 Printer, an excellent laser printer for the cost-co...  
high-quality LaserJet printing in a compact size, and at a price you can...

[user reviews](#) [product details](#) [expert reviews](#) [compare prices](#)

**user reviews**

view: **positive comments (44)**

speed  96%

The quality is as good as any laserjet printer I've used and the speed is fast.  
Love Reading [www.amazon.com](http://www.amazon.com) 3/17/2006 [more...](#)

Quick and fast transaction.  
Arthur L. Taylor [www.amazon.com](http://www.amazon.com) 2/5/2008 [more...](#)

It's small and fast and very reliable.  
Muffinhead's mom [www.amazon.com](http://www.amazon.com) 1/9/2007 [more...](#)

# Google Product Search (Blair-Goldensohn et al 2008 ?)

**Google products**

## Sony Cyber-shot DSC-W370 14.1 MP Digital Camera (Silver)

[Overview](#) - [Online stores](#) - [Nearby stores](#) - [Reviews](#) - [Technical specifications](#) - [Similar items](#) - [Accessories](#)

 \$140 [online](#), \$170 [nearby](#)  
 159 reviews  0

### Reviews

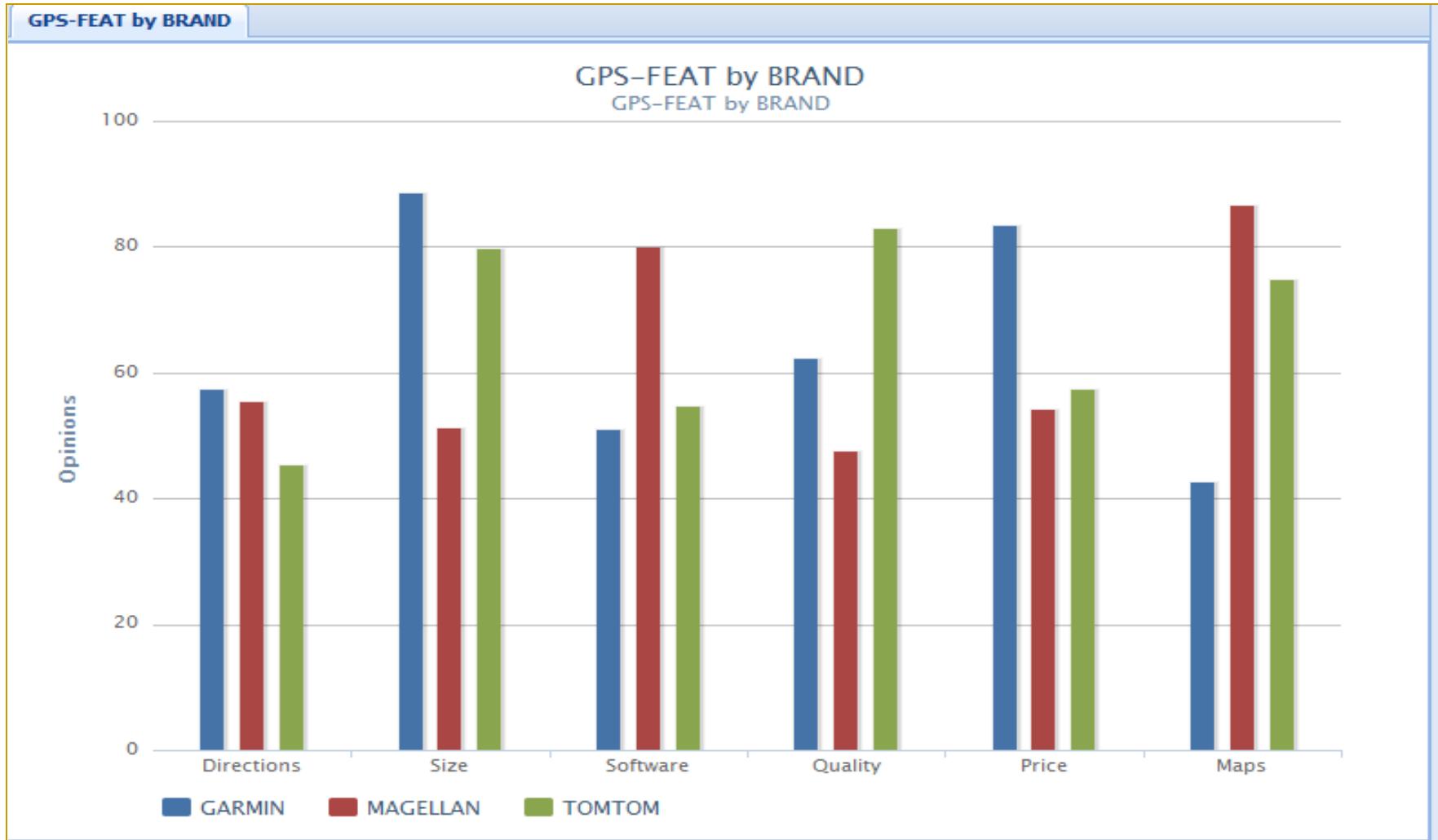
**Summary** - Based on 159 reviews

[1](#) [2](#) [3 stars](#) [4 stars](#) [5 stars](#)

#### What people are saying

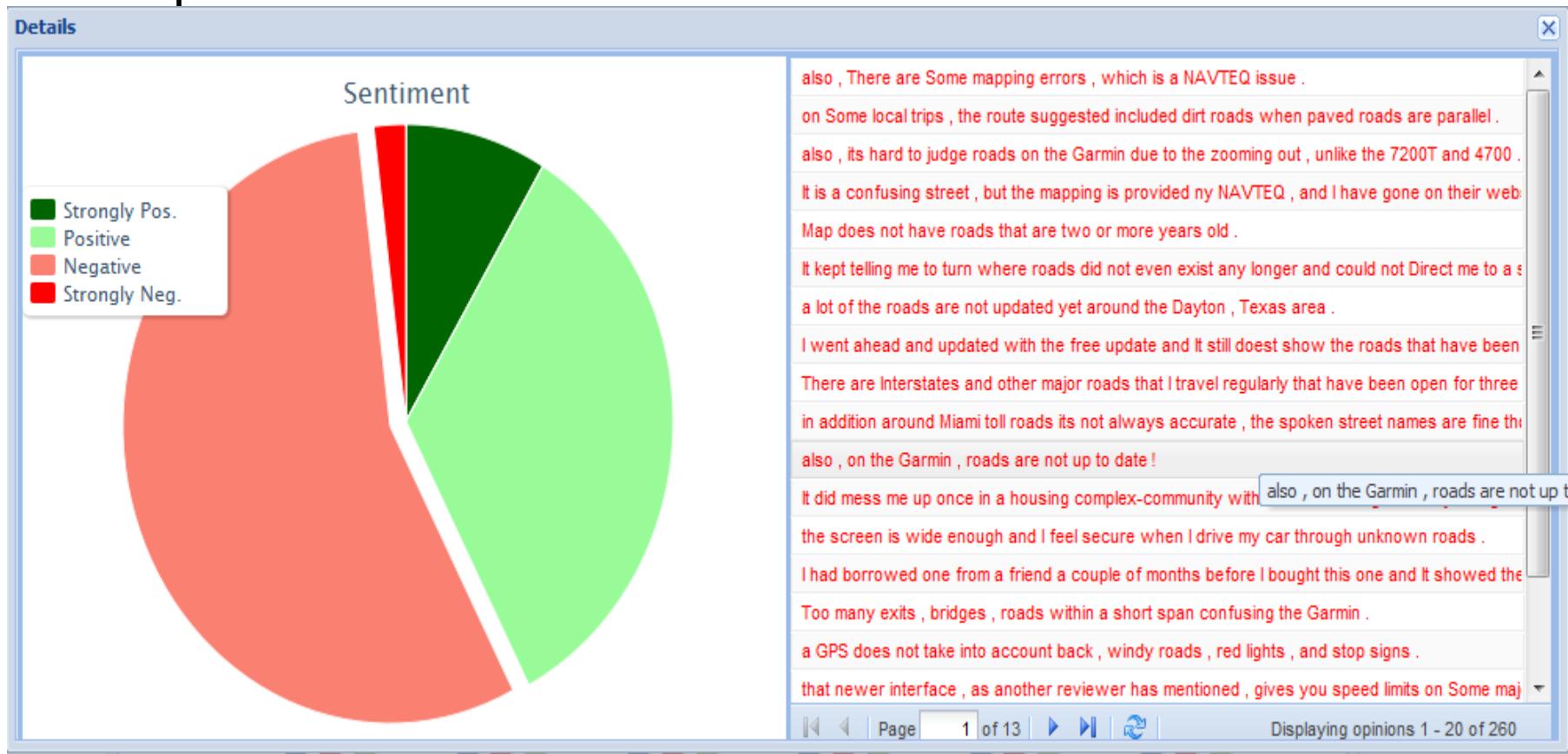
<a href="#">pictures</a>	 	"We use the product to take quickly photos."
<a href="#">features</a>	 	"Impressive panoramic feature."
<a href="#">zoom/lens</a>	 	"It also record better and focus better on sunny days."
<a href="#">design</a>	 	"It has the slightest grip but it's sufficient."
<a href="#">video</a>	 	"Video zoom is choppy."
<a href="#">battery life</a>	 	"Even better, the battery lasts long."
<a href="#">screen</a>	 	"I Love the Sony's 3" screen which I really wanted."

# Some examples from OpinionEQ

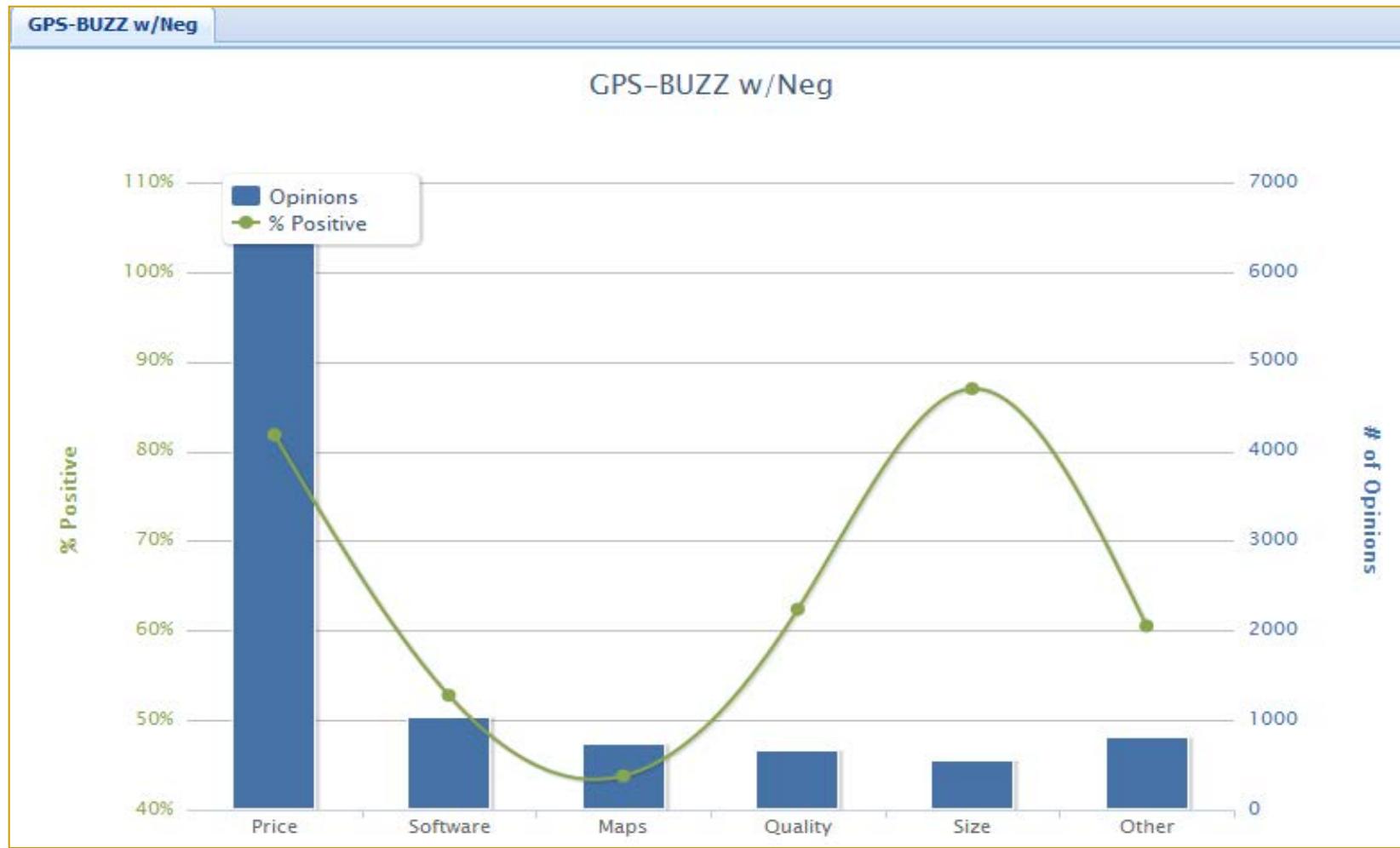


# Detail opinion sentences

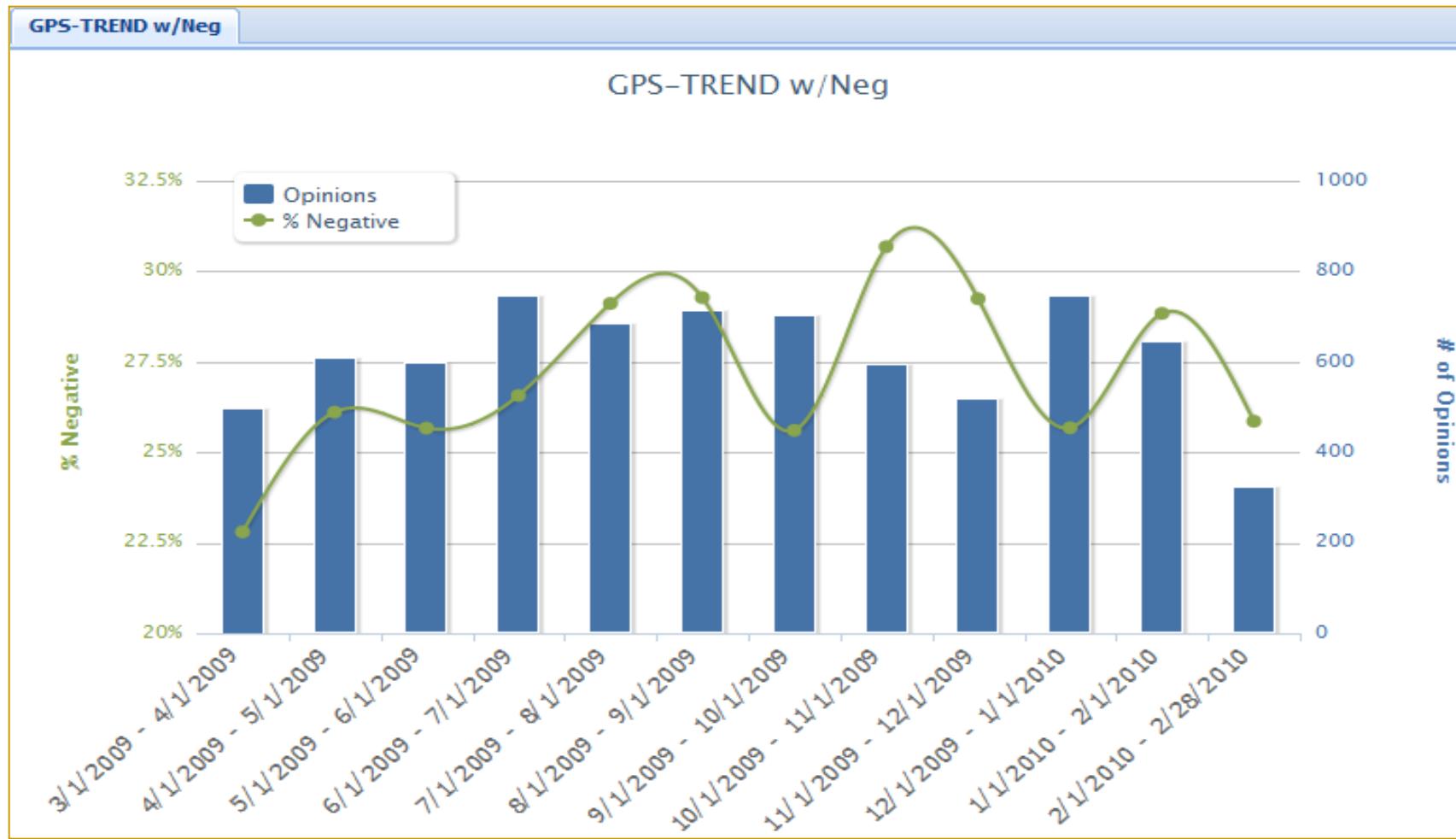
- Click on any bar (previous slide) to see the opinion sentences. Here are negative opinion sentences on the maps feature of Garmin.



# % of +ve opinion and # of opinions

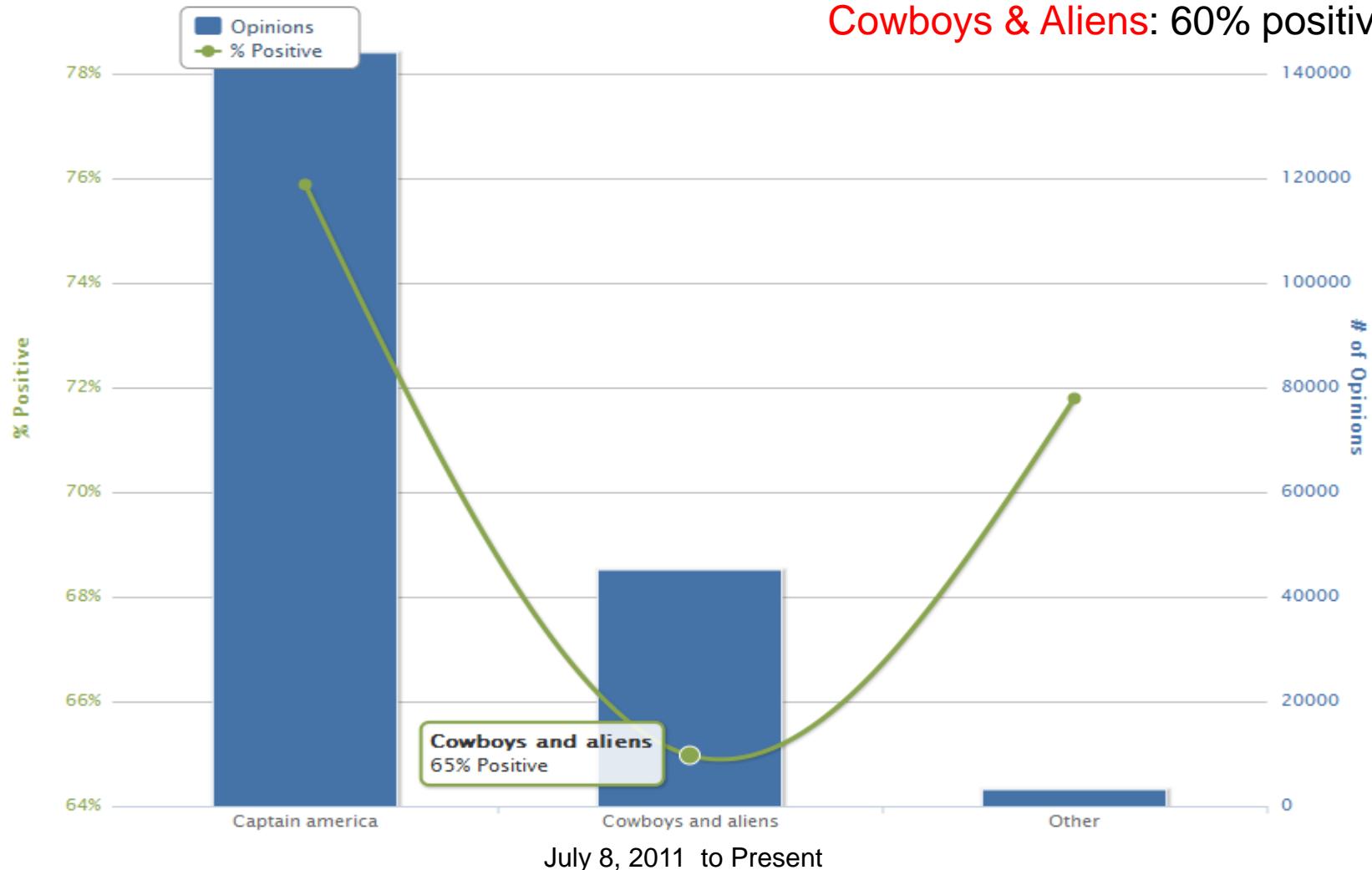


# Aggregate opinion trend



# Live tracking of two movies (Twitter)

User ratings from Rotten Tomatoes: Captain America: 81% positive  
Cowboys & Aliens: 60% positive



# Not just ONE problem

- $(e_j, a_{jk}, so_{ijkl}, h_i, t_l)$ ,
  - $e_j$  - a target entity: Named Entity Extraction (more)
  - $a_{jk}$  – an aspect of  $e_j$ : Information Extraction
  - $so_{ijkl}$  is sentiment: Sentiment Identification
  - $h_i$  is an opinion holder: Information/Data Extraction
  - $t_l$  is the time: Information/Data Extraction
  - 5 pieces of information must match
- Coreference resolution
- Synonym match (voice = sound quality)
- ...

# Opinion mining is hard!

- “*This past Saturday, I bought a Nokia phone and my girlfriend bought a Motorola phone with Bluetooth. We called each other when we got home. The voice on my phone was not so clear, worse than my previous Samsung phone. The battery life was short too. My girlfriend was quite happy with her phone. I wanted a phone with good sound quality. So my purchase was a real disappointment. I returned the phone yesterday.*”

# Easier and harder problems

- Tweets from Twitter are the easiest
  - short and thus usually straight to the point
- Reviews are next
  - entities are given (almost) and there is little noise
- Discussions, comments, and blogs are hard.
  - Multiple entities, comparisons, noisy, sarcasm, etc
- Determining sentiments seems to be easier.
- Extracting entities and aspects is harder.
- Combining them is even harder.

# Opinion mining in the real world

- Source the data, e.g., reviews, blogs, etc
  - (1) Crawl all data, store and search them, or
  - (2) Crawl only the target data
- Extract the right entities & aspects
  - Group entity and aspect expressions,
    - Moto = Motorola, photo = picture, etc ...
- Aspect-based opinion mining (sentiment analysis)
  - Discover all quintuples
    - (Store the quintuples in a database)
- Aspect based opinion summary

# Roadmap

- Opinion Mining Problem
-  ■ **Document sentiment classification**
- Sentence subjectivity & sentiment classification
- Aspect-based sentiment analysis
- Aspect-based opinion summarization
- Opinion lexicon generation
- Mining comparative opinions
- Some other problems
- Opinion spam detection
- Utility or helpfulness of reviews
- Summary

# Sentiment classification

- Classify a whole opinion document (e.g., a review) based on the overall sentiment of the opinion holder (Pang et al 2002; Turney 2002)
  - Classes: Positive, negative (possibly neutral)
  - Neutral or no opinion is hard. Most papers ignore it.
- An example review:
  - *“I bought an iPhone a few days ago. It is such a nice phone, although a little large. The touch screen is cool. The voice quality is clear too. I simply love it!”*
  - Classification: positive or negative?
- Perhaps the most widely studied problem.

# A text classification task

- It is basically a text classification problem
- But different from topic-based text classification.
  - In topic-based text classification (e.g., computer, sport, science), topic words are important.
  - But in sentiment classification, opinion/sentiment words are more important, e.g., great, excellent, horrible, bad, worst, etc.
- Opinion/sentiment words
  - Words and phrases that express desired or undesired states or qualities.

# Assumption and goal

- **Assumption:** The doc is written by a single person and express opinion/sentiment on a single entity.
- **Goal:** discover  $(\_, \_, \text{so}, \_, \_)$ ,  
where e, a, h, and t are ignored
- **Reviews usually satisfy the assumption.**
  - Almost all papers use reviews
  - Positive: 4 or 5 stars, negative: 1 or 2 stars
- Many forum postings and blogs do not
  - They can mention and compare multiple entities
  - Many such postings express no sentiments

# Some Amazon reviews

248 of 263 people found the following review helpful:

★★★★★ **This is one to get if you want 5MP**, April 14, 2004

By [Gadgester "No Time, No Money"](#) (Mother Earth) - [See all my reviews](#)

TOP 100 REVIEWER

Amazon Verified Purchase ([What's this?](#))

This review is from: **Canon PowerShot S500 5MP Digital Elph with 3x Optical Zoom (Electronics)**

The new Canon PowerShot S500 is a 5MP upgrade to the immensely popular S400 model, which was a 4MP digital camera. The S500 produces excellent images, is easy to use, and is compact enough to carry in a pocket. 3X optical zoom is standard on these cameras. Besides shooting still photos, you can record low-res video clips as well as audio clips, but don't expect high quality on either.

For a hundred bux less, you can get the 4MP S410 model which is otherwise identical to the S500. Should you go for this or the S410? I think for most consumers 4MP is plenty enough, with room for cropping and enlargements. 5MP is only necessary if you really crop a lot \*and\* plan to blow up the cropped images. The S410 strikes a great balance between pixel count and price -- it's a better value.

Help other customers find the most helpful reviews

[Report abuse](#) | [Permalink](#)

Was this review helpful to you?

[Comment](#)

41 of 41 people found the following review helpful:

★★★★☆ **E18 Error / problem with the lens**, September 29, 2004

By [Johnathan Parker](#) (Springdale, AR USA) - [See all my reviews](#)

REAL NAME

This review is from: **Canon PowerShot S500 5MP Digital Elph with 3x Optical Zoom (Electronics)**

This is my second Canon digital elph camera. Both were great cameras. Recently upgraded to the S500. About 6 months later I get the dreaded E18 error. I searched the Internet and found numerous people having problems. When I determined the problem to be the lens not fully extending I decided to give it a tug. It clicked and the camera came on,

# Unsupervised classification

(Turney, 2002)

- Data: reviews from [epinions.com](http://epinions.com) on automobiles, banks, movies, and travel destinations.
- The approach: Three steps
- Step 1:
  - Part-of-speech (POS) tagging
  - Extracting two consecutive words (**two-word phrases**) from reviews if their tags conform to some given patterns, e.g., (1) JJ, (2) NN.

# Patterns of POS tags

First word	Second word	Third word (Not Extracted)
1. JJ	NN or NNS	anything
2. RB, RBR, or RBS	JJ	not NN nor NNS
3. JJ	JJ	not NN nor NNS
4. NN or NNS	JJ	not NN nor NNS
5. RB, RBR, or RBS	VB, VBD, VBN, or VBG	anything

- Step 2: Estimate the sentiment orientation (SO) of the extracted phrases

- Use Pointwise mutual information

$$PMI(word_1, word_2) = \log_2 \left( \frac{P(word_1 \wedge word_2)}{P(word_1)P(word_2)} \right)$$

- Semantic orientation (SO):

$$SO(phrase) = PMI(phrase, "excellent")$$

$$- PMI(phrase, "poor")$$

- Using AltaVista near operator to do search to find the number of hits to compute PMI and SO.

- Step 3: Compute the average SO of all phrases
  - classify the review as **positive** if average SO is positive, **negative** otherwise.
- Final classification accuracy:
  - automobiles - 84%
  - banks - 80%
  - movies - 65.83
  - travel destinations - 70.53%

# Supervised learning (Pang et al, 2002)

- Directly apply supervised learning techniques to classify reviews into positive and negative.
  - Like a text classification problem
- Three classification techniques were tried:
  - Naïve Bayes
  - Maximum entropy
  - Support vector machines
- Pre-processing:
  - Features: negation tag, unigram (single words), bigram, POS tag, position.

# Supervised learning

- Training and test data
  - Movie reviews with star ratings
    - 4-5 stars as positive
    - 1-2 stars as negative
- Neutral is ignored.
- SVM gives the best classification accuracy based on balance training data
  - 83%
  - Features: unigrams (bag of individual words)

# Features for supervised learning

- The problem has been studied by numerous researchers subsequently
  - Probably the most extensive studied problem
    - Including domain adaption and cross-lingual, etc.
- Key: feature engineering. A large set of features have been tried by researchers. E.g.,
  - Terms frequency and different IR weighting schemes
  - Part of speech (POS) tags
  - Opinion words and phrases
  - Negations
  - Syntactic dependency

# A large number of related papers

- Bickerstaffe and Zukerman (2010) used a hierarchical multi-classifier considering inter-class similarity
- Burfoot, Bird and Baldwin (2011) sentiment-classified congressional floor debates
- Cui et al. (2006) evaluated some sentiment classification algorithms
- Das and Chen (2001) extracted market sentiment from stock message boards
- Dasgupta and Ng (2009) used semi-supervised learning
- Dave, Lawrence & Pennock (2003) designed a custom function for classification
- Gamon (2004) classified customer feedback data

# A large number of related papers

- Goldberg and Zhu (2006) used semi-supervised learning.
- Kim, Li and Lee (2009) and Paltoglou and Thelwall (2010) studied different IR term weighting schemes
- Li, Lee, et al (2010) made use of different polarity shifting.
- Li, Huang, Zhou and Lee (2010) used personal (I, we) and impersonal (they, it, this product) sentences to help
- Maas et al (2011) used word vectors which are latent aspects of the words.
- Mullen and Collier (2004) used PMI, syntactic relations and other attributes with SVM.
- Nakagawa, Inui and Kurohashi (2010) used dependency relations and CRF.

# A large number of related papers

- Ng, Dasgupta and Arifin (2006) identified reviews and classified sentiments of reviews
- Pang and Lee (2004) used minimum cuts
- Qiu, Zhang, Hu and Zhao (2009) proposed a lexicon-based and self-supervision approach
- Tong (2001) used a set of domain specific phrases
- Yessenalina, Choi and Cardie (2010) automatically generated annotator rationales to help classification
- Yessenalina, Yue and Cardie (2010) found subjective sentences and then used them for model building
- Zhou, Chen and Wang (2010) used semi-supervised and active learning

# Review rating prediction

- Apart from classification of positive or negative sentiments,
  - research has also been done to **predict the rating scores** (e.g., 1–5 stars) of reviews (Pang and Lee, 2005; Liu and Seneff 2009; Qu, Ifrim and Weikum 2010; Long, Zhang and Zhu, 2010).
  - Training and testing are reviews with star ratings.
- **Formulation:** The problem is formulated as regression since the rating scores are ordinal.
- Again, feature engineering and model building.

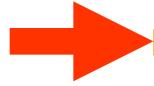
# Domain adaptation (transfer learning)

- Sentiment classification is sensitive to the domain of the training data.
  - A classifier trained using reviews from one domain often performs poorly in another domain.
    - words and even language constructs used in different domains for expressing opinions can be quite different.
    - same word in one domain may mean positive but negative in another, e.g., “*this vacuum cleaner really sucks.*”
- Existing research has used labeled data from one domain and unlabeled data from the target domain and general opinion words for learning (Aue and Gamon 2005; Blitzer et al 2007; Yang et al 2006; Pan et al 2010; Wu, Tan and Cheng 2009; Bollegala, Wei and Carroll 2011; He, Lin and Alani 2011).

# Cross-lingual sentiment classification

- Useful in the following scenarios:
  - E.g., there are many English sentiment corpora, but for other languages (e.g. Chinese), the annotated sentiment corpora may be limited.
  - Utilizing English corpora for Chinese sentiment classification can relieve the labeling burden.
- Main approach: use available language corpora to train sentiment classifiers for the target language data.  
Machine translation is typically employed
  - (Banea et al 2008; Wan 2009; Wei and Pal 2010; Kim et al. 2010; Guo et al 2010; Mihalcea & Wiebe 2010; Boyd-Graber and Resnik 2010; Banea et al 2010; Duh, Fujino & Nagata 2011; Lu et al 2011)

# Roadmap

- Opinion Mining Problem
- Document sentiment classification
-  ■ **Sentence subjectivity & sentiment classification**
- Aspect-based sentiment analysis
- Aspect-based opinion summarization
- Opinion lexicon generation
- Mining comparative opinions
- Some other problems
- Opinion spam detection
- Utility or helpfulness of reviews
- Summary

# Subjectivity classification

- Document-level sentiment classification is too coarse for most applications.
- We now move to the sentence level.
- Much of the early work on sentence level analysis focuses on identifying **subjective sentences**.
- **Subjectivity classification:** classify a sentence into one of the **two classes** (Wiebe et al 1999)
  - Objective and subjective.
- Most techniques use supervised learning.
  - E.g., a naïve Bayesian classifier (Wiebe et al. 1999).

# Sentence sentiment analysis

- Usually consist of two steps
  - Subjectivity classification
    - To identify subjective sentences
  - Sentiment classification of subjective sentences
    - Into two classes, positive and negative
- But bear in mind
  - Many objective sentences can imply sentiments
  - Many subjective sentences do not express positive or negative sentiments/opinions
    - E.g., "I believe he went home yesterday."

# As an intermediate step

- We do not use the quintuple ( $e, a, so, h, t$ ) to define the problem here because
  - sentence classification is an intermediate step.
- Knowing that some sentences have positive or negative opinions are not sufficient.
- However, it helps
  - filter out sentences with no opinions (mostly)
  - determine (to some extend) if sentiments about entities and their aspects are positive or negative.
    - But not enough

# Assumption

- **Assumption:** Each sentence is written by a single person and expresses a single positive or negative opinion/sentiment.
- **True for simple sentences**, e.g.,
  - “I like this car”
- **But not true for compound and “complex” sentences**, e.g.,
  - “I like the picture quality but battery life sucks.”
  - “Apple is doing very well in this lousy economy.”

# Subjectivity classification using patterns

(Riloff and Wiebe, 2003)

## ■ A bootstrapping approach.

- A high precision classifier is first used to automatically identify some subjective and objective sentences.
  - Two high precision (but low recall) classifiers are used,
    - a high precision subjective classifier
    - A high precision objective classifier
    - Based on manually collected lexical items, single words and n-grams, which are good subjective clues.
- A set of patterns are then learned from these identified subjective and objective sentences.
  - Syntactic templates are provided to restrict the kinds of patterns to be discovered, e.g., <subj> passive-verb.
- The learned patterns are then used to extract more subject and objective sentences (the process can be repeated).

# Subjectivity and sentiment classification

(Yu and Hazivassiloglou, 2003)

- **Subjective sentence identification:** a few methods were tried, e.g.,
  - Sentence similarity.
  - Naïve Bayesian classification.
- **Sentiment classification (positive, negative or neutral)** (also called **polarity**): it uses a similar method to (Turney, 2002), but
  - with more seed words (rather than two) and based on log-likelihood ratio (LLR).
  - For classification of each word, it takes the average of LLR scores of words in the sentence and use cutoffs to decide positive, negative or neutral.

# Segmentation and classification

- Since a single sentence may contain multiple opinions and subjective and factual clauses
- A study of automatic clause sentiment classification was presented in (Wilson et al 2004)
  - to classify clauses of every sentence by the *strength* of opinions being expressed in individual clauses, down to four levels
    - *neutral, low, medium, and high*
- Clause-level may not be sufficient
  - “Apple is doing very well in this lousy economy.”

# Some other related work

- Abdul-Mageed, Diab and Korayem (2011) carried out subjectivity and sentiment analysis of Arabic sentences
- Alm (2011) analyzed subjectivity research motivations, applications, characteristics, etc
- Barbosa and Feng (2010) and Davidov, Tsur and Rappoport (2010) performed Twitter subjectivity and sentiment classification using many features, hashtags, and smileys
- Eguchi and Lavrendo (2006) studied sentiment sentence retrieval
- Gamon et al. (2005) used semi-supervised learning
- Hassan, Qazvinian, Radev (2010) found attitude sentences
- Kim and Hovy (2004) summed up orientations of opinion words in a sentence (or within some word window).
- Hatzivassiloglou & Wiebe (2000) considered gradable adjectives

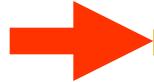
# Some other related work

- Johansson and Moschitti (2011) extracted opinion expressions and sentiments
- Joshi and Penstein-Rose (2009) used dependency triples with “back-off” using POS tags rather than words
- Kim and Hovy (2006a) automatically identified pro and con reasons
- Kim and Hovy (2006b) Identified judgment opinions
- Kim and Hovy (2007) mined predictive opinions in election postings
- Kim, Li and Lee (2010) compared subjectivity analysis tools
- McDonald et al (2007) performed sentence to document sentiment classification
- Mukund and Srihari (2010) performed subjectivity classification with co-training

# Some other related work

- Nasukawa and Yi (2003) captured favorability
- Nigam and Hurst (2005) classified subjective and topic sentences
- Tackstrom & McDonald (2011) performed sentence sentiment classification
- Wiebe et al (2004) learned subjective language
- Wiebe and Riloff (2005) used semi-supervised learning with a initial training set identified by some strong patterns
- Wiebe and Mihalcea (2006) studied word sense and subjectivity
- Wilson, Wiebe and Hwa (2006) recognized strong and weak opinion clauses
- Wilson et al. (2004, 2005) found strength of sentiments/opinions in clauses

# Roadmap

- Opinion Mining Problem
- Document sentiment classification
- Sentence subjectivity & sentiment classification
-  ■ **Aspect-based sentiment analysis**
- Aspect-based opinion summarization
- Opinion lexicon generation
- Mining comparative opinions
- Some other problems
- Opinion spam detection
- Utility or helpfulness of reviews
- Summary

# We need to go further

- Sentiment classification at both the document and sentence (or clause) levels are useful, **but**
  - They do not find what people liked and disliked.
- **They do not identify the targets of opinions, i.e.,**
  - Entities and their aspects
  - Without knowing targets, opinions are of limited use.
- **We need to go to the entity and aspect level.**
  - *Aspect-based opinion mining and summarization* (Hu and Liu 2004).
  - We thus need the full opinion definition.

# Recall an opinion is a quintuple

## ■ An *opinion* is a quintuple

$$(e_j, a_{jk}, so_{ijkl}, h_i, t_i),$$

where

- $e_j$  is a target entity.
- $a_{jk}$  is an aspect/feature of the entity  $e_j$ .
- $so_{ijkl}$  is the sentiment value of the opinion of the opinion holder  $h_i$  on feature  $a_{jk}$  of entity  $e_j$  at time  $t_i$ .  
 $so_{ijkl}$  is +ve, -ve, or neu, or a more granular rating.
- $h_i$  is an opinion holder.
- $t_i$  is the time when the opinion is expressed.

# Aspect-based sentiment analysis

- Much of the research is based on online reviews
- For reviews, aspect-based sentiment analysis is easier because the entity (i.e., product name) is usually known
  - Reviewers simply express positive and negative opinions on different aspects of the entity.
- For blogs, forum discussions, etc., it is harder:
  - both entity and aspects of entity are unknown,
  - there may also be many comparisons, and
  - there is also a lot of irrelevant information.

# Find entities (entity set expansion)

- Although similar, it is somewhat different from the traditional named entity recognition (NER).
- E.g., one wants to study opinions on phones
  - given Motorola and Nokia, find all phone brands and models in a corpus, e.g., Samsung, Moto,
- Formulation: Given a set  $Q$  of seed entities of class  $C$ , and a set  $D$  of candidate entities, we wish to determine which of the entities in  $D$  belong to  $C$ .
  - A classification problem. It needs a binary decision for each entity in  $D$  (belonging to  $C$  or not)
  - But it's often solved as a ranking problem

# Some methods (Li, Zhang et al 2010, Zhang and Liu 2011)

- **Distributional similarity**: This is the traditional method used in NLP. It compares the surrounding text of candidates using cosine or PMI.
  - It performs poorly.
- **PU learning**: learning from positive and unlabeled examples.
  - S-EM algorithm (Liu et al. 2002)
- **Bayesian Sets**: We extended the method given in (Ghahramani and Heller, 2006).

# Aspect extraction

- **Goal:** Given an opinion corpus, extract all aspects
- **A frequency-based approach** (Hu and Liu, 2004): nouns (NN) that are frequently talked about are likely to be true **aspects** (called frequent aspects) .
- **Why the frequency based approach?**
  - Different reviewers tell different stories (irrelevant)
  - When product aspects/features are discussed, the words they use converge.
  - They are the main aspects.
- Sequential/association pattern mining finds **frequent nouns and noun phrases**.

# An example review

**GREAT Camera.**, Jun 3, 2004

Reviewer: **jprice174** from Atlanta, Ga.

I did a lot of research last year before I bought this camera... It kinda hurt to leave behind my beloved nikon 35mm SLR, but I was going to Italy, and I needed something smaller, and digital.

The **pictures** coming out of this camera are amazing. The 'auto' feature takes great **pictures** most of the time. And with digital, you're not wasting film. ....

....

# Infrequent aspect extraction

- To improve recall due to loss of infrequent aspects. It uses opinions words to extract them
- **Key idea:** opinions have targets, i.e., opinion words are used to modify aspects and entities.
  - “The pictures are absolutely amazing.”
  - “This is an amazing software.”
- The modifying relation was approximated with the nearest noun to the opinion word.
- The idea was generalized to dependency in (Zhuang et al 2006) and double propagation in (Qiu et al 2009;2011).
  - It has been used in many papers and practical systems

# Using part-of relationship and the Web

(Popescu and Etzioni, 2005)

- Improved (Hu and Liu, 2004) by removing those frequent noun phrases that may not be aspects: better precision (a small drop in recall).
- It identifies **part-of** relationship
  - Each noun phrase is given a pointwise mutual information score between the phrase and **part discriminators** associated with the product class, e.g., a scanner class.
  - E.g., “of scanner”, “scanner has”, etc, which are used to find parts of scanners by searching on the Web:

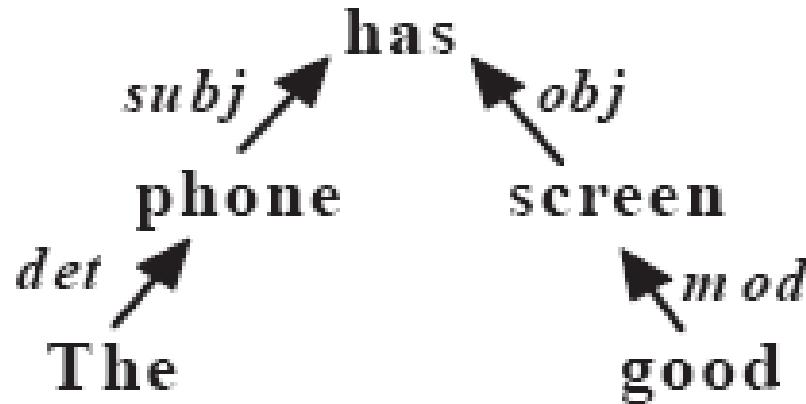
$$PMI(a, d) = \frac{hits(a \wedge d)}{hits(a)hits(d)},$$

# Extract aspects using DP (Qiu et al. 2009; 2011)

- A *double propagation* (DP) approach proposed
- Based on the definition earlier, **an opinion should have a target**, entity or aspect.
- Use dependency of opinions & aspects to extract both aspects & opinion words.
  - Knowing one helps find the other.
  - E.g., “*The rooms* are *spacious*”
- It extracts both aspects and opinion words.
  - A domain independent method.

# The DP method

- DP is a bootstrapping method
  - Input: a set of seed opinion words,
  - no aspect seeds needed
- Based on dependency grammar (Tesniere 1959).
  - “This phone has good screen”



# Rules from dependency grammar

	Relations and Constraints	Output	Examples
R1 <sub>1</sub>	$O \rightarrow O\text{-}Dep \rightarrow F$ s.t. $O \in \{O\}$ , $O\text{-}Dep \in \{MR\}$ , $POS(F) \in \{NN\}$	$f = F$	<i>The phone has a <u>good</u> “screen”.</i> <i>good → mod → screen</i>
R1 <sub>2</sub>	$O \rightarrow O\text{-}Dep \rightarrow H \leftarrow F\text{-}Dep \leftarrow F$ s.t. $O \in \{O\}$ , $O/F\text{-}Dep \in \{MR\}$ , $POS(F) \in \{NN\}$	$f = F$	<i>“iPod” is the <u>best</u> mp3 player.</i> <i>best → mod → player ← subj ← iPod</i>
R2 <sub>1</sub>	$O \rightarrow O\text{-}Dep \rightarrow F$ s.t. $F \in \{F\}$ , $O\text{-}Dep \in \{MR\}$ , $POS(O) \in \{JJ\}$	$o = O$	same as R1 <sub>1</sub> with <i>screen</i> as the known word and <i>good</i> as the extracted word
R2 <sub>2</sub>	$O \rightarrow O\text{-}Dep \rightarrow H \leftarrow F\text{-}Dep \leftarrow F$ s.t. $F \in \{F\}$ , $O/F\text{-}Dep \in \{MR\}$ , $POS(O) \in \{JJ\}$	$o = O$	same as R1 <sub>2</sub> with <i>iPod</i> is the known word and <i>best</i> as the extract word.
R3 <sub>1</sub>	$F_{i(j)} \rightarrow F_{i(j)}\text{-}Dep \rightarrow F_{j(i)}$ s.t. $F_{j(i)} \in \{F\}$ , $F_{i(j)}\text{-}Dep \in \{CONJ\}$ , $POS(F_{i(j)}) \in \{NN\}$	$f = F_{i(j)}$	<i>Does the player play dvd with <u>audio</u> and “video”?</i> <i>video → conj → audio</i>
R3 <sub>2</sub>	$F_i \rightarrow F_i\text{-}Dep \rightarrow H \leftarrow F_j\text{-}Dep \leftarrow F_j$ s.t. $F_i \in \{F\}$ , $F_i\text{-}Dep = F_j\text{-}Dep$ , $POS(F_j) \in \{NN\}$	$f = F_j$	<i>Canon “G3” has a great <u>len</u>.</i> <i>len → obj → has ← subj ← G3</i>
R4 <sub>1</sub>	$O_{i(j)} \rightarrow O_{i(j)}\text{-}Dep \rightarrow O_{j(i)}$ s.t. $O_{j(i)} \in \{O\}$ , $O_{i(j)}\text{-}Dep \in \{CONJ\}$ , $POS(O_{i(j)}) \in \{JJ\}$	$o = O_{i(j)}$	<i>The camera is <u>amazing</u> and “easy” to use.</i> <i>easy → conj → amazing</i>
R4 <sub>2</sub>	$O_i \rightarrow O_i\text{-}Dep \rightarrow H \leftarrow O_j\text{-}Dep \leftarrow O_j$ s.t. $O_i \in \{O\}$ , $O_i\text{-}Dep = O_j\text{-}Dep$ , $POS(O_j) \in \{JJ\}$	$o = O_j$	<i>If you want to buy a <u>sexy</u>, “cool”, accessory-available mp3 player, you can choose iPod.</i> <i>sexy → mod → player ← mod ← cool</i>

# Explicit and implicit aspects

(Hu and Liu 2004)

- **Explicit aspects**: Aspects explicitly mentioned as nouns or noun phrases in a sentence
  - The **picture quality** is of this phone is great.
- **Implicit aspects**: Aspects not explicitly mentioned in a sentence but are implied
  - “This car is so **expensive**.”
  - “This phone will not easily **fit in a pocket**.”
  - “Included **16MB** is stingy”
- Not much work has been done on mining or mapping implicit aspects.

# Implicit aspect mapping

- There are many types of implicit aspect expressions. Adjectives and adverbs are perhaps the most common type.
  - Most adjectives modify or describe some specific attributes of entities.
  - “expensive” ⇒ aspect “price,” “beautiful” ⇒ aspect “appearance”, “heavy” ⇒ aspect “weight”
- Although manual mapping is possible, in different contexts, the meaning can be different.
  - E.g., “The computation is expensive”.

# A mutual reinforcement method

(Su et al. 2009)

- It proposed an unsupervised approach which exploits the mutual reinforcement relationship between aspects and opinion words.
  - Specifically, it uses the co-occurrence of aspect and opinion word pair in a sentence.
- The algorithm iteratively clusters the set of aspects and the set of opinion words separately,
  - but before clustering each set, clustering results of the other set is used to update the pairwise weight of the set.
  - The model is based on a bipartite graph.

# Other papers on aspect extraction

We will discuss topic modeling based methods later.

- Carvalho et al (2011) annotated political debates with aspects and others.
- Choi and Cardie (2010) used a CRF based approach.
- Jin and Ho (2009) proposed a HMM-based method
- Jakob and Gurevych (2010) used anaphora (or coreference) resolution to help find aspects that are mentioned in previous sentences but are referred to as pronouns in the next sentences.
  - E.g., “I took a few pictures yesterday. They look great.”
  - There is almost no improvement with anaphora resolution, higher recall but lower precision.

# Other papers on aspect extraction

- Jakob and Gurevych (2010) used CRF to train on review sentences from different domains for a more domain independent extraction. A set of domain independent features were used, e.g. tokens, POS tags, dependency, word distance, and opinion sentences.
- Kobayashi et al (2006) extracted subject-attribute-value
- Kobayashi et al (2007) extracted aspect-evaluation and aspect-of relations using mined patterns.
- Ku et al. (2006a, 2006b) performed the extraction from Chinese reviews and news.

# Other papers on aspect extraction

- Li et al (coling-2010) integrated Skip-CRF and Tree-CRF to extract aspects and opinions. It was able to exploit structure features
- Long, Zhang and Zhu (2010) extracted aspects (nouns) based on frequency and the Web, and dependent words (adjectives). These words are then used to select reviews which discuss an aspect most.
- Ma and Wan (2010) used centering theory for extraction in news comments. It also exploited aspects in the news title and contents.
- Meng and Wang (2009) extracted aspects from product specifications, which are usually structured data.

# Other papers on aspect extraction

- Scaffidi et al (2007) extracted frequent nouns and noun phrases but compare their frequency in a review corpus with their occurrence rates in generic English to identify true aspects
- Somasundaran and Wiebe (2009) also used syntactic dependency for aspect and opinion extraction.
- Toprak, Jakob and Gurevych (2010) designed a comprehensive annotation scheme for aspect-based opinion annotation. Earlier annotations are partial and mainly for individual papers.
- Yi et al (2003) used language models to extract product features.

# Other papers on aspect extraction

- Yu et al (2011) ranked aspects by considering their frequency and contribution to the overall review rating
- Zhu et al (CIKM-2009) used a method for finding multi-word terms, called cvalue, to find aspects.
  - The method also segments a sentence with multiple aspects.

# Identify aspect synonyms (Carenini et al 2005)

- Once aspect expressions are discovered, group them into aspect categories.
  - E.g., **power usage** and **battery life** are the same.
- It proposed a method based on some similarity metrics, but it needs a taxonomy of aspects.
  - The system merges each discovered aspect to a aspect node in the taxonomy.
  - Similarity metrics: string similarity, synonyms and other distances measured using WordNet.
- Many ideas in **Web information integration** are applicable.

# Multilevel latent categorization

(Guo et al 2009)

- This method performs multilevel latent semantic analysis to group aspects expressions.
  - At the first level, all the words in aspect expressions are grouped into a set of concepts using LDA. The results are used to build *latent topic structures* for aspect expressions, e.g.,
    - *touch screen*: topic-1, topic-2
  - At the second level, aspect expressions are grouped by LDA again according to
    - their latent topic structures produced from level 1 and
    - context snippets in reviews.

# Group aspect synonyms (Zhai et al. 2011a, b)

- A variety of information/similarities are used to cluster aspect expressions into aspect categories.
  - Lexical similarity based on WordNet
  - Distributional information (surrounding words context)
  - Syntactical constraints (sharing words, in the same sentence)
- Two unsupervised learning methods were used:
  - Clustering: EM-based.
  - Constrained topic modeling: Constrained-LDA
    - By intervening Gibbs sampling.

# The EM method

## ■ WordNet similarity

$$Jcn(w_1, w_2) = \frac{1}{IC(w_1) + IC(w_2) - 2 \times Res(w_1, w_2)}$$

## ■ EM-based probabilistic clustering

$$P(w_t|c_j) = \frac{1 + \sum_{i=1}^{|D|} N_{ti} P(c_j|d_i)}{|V| + \sum_{m=1}^{|V|} \sum_{i=1}^{|D|} N_{mi} P(c_j|d_i)}$$

$$P(c_j) = \frac{1 + \sum_{i=1}^{|D|} P(c_j|d_i)}{|\mathcal{C}| + |D|}$$

$$P(c_j|d_i) = \frac{P(c_j) \prod_{k=1}^{|d_i|} P(w_{d_i,k}|c_j)}{\sum_{r=1}^{|\mathcal{C}|} P(c_r) \prod_{k=1}^{|d_i|} P(w_{d_i,k}|c_r)}$$

# Aspect sentiment classification

- For each aspect, identify the sentiment or opinion expressed on it.
- Work based on sentences, but also consider,
  - A sentence can have multiple aspects with different opinions.
  - E.g., The battery life and picture quality are *great* (+), but the view founder is *small* (-).
- Almost all approaches make use of **opinion words and phrases**. But notice:
  - Some opinion words have context independent orientations, e.g., “good” and “bad” (almost)
  - Some other words have context dependent orientations, e.g., “small” and “sucks” (+ve for vacuum cleaner)

# Some approaches

- **Supervised learning**
  - Sentence level classification can be used, but ...
  - Need to consider target and thus to segment a sentence (e.g., Jiang et al. 2011)
- **Lexicon-based approach** (Ding, Liu and Yu, 2008)
  - **Need parsing to deal with:** Simple sentences, compound sentences, comparative sentences, conditional sentences, questions; different verb tenses, etc.
  - Negation (not), contrary (but), comparisons, etc.
  - A large opinion lexicon, context dependency, etc.
  - **Easy:** “*Apple* is doing well in this bad *economy*.”

# A lexicon-based method (Ding, Liu and Yu 2008)

- **Input:** A set of opinion words and phrases. A pair  $(a, s)$ , where  $a$  is an aspect and  $s$  is a sentence that contains  $a$ .
- **Output:** whether the opinion on  $a$  in  $s$  is +ve, -ve, or neutral.
- Two steps:
  - Step 1: split the sentence if needed based on BUT words (but, except that, etc).
  - Step 2: work on the segment  $s_f$  containing  $a$ . Let the set of opinion words in  $s_f$  be  $w_1, \dots, w_n$ . Sum up their orientations (1, -1, 0), and assign the orientation to  $(a, s)$  accordingly.

$$\sum_{i=1}^n \frac{w_i \cdot o}{d(w_i, a)}$$

where  $w_i \cdot o$  is the opinion orientation of  $w_i$ .  $d(w_i, a)$  is the distance from  $a$  to  $w_i$ .

# Sentiment shifters (e.g., Polanyi and Zaenen 2004)

- Sentiment/opinion shifters (also called **valence shifters**) are words and phrases that can shift or change opinion orientations.
- Negation words like *not*, *never*, *cannot*, etc., are the most common type.
- Many other words and phrases can also alter opinion orientations. E.g., **modal auxiliary verbs** (e.g., *would*, *should*, *could*, etc)
  - “The brake could be improved.”

# Sentiment shifters (contd)

- Some **presuppositional** items also can change opinions, e.g., *barely* and *hardly*
  - “It hardly works.” (comparing to “it works”)
  - It presupposes that better was expected.
- Words like *fail*, *omit*, *neglect* behave similarly,
  - “This camera fails to impress me.”
- Sarcasm changes orientation too
  - “What a great car, it did not start the first day.”
- Jia, Yu and Meng (2009) designed some rules based on parsing to find the scope of negation.

# Basic rules of opinions (Liu, 2010)

- Opinions/sentiments are governed by many rules, e.g.,
  - *Opinion word or phrase*, ex: “I love this car”
    - P ::= a positive opinion word or phrase
    - N ::= an negative opinion word or phrase
  - *Desirable or undesirable facts*, ex: “After my wife and I slept on it for two weeks, I noticed a mountain in the middle of the mattress”
    - P ::= desirable fact
    - N ::= undesirable fact

# Basic rules of opinions

- *High, low, increased and decreased quantity of a positive or negative potential item, ex: “The battery life is long.”*

PO ::= no, low, less or decreased quantity of NPI  
| large, larger, or increased quantity of PPI

NE ::= no, low, less, or decreased quantity of PPI  
| large, larger, or increased quantity of NPI

NPI ::= a negative potential item

PPI ::= a positive potential item

# Basic rules of opinions

- *Decreased and increased quantity of an opinionated item, ex: “This drug reduced my pain significantly.”*

PO ::= less or decreased N

| more or increased P

NE ::= less or decreased P

| more or increased N

- *Deviation from the desired value range: “This drug increased my blood pressure to 200.”*

PO ::= within the desired value range

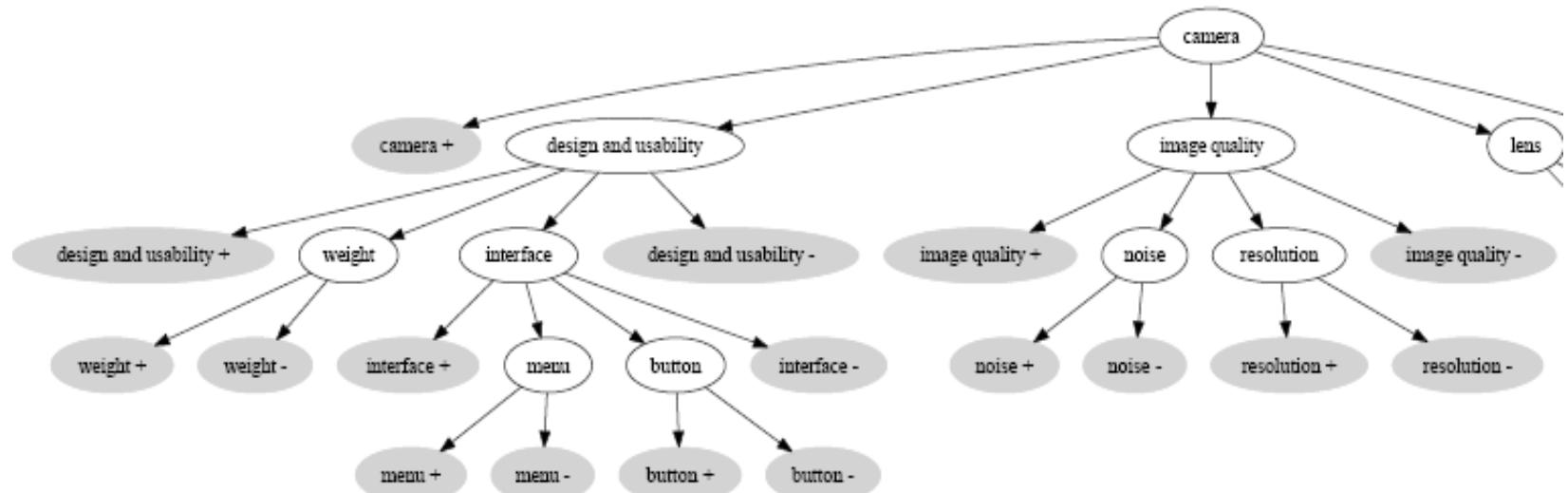
NE ::= above or below the desired value range

# Basic rules of opinions

- *Producing and consuming resources and wastes, ex:*  
“This washer uses a lot of water”
  - PO ::= produce a large quantity of or more resource
    - | produce no, little or less waste
    - | consume no, little or less resource
    - | consume a large quantity of or more waste
  - NE ::= produce no, little or less resource
    - | produce some or more waste
    - | consume a large quantity of or more resource
    - | consume no, little or less waste

# Sentiment ontology tree (Wei and Gulla, 2010)

- Recall in the definition of opinions, we simplified the tree structure to two levels (entity & aspects).
- This paper uses a full tree ontology to denote the relationships of aspects of a product.



# Sentiment ontology tree (contd)

- The leaves of the tree are positive or negative sentiments.
- It then uses a hierarchical classification model to learn to assign an sentiment to each node, which is reflected as a child leaf node.
  - Hierarchical classifier is useful here because it considers parents when classifying children.
- However, the ontology for each product has to be built manually.

# Aspect-sentiment statistical models

- This direction of research is mainly based on **topic models**:
  - pLSA: Probabilistic Latent Semantic Analysis (Hofmann 1999)
  - LDA: Latent Dirichlet allocation (Blei, Ng & Jordan, 2003; Griffiths & Steyvers, 2003; 2004)
- Topic models:
  - documents are mixtures of topics
  - a topic is a probability distribution over words.
- A topic model is a document **generative model**
  - it specifies a simple probabilistic procedure by which documents can be generated.

# Aspect-sentiment model (Mei et al 2007)

- This model is based on pLSA (Hofmann, 1999).
- It builds a topic (aspect) model, a positive sentiment model, and a negative sentiment model.
- A training data is used to build the initial models.
  - Training data: topic queries and associated positive and negative sentences about the topics.
- The learned models are then used as priors to build the final models on the target data.
- Solution: log likelihood and EM algorithm

# Multi-Grain LDA to extract aspects

(Titov and McDonald, 2008a, 2008b)

- Unlike a diverse document set used for traditional topic modeling. All reviews for a product talk about the same topics/aspects. It makes applying PLSA or LDA in the traditional way problematic.
- Multi-Grain LDA (MG-LDA) models global topics and local topics (Titov and McDonald, 2008a).
  - Global topics are entities (based on reviews)
  - Local topics are aspects (based on local context, sliding windows of review sentences)
- MG-LDA was extended to MAS model to give aspect rating (Titov and McDonald, 2008b).

# Aspect-rating of short text (Lu et al 2009)

- This work makes use of short phrases, head terms ( $w_h$ ) and their modifiers ( $w_m$ ), i.e.
  - ( $w_m, w_h$ )
  - E.g., great shipping, excellent seller
- Objective: (1) extract aspects and (2) compute their ratings in each short comment.
- It uses pLSA to extract and group aspects
- It uses existing rating for the full post to help determine aspect ratings.

# Aspect-rating regression

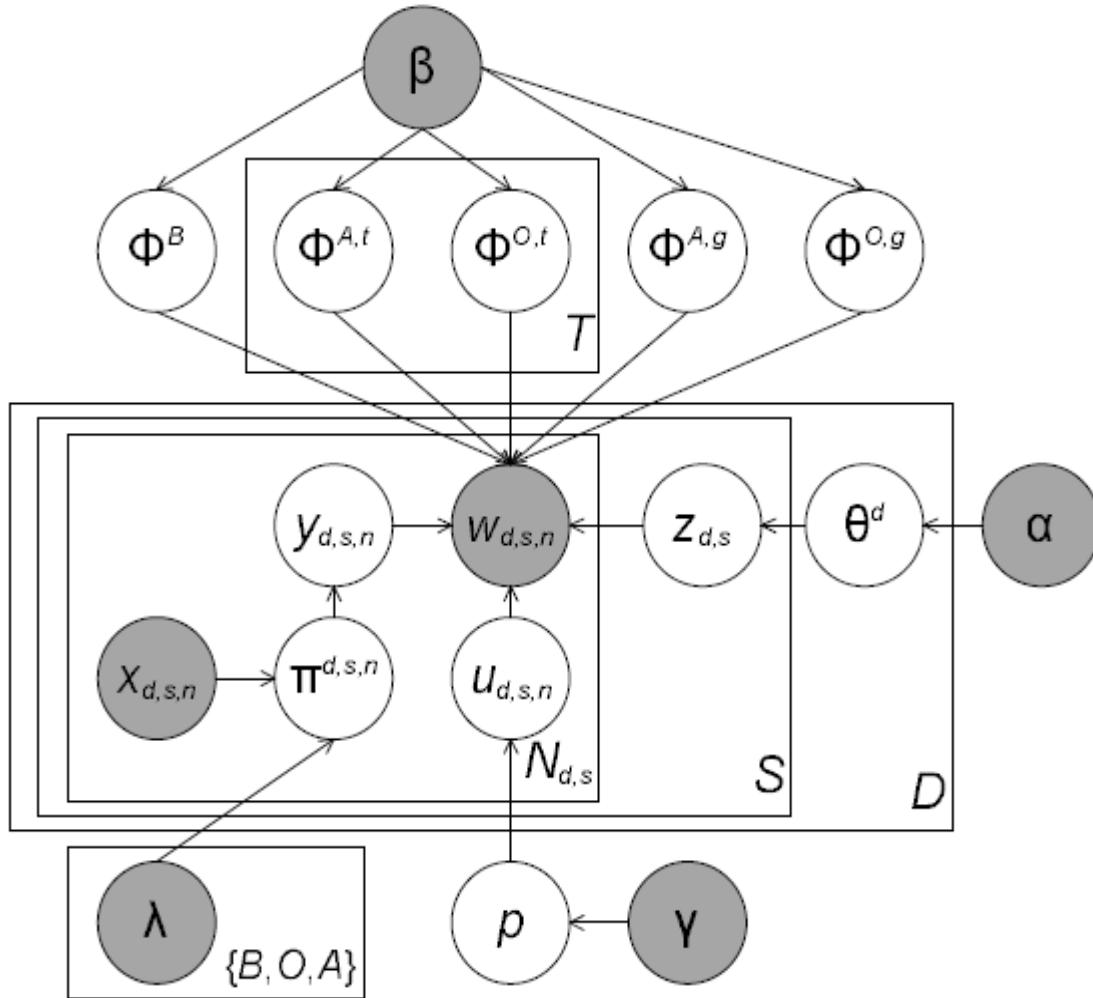
(Wang, Lu, and Zhai, 2010)

- In this work, some seed aspects are given. Its first step finds more aspect words using a heuristic bootstrapping method.
- Its regression model makes use of the review rating and assumes the overall review rating is a linear combination of its aspect ratings.
- The problem is model as a Bayesian regression problem.
  - It is solved using log-likelihood and EM.

# MaxEnt-LDA Hybrid (Zhao et al. 2010)

We now describe the generative process of the model. First, we draw several multinomial word distributions from a symmetric Dirichlet prior with parameter  $\beta$ : a background model  $\phi^B$ , a general aspect model  $\phi^{A,g}$ , a general opinion model  $\phi^{O,g}$ ,  $T$  aspect models  $\{\phi^{A,t}\}_{t=1}^T$  and  $T$  aspect-specific opinion models  $\{\phi^{O,t}\}_{t=1}^T$ . All these are multinomial distributions over the vocabulary, which we assume has  $V$  words. Then for each review document  $d$ , we draw a topic distribution  $\theta^d \sim \text{Dir}(\alpha)$  as in standard LDA. For each sentence  $s$  in document  $d$ , we draw an aspect assignment  $z_{d,s} \sim \text{Multi}(\theta^d)$ .

# Graphical model (plate)



- $y_{d,s,n}$  indicates
  - Background word
  - Aspect word, or
  - Opinion word
- MaxEnt is used to train a model using training set
  - $\pi^{d,s,n}$
  - $x_{d,s,n}$  feature vector
- $U_{d,s,n}$  indicates
  - General or
  - Aspect-specific

# Topic model of snippets

(Sauper, Haghghi and Barzilay, 2011)

- This method works on short snippets already extracted from reviews.
  - “battery life is the best I’ve found”
- The model is a variation of LDA but with seeds for sentiment words as priors,
  - but it also has HMM for modeling the sequence of words with types (aspect word, sentiment word, or background word).
- Inference: variational technique

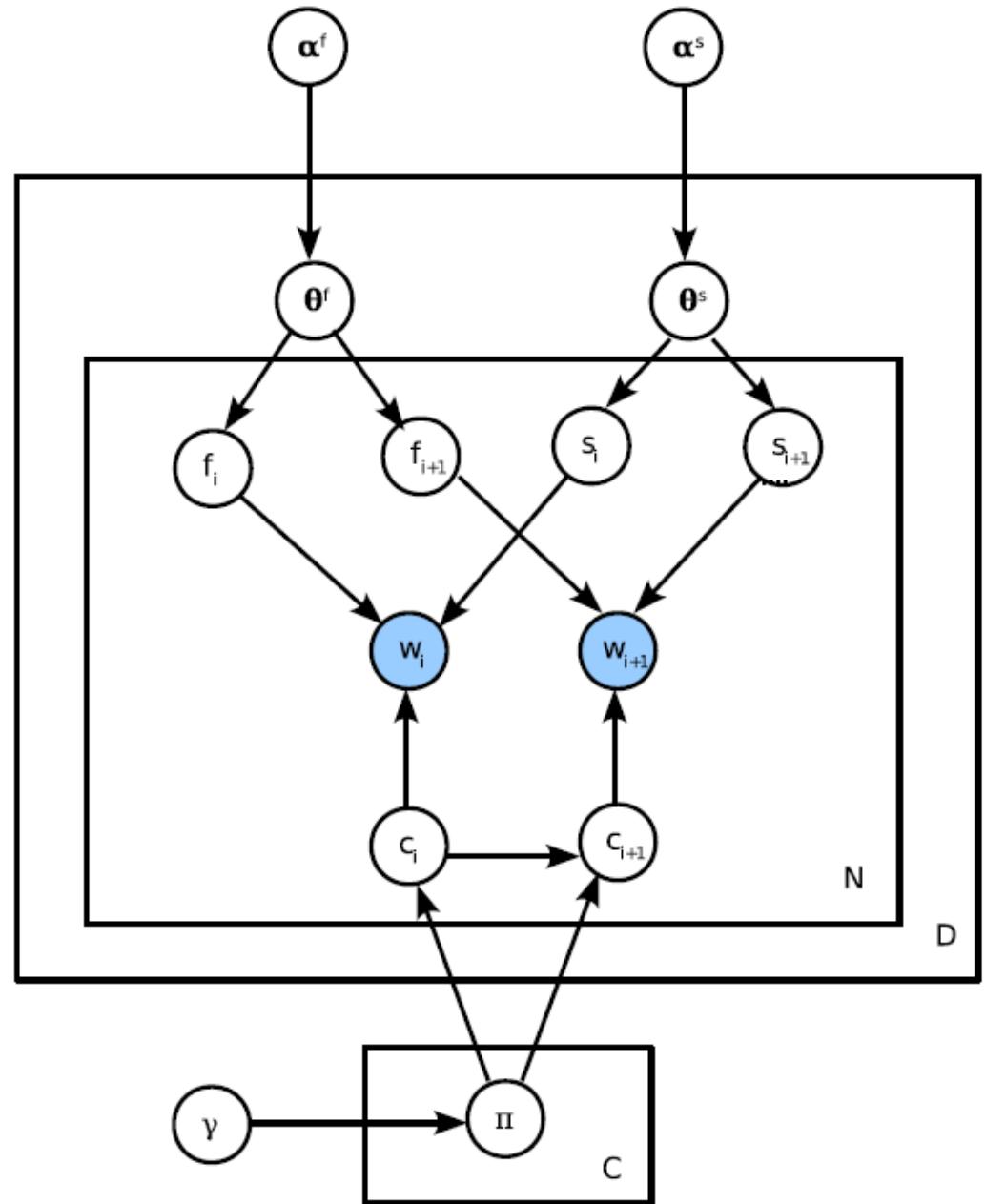
# Considering both syntax and semantics

(Lakkaraju et al. 2011)

- This work is based the composite model of HMM-LDA of Griffiths et al. (2005), which consider both word sequence and word-bag
- It captures both syntactic structure and semantic dependencies (similar to the previous paper)
- A class label is used for each word to represent the syntactic category of the word, whether it is
  - an aspect word,
  - a sentiment word, or
  - some other category.

# FACTS model

- Words:  $w_{d,1}, w_{d,2} \dots w_{d,N}$
- Hidden variables
  - Class:  $c_{d,i}$ 
    - 1: aspect word
    - 2: sentiment word
    - others
  - Aspect cat.:  $f_{d,i}$
  - Sentiment cat.:  $s_{d,i}$
- It also has more sophisticated models
  - CFACTS: consider neighboring windows
  - CFACTS-R: consider ratings



# About topic model based methods

- There several other similar topic model based methods (e.g., Brody and Elhadad, 2010; Lu et al. 2011; Jo and Oh, 2011; Lin and He 2009; Liu et al, 2007).
- These methods tend to need a large number reviews (10000 and more) to make it statistically stable. They are hard to use for most specific products, which often have <100 reviews.
- They also need a lot of parameter tuning.
- The results usually are quite coarse, not precise enough for practical needs.

# Roadmap

- Opinion Mining Problem
- Document sentiment classification
- Sentence subjectivity & sentiment classification
- Aspect-based sentiment analysis
- ➡ ■ **Aspect-based opinion summarization**
- Opinion lexicon generation
- Mining comparative opinions
- Some other problems
- Opinion spam detection
- Utility or helpfulness of reviews
- Summary

# Aspect-based opinion summarization

- A multi-document summarization problem.
  - An opinion from a single person is usually not sufficient for action unless from a VIP (e.g., President)
- Key Idea: Use aspects as basis for a summary
  - Not done in traditional multi-document summarization.
- We have discussed the aspect-based summary using quintuples earlier (Hu and Liu 2004; Liu, 2010).
  - Also called: *Structured Summary*
- Similar approaches are also taken in
  - (e.g., Ku et al 2006; Carenini, Ng and Paul 2006) and
  - By most topic model based methods

# Text summary of opinions

- One can also generate a summary in the **tradition fashion**, e.g., producing a short text summary (Lerman et al 2009), by extracting some important sentences, etc.
  - Weakness: It is only qualitative but not quantitative.
- One can generate sentences based on aspects and opinions using some templates.
  - E.g., 60% of the people like the picture quality.

# Select and order sentences

(Tata and Di Eugenio, 2010)

- If we produce summary as a list of sentences for each aspect and each sentiment (+ or -), it is useful to
  - Select a representative sentence for each group: it selects a sentence that mention fewest aspects (the sentence is focused).
  - Order the sentences: It uses an ontology to map sentences to the ontology nodes (domain concepts).

# Informativeness and Readability

(Nishikawa et al. 2010)

- It summarizes by considering both informativeness and readability.
- It uses frequency  $f(\cdot)$  of (aspect, opinion), but it is more like a traditional summary.
- It is not quantitative. Note: Lerman et al (2009) used +ve/-ve proportions.

- $S^*$  is the summary

$$S^* = \underset{S \in T}{\operatorname{argmax}} [\operatorname{Info}(S) + \lambda \operatorname{Read}(S)]$$

s.t.  $\operatorname{length}(S) \leq K$

$$\operatorname{Info}(S) = \sum_{e \in E(S)} f(e)$$

$$\operatorname{Read}(S) = \sum_{i=0}^n \mathbf{w}^\top \phi(s_i, s_{i+1})$$

# Summarization using an ontology

(Lu et al. Coling-2010)

- This work uses existing online ontologies of entities and aspects to organize opinions
  - Given an entity and an online ontology of the entity
  - **Goal:** Generate a structured summary of opinions.
- It performs
  - Aspect selection to capture major opinions
  - Aspect ordering that is natural for human viewing
  - Suggest new aspects to add to ontology

# Summarization using an ontology (contd)

- Aspect selection
  - E.g., by frequency, by opinion coverage (no redundancy), or by conditional entropy
- Ordering aspects and their corresponding sentences based on their appearance in their original posts, called **coherence**

$$Co(A_i, A_j) = \frac{\sum_{S_{i,k} \in S_i, S_{j,l} \in S_j} Co(S_{i,k}, S_{j,l})}{|S_i||S_j|}$$

$$\hat{\pi}(A') = \arg \max_{\pi(A')} \sum_{A_i, A_j \in A', A_i \prec A_j} Co(A_i, A_j)$$

# Some other summarization papers

- Carenini, Ng and Pauls (2006) evaluated different summarization methods using human judges.
- Huang, Wan and Xiao (2011) generated contrast summaries of news.
- Kim and Zhai (2009) generated contrast opinion sentence pairs.
- Lerman and McDonald (2009) generated summaries to contrast opinions about two different products.
- Lerman, Blair-Goldensohn and McDonald (2009) designed three summarizers and evaluated them with human raters.
- Paul, Zhai and Girju (2010) found opposing views.
- Park, Lee and Song (2011) also found opposing views
- Wang and Liu (2011) generated opinion summary for conversations.

# Roadmap

- Opinion Mining Problem
- Document sentiment classification
- Sentence subjectivity & sentiment classification
- Aspect-based sentiment analysis
- Aspect-based opinion summarization
- **Opinion lexicon generation**
- Mining comparative opinions
- Some other problems
- Opinion spam detection
- Utility or helpfulness of reviews
- Summary

# Opinion (or sentiment) lexicon

- **Opinion lexicon:** lists of words and expressions used to express people's subjective feelings and sentiments/opinions.
  - Not just individual words, but also phrases and idioms, e.g., "cost an arm and a leg"
- They are instrumental for opinion mining.
- There seems to be endless variety of sentiment bearing expressions.
  - We have compiled more than 6,700 individual words.
  - There are also a large number of phrases.

# Opinion lexicon

- Opinion words or phrases (also called polar words, opinion bearing words, etc). E.g.,
  - Positive: beautiful, wonderful, good, amazing,
  - Negative: bad, poor, terrible, cost an arm and a leg.
- Many of them are context dependent, not just application domain dependent.
- Three main ways to compile such lists:
  - Manual approach: not a bad idea, only an one-time effort
  - Corpus-based approach
  - Dictionary-based approach

# Corpus-based approaches

- Rely on syntactic patterns in large corpora.  
(Hazivassiloglou and McKeown, 1997; Turney, 2002; Yu and Hazivassiloglou, 2003; Kanayama and Nasukawa, 2006; Ding, Liu and Yu, 2008)
  - Can find domain dependent orientations (positive, negative, or neutral).
- (Turney, 2002) and (Yu and Hazivassiloglou, 2003) are similar.
  - Assign opinion orientations (polarities) to words/phrases.
  - (Yu and Hazivassiloglou, 2003) is slightly different from (Turney, 2002)
    - use more seed words (rather than two) and use log-likelihood ratio (rather than PMI).

# Corpus-based approaches (contd)

- **Sentiment consistency:** Use conventions on connectives to identify opinion words (Hazivassiloglou and McKeown, 1997). E.g.,
  - **Conjunction:** conjoined adjectives usually have the same orientation.
    - E.g., “This car is *beautiful* and *spacious*.” (conjunction)
  - AND, OR, BUT, EITHER-OR, and NEITHER-NOR have similar constraints.
  - **Learning using**
    - **log-linear model:** determine if two conjoined adjectives are of the same or different orientations.
    - **Clustering:** produce two sets of words: positive and negative

# Find domain opinion words

- A similar approach was also taken in (Kanayama and Nasukawa, 2006) but for Japanese words:
  - Instead of only based on intra-sentence sentiment consistency, the new method also looks at the previous and next sentence, i.e., inter-sentence sentiment consistency.
  - Have an initial seed lexicon of positive and negative words.

# Context dependent opinion

- Find domain opinion words is insufficient. A word may indicate different opinions in same domain.
  - “The battery life is *long*” (+) and “It takes a *long* time to focus” (-).
- Ding, Liu and Yu (2008) and Ganapathibhotla and Liu (2008) exploited sentiment consistency (both inter and intra sentence) based on contexts
  - It finds context dependent opinions.
  - Context: (adjective, aspect), e.g., (long, battery\_life)
  - It assigns an opinion orientation to the pair.

# The Double Propagation method

(Qiu et al 2009, 2011)

- The same DP method can also use dependency of opinions & aspects to extract new opinion words.
- Based on dependency relations
  - Knowing an aspect can find the opinion word that modifies it
    - E.g., “The **rooms** are **spacious**”
  - Knowing some opinion words can find more opinion words
    - E.g., “The **rooms** are **spacious** and **beautiful**”

# Opinions implied by objective terms

(Zhang and Liu, 2011a)

- Most opinion words are adjectives and adverbs, e.g., good, bad, etc
  - There are also many subjective and opinion verbs and nouns, e.g., hate (VB), love (VB), crap (NN).
- **But objective nouns can imply opinions too.**
  - E.g., “After sleeping on the mattress for one month, a **valley/body impression** has formed in the middle.”
- How to discover such nouns in a domain or context?

# The technique

- Sentiment analysis to determine whether the context is +ve or –ve.
  - E.g., “I saw a **valley** in two days, which is terrible.”
  - This is a negative context.
- Statistical test to find +ve and –ve candidates.

$$Z = \frac{P - P_0}{\sqrt{\frac{P_0(1 - P_0)}{n}}}$$

- Pruning to move those unlikely ones though *sentiment homogeneity*.

# Pruning

- For an aspect with an implied opinion, it has a fixed opinion, either +ve or -ve, but not both.
- We find two direct modification relations using a dependency parser.
  - Type 1:  $O \rightarrow O\text{-}Dep \rightarrow A$ 
    - e.g. “*This TV has **good** picture quality.*”
  - Type 2:  $O \rightarrow O\text{-}Dep \rightarrow H \leftarrow A\text{-}Dep \leftarrow A$ 
    - e.g. “*The **springs** of the mattress **are bad**.*”
- If an aspect has mixed opinions based on the two dependency relations, prune it.

# Opinions implied by resource usage

(Zhang and Liu, 2011b)

- Resource usage descriptions may also imply opinions (as mentioned in rules of opinions)
  - E.g., “This washer uses a lot of water.”
- Two key roles played by resources usage:
  - An important aspect of an entity, e.g., water usage.
  - Imply a positive or negative opinion
- Resource usages that imply opinions can often be described by a triple.  
(verb, quantifier, noun\_term),
  - Verb: uses, quantifier: “a lot of”, noun\_term: water

# The proposed technique

- The proposed method is graph-based.
  - Stage 1: Identifying Some Global Resource Verbs
    - Identify and score common resource usage verbs used in almost any domain, e.g., “use” and “consume”
  - Stage 2: Discovering Resource Terms in each Domain Corpus
    - Use a graph-based method considering occurrence probabilities.
    - With resource verbs identified from stage 1 as the seeds.
    - Score domain specific resource usage verbs and resource terms.

# Dictionary-based methods

- Typically use WordNet's synsets and hierarchies to acquire opinion words
  - Start with a small seed set of opinion words.
  - Bootstrap the set to search for synonyms and antonyms in WordNet iteratively (Hu and Liu, 2004; Kim and Hovy, 2004; Kamps et al 2004).
- Use additional information (e.g., glosses) from WordNet (Andreevskaia and Bergler, 2006) and learning (Esuti and Sebastiani, 2005). (Dragut et al 2010) uses a set of rules to infer orientations.

# Semi-supervised learning

(Esuti and Sebastiani, 2005)

- Use supervised learning
  - Given two seed sets: positive set P, negative set N
  - The two seed sets are then expanded using synonym and antonymy relations in an online dictionary to generate the expanded sets P' and N'.
- P' and N' form the training sets.
- Using all the glosses in a dictionary for each term in  $P' \cup N'$  and converting them to a vector
- Build a binary classifier
  - Tried various learners.

# Multiple runs of bootstrapping

(Andreevskaia and Bergler, 2006)

- Basic bootstrapping with given seeds sets (adjectives)
  - First pass: seed sets are expanded using synonym, antonymy, and hyponyms relations in WordNet.
  - Second pass: it goes through all WordNet glosses and identifies the entries that contain in their definitions the sentiment-bearing words from the extended seed set and adds these head words to the corresponding category (+ve, -ve, neutral)
  - Third pass: clean up using a POS tagger to make sure the words are adjectives and remove contradictions.

# Multiple runs of bootstrapping (contd)

- Each word is then assigned a fuzzy score reflecting the degree of certainty that the word is opinionated (+ve/-ve).
- The method performs multiple runs of bootstrapping using non-overlapping seed sets.
  - A net overlapping score for each word is computed based on how many times the word is discovered in the runs as +ve (or -ve)
  - The score is normalized based on the fuzzy membership.

# Which approach to use?

- Both corpus and dictionary based approaches are needed.
- Dictionary usually does not give domain or context dependent meaning
  - Corpus is needed for that
- Corpus-based approach is hard to find a very large set of opinion words
  - Dictionary is good for that
- In practice, corpus, dictionary and manual approaches are all needed.

# Some other related papers

- Choi and Cardie (2009) adapting a lexicon to domain specific need using integer linear programming
- Du and Tan (2009) and Du, Tan, Cheng and Yun (2010) clustered sentiment words
- Hassan and Radev (2010) built a word graph based on synonyms and then used a number of random walks to hit known seed words
- Hassan et al. (2011) found sentiment orientations of foreign words. It first created a multilingual word network and then did random walk similar to the above paper.
- Jijkoun, Rijke and Weerkamp (2010) used target and sentiment word relationship. Similar to that in (Qiu et al 2009).

# Some other related papers

- Kaji and Kitsuregawa (2006, 2007) and Velikovich et al (2010) used text on the web to generate lexicons.
- Lu et al (2011) dealt with the same problem as (Ding et al 2008) but used various constraints in optimization.
- Mohammad, Dunne, and Dorr, (2009) used seeds and thesaurus.
- Rao and Ravichandran (2009) used WordNet and OpenOffice thesaurus and semi-supervised learning
- Wu and Wen (2010) found context adjectives like *large* and *small* by mining the web using lexico-syntactic patterns. They solved the same problem as (Ding et al 2008)

# Roadmap

- Opinion Mining Problem
- Document sentiment classification
- Sentence subjectivity & sentiment classification
- Aspect-based sentiment analysis
- Aspect-based opinion summarization
- Opinion lexicon generation
- **Mining comparative opinions**
- Some other problems
- Opinion spam detection
- Utility or helpfulness of reviews
- Summary

# Comparative Opinions

(Jindal and Liu, 2006)

## ■ *Gradable*

- *Non-Equal Gradable*: Relations of the type *greater or less than*
  - Ex: “*optics of camera A is better than that of camera B*”
- *Equative*: Relations of the type *equal to*
  - Ex: “*camera A and camera B both come in 7MP*”
- *Superlative*: Relations of the type *greater or less than all others*
  - Ex: “*camera A is the cheapest in market*”

# Analyzing Comparative Opinions

- **Objective:** Given an opinionated document  $d$ ,  
**Extract comparative opinions:**  
 $(E_1, E_2, A, po, h, t)$ ,  
where  $E_1$  and  $E_2$  are the entity sets being  
compared based on their shared aspects  $A$ ,  $po$  is  
the preferred entity set of the opinion holder  $h$ ,  
and  $t$  is the time when the comparative opinion is  
expressed.
- **Note:** not positive or negative opinions.

# An example

- Consider the comparative sentence
  - “*Canon’s optics is better than those of Sony and Nikon.*”
  - Written by John in 2010.
- The extracted comparative opinion/relation:
  - ( $\{\text{Canon}\}$ ,  $\{\text{Sony, Nikon}\}$ ,  $\{\text{optics}\}$ ,  
 $\text{preferred}:\{\text{Canon}\}$ , John, 2010)

# Common comparatives

- In English, comparatives are usually formed by adding *-er* and superlatives are formed by adding *-est* to their **base adjectives** and **adverbs**
- Adjectives and adverbs with two syllables or more and not ending in *y* do not form comparatives or superlatives by adding *-er* or *-est*.
  - Instead, *more*, *most*, *less*, and *least* are used before such words, e.g., *more beautiful*.
- Irregular comparatives and superlatives, i.e., *more*, *most*, *less*, *least*, *better*, *best*, *worse*, *worst*, etc

# Some techniques (Jindal and Liu, 2006, Ding et al, 2009)

- Identify comparative sentences
  - Using class sequential rules as attributes in the data, and then perform
  - Supervised learning
- Extraction of different items
  - Label sequential rules
  - conditional random fields
- Determine preferred entities (opinions)
  - Parsing and opinion lexicon

# Analysis of comparative opinions

- Grable comparative sentences can be dealt with *almost* as normal opinion sentences.
  - E.g., “*optics of camera A is better than that of camera B*”
  - Positive: “*optics of camera A*”
  - Negative: “*optics of camera B*”
- Difficulty: recognize non-standard comparatives
  - E.g., “I am so happy because my new iPhone is nothing like my old slow ugly Droid.”
  - ?

# Identifying preferred entities

(Ganapathibhotla and Liu, 2008)

## ■ The following rules can be applied

Comparative Negative ::= increasing comparative N

| decreasing comparative P

Comparative Positive ::= increasing comparative P

| decreasing comparative N

- E.g., “Coke tastes better than Pepsi”
- “Nokia phone’s battery life is longer than Moto phone”

## ■ Context-dependent comparative opinion words

- Using context pair: (aspect, JJ/JJR)
- Deciding the polarity of (battery\_life, longer) in a corpus

# Some other work

- Bos and Nissim (2006) proposed a method to extract items from superlative sentences, but does not study sentiments.
- Fiszman et al (2007) tried to identify which entity has more of a certain property in comparisons.
- Li et al (2010) finds comparative questions and compared entities using sequence patterns.
- Yang and Ko (2009, 2011) worked on Korean comparative sentences.
- Zhang, Narayanan and Choudhary (2010) found comparative sentences based on a set of rules, and the sentences must also mention at least two product names explicitly or implicitly (comparing with the product being reviewed).

# Roadmap

- Opinion Mining Problem
- Document sentiment classification
- Sentence subjectivity & sentiment classification
- Aspect-based sentiment analysis
- Aspect-based opinion summarization
- Opinion lexicon generation
- Mining comparative opinions
- ➡ ■ **Some other problems**
  - Opinion spam detection
  - Utility or helpfulness of reviews
  - Summary

# Coreference resolution: semantic level?

- Coreference resolution (Ding and Liu, 2010)
  - “I bought the Sharp tv a month ago. The picture quality is so bad. Our other Sony tv is much better than this Sharp. ***It* is also so expensive**”.
    - “it” means “Sharp”
  - “I bought the Sharp tv a month ago. The picture quality is so bad. Our other Sony tv is much better than this Sharp. ***It* is also very reliable.**”
    - “it” means “Sony”
- Sentiment consistency.

# Coreference resolution (contd)

- “The picture quality of this Canon camera is very good. ***It*** is not expensive either.”
  - Does “it” mean “Canon camera” or “Picture Quality”?
    - Clearly it is Canon camera because picture quality cannot be expensive.
    - Commonsense knowledge, but can be discovered.
- For coreference resolution, we actually need to
  - do sentiment analysis first, and
  - mine adjective-noun associations using dependency
- Finally, use supervised learning

# Find evaluative opinions in discussions

(Zhai et al. 2011c)

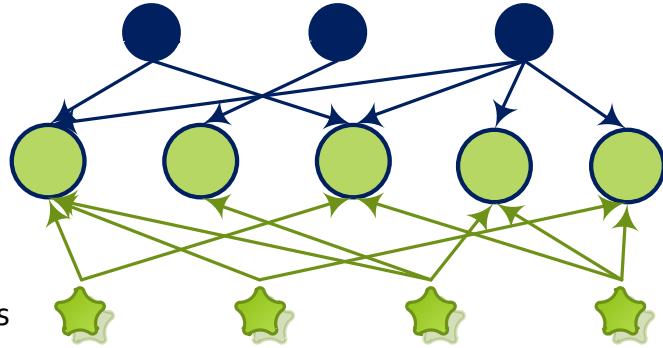
- Existing research focuses on product reviews
  - reviews are opinion-rich and
  - contain little irrelevant information.
- Not true for online discussions.
  - Many postings express no opinions about the topic, but emotional statements and others.
  - **Evaluative opinions**, “*The German defense is strong.*”
  - **Non-evaluative opinions**, “*I feel so sad for Argentina.*”  
“*you know nothing about defense*”
- **Goal:** identify evaluative opinion sentences.

# Aspects, evaluation words and emotion words interaction

Emotion words

Aspects

Evaluation words



$$asp(a_i) = \lambda \times \sum_{(i,j) \in E_{va-a}} eva(va_j) - (1 - \lambda) \times \sum_{(i,k) \in E_{mo-a}} emo(mo_k)$$

$$eva(va_j) = \sum_{(i,j) \in E_{va-a}} asp(a_i)$$

- ❖ An extracted aspect that is modified by many *evaluation words* is more likely to indicate an evaluative sentence.
- ❖ An extracted aspect that is modified by many *emotion words* is not a good indicator of an evaluative sentence.
- ❖ An evaluation word that does not modify *good* (high scored) aspects are likely to be a wrong evaluation word.
- ❖ The more evaluative the aspects are, the less emotional their associated emotion words should be.

$$(1) \quad (2)$$

$$tmp(mo_k) = \sum_{(i,k) \in E_{mo-a}} asp(a_i) \quad (3)$$

$$emo(mo_k) \propto -tmp(mo_k) \quad (4)$$

$$emo(mo_k) = -tmp(mo_k) + max = max - tmp(mo_k) \quad (5)$$

$$max = \max\{tmp(mo_1), tmp(mo_2), \dots, tmp(mo_{|V_{mo}|})\} \quad (6)$$

# Some interesting sentences

- “Trying out Google chrome because Firefox keeps crashing.”
  - The opinion about Firefox is clearly negative, but for Google chrome, there is no opinion.
  - We need to segment the sentence into clauses to decide that “crashing” only applies to Firefox.
  - “Trying out” also indicates no opinion.
- How about this
  - “I changed to Audi because BMW is so expensive.”

# Some interesting sentences (contd)

- Conditional sentences are hard to deal with  
(Narayanan et al. 2009)
  - “If I can find a good camera, I will buy it.”
  - But conditional sentences can have opinions
    - “If you are looking for a good phone, buy Nokia”
- Questions may or may not have opinions
  - No sentiment
    - “Are there any great perks for employees?”
  - With sentiment
    - “Any idea how to repair this lousy Sony camera?”

# Some interesting sentences (contd)

## ■ Sarcastic sentences

- “What a great car, it stopped working in the second day.”

## ■ Sarcastic sentences are very common in political blogs, comments and discussions.

- They make political blogs difficult to handle
- Many political aspects can also be quite complex and hard to extract because they cannot be described using one or two words.

## ■ Some initial work by (Tsur, Davidov, Rappoport 2010)

# Some interesting sentences (contd)

- See these two sentences in a medical domain:
  - “I come to see my doctor because of severe pain in my stomach”
  - “After taking the drug, I got severe pain in my stomach”
- If we are interested in opinions on a drug, the first sentence has no opinion, but the second implies negative opinion on the drug.
  - Some understanding seems to be needed?

# Some interesting sentences (contd)

- The following two sentences are from reviews in the paint domain.
  - “For paint\_X, one coat can cover the wood color.”
  - “For paint\_Y, we need three coats to cover the wood color.
- We know that paint\_X is good and Paint\_Y is not, but how by a system.
  - Do we need commonsense knowledge and understanding of the text?

# Some more interesting/hard sentences

- “My goal is to have a high quality tv with decent sound”
- “The top of the picture was much brighter than the bottom.”
- “Google steals ideas from Bing, Bing steals market shares from Google.”
- “When I first got the airbed a couple of weeks ago it was wonderful as all new things are, however as the weeks progressed I liked it less and less.”

# Roadmap

- Opinion Mining Problem
- Document sentiment classification
- Sentence subjectivity & sentiment classification
- Aspect-based sentiment analysis
- Aspect-based opinion summarization
- Opinion lexicon generation
- Mining comparative opinions
- Some other problems
- ■ **Opinion spam detection**
- Utility or helpfulness of reviews
- Summary

# Opinion spam detection

(Jindal and Liu 2007, 2008)

- Opinion spamming refers to people giving fake or untruthful opinions, e.g.,
  - Write undeserving positive reviews for some target entities in order to promote them.
  - Write unfair or malicious negative reviews for some target entities in order to damage their reputations.
- Opinion spamming has become a business in recent years.
- Increasing number of customers are wary of fake reviews (biased reviews, paid reviews)

# Problem is wide-spread

## Professional Fake Review Writing Services

- [Post positive reviews](#)
- [Fake review writer](#)
- [Product review writer for hire](#)
- [Hire a content writer](#)

## Manipulating Social Media (sock puppets - fake identities - fake personas)

- [Revealed: US spy operation that manipulates social media](#), Guardian.co.uk, Thursday 17 March 2011.
- [America's absurd stab at systematising sock puppetry](#), Guardian.co.uk, Thursday 17 March 2011.

## China's Internet "Water Army" (Shuijun) - Opinion Spammers

- You can hire people to write and post fake reviews or comments, and even bribe staff at review, forum
- ['Water Army' Whistleblower Threatened](#), January 7, 2011, People's Daily.
- [The Chinese Online "Water Army"](#), June 25, 2010, Wired.com.
- If you read Chinese, see [this description](#) from Baidu Baike at baidu.com.

# An example practice of review spam

## Belkin International, Inc

- Top networking and peripherals manufacturer | Sales ~ \$500 million in 2008
- Posted an ad for writing fake reviews on amazon.com (65 cents per review)

The image shows a screenshot of a Mechanical Turk HIT interface. At the top, there is a timer indicating "00:00:00 of 60 minutes". To the right are buttons for "Accept HIT" and "Skip HIT". Below the timer, the task description is: "Write Product Reviews 25-50 Words". It specifies the requester as "Mike Bayard" and lists "Qualifications Required: HIT approval rate (%) is not less than 95". A red annotation "Jan 2009" is placed next to the requester's name. The main task area has a heading "Write a Positive 5/5 Review for Product on Website" and instructions: "Positive review writing." followed by a bulleted list of guidelines. Below the guidelines is a box containing detailed instructions about reading the product features and writing a review.

Timer: 00:00:00 of 60 minutes

Want to work on this HIT? **Accept HIT** Want to see other HITs? **Skip HIT**

Write Product Reviews 25-50 Words

Requester: Mike Bayard

Qualifications Required: HIT approval rate (%) is not less than 95

Jan 2009

### Write a Positive 5/5 Review for Product on Website

Positive review writing.

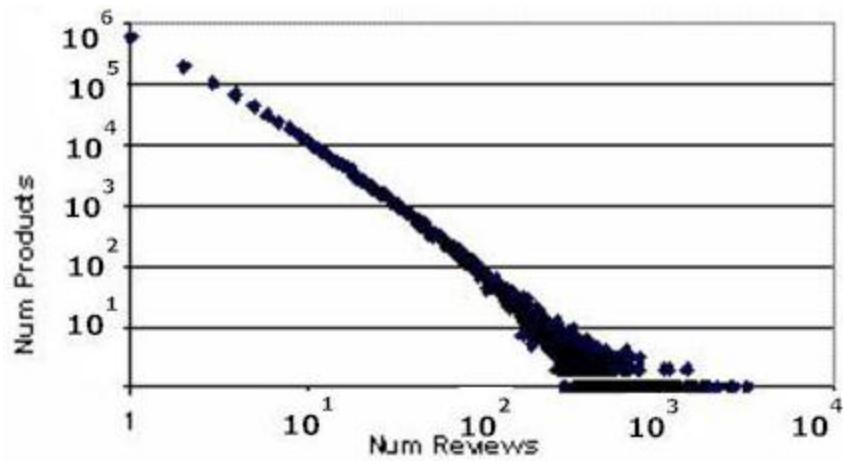
- Use your best possible grammar and write in US English only
- Always give a 100% rating (as high as possible)
- Keep your entry between 25 and 50 words
- Write as if you own the product and are using it
- Tell a story of why you bought it and how you are using it
- Thank the website for making you such a great deal
- Mark any other negative reviews as "not helpful" once you post yours

Instructions:

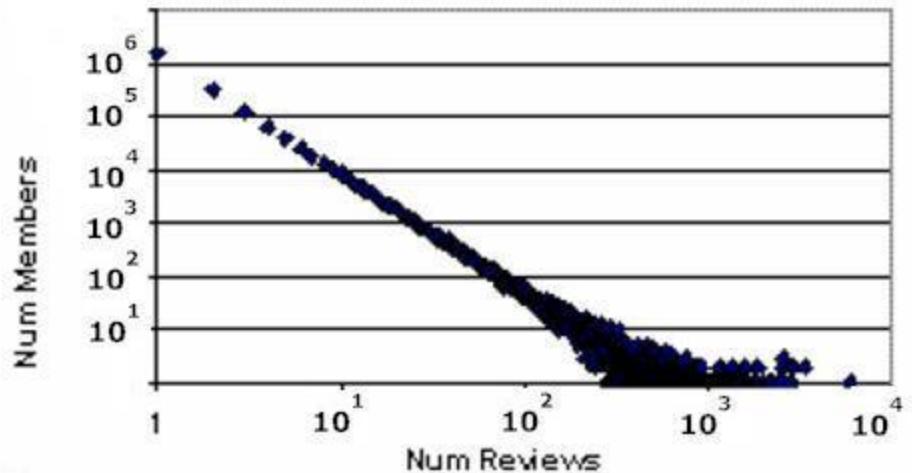
The link below leads to a product on a website. Read-through the product's features and write a positive review for it using the guidelines above to the best of your ability. I have also provided the part number for this product and you can click on the links below to see it on several alternative websites. In order to post some reviews you will need to create an account on the site. You can use your own email address or open a new free webmail account (gmail, yahoo...) and use it to post with.

# Log-log plot

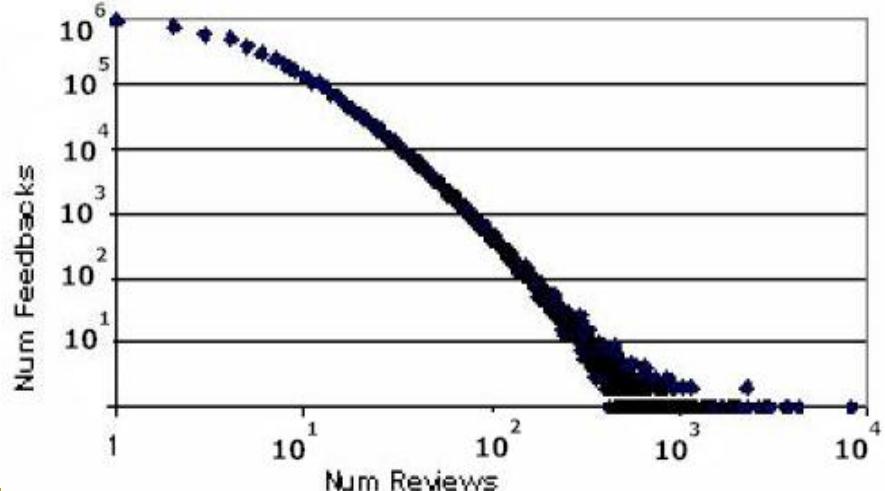
## Amazon reviews, reviewers and products



■ Fig. 2 reviews and products



■ Fig. 1 reviews and reviewers



■ Fig. 3 reviews and feedbacks

# Categorization of opinion spam

(Jindal and Liu 2008)

- Type 1 (fake reviews)

Ex:

- Type 2 (Reviews on Brands Only) (?)

Ex: “*I don’t trust HP and never bought anything from them*”

- Type 3 (Non-reviews)

- Advertisements

Ex: “*Detailed product specs: 802.11g, IMR compliant, ...*”

“*...buy this product at: compuplus.com*”

- Other non-reviews

Ex: “*What port is it for*”

“*The other review is too funny*”

“*Go Eagles go*”

# Type 1 Spam Reviews

- Hype spam – promote one's own products
- Defaming spam – defame one's competitors' products

**Table 4. Spam reviews vs. product quality**

	Positive spam review	Negative spam review
Good quality product	1	2
Bad quality product	3	4
Average quality product	5	6

The diagram shows a 3x2 grid representing spam reviews versus product quality. The columns are labeled 'Positive spam review' and 'Negative spam review'. The rows are labeled by product quality: 'Good quality product', 'Bad quality product', and 'Average quality product'. The cells contain numbers 1 through 6. Arrows point from the bottom-right corner of the grid (containing number 6) to the text 'Harmful Regions' located below the table.

- Very hard to detect manually

Harmful Regions

# Harmful spam are outlier reviews?

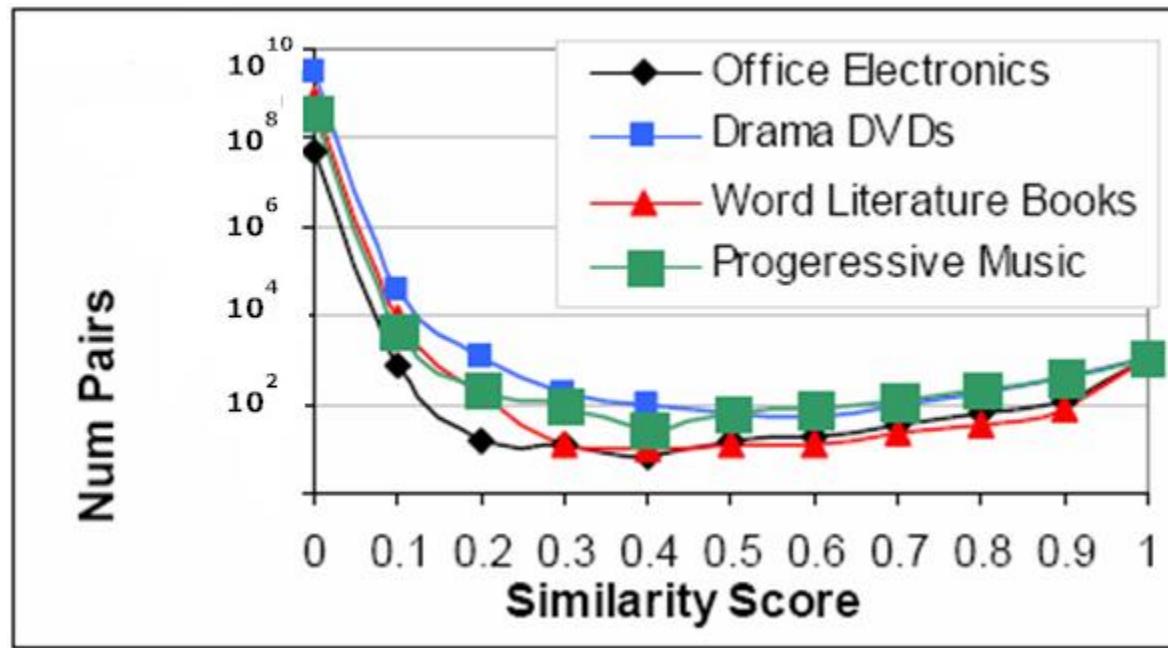
- **Assumption:** Most reviewers and reviews are honest,
  - Not true when a group of people spam on a product (called group spam, discussed later).
- **Outliers reviews:** Reviews which deviate a great deal from the average product rating
- **Harmful spam reviews:**
  - Outliers are necessary but not sufficient condition for harmful spam reviews.
  - This idea helps us identify learning features.

# Spam detection

- Type 2 and Type 3 spam reviews are relatively easy to detect
  - Supervised learning, e.g., logistic regression
  - It performs quite well, and not discuss it further.
- Type 1 spam (fake) reviews
  - **Manual labeling is extremely hard**
  - Propose to use duplicate and near-duplicate reviews as positive training data

# Duplicate reviews

Two reviews which have similar contents are called duplicates



# Four types of duplicates

1. Same userid, same product
  2. Different userid, same product
  3. Same userid, different products
  4. Different userid, different products
- The last three types are very likely to be fake!

# Supervised model building

- Logistic regression
  - Training: duplicates as spam reviews (positive) and the rest as non-spam reviews (negative)
- Use the follow data attributes
  - Review centric features (content)
    - About reviews (contents (n-gram), ratings, etc)
  - Reviewer centric features
    - About reviewers (different unusual behaviors, etc)
  - Product centric features
    - Features about products reviewed (sale rank, etc)

# Predictive power of duplicates

- Representative of all kinds of spam
- Only 3% duplicates accidental
- Duplicates as positive examples, rest of the reviews as negative examples

**Table 5.** AUC values on duplicate spam reviews.

Features used	AUC
All features	78%
Only review features	75%
Only reviewer features	72.5%
Without feedback features	77%
Only text features	63%

- reasonable predictive power
- **Maybe we can use duplicates as type 1 spam reviews(?)**

# Tentative classification results

- Negative outlier reviews tend to be heavily spammed
- Those reviews that are the only reviews of products are likely to be spammed
- Top-ranked reviewers are more likely to be spammers
- Spam reviews can get good helpful feedbacks and non-spam reviews can get bad feedbacks

# Detecting deceptive reviews (Ott et al 2011)

- Detecting deceptive language has been studied in psychology, communication and linguistics (Newman et al 2003; Zhou, Shi and Zhang 2008; Mihalcea and Strapparava 2009).
- Ott et al (2011) used the idea to detect deceptive opinion spam reviews with supervised learning.
  - Manually labeled a dataset
  - Various features on genre, psycholinguistic, n-grams
- Yoo and Gretzel (2009) also studied deceptive reviews.

# Finding unexpected reviewer behavior

- Since in general it is hard to manually label spam review for learning, it is thus difficult to detect fake reviews based on review contents.
- Lim et al (2010) and Nitin et al (2010) analyze the behavior of reviewers
  - identifying *unusual review patterns* which may indicate suspicious behaviors of reviewers.
- The problem is formulated as finding **unexpected rules and rule groups**.

# Spam behavior models (Lim et al 2010)

- Several unusual reviewer behavior models were identified.
  - Targeting products
  - Targeting groups
  - General rating deviation
  - Early rating deviation
- Their scores for each reviewer are then combined to produce the final spam score.
- Ranking and user evaluation

# Finding unexpected rules (Jindal, Liu, Lim 2010)

- For example, if a reviewer wrote all positive reviews on products of a brand but all negative reviews on a competing brand ...
- Finding unexpected rules,
  - Data: *reviewer-id*, *brand-id*, *product-id*, and a *class*.
  - Mining: class association rule mining
  - Finding unexpected rules and rule groups, i.e., showing atypical behaviors of reviewers.

Rule1: Reviewer-1, brand-1 -> positive (confid=100%)

Rule2: Reviewer-1, brand-2 -> negative (confid=100%)

# The example (cont.)

**Expectation:** Let the subset of data with  $A_j = v_{jk}$  be  $D^{jk}$ . We have

$$E(v_{jk}, A_g \rightarrow C) = \text{entropy}(D^{jk}) \quad (24)$$

**Attribute unexpectedness:** To compute attribute unexpectedness, we first compute the entropy after adding the  $A_g$  attribute:

$$\text{entropy}_{A_g}(D^{jk}) = -\sum_{h=1}^{|A_g|} \frac{|D^{jk}_h|}{|D^v|} \text{entropy}(D^{jk}_h) \quad (25)$$

The unexpectedness is computed as follows (information gain):

$$Au(v_{jk}, A_g \rightarrow C) = \text{entropy}(D^{jk}) - \text{entropy}_{A_g}(D^{jk}) \quad (26)$$

# Confidence unexpectedness

Rule: reviewer-1, brand-1 → positive [sup = 0.1, conf = 1]

- If we find that on average reviewers give brand-1 only 20% positive reviews (expectation), then reviewer-1 is quite unexpected.

$$Cu(v_{jk} \rightarrow c_i) = \frac{\Pr(c_i | v_{jk}) - E(\Pr(c_i | v_{jk}))}{E(\Pr(c_i | v_{jk}))}$$

$$E(\Pr(c_i | v_{jk}, v_{gh})) = \frac{\Pr(c_i | v_{jk}) \Pr(c_i | v_{gh})}{\Pr(c_i) \sum_{r=1}^m \frac{\Pr(c_r | v_{jk}) \Pr(c_r | v_{gh})}{\Pr(c_r)}}$$

# Support unexpectedness

Rule: reviewer-1, product-1 -> positive [sup = 5]

- Each reviewer should write only one review on a product and give it a positive or negative rating (expectation).
- This unexpectedness can detect those reviewers who review the same product multiple times, which is unexpected.
  - These reviewers are likely to be spammers.
- Can be defined probabilistically as well.

# Detecting group spam (Mukherjee et al 2011)

- A group of people (could be a single person with multiple ids, *sockpuppets*) work together to promote a product or to demote a product.
- Such spam can be very harmful as
  - they can take control of sentiment on a product
- The algorithm has three steps
  - Frequent pattern mining: find groups of people who reviewed a number of products together.
  - A set of feature indicators are identified
  - Ranking is performed with a learning to rank algo.

# Roadmap

- Opinion Mining Problem
- Document sentiment classification
- Sentence subjectivity & sentiment classification
- Aspect-based sentiment analysis
- Aspect-based opinion summarization
- Opinion lexicon generation
- Mining comparative opinions
- Some other problems
- Opinion spam detection
- ➡ ■ **Utility or helpfulness of reviews**
- Summary

# Utility or quality of reviews

- **Goal:** Determining the usefulness, helpfulness, or utility of each review.
  - It is desirable to rank reviews based on utilities or qualities when showing them to users, with the highest quality review first.
- Many review aggregation sites have been practicing this, e.g., amazon.com.
  - “*x of y people found the following review helpful.*”
  - Voted by user - “*Was the review helpful to you?*”

# Application motivations

- Although review sites use helpfulness feedback to rank,
  - A review takes a long time to gather enough feedback.
    - New reviews will not be read.
  - Some sites do not provide feedback information.
- It is thus beneficial to score each review once it is submitted to a site.

# Regression formulation

(Zhang and Varadarajan, 2006; Kim et al. 2006)

- **Formulation:** Determining the utility of reviews is usually treated as a **regression** problem.
  - A set of features is engineered for model building
  - The learned model assigns an utility score to each review, which can be used in review ranking.
- Unlike fake reviews, the ground truth data used for both training and testing are available
  - Usually the user-helpfulness feedback given to each review.

# Features for regression learning

- Example features include
  - review length, review rating, counts of some POS tags, opinion words, tf-idf scores, wh-words, product aspect mentions, comparison with product specifications, timeliness, etc (Zhang and Varadarajan, 2006; Kim et al. 2006; Ghose and Ipeirotis 2007; Liu et al 2007)
- Subjectivity classification was applied in (Ghose and Ipeirotis 2007).
- Social context was used in (O'Mahony and Smyth 2009; Lu et al. 2010).

# Classification formulation

- **Binary classification:** Instead of using the original helpfulness feedback as the target or dependent variable,
  - Liu et al (2007) performed manual annotation of two classes based on whether the review evaluates many product aspects or not.
- **Binary class classification** is also used in (O'Mahony and Smyth 2009)
  - Classes: Helpful and not helpful
  - Features: helpfulness, content, social, and opinion

# Roadmap

- Opinion Mining Problem
- Document sentiment classification
- Sentence subjectivity & sentiment classification
- Aspect-based sentiment analysis
- Aspect-based opinion summarization
- Opinion lexicon generation
- Mining comparative opinions
- Some other problems
- Opinion spam detection
- Utility or helpfulness of reviews
- **Summary**

# Summary

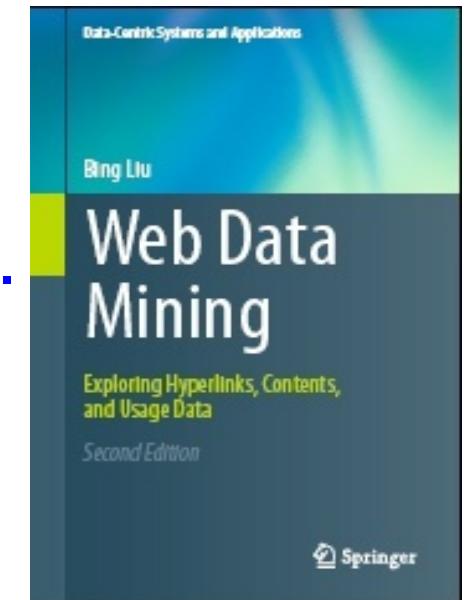
- **This tutorial presented**
  - The problem of sentiment analysis and opinion mining
    - It provides a structure to the unstructured text.
    - It shows that summarization is crucial.
  - Main research directions and their representative techniques.
  - By no means exhaustive, a large body of work.
- Still many problems not attempted or studied.
- **None of the problem is solved.**

# Summary (contd)

- It is a fascinating NLP or text mining problem.
  - Every sub-problem is highly challenging.
  - But it is also highly restricted (semantically).
- Despite the challenges, applications are flourishing!
  - It is useful to every organization and individual.
- The general NLP is probably too hard, but can we solve this highly restricted problem?
  - I am optimistic

# References

- The references will be available from this link
  - <http://www.cs.uic.edu/~liub/FBS/AAAI-2011-tutorial-references.pdf>
- The main content is from Chapter 11 of the following book,
  - B. Liu. *Web Data Mining: Exploring Hyperlinks, Contents and Usage Data. Second Edition*, Springer, July 2011.
  - But updated based on 2011 papers.



# Why Teleports Solve the Problem?

$$r^{(t+1)} = Mr^{(t)}$$

## Markov chains

- Set of states  $X$
- Transition matrix  $P$  where  $P_{ij} = P(X_t=i \mid X_{t-1}=j)$
- $\pi$  specifying the stationary probability of being at each state  $x \in X$
- Goal is to find  $\pi$  such that  $\pi = P\pi$

# Why is This Analogy Useful?

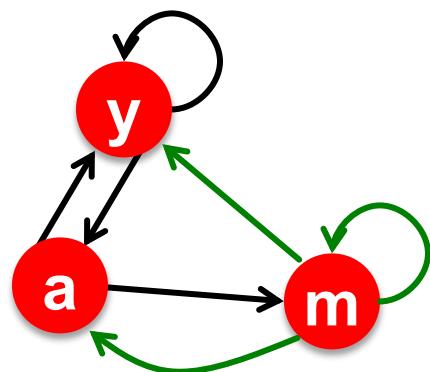
- **Theory of Markov chains**
- Fact: For **any start vector**, the power method applied to a Markov transition matrix  $P$  will **converge** to a **unique** positive stationary vector as long as  $P$  is **stochastic, irreducible** and **aperiodic**.

# Make M Stochastic

- **Stochastic:** Every column sums to 1
- **Solution:** Add **green** links

$$A = M + a^T \left( \frac{1}{n} e \right)$$

- $a_i = 1$  if node  $i$  has out deg 0, =0 else
- $e$  vector of all 1s



	y	a	m
y	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{3}$
a	$\frac{1}{2}$	0	$\frac{1}{3}$
m	0	$\frac{1}{2}$	$\frac{1}{3}$

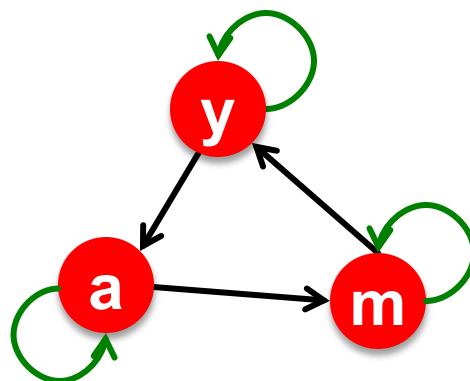
$$\mathbf{r}_y = \mathbf{r}_y/2 + \mathbf{r}_a/2 + \mathbf{r}_m/3$$

$$\mathbf{r}_a = \mathbf{r}_y/2 + \mathbf{r}_m/3$$

$$\mathbf{r}_m = \mathbf{r}_a/2 + \mathbf{r}_m/3$$

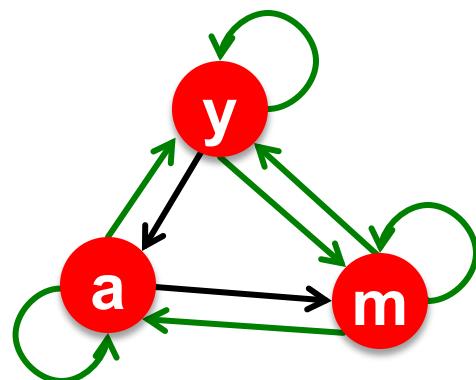
# Make M Aperiodic

- A chain is **periodic** if there exists  $k > 1$  such that the interval between two visits to some state  $s$  is always a multiple of  $k$
- **Solution:** Add **green** links



# Make $M$ Irreducible

- From any state, there is a non-zero probability of going from any one state to any another
- Solution:** Add green links



# Solution: Random Jumps

- Google's solution that does it all:
  - Makes  $M$  stochastic, aperiodic, irreducible
- At each step, random surfer has two options:
  - With probability  $\beta$ , follow a link at random
  - With probability  $1-\beta$ , jump to some random page
- PageRank equation [Brin-Page, 98]

$$r_j = \sum_{i \rightarrow j} \beta \frac{r_i}{d_i} + (1 - \beta) \frac{1}{n}$$

$d_i$  ... out-degree  
of node i

The above formulation assumes that  $M$  has no dead ends. We can either preprocess matrix  $M$  (bad!) or explicitly follow random teleport links with probability 1.0 from dead-ends.

# The Google Matrix

- **PageRank equation** [Brin-Page, 98]

$$r_j = \sum_{i \rightarrow j} \beta \frac{r_i}{d_i} + (1 - \beta) \frac{1}{n}$$

- **The Google Matrix  $A$ :**

$$A = \beta M + (1 - \beta) \frac{1}{n} \mathbf{e} \cdot \mathbf{e}^T$$

$\mathbf{e}$  vector of all 1s

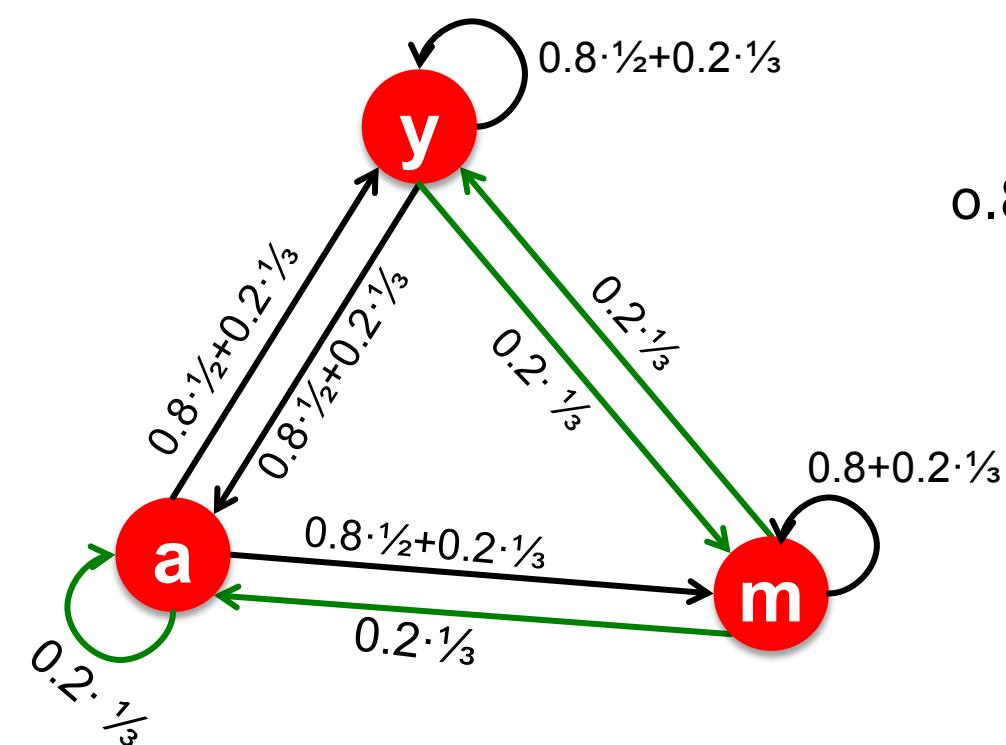
- **$A$  is stochastic, aperiodic and irreducible, so**

$$\mathbf{r}^{(t+1)} = A \cdot \mathbf{r}^{(t)}$$

- **What is  $\beta$ ?**

- In practice  $\beta = 0.8, 0.9$  (make 5 steps and jump)

# Random Teleports ( $\beta = 0.8$ )



$$\begin{array}{c}
 M \\
 \begin{array}{|ccc|} \hline & 1/2 & 1/2 & 0 \\ & 1/2 & 0 & 0 \\ & 0 & 1/2 & 1 \\ \hline \end{array} \\
 + 0.2 \quad \begin{array}{|ccc|} \hline & 1/3 & 1/3 & 1/3 \\ & 1/3 & 1/3 & 1/3 \\ & 1/3 & 1/3 & 1/3 \\ \hline \end{array} \\
 \begin{array}{c} y \\ a \\ m \end{array} \quad \begin{array}{|ccc|} \hline & 7/15 & 7/15 & 1/15 \\ & 7/15 & 1/15 & 1/15 \\ & 1/15 & 7/15 & 13/15 \\ \hline \end{array} \\
 A
 \end{array}$$

y	1/3	0.33	0.24	0.26	...	7/33
a	1/3	0.20	0.20	0.18	...	5/33
m	1/3	0.46	0.52	0.56		21/33