

# Graph and Web Mining - Motivation, Applications and Algorithms



---

Prof. Ehud Gudes

Department of Computer Science

Ben-Gurion University, Israel

# Graph and Web Mining - Motivation, Applications and Algorithms



---

Co-Authors: Natalia Vanetik, Moti Cohen,  
Eyal Shimony

Some slides taken with thanks from:  
J. Han, X. Yan, P. Yu, G. Karypis



# General

---

Whereas data-mining in structured data focuses on frequent data values, in semi-structured and graph data mining, the structure of the data is just as important as its content.

We study the problem of discovering typical patterns of graph data. The discovered patterns can be useful for many applications, including: compact representation of the information, finding strongly connected groups in social networks and in several scientific domains like finding frequent molecular structures.

The discovery task is impacted by structural features of graph data in a non-trivial way, making traditional data mining approaches inapplicable. Difficulties result from the complexity of some of the required sub-tasks, such as graph and sub-graph isomorphism, which are hard problems.

This course will discuss first the motivation and applications of Graph mining, and then will survey in detail the common algorithms for this task, including: FSG, GSPAN and other recent algorithms by the Presentor. The last part of the course will deal with Web mining. Graph mining is central to web mining because the web links form a huge graph and mining its properties has a large significance.



# Course Outline

---

- Basic concepts of Data Mining and Association rules
  - Apriori algorithm
  - Sequence mining
- Motivation for Graph Mining
- Applications of Graph Mining
- Mining Frequent Subgraphs - Transactions
  - BFS/Apriori Approach (FSG and others)
  - DFS Approach (gSpan and others)
  - Diagonal and Greedy Approaches
  - Constraint-based mining and new algorithms
- Mining Frequent Subgraphs – Single graph
  - The support issue
  - The Path-based algorithm



# Course Outline ( Cont.)

---

- Searching Graphs and Related algorithms
  - Sub-graph isomorphism (Sub-sea)
  - Indexing and Searching – graph indexing
  - A new sequence mining algorithm
- Web mining and other applications
  - Document classification
  - Web mining
  - Short student presentation on their projects/papers
- Conclusions



# Important References

---

- [1] T. Washio and H. Motoda, "*State of the art of graph-based data mining*", SIGKDD Explorations, 5:59-68, 2003
- [2] X. Yan and J. Han, "*gSpan: Graph-Based Substructure Pattern Mining*", ICDM'02
- [3] X. Yan and J. Han, "*CloseGraph: Mining Closed Frequent Graph Patterns*", KDD'03
- [4] [5] M. Kuramochi, G. Karypis, "*An Efficient Algorithm for Discovering Frequent Subgraphs*" IEEE TKDE, September 2004 (vol. 16 no. 9)
- [5] N. Vanetik, E. Gudes, and S. E. Shimony, "*Computing Frequent Graph Patterns from Semistructured Data*", Proceedings of the 2002 IEEE ICDM'02
- [6] [4] X. Yan, P. S. Yu, and J. Han, "*Graph Indexing: A Frequent Structure-based Approach*", SIGMOD'04
- [7] J. Han and M. Kamber, "*Data mining – Concepts and Techniques*, 2<sup>nd</sup> Edition, Morgan Kaufmann Publishers, 2006
- [8] Bing Liu, "*Web Data Mining: Exploring Hyperlinks, Contents, and Usage Data*", Springer publishing, 2009



# Course Requirements

---

- The main requirement of this course (in addition to attending lectures) is a final project or a final paper to be submitted a month after the end of the course. In addition the students will be required to answer few homework questions.
- In the **final project** the students (mostly 2) will implement one of the studied graph mining algorithms and will test it on some public available data. In addition to the software , a report detailing the problem, algorithm, software structure and test results is expected.
- In the **final paper** the student(mostly 1) will review at least two recent papers in graph mining not presented in class and explain them in detail.
- Topics for projects and papers will be presented during the course. The last hour of the course will be dedicated for students for presenting their selected project/paper (about 8-10 mins. each )



# What is Data Mining?

---

*Data Mining*, also known as *Knowledge Discovery in Databases* (KDD), is the process of extracting useful hidden information from very large databases in an unsupervised manner.





# What is Data Mining?

---

There are many data mining methods including:

- Clustering and Classification
- Decision Trees
- Finding frequent patterns and Association rules



# Mining Frequent Patterns: What is it good for?

---

- **Frequent Pattern:** a pattern (a set of items, subsequences, substructures, etc.) that occurs frequently in a data set
- **Motivation:** Finding inherent regularities in data
  - What products were often purchased together?
  - What are the subsequent purchases after buying a PC?
  - What kinds of DNA are sensitive to this new drug?
  - Can we classify web documents using frequent patterns?



# What Is Association Mining?

---

- Finding regularities in Transactional DB
- Rules expressing relationships between items
- Example:

{diaper }  $\Rightarrow$  { beer }

{milk, tea}  $\Rightarrow$  {cookies}

<i>TID</i>	<i>items</i>
T001	diaper, milk, candy
T002	milk, egg
T003	milk, beer
T004	diaper, milk, egg
T005	diaper, beer
T006	milk, beer
T007	diaper, beer
T008	diaper, milk, beer, candy
T009	diaper, milk, beer
...	...



# Basic Concepts:

---

- Set of items  $I = \{i_1, i_2, \dots, i_m\}$

- Transaction  $T \subseteq I$

- Set of transactions (i.e., our data)

$$D = \{T_1, T_2, \dots, T_k\}$$

- Association rule

$$A \Rightarrow B \quad A, B \subset I \quad A \cap B = \emptyset$$

- Frequency function

$$\text{Frequency}(A, D) = |\{T \in D \mid A \subset T\}|$$



# Interestingness Measures

---

- Rules ( $A \Rightarrow B$ ) are included/excluded based on two metrics given by user
  - *Minimum support ( $0 < minSup < 1$ )*

How frequently all of the items in a rule appear in transactions
  - *Minimum confidence ( $0 < minConf < 1$ )*

How frequently the left hand side of a rule implies the right hand side



# Measuring Interesting Rules

---

- Support

- Ratio of # of transactions containing  $A$  and  $B$  to the total # of transactions

$$\text{support } (A \Rightarrow B) = \frac{\text{Frequency } (A \cup B, D)}{|D|}$$

- Confidence

- Ratio of # of transactions containing  $A$  and  $B$  to #of transactions containing  $A$

$$\text{confidence } (A \Rightarrow B) = \frac{\text{Frequency } (A \cup B, D)}{\text{Frequency } (A, D)}$$



# Frequent Itemsets

---

Given  $D$  and  $minSup$

- A set  $\alpha$  is *frequent itemset* if:

$$Frequency(\alpha, D) > minSup$$

- Suppose we know all frequent itemsets and their exact frequency in  $D$
- How then, can it help us find all associations rules?
- By computing the confidence of the various combinations of the two sides
- **Therefore the main problem: Finding frequent Itemsets (Patterns)!**

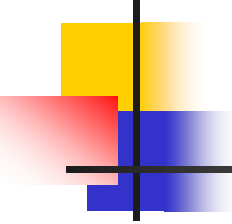


# Frequent Itemsets: A Naïve Algorithm

---

- First try
  - Keep a running count for each possible itemset
  - For each transaction  $T$ , and for each itemset  $X$ , if  $T$  contains  $X$  then increment the count for  $X$
  - Return itemsets with large enough counts
- Problem: The number of itemsets is huge!
  - Worst case:  $2^n$ , where  $n$  is the number of items





# The Apriori Principle: Downward Closure Property

---

- All subsets of a frequent itemset must also be frequent
  - Because any transaction that contains  $X$  must also contain any subset of  $X$
- If we have already verified that  $X$  is infrequent, there is no need to count  $X$  supersets because they must be infrequent too.

# Apriori Algorithm (Agrawal & Srikant, 1994)

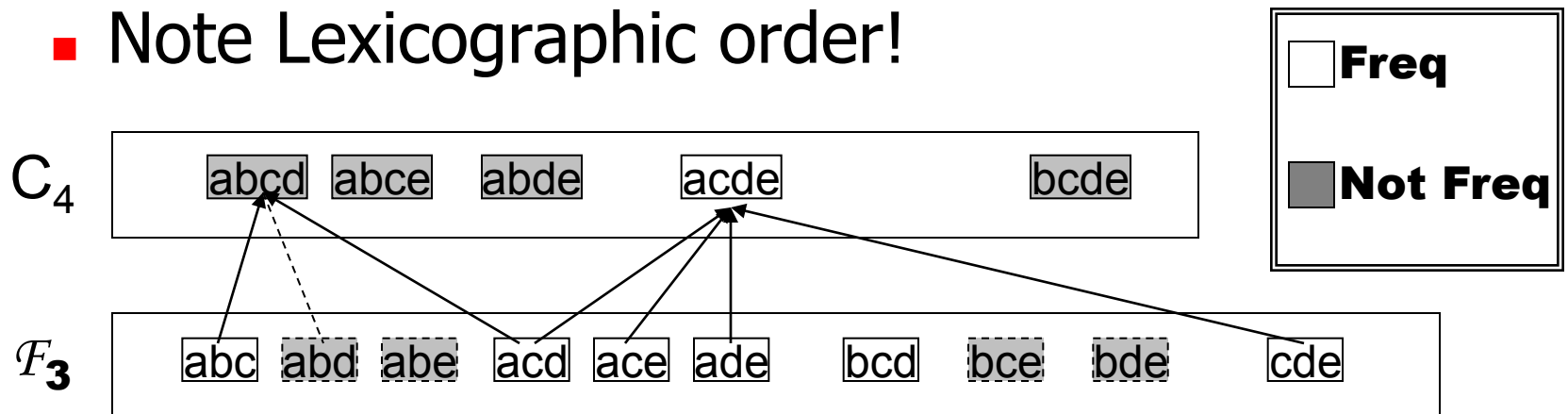
**Init:** Scan the transactions to find  $F_1$ , the set of all frequent 1-itemsets, together with their counts;

For ( $k=2$ ;  $F_{k-1} \neq \emptyset$  ;  $k++$ )

- 1) **Candidate Generation** -  $C_k$  the set of candidate  $k$ -itemsets, from  $F_{k-1}$ , the set of frequent  $(k-1)$ -itemsets found in the previous step
  - 2) **Candidates pruning** - a necessary condition of candidate to be frequent is that each of its  $(k-1)$ -itemset is frequent.
  - 3) **Frequency counting** - Scan the transactions to count the occurrences of itemsets in  $C_k$
  - 4)  $F_k = \{ c \in C_k \mid c \text{ has counts no less than } \#minSup \}$
- Return  $F_1 \cup F_2 \cup \dots \cup F_k$  ( $= F$ )

# Itemsets: Candidate Generation

- From  $\mathcal{F}_{k-1}$  to  $\mathcal{C}_k$ 
  - **Join**: combine frequent (k-1)-itemsets to form k-itemsets using a common core (of size k-2)
  - **Prune**: ensure every size (k-1) subset of a candidate is frequent
  - Note Lexicographic order!



# pass 1

**DB**

TID	items
T001	A, B, E
T002	B, D
T003	B, C
T004	A, B, D
T005	A, C
T006	B, C
T007	A, C
T008	A, B, C, E
T009	A, B, C
T010	F

**Transactions**

$F_1$

itemsets	count
{A}	6
{B}	7
{C}	6
{D}	2
{E}	2

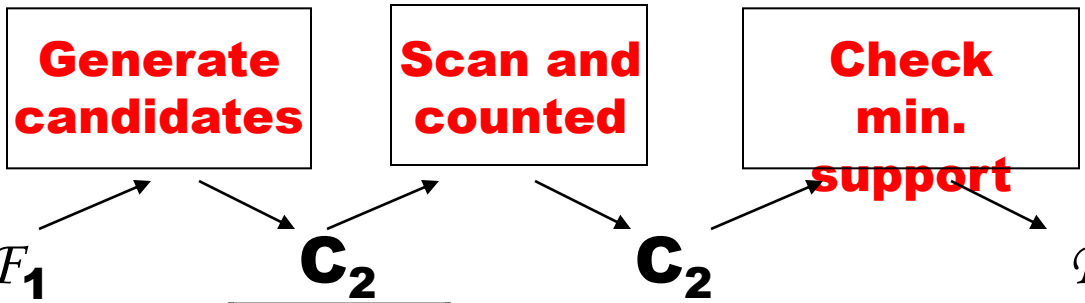
**Itemset {F} is infrequent**

**minSup = 20%**

**DB**

TID	items
T001	A, B, E
T002	B, D
T003	B, C
T004	A, B, D
T005	A, C
T006	B, C
T007	A, C
T008	A, B, C, E
T009	A, B, C
T010	F

**pass 2**



$F_1$

itemsets	count
{A}	6
{B}	7
{C}	6
{D}	2
{E}	2

$C_2$

itemsets	count
{A, B}	4
{A, C}	4
{A, D}	1
{A, E}	2
{B, C}	4
{B, D}	2
{B, E}	2
{C, D}	0
{C, E}	1
{D, E}	0

$C_2$

itemsets	count
{A, B}	4
{A, C}	4
{A, D}	1
{A, E}	2
{B, C}	4
{B, D}	2
{B, E}	2
{C, D}	0
{C, E}	1
{D, E}	0

$F_2$

itemsets	count
{A, B}	4
{A, C}	4
{A, E}	2
{B, C}	4
{B, D}	2
{B, E}	2

**minSup = 20%**

**DB**

TID	items
T001	A, B, E
T002	B, D
T003	B, C
T004	A, B, D
T005	A, C
T006	B, C
T007	A, C
T008	A, B, C, E
T009	A, B, C
T010	F

**pass 3**

**Generate candidates**

$F_2$

$C_3$

itemsets	count
{A, B}	4
{A, C}	4
{A, E}	2
{B, C}	4
{B, D}	2
{B, E}	2

itemsets
{A, B, C}
{A, B, D}
{A, B, E}
{A, C, E}
{B, C, D}
{B, C, E}
{B, D, E}

**(A,B,C) is generated from (A,B) joined (A,C) using common core A**

**(A,C,E) is generated from (A,C) joined (A,E) but eliminated because (C,E) is not frequent**

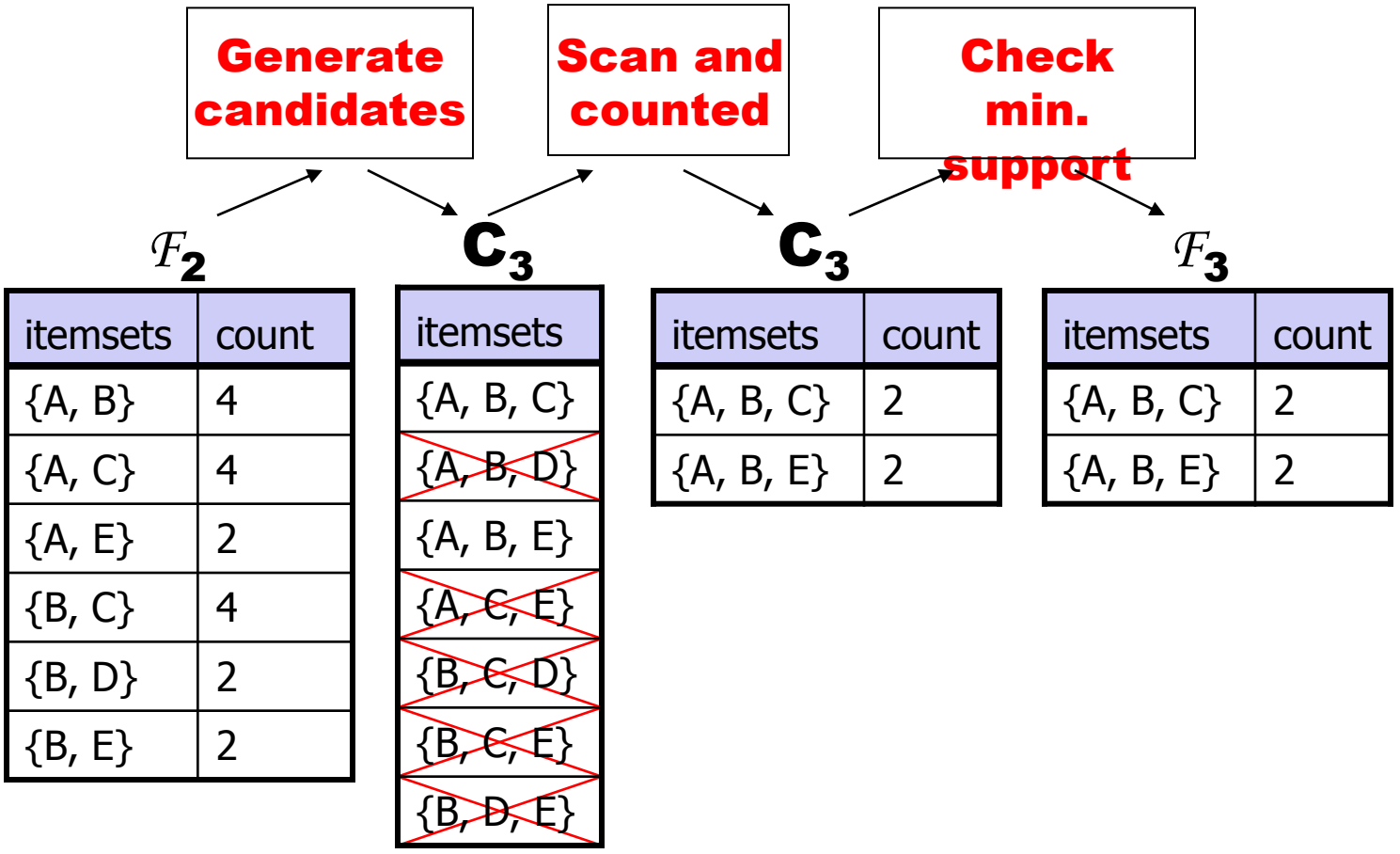
**minSup = 20%**

**The notion of Core**

**DB**

TID	items
T001	A, B, E
T002	B, D
T003	B, C
T004	A, B, D
T005	A, C
T006	B, C
T007	A, C
T008	A, B, C, E
T009	A, B, C
T010	F

**pass 3**



**minSup = 20%**

**DB**

TID	items
T001	A, B, E
T002	B, D
T003	B, C
T004	A, B, D
T005	A, C
T006	B, C
T007	A, C
T008	A, B, C, E
T009	A, B, C
T010	F

# pass 4

**Generate candidates**

$F_3$

itemsets	count
{A, B, C}	2
{A, B, E}	2

$C_4$

itemsets
<del>{A, B, C, E}</del>

e.g. (A,C,E) is not frequent

**$C_4$  is empty. Stop!**

**minSup = 20%**



# Final Answer

(All Frequent Itemsets when minSup=20%)

$\mathcal{F}_1$

itemsets	count
{A}	6
{B}	7
{C}	6
{D}	2
{E}	2

$\mathcal{F}_2$

itemsets	count
{A, B}	4
{A, C}	4
{A, E}	2
{B, C}	4
{B, D}	2
{B, E}	2

$\mathcal{F}_3$

itemsets	count
{A, B, C}	2
{A, B, E}	2

# FP-growth: Another Method for Frequent Itemset Generation



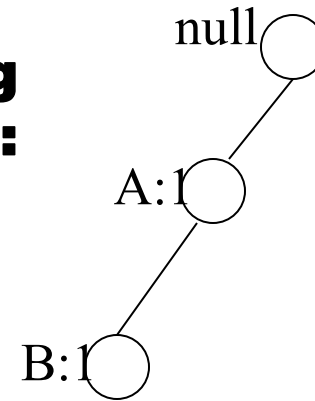
---

- Use a compressed representation of the database using an **FP-tree**
- Once an FP-tree has been constructed, FP-growth uses a recursive divide-and-conquer approach to mine the frequent itemsets

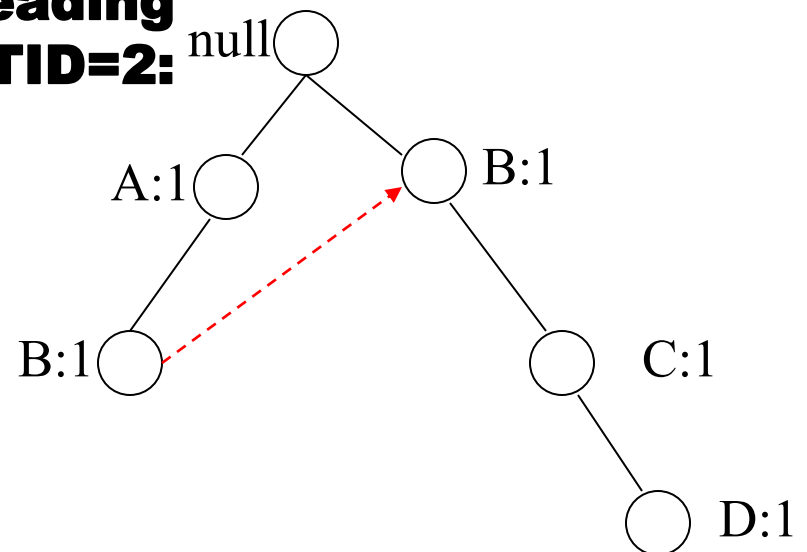
# FP-Tree Construction

TID	Items
1	{A,B}
2	{B,C,D}
3	{A,C,D,E}
4	{A,D,E}
5	{A,B,C}
6	{A,B,C,D}
7	{B,C}
8	{A,B,C}
9	{A,B,D}
10	{B,C,E}

**After reading  
TID=1:**



**After reading  
TID=2:**



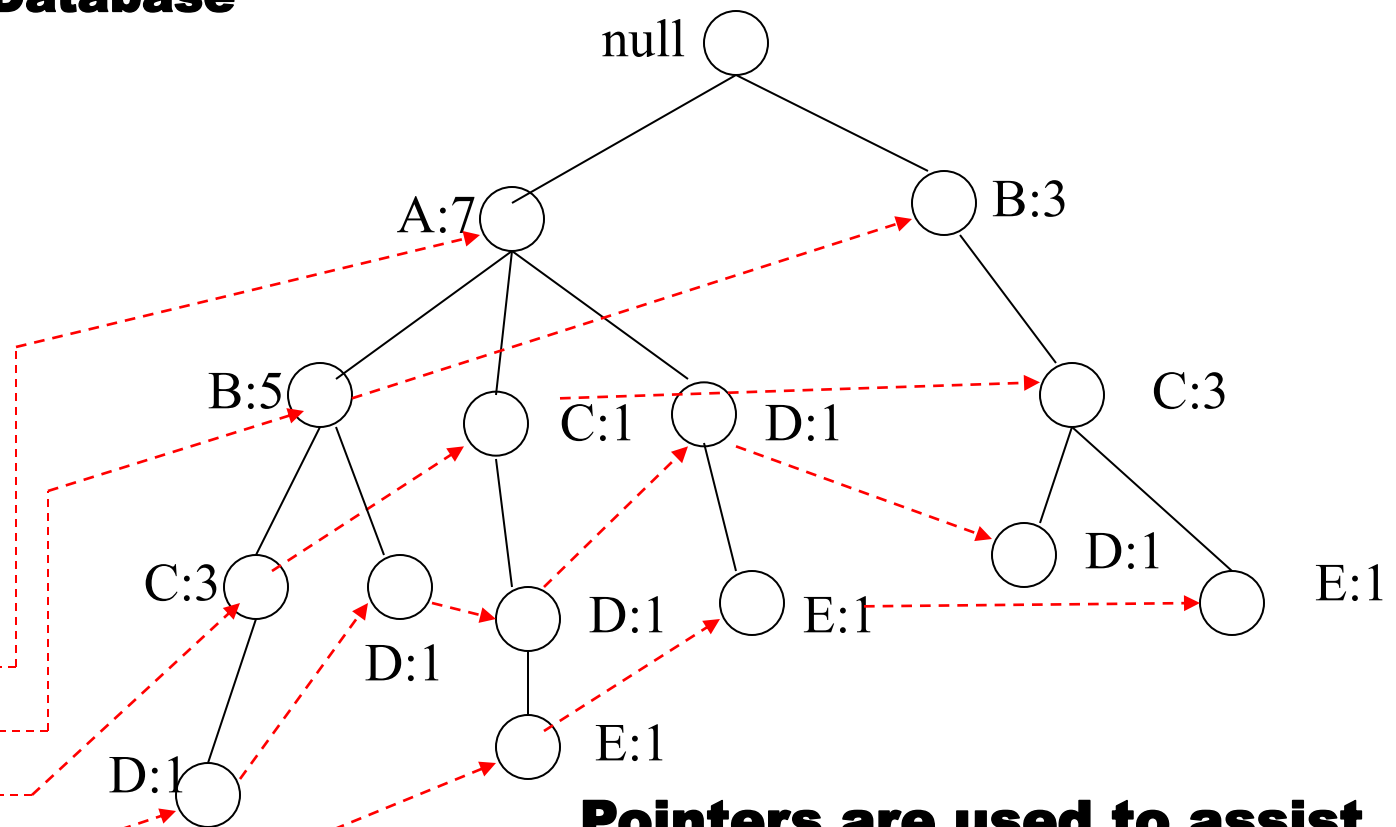
# FP-Tree Construction

TID	Items
1	{A,B}
2	{B,C,D}
3	{A,C,D,E}
4	{A,D,E}
5	{A,B,C}
6	{A,B,C,D}
7	{B,C}
8	{A,B,C}
9	{A,B,D}
10	{B,C,E}

**Transaction Database**

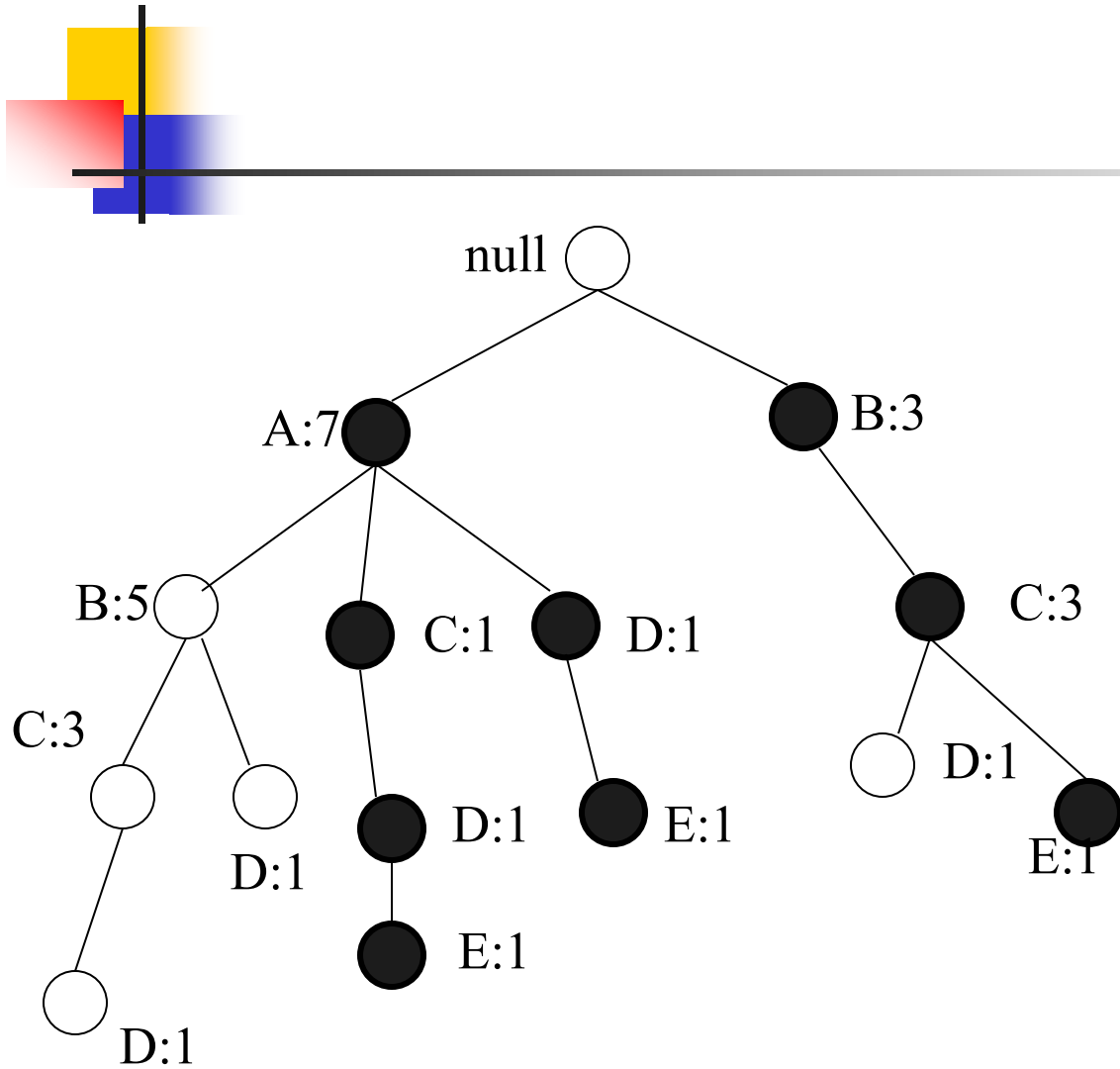
**Header table**

Item	Pointer
A	
B	
C	
D	
E	



**Pointers are used to assist frequent itemset generation**

# FP-growth

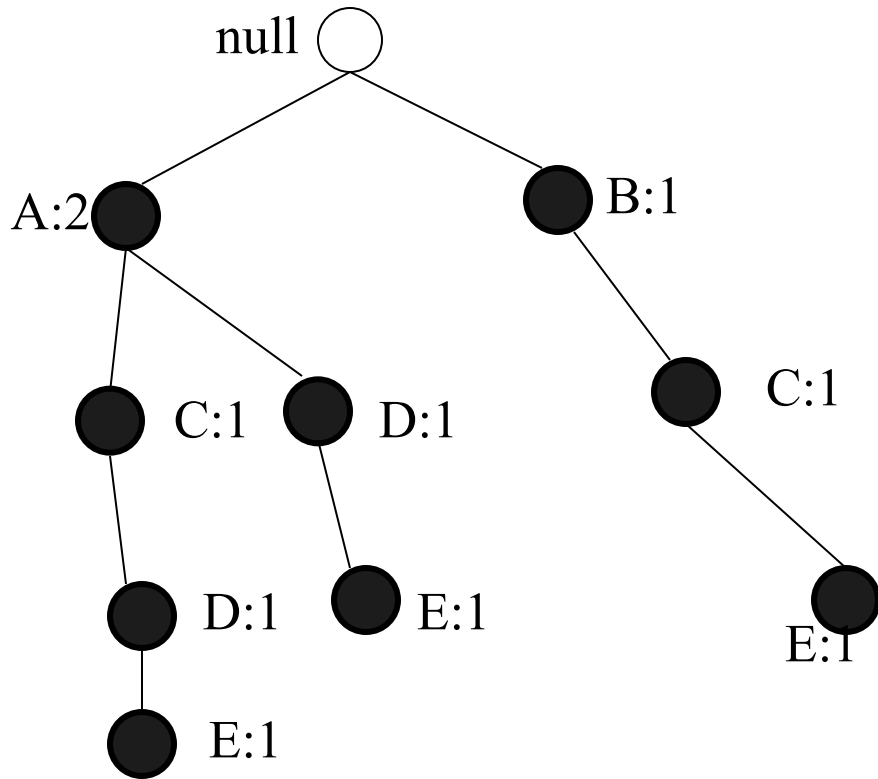


**Build conditional  
pattern base for E:  
 $P = \{(A:1,C:1,D:1),$   
 $(A:1,D:1),$   
 $(B:1,C:1)\}$**

**Recursively apply FP-  
growth on P**

# FP-growth

## Conditional tree for E: minSupp is 2



**Conditional Pattern base for E:**

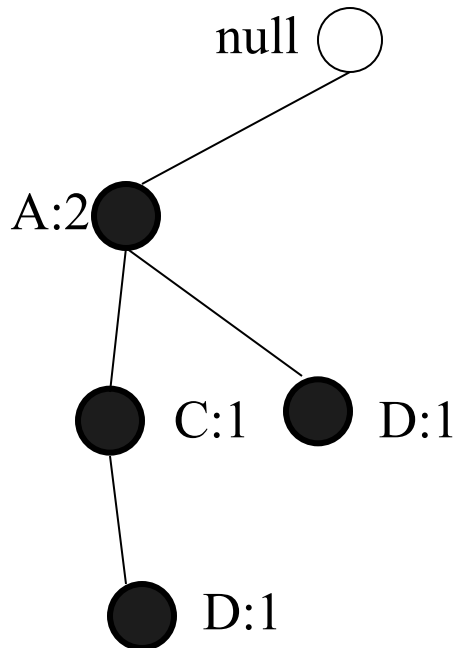
**P =**  
**{(A:1,C:1,D:1,E:1),**  
**(A:1,D:1,E:1),**  
**(B:1,C:1,E:1)}**

**Count for E is 3: {E} is frequent itemset**

**Recursively apply FP-growth on P**

# FP-growth

## Conditional tree for D within conditional tree for E:



**Conditional pattern  
base for D within  
conditional base for E:**

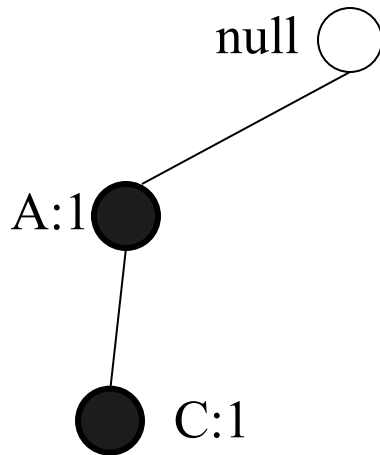
**$P = \{(A:1,C:1,D:1),$   
 $(A:1,D:1)\}$**

**Count for D is 2:  
therefore {D,E} is a  
frequent itemset**

**Recursively apply FP-  
growth on P**

# FP-growth

## Conditional tree for C within D within E:



**Conditional pattern base for C within D within E:**

$$P = \{(A:1, C:1)\}$$

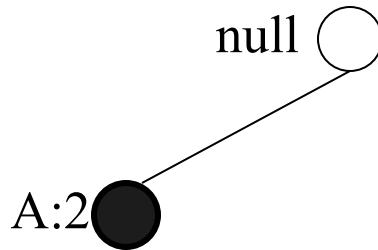
**Count for C is 1:  
{C,D,E} is NOT  
frequent itemset**



# FP-growth

## Conditional tree for A within D within E:

---



**Count for A is 2:  
{A,D,E} is frequent  
itemset**

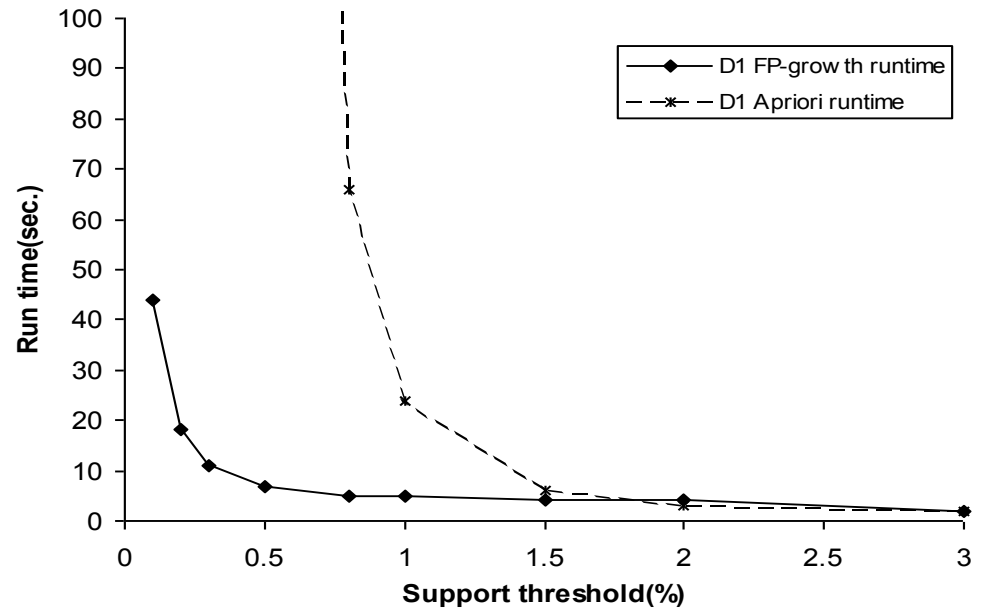
**Next step:**

**Construct conditional  
tree C within  
conditional tree E**

**Continue until  
exploring conditional  
tree for A (which has  
only node A)**

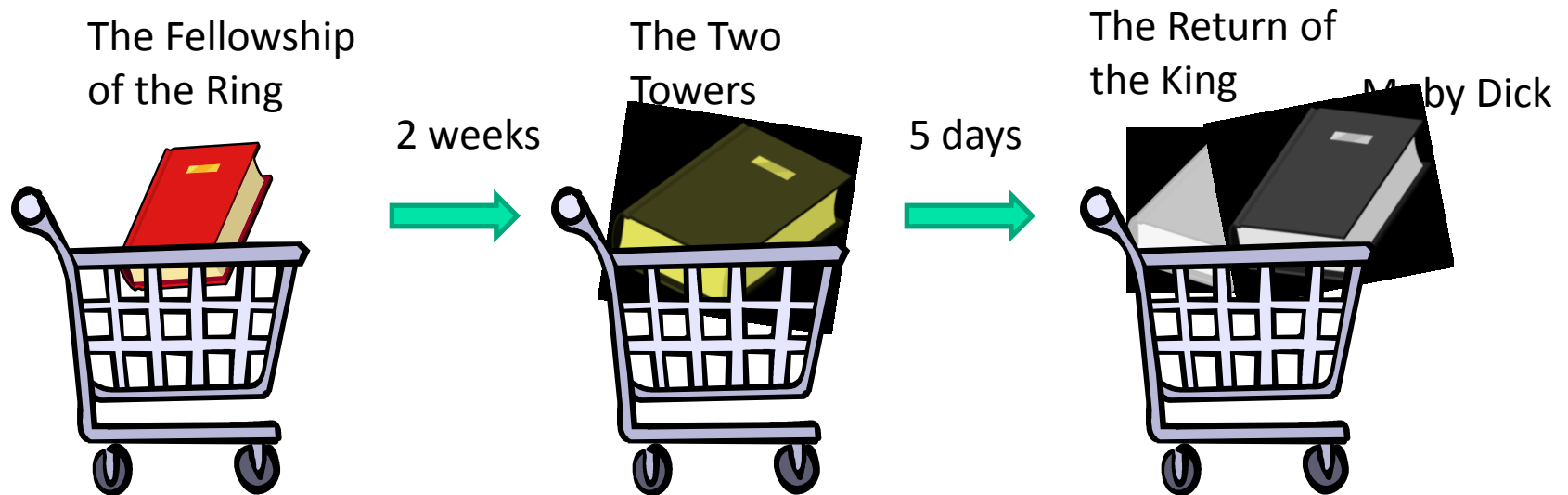
# Benefits of the FP-tree Structure

- Performance study shows
  - FP-growth is an order of magnitude faster than Apriori, and is also faster than tree-projection
- Reasoning
  - No candidate generation, no candidate test
  - Use compact data structure
  - Eliminate repeated database scan
  - Basic operation is counting and FP-tree building



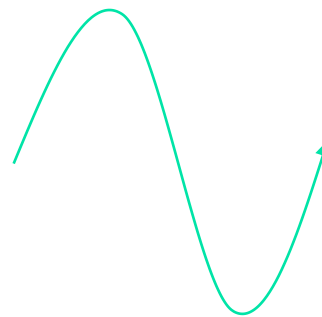
# Sequential Patterns Mining

- Given a set of sequences, find the complete set of frequent subsequences



# More Detailed Example

SID	sequence
10	<a(abc)(ac)d(cf)>
20	<(ad)c(bc)(ae)>
30	<(ef)(ab)(df)cb>
40	<eg(af)cbc>



Min Support = 0.5

## Frequent Sequences

<a>

<(a)(a)>

<(a)(c)>

<(a)(bc)>

<(e)(a)(c)>

...

# Motivation

- Business:
  - Customer shopping patterns
  - telephone calling patterns
  - Stock market fluctuation
  - Weblog click stream analysis
- Medical Domains:
  - Symptoms of a diseases
  - DNA sequence analysis





# Definitions

---

- **Items:** a set of literals  $\{i_1, i_2, \dots, i_m\}$
- **Itemset** (or event): a non-empty set of items.
- **Sequence:** an ordered list of itemsets, denoted as  $\langle (abc)(aef)(b) \rangle$
- A sequence  $\langle a_1 \dots a_n \rangle$  is a **subsequence** of sequence  $\langle b_1 \dots b_m \rangle$  if there exists integers  $i_1 < \dots < i_n$  such that  $a_1 \in b_{i_1}, \dots, a_n \in b_{i_n}$

# Definitions

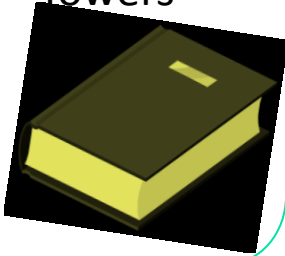
The Fellowship  
of the Ring



2 weeks



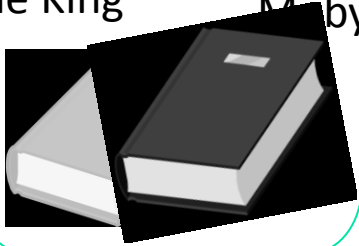
The Two  
Towers



5 days



The Return of  
the King



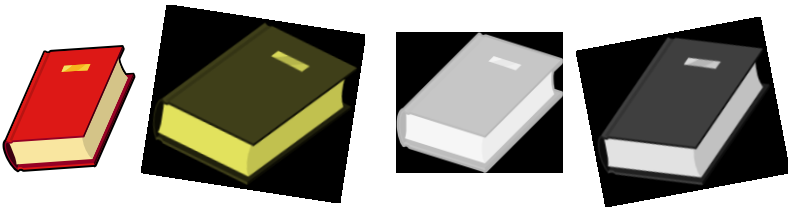
Moby Dick

event

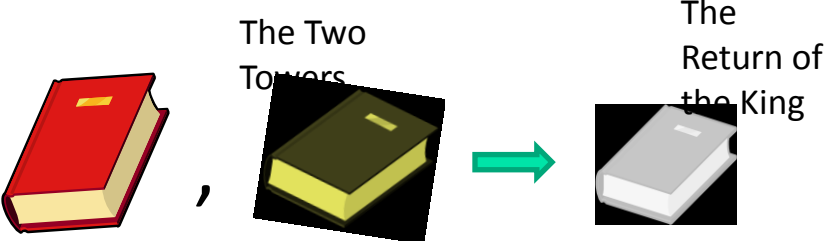
event

event

Items:



subsequences:





# More Definitions

---

- *Support* is the number of **sequences** that contain the pattern. (as in frequent itemsets, the concept of *confidence* is **not** defined)
- A sequential pattern is a sub-sequence appearing in more than **minSup** sequences



# Definitions

A sequence database

Seq. ID	Sequence
10	<(bd)cb(ac)>
20	<(bf)(ce)b(fg)>
30	<(ah)(bf)abf>
40	<(be)(ce)d>
50	<a(bd)bcb(ade)>

A sequence : <(bd) c b (ac)>

Events

<ad(ae)> is a subsequence of  
<a(bd)bcb(ade)>

Given support threshold  $min\_sup = 2$ ,  
<(bd)cb> is a sequential pattern

# Much Harder than Frequent Itemsets!

$2^{m*n}$  possible candidates!



Where  $m$  is the number of items, and  $n$  is the number of transactions in the longest sequence.



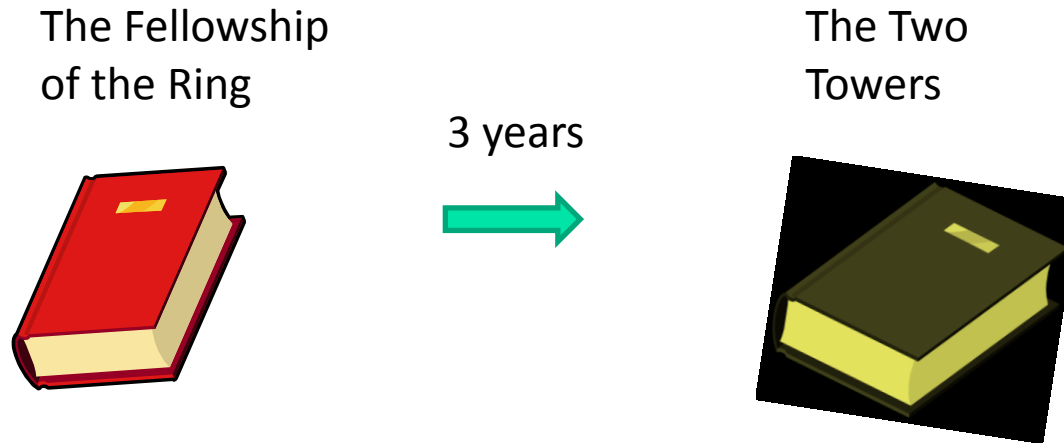
# Aside: Constraints

---

- Problem: most frequent sequences are not useful
- Solution: remove them
- The trick: do so while mining them to reduce time and narrow search space

# Example for Constraints

- **Min/Max Gap:** maximum and/or minimum time gaps between adjacent elements.



# More Constraints

- **Sliding Windows:** consider two transactions as one as long as they are in the same time-windows.

The Fellowship  
of the Ring



1 day



The Two  
Towers



2 weeks



The Return of  
the King



The Fellowship  
of the Ring



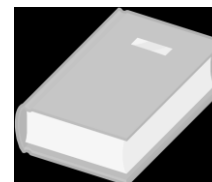
The Two  
Towers



2 weeks



The Return of  
the King





# The GSP Algorithm

---

- Developed by Srikant and Agrawal in 1996.
- Multiple-passes over the database.
- Uses generate-and-test approach.

# The SPADE Algorithm



- **SPADE** (**S**equential **P**attern **D**iscovery using **E**quivalent Class) developed by Zaki 2001.
- A vertical format sequential pattern mining method.
- A sequence database is mapped to a large set of
  - Item: <SID, EID>
- Sequential pattern mining is performed by
  - growing the subsequences (patterns) one item at a time by Apriori candidate generation



# Existing Algorithms

---

- Apriori based: GSP (96), SPADE (01)
- Pattern growth (similar to FP-growth ): PrefixSpan (04)
- All don't perform well on long sequences





# CAMLS (Gudes et. Al.)


---

- **C**onstraint-based **A**priori algorithm for **M**ining **L**ong **S**equences
- Designed especially for efficient mining of long sequences
- Uses constraints to increase efficiency
- Outperforms both SPADE and Prefix Span on both synthetic and real data



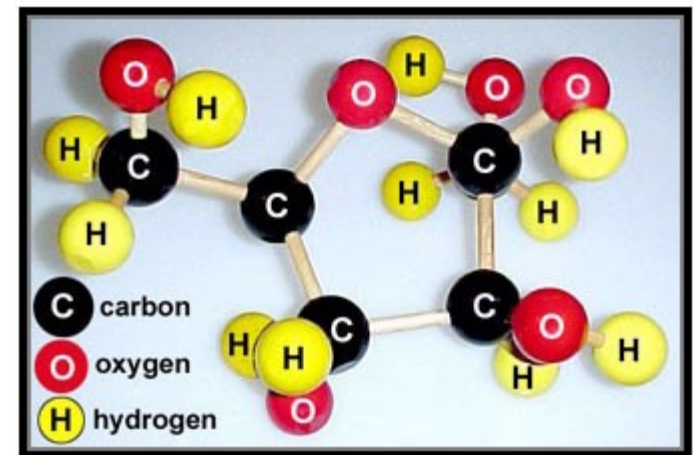
# Outline

---

- Basic concepts of Data Mining and Association rules
  - Apriori algorithm
  - Sequence mining
- Motivation for Graph Mining 
- Applications of Graph Mining
- Mining Frequent Subgraphs - Transactions
  - BFS/Apriori Approach (FSG and others)
  - DFS Approach (gSpan and others)
  - Diagonal Approach
  - Constraint-based mining and new algorithms
- Mining Frequent Subgraphs – Single graph
  - The support issue
  - The Path-based algorithm

# What Graphs are good for?

- Most of existing data mining algorithms are based on **Flat transaction representation**, i.e., sets of items.
- Datasets with structures, layers, hierarchy and/or geometry often do not fit well in this flat transaction setting. For example:
  - Numerical simulations
  - 3D protein structures
  - Chemical compounds
  - Generic XML files



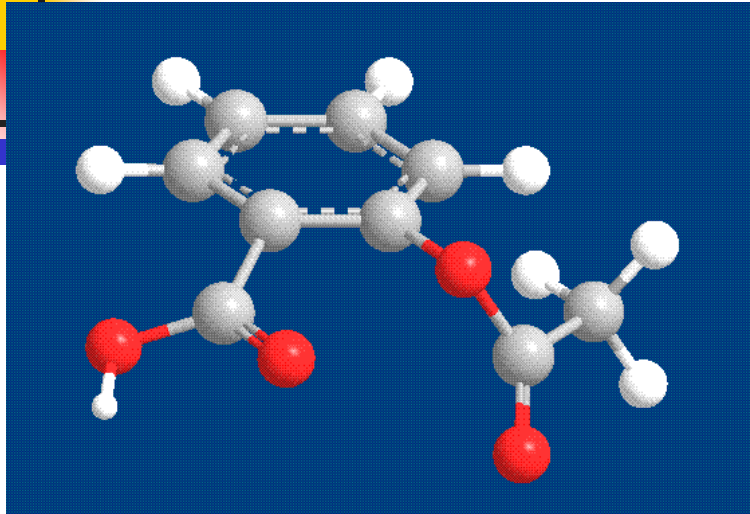


# Graph Based Data Mining

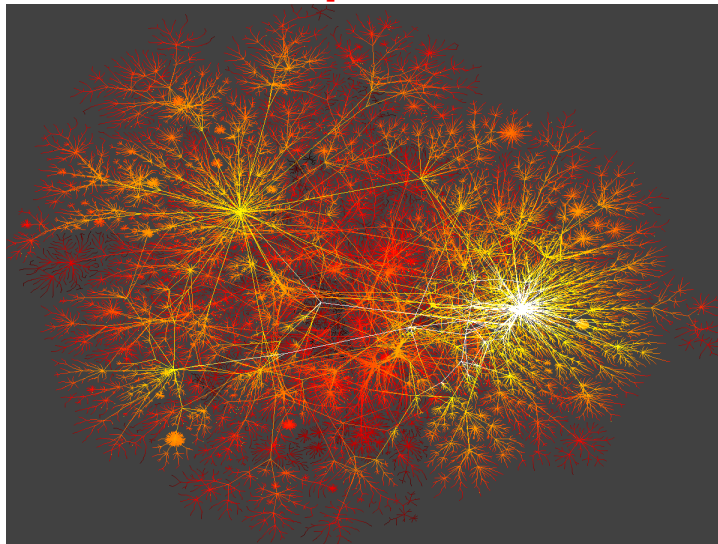
---

- Graph Mining (GM) is essentially the problem of discovering repetitive subgraphs occurring in the input graphs
- Motivation
  - Finding subgraphs capable of compressing the data by abstracting instances of the substructures
  - Identifying conceptually interesting patterns

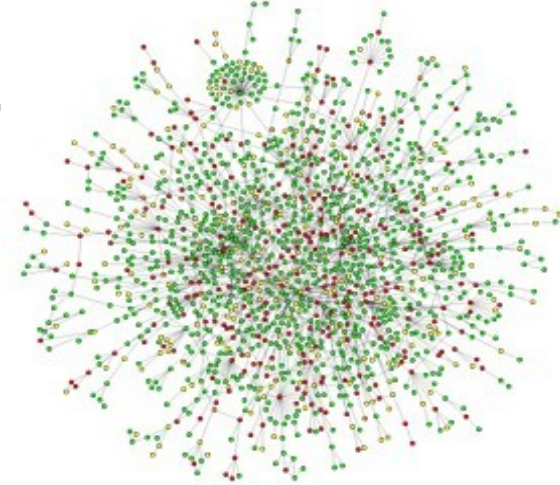
# Graph, Graph, Everywhere



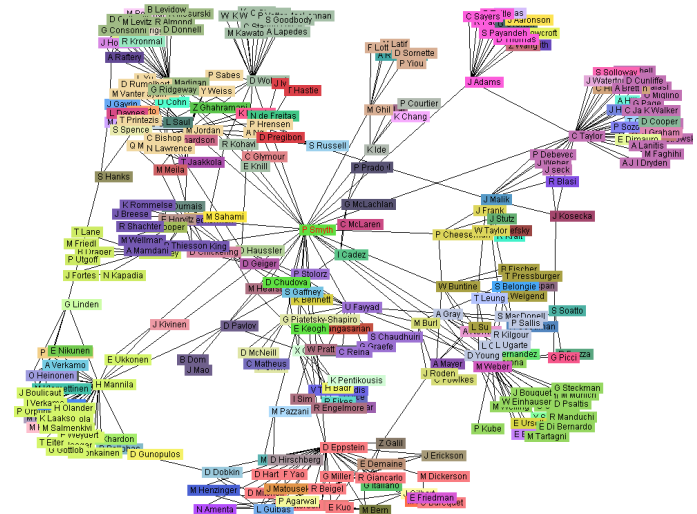
**Aspirin**



**Internet**



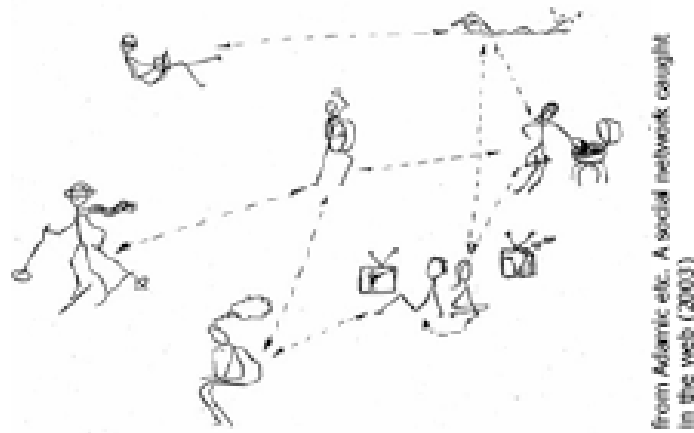
**Yeast protein interaction network**



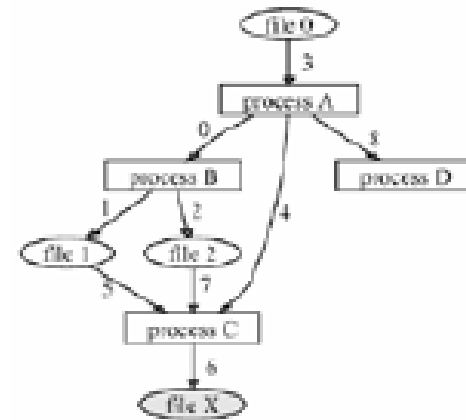
**Co-author network**

from H. Jeong et al Nature 411, 41 (2001)

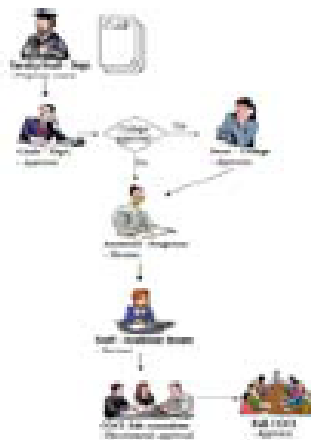
# Graph, Graph, Everywhere (cont.)



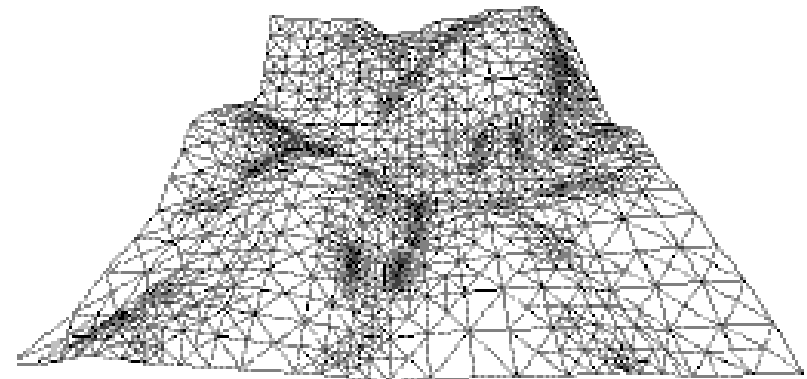
Social Network



Event Log Graph



Workflow



Mesh



# Why Graph Mining?

---

- Graphs are ubiquitous
  - Chemical compounds (Cheminformatics)
  - Protein structures, biological pathways/networks (Bioinformatics)
  - Program control flow, traffic flow, and workflow analysis
  - XML databases, Web, and social network analysis
- Graph is a general model
  - Trees, lattices, sequences, and items are degenerated graphs
- Diversity of graphs
  - Directed vs. undirected, labeled vs. unlabeled (edges & vertices), weighted, with angles & geometry (topological vs. 2-D/3-D)
- Complexity of algorithms: many problems are of high complexity (NP complete!)

# Modeling Data With Graphs...

## Going Beyond Transactions

Graphs are suitable for capturing arbitrary relations between the various elements.

<u>Data Instance</u>		<u>Graph Instance</u>
Element	↔	Vertex
Element's Attributes	↔	Vertex Label
Relation Between Two Elements	↔	Edge
Type Of Relation	↔	Edge Label
<hr/>		
Relation between a Set of Elements	↔	Hyper Edge

**Provide enormous flexibility for modeling the underlying data as they allow the modeler to decide on what the elements should be and the type of relations to be modeled**





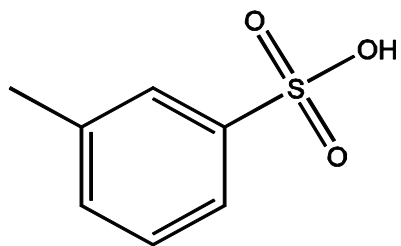
# Graph Pattern Mining

---

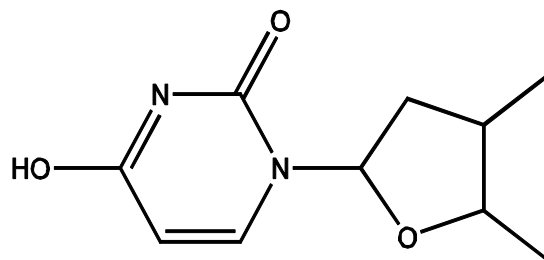
- *Frequent* subgraphs
  - A (sub)graph is *frequent* if its *support* (occurrence frequency) in a given dataset is no less than a *minimum support* threshold
- **What is Support?** – intuitively the number of transactions containing a single occurrence
- We'll see other definitions later

# Example: Frequent Subgraphs

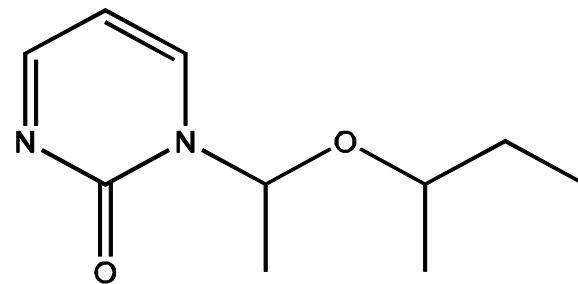
## GRAPH DATASET



(A)



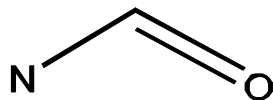
(B)



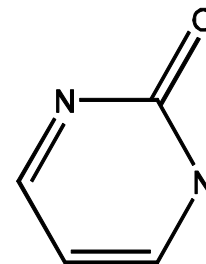
(C)

## FREQUENT PATTERNS (MIN SUPPORT IS 2)

(1)

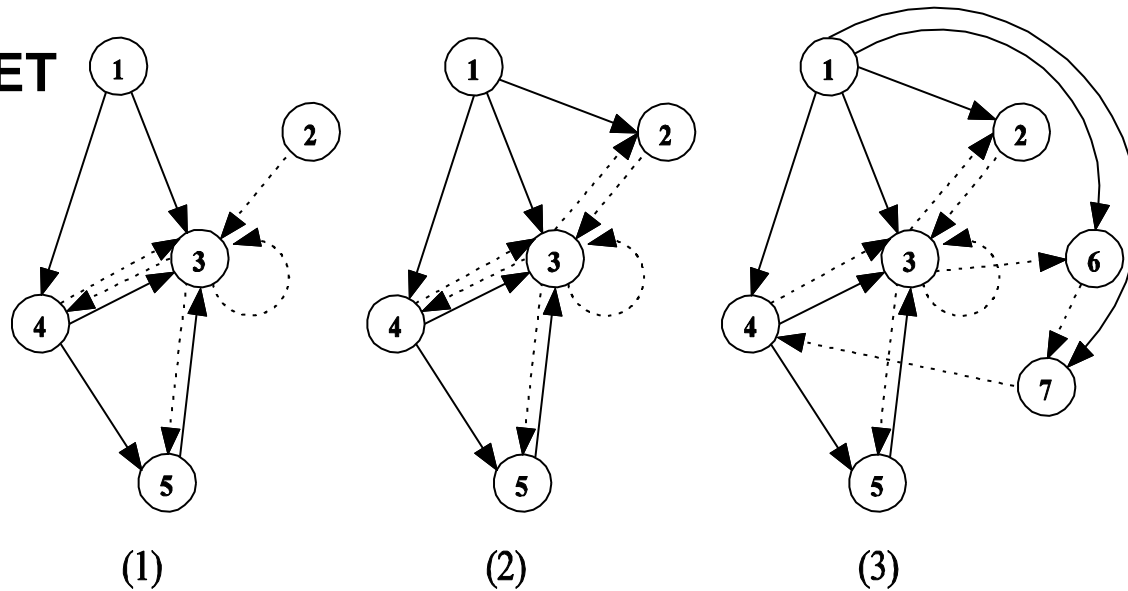


(2)



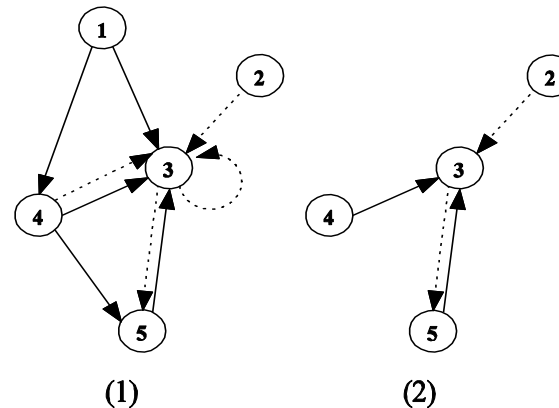
# Example (II) – Execution flow

## GRAPH DATASET



- 1: makepat
- 2: esc
- 3: addstr
- 4: getccl
- 5: dodash
- 6: in\_set\_2
- 7: stclose

## FREQUENT PATTERNS (MIN SUPPORT IS 2)



# Association rules vs. Graph patterns

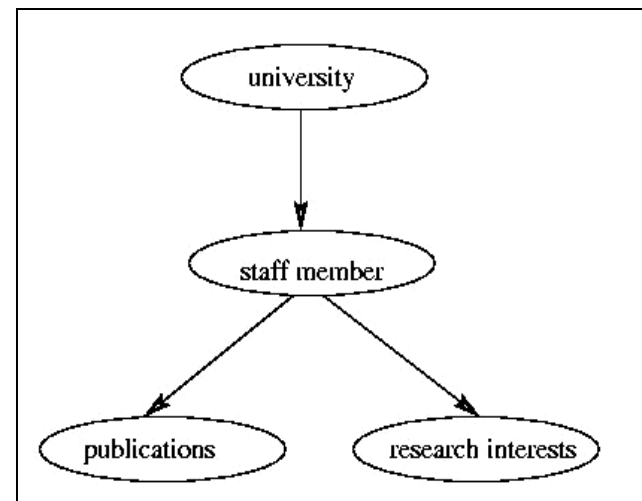
- **Rule-based patterns**

Patterns of form  $A_1, A_2, \dots, A_n \Rightarrow B$  where  $A_1, \dots, A_n, B$  are atomic values. *Example:* "diapers  $\Rightarrow$  beer"

- **Topology-based patterns**

Patterns that have structure in addition to atomic values

*Example:* graph pattern  
(no concept of implication)





# Semi-structured data as Graphs

---

- **Semi-structured** data is data that can be modeled as a labeled graph. For example, XML and HTML data.
- **Frequent patterns discovered from semi-structured data are useful for:**
  - Improving database design (A. Deutsch, M. Fernandez, D.Suciu "*Storing Semistructured Data with STORED*", SIGMOD'99)
  - Efficient indexing (Apex Index for XML)
  - User behavior predictions and User preference based applications
  - Social networks analysis
  - Chemical and Bioinformatics applications

# Semi-structured Data Mining Algorithms



---


- Simple path patterns (Chen, Park, Yu 98)
- Generalized path patterns (Nanopoulos, Manolopoulos 01)
- Simple tree patterns (Lin, Liu, Zhang, Zhou 98)
- Tree-like patterns (Wang, Huiqing, Liu 98)
- General graph patterns (Kuramochi, Karypis 01, Han 02)

**We are interested in general graph mining!**



# Outline

---

- Basic concepts of Data Mining and Association rules
  - Apriori algorithm
  - Sequence mining
- Motivation for Graph Mining
- Applications of Graph Mining 
- Mining Frequent Subgraphs - Transactions
  - BFS/Apriori Approach (FSG and others)
  - DFS Approach (gSpan and others)
  - Diagonal Approach
  - Constraint-based mining and new algorithms
- Mining Frequent Subgraphs – Single graph
  - The support issue
  - The Path-based algorithm



# Applications of Graph Mining – two examples

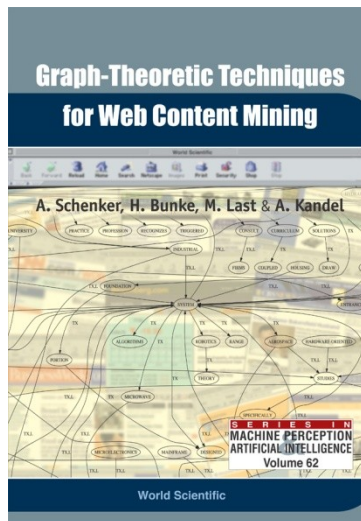
---

- Document Classification (Last & Kandel)
- Drug development (Christian Borgelt )
- Representing information (Toivonen – here...)



# Documents Classification

## Alternative Representation of Multilingual Web Documents: The Graph-Based Model



Introduced in A. Schenker, H. Bunke, M. Last, A. Kandel, *Graph-Theoretic Techniques for Web Content Mining*, World Scientific, 2005



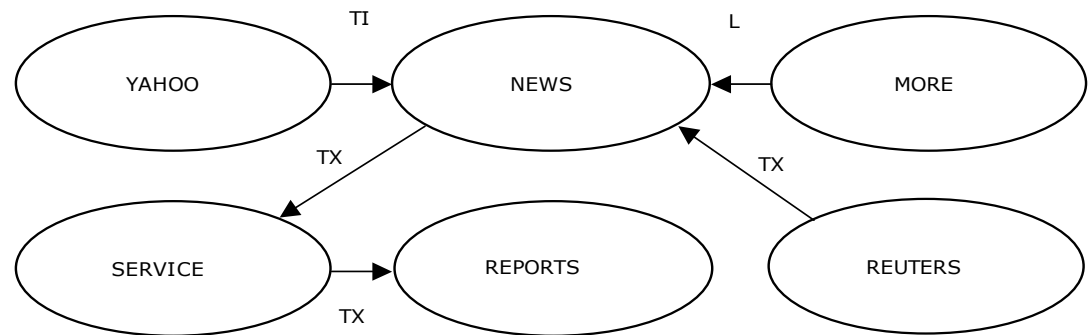
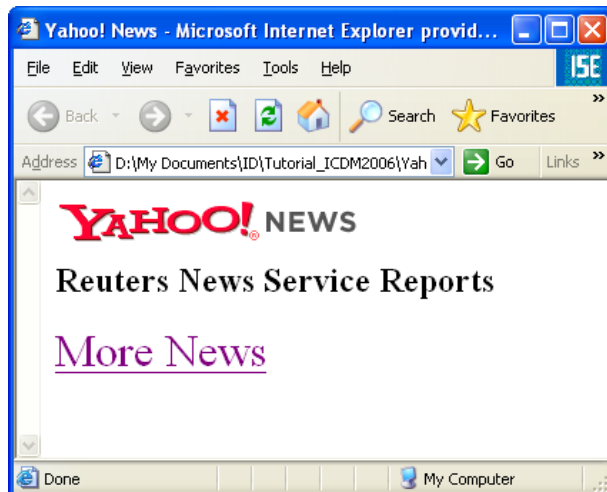
# The Graph-Based Model of Web Documents

---

- Basic ideas
  - One node for each unique term
  - If word B follows word A, there is an edge from A to B
    - In the presence of terminating punctuation marks (periods, question marks, and exclamation points) no edge is created between two words
  - Graph size is limited by including only the most frequent terms
  - Several variations for node and edge labeling (see the next slides)
- Pre-processing steps
  - Stop words are removed
  - Lemmatization
    - Alternate forms of the same term (singular/plural, past/present/future tense, etc.) are mapped to the most frequently occurring form

# The *Graph* Representation

- Edges are labeled according to the document section where the words are followed by each other
  - *Title (TI)* contains the text related to the document's title and any provided keywords (meta-data);
  - *Link (L)* is the "anchor text" that appears in clickable hyperlinks on the document;
  - *Text (TX)* comprises any of the visible text in the document (this includes anchor text but not title and keyword text)



# Graph Based Document Representation – Detailed Example

Source: [www.cnn.com](http://www.cnn.com), May 24, 2005



## **Iraq bomb: Four dead, 110 wounded**

A car bomb has exploded outside a popular Baghdad restaurant, killing three Iraqis and wounding more than 110 others, police officials said. Earlier an aide to the office of Iraqi Prime Minister Ibrahim al-Jaafari and his driver were killed in a drive-by shooting.

**[FULL STORY](#)**

# Graph Based Document Representation - Parsing

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"
<!-- saved from url=(0023)http://edition.cnn.com/ -->
<HTML lang=en><HEAD><TITLE>CNN.com International</TITLE>
<META http-equiv=content-type content="text/html; charset=iso-8859-1">
<META http-equiv=refresh content=1800><LINK href="/" rel=Start><LINK
```

title

```
<DIV class=cnnSectionT1
style="PADDING-RIGHT: 6px; PADDING-LEFT: 6px; PADDING-BOTTOM: 6px; PADDING-TOP: 3px">
<H2><A style="COLOR: #000"
href="http://edition.cnn.com/2005/WORLD/meast/05/23/iraq.main/index.html">Iraq
```

link

```
bomb: Four dead, 110 wounded</A></H2>
```

```
<P>A car bomb has exploded outside a popular Baghdad restaurant, killing
three Iraqis and wounding more than 110 others, police officials said.
Earlier an aide to the office of Iraqi Prime Minister Ibrahim al-Jaafari
and his driver were killed in a drive-by shooting.</P>
```

```
<P><A class=cnnt1link
```

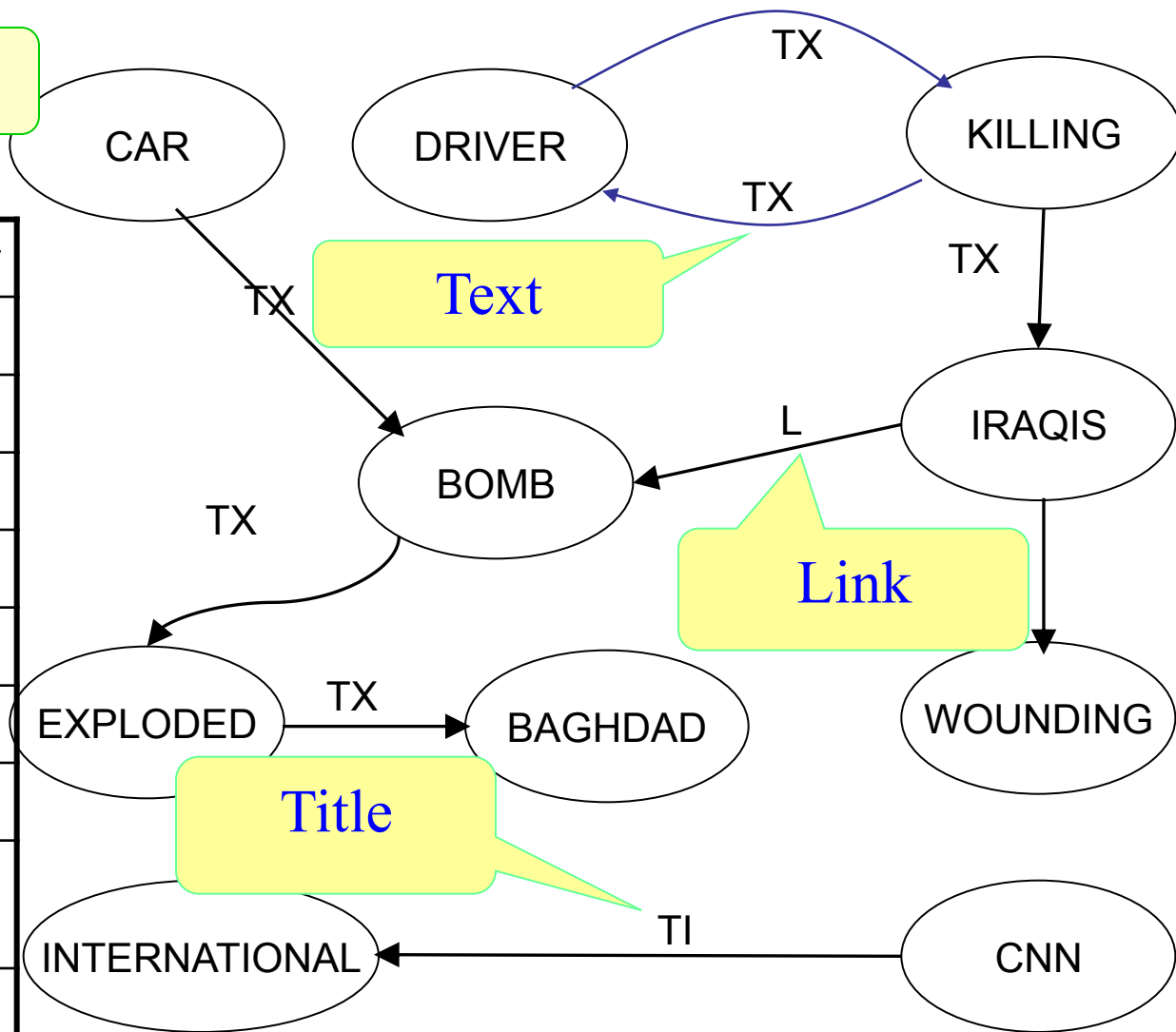
```
href="http://edition.cnn.com/2005/WORLD/meast/05/23/iraq.main/index.htm
STORY</A></P>
```

text

# Standard Graph Based Document Representation

Ten most frequent terms are used

Word	Frequency
Iraqis	3
Killing	2
Bomb	2
Wounding	2
Driver	2
Exploded	1
Baghdad	1
International	1
CNN	1
Car	1





# Classification Using Graphs

---

- Basic idea

- Mine the frequent sub-graphs, call them terms
- Use TF-IDF like measure for assigning the most characteristic terms to documents
- Use Clustering and K-nearest neighbors classification



# Subgraph Extraction

---

- Input
  - $\mathbf{G}$  – training set of directed, unique nodes graphs
  - $CR_{min}$  - Minimum Classification Rate
- Output
  - Set of classification-relevant sub-graphs
- Process:
  - For each class find sub-graphs with  $CR > CR_{min}$
  - Combine all sub-graphs into one set
- Basic Assumption
  - **Classification-Relevant Sub-Graphs** are more frequent in a specific category than in other categories





# Computing the Classification Rate

---

- Subgraph Classification Rate

$$CR (g'_k(c_i)) = SCF (g'_k(c_i)) \times ISF (g'_k(c_i))$$

- $SCF (g'_k(c_i))$  - Subgraph Class Frequency of subgraph  $g'_k$  in category  $c_i$
- $ISF (g'_k(c_i))$  - Inverse Subgraph Frequency of subgraph  $g'_k$  in category  $c_i$
- **Classification Relevant Feature** is a feature that best explains a specific category, or frequent in this category more than in all others

# Calculation of ISF

$g'_{kf}(c_i)$  - Number of graphs containing a sub-graph  $g'_k$  in category  $c_i$ .

$N(c_i)$  - Number of graphs in category  $c_i$ .

*ISF - Inverse Sub-graph Frequency:*

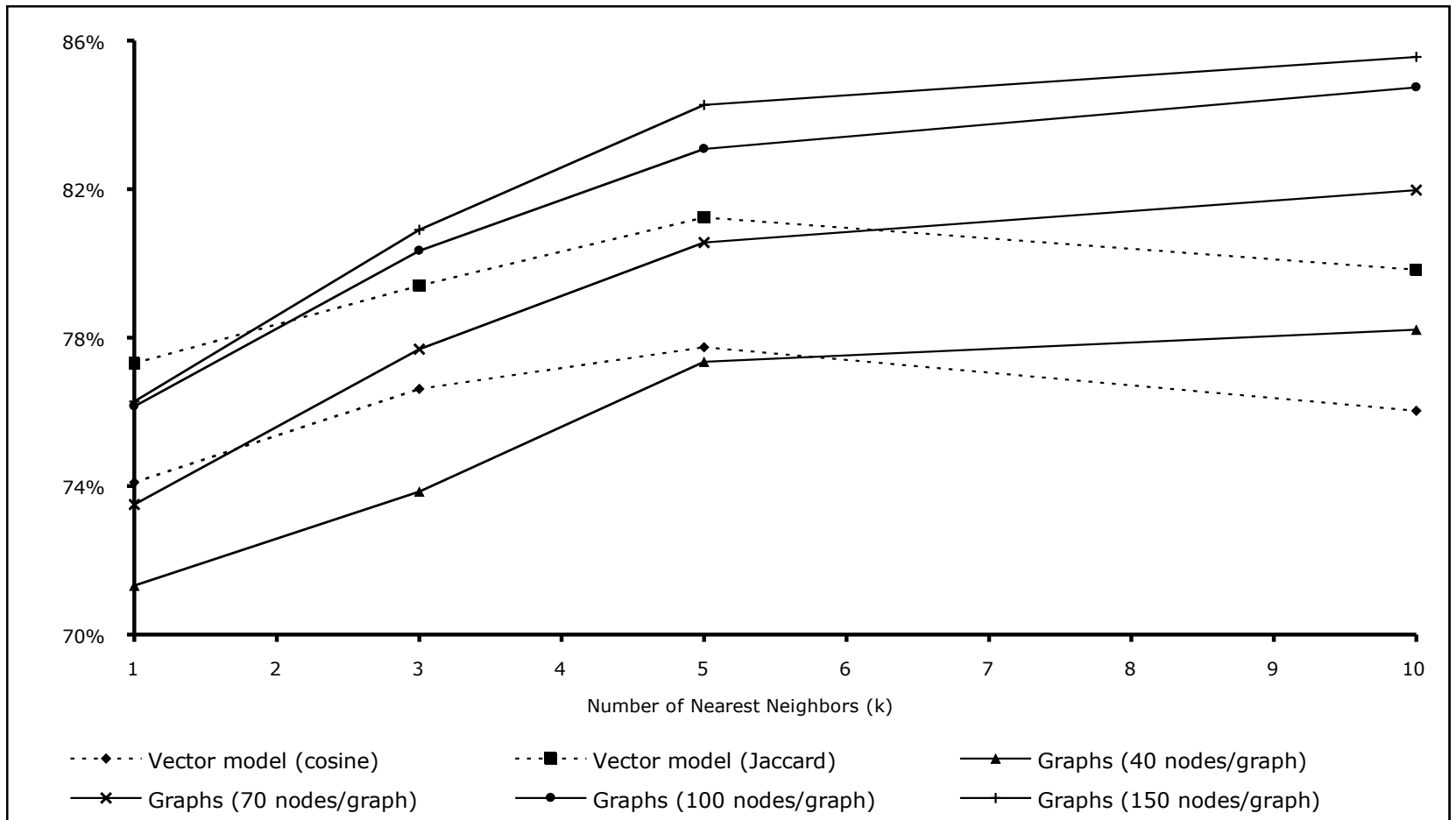
$$ISF(g'_k(c_i)) = \begin{cases} \log_2 \left( \frac{\sum N(c_j)}{\sum g'_{kf}(c_j)} \right) & \text{if } \sum g'_{kf}(c_j) > 0 \\ \log_2(2 \times \sum N(c_j)) & \text{if } \sum g'_{kf}(c_j) = 0 \end{cases} \quad \{\forall c_j \in C; j \neq i\}$$

$ISF(g'_k(c_i))$  - Measure for inverse frequency of sub-graph  $g'_k$  in category  $c_i$ .

$N(c_j)$  - Number of graphs in category  $c_j$ .

$g'_{kf}(c_j)$  - Number of graphs containing  $g'_k$  in category  $c_j$ .

# k-nearest neighbors with graphs — Accuracy vs. Graph Size (Graph model was more effective than Vector model )



# Frequent Pattern Mining

## Christian Borgelt

Intelligent Data Analysis and Graphical Models  
Research Unit

European Center for Soft Computing

c/ Gonzalo Gutierrez Quiros s/n, 33600 Mieres, Spain

[christian.borgelt@softcomputing.es](mailto:christian.borgelt@softcomputing.es)

<http://www.softcomputing.es/>

<http://www.borgelt.net/>

<http://www.borgelt.net/teach/fpm/>

Christian

# Frequent Pattern Mining

## Christian Borgelt

Application:  
Molecular Fragment Mining

# Frequent Pattern Mining

## Drugs Development

- Developing a new drug can take 10 to 12 years  
(from the choice of the target to the introduction into the market).
- In recent years the duration of the drug development processes increased continuously; at the same time, the number of substances under development has gone down drastically.
- Due to high investments, pharmaceutical companies must secure their market position and competitiveness by only a few, highly successful drugs.
- As a consequence, the chances for the development of drugs for target groups with rare diseases or with special diseases in developing countries are considerably reduced.
- A significant reduction of the development time could mitigate this trend or even reverse it.

# Frequent Pattern Mining

## Christian Borgelt

- Motivation: Accelerating Drug Development
  - Phases of drug development: pre-clinical and clinical
  - Data gathering by high-throughput screening:  
building molecular databases with activity information
  - Acceleration potential by intelligent data analysis of the **pre-clinical** phase:  
(quantitative) structure-activity relationship discovery
- Mining Molecular Databases
  - Example data: NCI DTP HIV Antiviral Screen data set
  - Description languages for molecules:  
SMILES, SLN, SDIe/Ctab etc.
  - Finding common molecular substructures
  - Finding discriminative molecular substructures

# Frequent Pattern Mining

## Christian Borgelt

- The length of the pre-clinical and clinical tests series can hardly be reduced,
- since they serve the purpose to ensure the safety of the patients.
- Therefore approaches to speed up the development process usually target the pre-clinical phase before the animal tests.
- In particular, it is tried to improve the search for new drug candidates
- Here Intelligent Data Analysis and Frequent Pattern Mining can help.
- One possible approach: With high-throughput screening a very large number of substances is tested automatically and their activity is determined.
- The resulting molecular databases are analyzed by trying to find common substructures of active substances.



# Frequent Pattern Mining

## Christian Borgelt

### Common Molecular Substructures

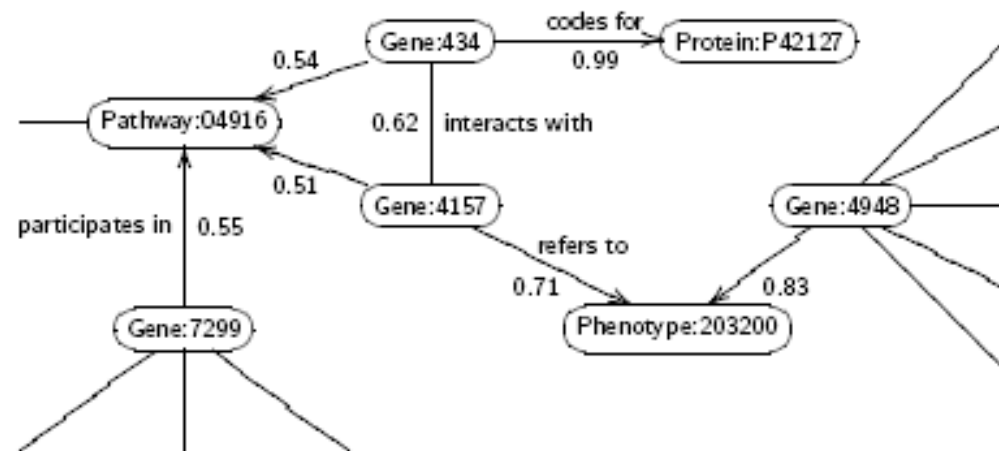
- Analyze only the active molecules.
- Find molecular fragments that appear frequently in the molecules.

### Discriminative Molecular Substructures

- Analyze the active and the inactive molecules.
- Find molecular fragments that appear frequently in the active molecules and only rarely in the inactive molecules.
- Rationale in both cases:
  - The found fragments can give hints which structural properties are responsible for the activity of a molecule.
  - This can help to identify drug candidates (so-called pharmacophores) and to guide future screening efforts

# Biomine – Representing Biological Information (Toivonen, Langohr and others... )

- **12 biological databases integrated** into a single large graph: Entrez Gene, UniProt, GO, HomoloGene, MIM, ...
- **About 1 million nodes:** genes, proteins, phenotypes, pathways, articles, genomic regions, tissues, protein families, ligands, drugs, ...
- **About 10 million edges:** codes for, homologous to, interacts with, participates in, ...
- **Edge weights** are functions of the reliability of the original data and the informativeness of the edge



# Biomine Queries – Subgraphs extraction

- **Biological entries** as query terms:  
Biomine finds **subgraph which connects query terms**
- Biomine selects the graphs to maximize the connectivity of query terms given the weights
- **One biological entry** as query term:  
Biomine finds **neighborhood**
  
- Biomine: <http://biomine.cs.helsinki.fi>
- Plants Biomine: <http://biomine.cs.helsinki.fi/plants>
- DBLP Biomine: <http://biomine.cs.helsinki.fi/dblp>

# Two more interesting Applications

1. Graph mining for detection of financial crimes (Jedrzejek et. Al.)

The illegal activity is represented as a graph, and that graph is searched in a large set of financial transactions (this is actually graph searching not graph mining)

2. Consumer behavior analysis by Graph mining (Yada et. Al.)

Representing the sequence of consumer purchases as a graph and searching for frequent patterns.

# Consumer behavior analysis by Graph mining

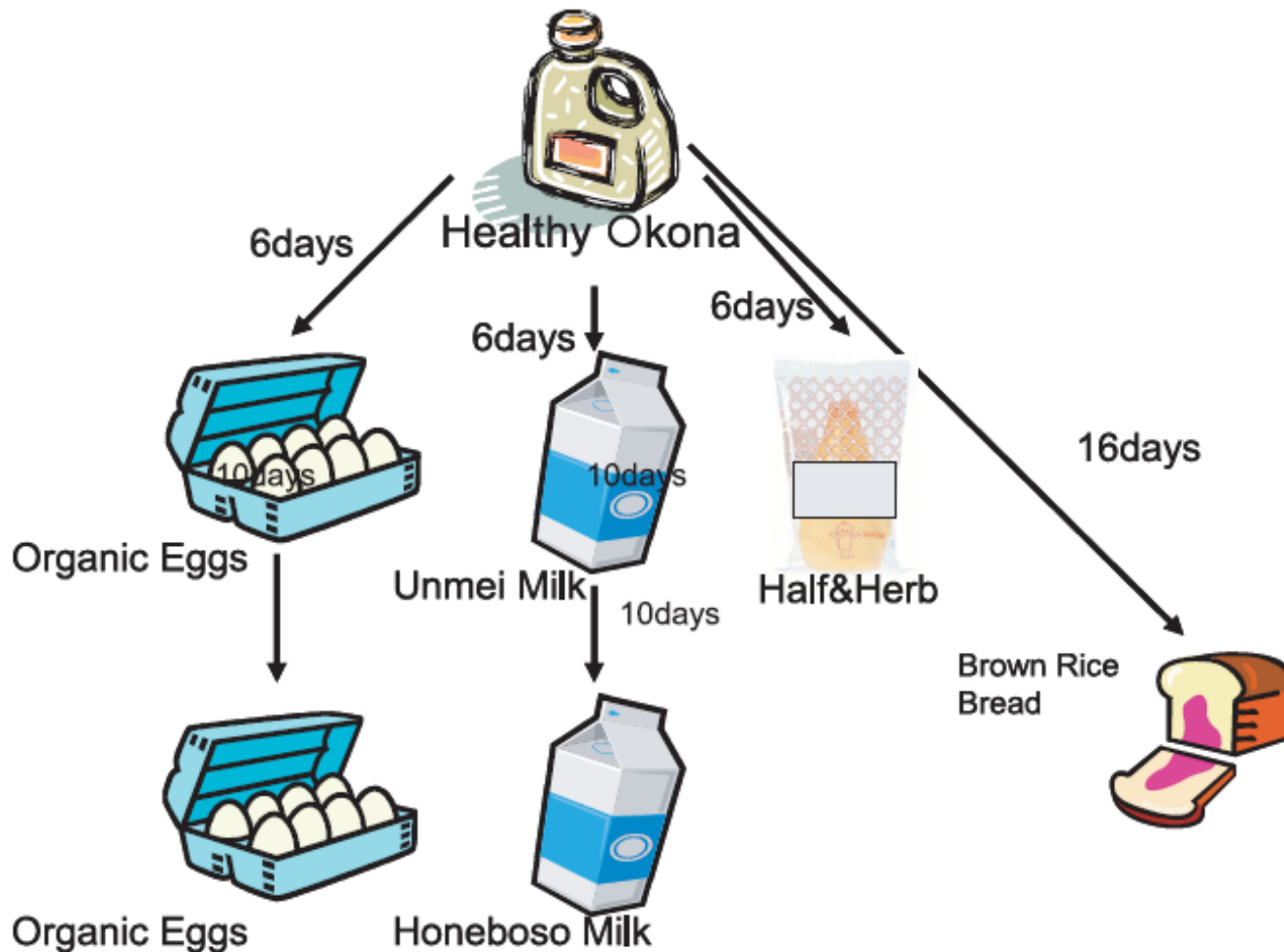


Fig. 1. Graph structured data and purchasing behavior.

# Graph Mining

