HELSINGIN YLIOPISTO
HELSINGFORS UNIVERSITET
UNIVERSITY OF HELSINKI

582364 Data mining, 4 cu
Lecture 7:
Sequential Patterns

Spring 2010

Lecturer: Juho Rousu

Teaching assistant: Taru Itäpelto

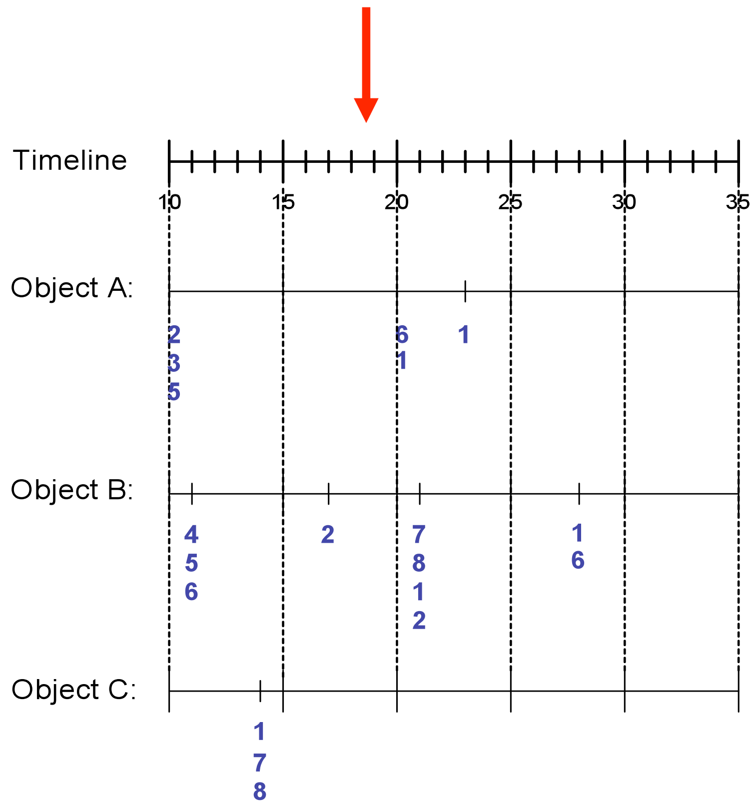
Sequential Patterns

- In many data mining tasks the *order* and *timing* of events contains important information
  - Credit card usage profile (10.4 €0, 11.4 €1000 12.4 €1500, ..)
  - Travel plan (Road E75 for 100km, Road 24 for 25km, Road 313 for 5km)
  - Process monitoring (Warning X at 1am, Crash Y at 2am,...)
- Frequent itemsets only capture the co-occurrences
  - No order between the items: {Bread, Milk} means the same as {Milk, Bread}
  - Order of transactions not considered: itemset support is a sum over a set of transactions

Sequence Data

- Each row ('transaction') records occurrences of *events* associated with a particular *object* at a given *time*
- Sorting the transactions using the timestamp, gives a *sequence* for each object with *elements* given by the collection of events ('items')

Object	Timestamp	Events
A	10	2, 3, 5
A	20	6, 1
A	23	1
B	11	4, 5, 6
B	17	2
B	21	7, 8, 1, 2
B	28	1, 6
C	14	1, 8, 7



Examples of Sequences

- Web sequence:

  < {Homepage} {Electronics} {Digital Cameras} {Canon Digital Camera} {Shopping Cart} {Order Confirmation} {Return to Shopping} >

- Sequence of books checked out at a library:

  <{Introduction to Data Mining} {Fellowship of the Ring} {The Two Towers, Return of the King}>

- Sequence of initiating events leading to the Three-Mile Island Nuclear Accident:

< {clogged resin} {outlet valve closure} {loss of feedwater} {condenser polisher outlet valve shut} {booster pumps trip} {main waterpump trips} {main turbine trips} {reactor pressure increases}>

Sequences Formally

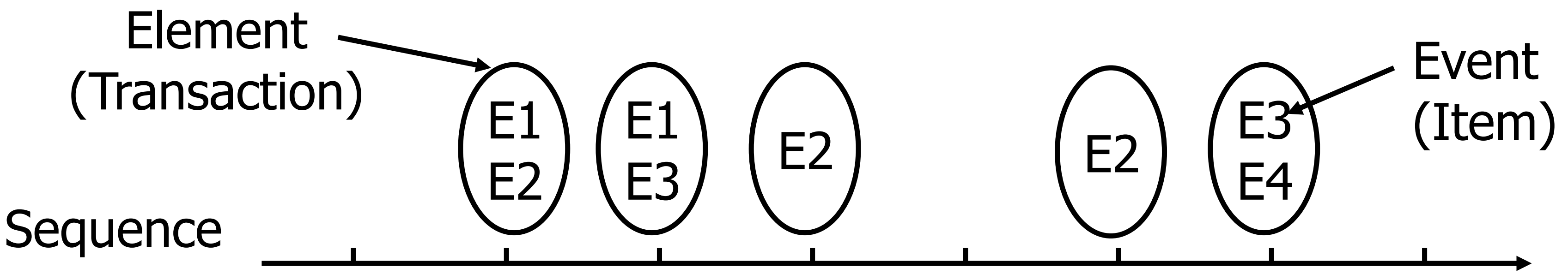
- A sequence is an ordered list of elements (transactions)

  $$s = < e_1\ e_2\ e_3 \ldots >$$

  - Each element contains a collection of events (items)

    $$e_i = \{i_1, i_2, \ldots, i_k\}$$

  - Each element is attributed to a specific *time* or *location*
- Two different measures of the ‘size’ of the sequence:
  - Length of a sequence, |s|, is given by the number of elements of the sequence
  - A k-sequence is a sequence that contains k events (items)
  - Below: a 8-sequence of length 5



Examples of Sequence Data

Sequence Database	Sequence	Ordering by	Element (Transaction)	Event (Item)
Customer	Purchase history of a given customer	Time	A set of items bought by a customer at time t	Books, diary products, CDs, etc
Web Data	Browsing activity of a particular Web visitor	Time	A collection of files/ frames viewed by a Web visitor after a single mouse click	Home page, index page, contact info, etc
Event data	History of events generated by a sensor	Time	Events triggered by a sensor at time t	Types of alarms generated by sensors
Genome sequences	DNA sequence of a particular species	Adjacency in the sequence	An element in the DNA sequence	Bases A,T,G,C
Journey planner	Public transport from A to B at time T	Time and Location	Using vehicle type X between two stops	Entering vehicle, exiting vehicle

Subsequences

- In sequential data mining, the central concept is a subsequence
- A subsequence is *contained* in a sequence it can be obtained from the original sequence by removing events or elements from it
- Formally, a sequence $<a_1\ a_2 \dots a_n>$ is contained in another sequence $<b_1\ b_2 \dots b_m>$ ($m \geq n$) if there exist integers $i_1 < i_2 < \dots < i_n$ such that $a_1 \subseteq b_{i1}$, $a_2 \subseteq b_{i1}$, …, $a_n \subseteq b_{in}$
- Example:

i1=1 i2=2 i3=4

b = {Milk,Bread}{Apples}{Sausages}{Beer,Bread}

a = {Milk}{Apples}{Bread}

Example: Subsequences

- In sequential data mining, the central concept is a subsequence
- Intuitively, a subsequence is contained in a sequence if it can be obtained from the original sequence by removing events or elements from it
- Formally, a sequence $<a_1\ a_2 \ldots a_n>$ is *contained* in another sequence $<b_1\ b_2 \ldots b_m>$ $(m \geq n)$ if there exist integers $i_1 < i_2 < \ldots < i_n$ such that $a_1 \subseteq b_{i1}$, $a_2 \subseteq b_{i1}$, …, $a_n \subseteq b_{in}$

Data sequence	Subsequence	Contain?
< {2,4} {3,5,6} {8} >	< {2} {3,5} >	Yes
< {1,2} {3,4} >	< {1} {2} >	No
< {2,4} {2,4} {2,5} >	< {2} {4} >	Yes

Sequential Pattern Mining

- Consider data set D that contain one or more data sequences
- Each data sequence relates to a particular object (e.g. on the right: A,B or C)
- The *support* of a sequence *s* is the fraction of all data sequences that contain *s*.
- Sequence s is a *frequent sequence* if it is support is greater than user-defined level *minsup*

Object	Timestamp	Events
A	10	2, 3, 5
A	20	6, 1
A	23	1
B	11	4, 5, 6
B	17	2
B	21	7, 8, 1, 2
B	28	1, 6
C	14	1, 8, 7

Sequential Pattern Mining: Task

- Given:
  - a database of sequences
  - a user-specified minimum support threshold, *minsup*
- Task:
  - Find all subsequences with support ≥ *minsup*

Object	Timestamp	Events
A	1	1,2,4
A	2	2,3
A	3	5
B	1	1,2
B	2	2,3,4
C	1	1, 2
C	2	2,3,4
C	3	2,4,5
D	1	2
D	2	3, 4
D	3	4, 5
E	1	1, 3
E	2	2, 4, 5

Minsup = 50%

Examples of Frequent Subsequences:

< {1,2} >	s=60%
< {2,3} >	s=60%
< {2,4}>	s=80%
< {3} {5}>	s=80%
< {1} {2} >	s=80%
< {2} {2} >	s=60%
< {1} {2,3} >	s=60%
< {2} {2,3} >	s=60%
< {1,2} {2,3} >	s=60%

Sequential Pattern Mining: Challenge

- Given a sequence: <{a b} {c d e} {f} {g h i}>
  - Examples of subsequences:

    <{a} {c d} {f} {g} >, < {c d e} >, < {b} {g} >, etc.
- How many k-subsequences can be extracted from a given n-sequence?
  - i.e. how many different ways there is to select 4 items out of 9

$$\text{Answer :}$$

$$\binom{n}{k} = \binom{9}{4} = 126$$

  - Exponential number in the number of items, as in itemset mining!

Sequential Pattern Mining: Challenge

- Number of candidate subsequences is even higher than the number of itemsets for the same set of items (events):
  - An item can appear only once in each itemset, but an event can appear several times in the same sequence (though not in the same element (transaction)
  - Order of items in a sequence does matter so all permutations of elements are considered different
  - Example:
    - 2-itemset {a,b}
    - possible 2-sequences of from the same items: <{a},{a}>,<{a}{b}>,<{b}{a}>,<{b}{b}>,<{a,b}>
    - possible sequences of length two: <{a},{a}>,<{a}{b}>,<{b}{a}>,<{b}{b}>,<{a,b},{a}>,<{a,b}{b}>,<{a,b},{a,b}>,<{a}{a,b}>,<{b}{a,b}>

Sequential Pattern Mining: Challenge

- Consider level-wise candidate generation to find all frequent subsequences (1-sequences, 2-sequences, 3-sequences,...)
- Given n events (items), we get
- Candidate 1-subsequences:

  $<\{i_1\}>, <\{i_2\}>, <\{i_3\}>, \ldots, <\{i_n\}>$
- Candidate 2-subsequences:

  $<\{i_1, i_2\}>, <\{i_1, i_3\}>, \ldots, <\{i_{n-1}, i_n\}>,$

  $<\{i_1\}\ \{i_1\}>, <\{i_1\}\ \{i_2\}>, \ldots, <\{i_n\}\ \{i_n\}>$
- Candidate 3-subsequences: $<\{i_1, i_2, i_3\}>, \ldots, <\{i_{n-2}, i_{n-1}, i_n\}>, <\{i_1, i_2\}\ \{i_1\}>, \ldots, <\{i_{n-1}, i_n\}\{i_n\}>, <\{i_1\}\ \{i_1, i_2\}>, \ldots, <\{i_n\}\ \{i_{n-1}, i_n\}>, \ldots, <\{i_1\}\ \{i_1\}\ \{i_1\}>, \ldots, <\{i_n\}\ \{i_n\}\{i_n\}>$
- Considerably more than the number of candidate itemsets for the same number of items!

Apriori principle for sequences

- **All subsequences of a frequent sequence are frequent**
- Easy to see:
  - if a data sequence of an arbitrary object A contains sequence *s*, it also contains any subsequence *t* of *s*
  - each data sequence that contains s adds to the support counts of s and t
- We can modify Apriori to work on the sequential patterns

Apriori approach for Sequential Pattern Mining

- **Step 1**:
  - Make the first pass over the sequence database D to yield all frequent 1-subsequences
- **Step 2**:

  Repeat until no new frequent sequences are found
  - **Candidate Generation**:
    - Merge pairs of frequent subsequences found in the (k-1)*th* pass to generate candidate sequences that contain k items
  - **Candidate Pruning**:
    - Prune candidate *k*-sequences that contain infrequent *(k-1)*-subsequences
  - **Support Counting**:
    - Make a new pass over the sequence database D to find the support for these candidate sequences
  - **Candidate Elimination**:
    - Eliminate candidate *k*-sequences whose actual support is less than *minsup*

Sequential Apriori: Overview

Frequent 3-sequences

< {1} {2} {3} >
< {1} {2 5} >
< {1} {5} {3} >
< {2} {3} {4} >
< {2 5} {3} >
< {3} {4} {5} >
< {5} {3 4} >

Candidate Generation

< {1} {2} {3} {4} >
< {1} {2 5} {3} >
< {1} {5} {3 4} >
< {2} {3} {4} {5} >
< {2 5} {3 4} >

Candidate Pruning

< {1} {2 5} {3} >

Candidate generation in sequential Apriori

- Merging two frequent 1-sequences $<\{i_1\}>$ and $<\{i_2\}>$ will produce three candidate 2-sequences: $<\{i_2\}\{i_1\}>$, $<\{i_1\}\{i_2\}>$ and $<\{i_1, i_2\}>$
- For k>2 the algorithm checks whether the sequences can be superimposed so that the 'middle' part is shared
  - let (k-1)-subsequence s_1 be the suffix of f_1 obtained by dropping the first event and let (k-1)-subsequence p_1 be the prefix of f_2 obtained by dropping the last event of f_2
  - if $p_2 = s_1$, f_1 is merged with f_2
    - <{1}{2}{3}> and <{2}{3}{4} can be merged into <{1}{2}{3}{4}>
    - <{1,5}{3}> and <{5}{3,4}} can be merged into <{1,5}{3,4}
    - <{1}{2}{3}> and <{1}{2}{5}} cannot be merged

Candidate generation

- The element structure of the middle part of the merged sequence is the same as the element structure in both s_1 and s_2.
- First element of the merged sequence will be the first element of the first sequence
- Last element of the merged sequence will be the last element of the second sequence
- e.g.
  - <{1}{2}{3}> and <{2}{3}{4} are merged to <{1}{2}{3}{4}>
    - <{1}{2}{3,4}>, <{1,2}{3}{4}>,... not generated this way,
  - <{1,5}{3}> and <{5}{3,4}} are merged into <{1,5}{3,4}>
    - <{1}{5}{3,4}>, <{1,5}{3}{4}>,<{1}{5}{3}{4}>,... not generated this way

Completeness of candidate generation

- Are all candidates generated by this approach?
- Given an arbitrary frequent k-sequence $s = <E_1,...,E_L>$ of length L the two frequent k-1 sequences s_1 and s_2 that are merged to produce s are the following
- Case k = 2: two subcases based on the structure of s:
  - If $s = <\{i,j\}>$ we have $s_1 = <\{i\}>$, $s_2 = <\{j\}>$
  - If $s = <\{i\}\{j\}>$ we also have $s_1 = <\{i\}>$, $s_2 = <\{j\}>$
- Case k > 2:
  - If E_l contains more than one event $s_1 =<E_1,...,E_{L-1},E'>$, where E' is obtained from E_l by dropping the last event, otherwise $s_1=<E_1,...,E_{L-1}>$
  - If E_1 contains more than one event $s_2 =<E'',...,E_L>$, where E'' is obtained from E_1 by dropping the first event, otherwise $s_2=<E_2,...,E_L>$

Candidate pruning & support counting

- Analogous principle to itemset Apriori
- Given a candidate k-sequence, we check if any of the k-1 subsequences are infrequent:
- e.g. 4-sequence <{1}{2}{3}{4}>
  - we know that <{1}{2}{3}> and <{2}{3}{4}> are frequent since they were used to generate the 4-sequence
  - we need to check is <{1}{2}{4}> and <{1}{3}{4}> are frequent
- If any infrequent subsequence is found the candidate is pruned
- Support counting is then performed for the remaining candidates and candidates below the *minsup* threshold are discarded

Timing constraints

- In some applications, relative timing of the transactions is crucial to define the pattern
- e.g. Consider a credit card company wanting to mine unusual patterns in purchasing behavior:
  - A fraudulent user of the card could easily buy similar items as the normal users would do, so the sequence of transactions might not discriminate enough
  - But the fraudulent user would do the purchases in short time interval to make maximum use of the card before it is close
- Constraining the patterns in temporal dimension is required to mine such patterns

Importance of timing: Examples

- Web sequence:

  < {Homepage} {Electronics} {Digital Cameras} {Canon Digital Camera} {Shopping Cart} {Order Confirmation} {Return to Shopping} >

  - Probably interesting only if happens *during a single session*

- Sequence of initiating events leading to the Three-Mile Island Nuclear Accident:

< {clogged resin} {outlet valve closure} {loss of feedwater} {condenser polisher outlet valve shut} {booster pumps trip} {main waterpump trips} {main turbine trips} {reactor pressure increases}>

  - Probably only relevant if all events happen *within 24 hours*

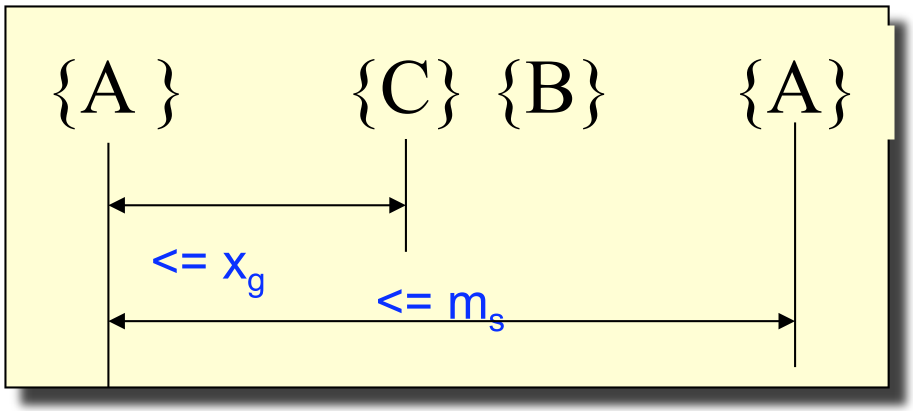
- Credit card database:

  <{Clothing Shop, 500€}{Jewellery shop, 500€}{Restaurant, 300€}>

  - Perhaps more alarming if happens *during a single day*

Timing Constraints

- We consider two kinds of constraints:
  - max-span constraint (m_s): maximum allowed time between the first element and the last element in the sequence
  - max-gap constraint (x_g): maximum length of a gap between two consecutive element

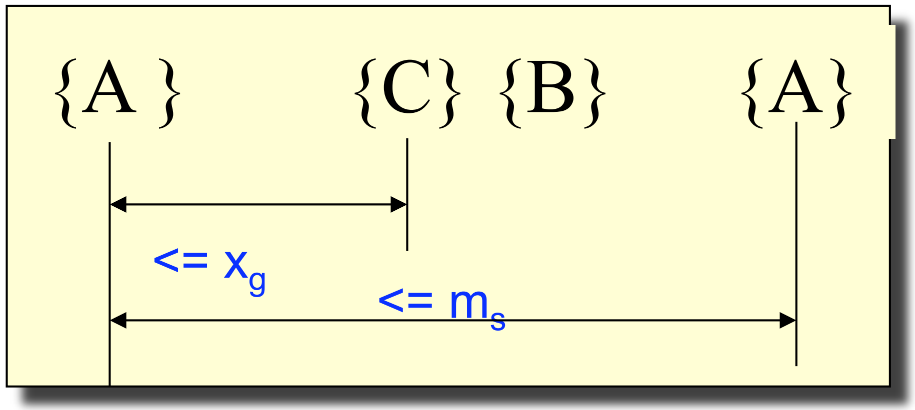


x_g: max-gap

m_s: maximum span

Timing Constraints: Example

- Assume parameters: $x_g = 2$, $n_g = 0$, $m_s = 4$
- Consider the data sequences below with element time stamps 1,2,3,...



Data sequence	Subsequence	Contain?
< {2,4} {3,5,6} {4,7} {4,5} {8} >	< {6} {5} >	Yes
< {1} {2} {3} {4} {5}>	< {1} {4} >	No (x_g=3)
< {1} {2,3} {3,4} {4,5}>	< {2} {3} {5} >	Yes
< {1,2} {3} {2,3} {3,4} {2,4} {4,5}>	< {1,2} {5} >	No (m_s=5)

Mining Sequential Patterns with Timing Constraints

- Approach 1:
  - Mine sequential patterns without timing constraints
  - Postprocess the discovered patterns
- Approach 2:
  - Modify the mining process to prune candidates that violate timing constraints during candidate generation
  - Question:
    - Does Apriori principle still hold?

Apriori Principle for Sequence Data

Object	Timestamp	Events
A	1	1,2,4
A	2	2,3
A	3	5
B	1	1,2
B	2	2,3,4
C	1	1, 2
C	2	2,3,4
C	3	2,4,5
D	1	2
D	2	3, 4
D	3	4, 5
E	1	1, 3
E	2	2, 4, 5

Suppose:

$x_g = 1$ (max-gap)

$m_s = 5$ (maximum span)

minsup = 60%

<{2} {5}> support = 40%

but

<{2} {3} {5}> support = 60%

Problem exists because of max-gap constraint!

Contiguous Subsequences

- The non-monotonicity caused by the maxgap constraint can be circumvented by considering contiguous subsequences
- Examples: s = < {1} {2} >
  - is a contiguous subsequence of

    < {1} {2 3}>, < {1 2} {2} {3}>, and < {3 4} {1 2} {2 3} {4} >
  - is not a contiguous subsequence of

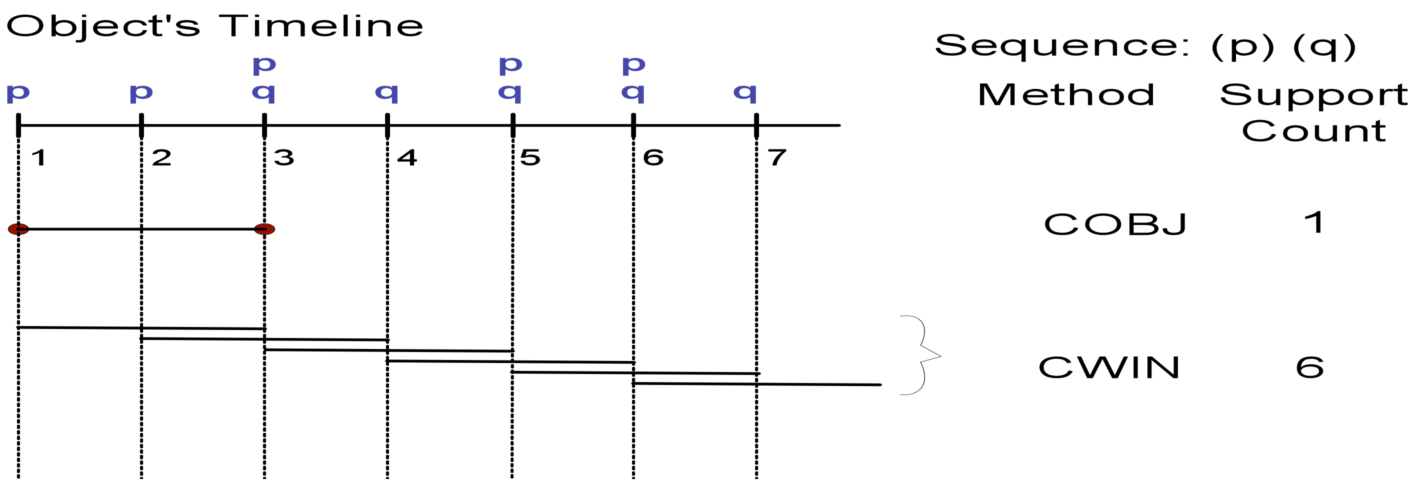
    < {1} {3} {2}> and < {2} {1} {3} {2}>
- A k-1-sequence t is a contiguous subsequence of k-sequence k if t can be constructed by
  - deleting events from the elements of s
  - while not allowing middle elements to get empty

Modified Sequential Apriori for timing constraints

- Modified Apriori principle: If a k-sequence is frequent, then all of its contiguous k-1-subsequences are frequent
- Modified algorithm:
  - Candidate generation step remains the same: we merge two frequent k-1 sequences that have the same middle part (excluding first and last event)
  - In Candidate pruning, we only need to verify contiguous k-1-sequences
    - e.g. Given 5-sequence <{1}{2,3}{4}{5}> we need to verify <{1}{2}{4}{5}>,<{1}{3}{4}{5}> and need not to verify <{1}{2,3}{5}>
  - In support counting need to check that *maxspan* constraint is not violated

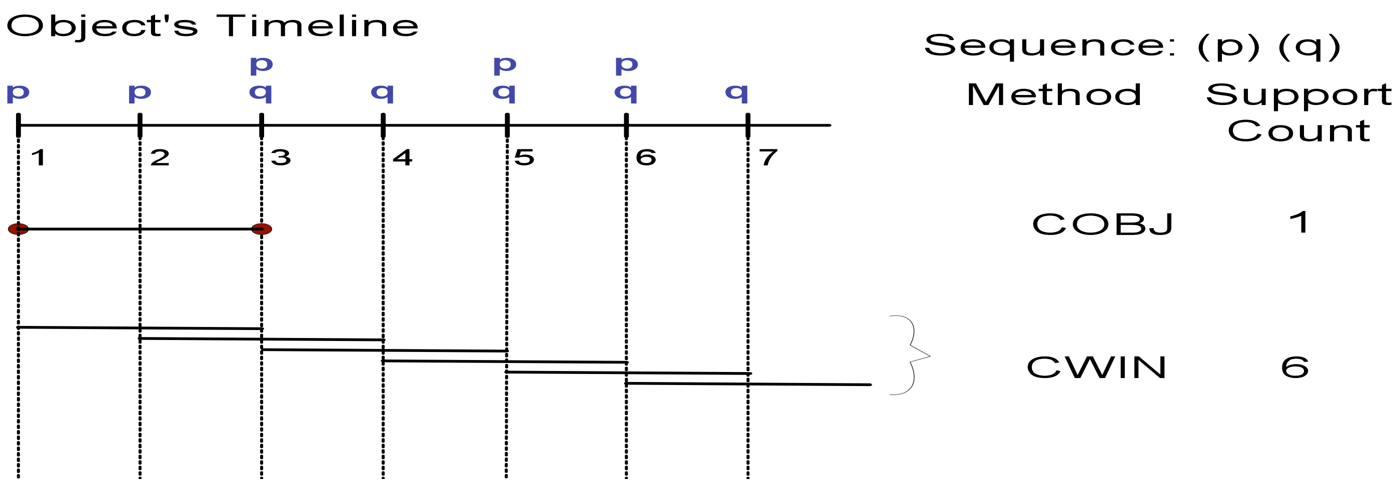
Support of a sequential pattern

- Support of a sequential pattern is not as clear cut as itemset support, due to the repetition of the items in the data sequence
- Many choices, two most important are
  1. One occurrence per object: ‘Customer X has bought Bread and then Milk’ in some *maxspan*=7-day interval
  2. One occurrence per sliding window: ‘Customer X has bough Bread and then Milk in 7-day interval in five occasions’



Support of a sequential pattern

- Important: the baseline 'N' for determining the support depends on the counting method
  - One occurrence per object: N = the number of objects (e.g. Customers)
  - One occurrence per sliding window: N = the number of possible positions for the sliding window in all objects



Text mining

- Text databases are an important form of sequential data
  - News databases
  - Blog archives
  - Scientific journals and abstract databases
- Many tasks:
  - Text categorization
  - Concept/entity extraction,
  - Sentiment analysis,
  - Document summarization, etc.
- How can frequent pattern mining help?

Phrases in text

- Two general types of phrases can be defined:
- Syntactical phrases: governed by the grammar of the language
  - noun phrases: ‘a green ball’,
  - verb phrases: ‘saw a ball’
  - ...not in the scope of this course
- Statistical phrases
  - frequent n-grams (frequent n-sequences of consecutive words) – basic tool in text analysis
  - frequent word sequences
    - of any length, gaps allowed
  - ...this we can do!

Finding frequent phrases in text

1. The Congress subcommittee backed away from mandating specific **retaliation against foreign** countries for **unfair foreign trade practices.**
2. He urged Congress to reject provisions that would mandate U.S. **retaliation against foreign unfair trade practices.**
3. Washington charged France West Germany the U.K. Spain and the EC Commission with **unfair practices** on behalf of Airbus.

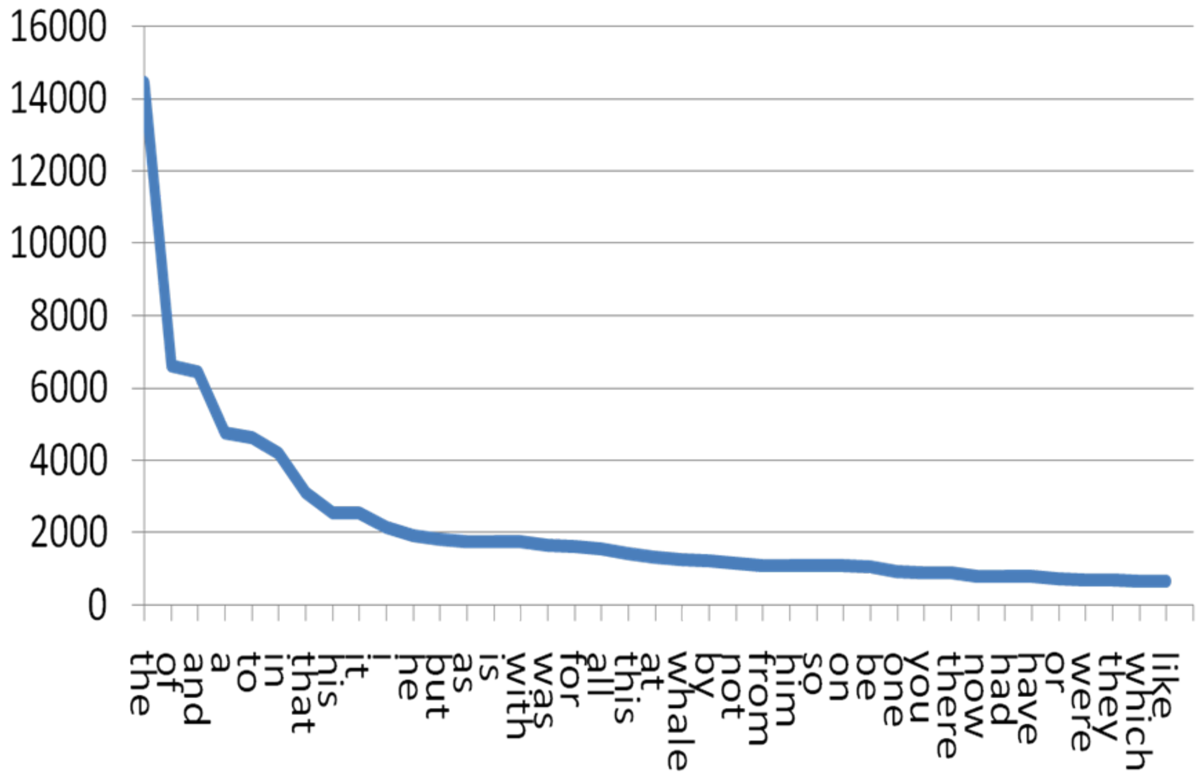
- Possible goal: find frequent phrases that capture topics among the documents

Finding frequent phrases

- The machinery for sequential pattern mining can be applied in principle
- We take documents as data sequences
  - Words as items (events),
  - Transactions (elements) consist of single words
  - Timestamp from the word order in the document
- Preprocessing phase is needed:
  - very common words are removed
  - some punctuation may be removed
  - numbers removed or converted
  - stemming
    - countries -> countr

Skewed support of natural language

- Word frequencies in ‘Moby Dick’: Top 20 words are ‘stop words’ i.e. generic words with little content
- Typical approach in text analysis is to remove such words



Finding maximal frequent sequences

- Example Document: ‘The Federal Reserve entered the U.S. Government securities market to arrange1.5 billion dlrs of customer repurchase agreements, a Fed spokesman said. Dealers said Federal funds were trading at 6- 3/16 pct when the Fed began its temporary and indirect supply of reserves to the banking system.’
- Maximal frequent sequences: federal reserve entered u.s. government securities market arrange repurchase agreements fed dealers federal funds trading fed began temporary supply reserves banking system (22 words)
- Paper #3: H. Ahonen-Myka: Finding all maximal frequent sequences in text. ICML-99 Workshop: Machine Learning in Text Data Analysis, 1999, pp. 11--17