# Fast Contextual Preference Scoring of Database Tuples


Kostas Stefanidis

Department of Computer Science,
University of Ioannina, Greece

Joint work with Evaggelia Pitoura

http://dmod.cs.uoi.gr

# Motivation

Today, a heterogeneous population of users have access to a heterogeneous collection of data

Personalization systems allow users to indicate preferences about interesting data items

Two different approaches for expressing preferences:

- The qualitative approach
- The quantitative approach

# Motivation

Two different approaches for expressing preferences:

- The qualitative approach
  - Preferences are specified using preference relations

    example: I prefer science fiction to western movies

 > 

- The quantitative approach
  - Preferences are expressed by using scoring functions

    example: I give to science fiction movies the interest score 0.9 and to western movies the score 0.5

F() = 0.9      F() = 0.5

# Motivation

But many times preferences vary depending on the circumstances

That is, preferences are context-dependent

For instance:

- I prefer to watch cartoons when with kids
  example: preference (kids, cartoons, 0.9)
- I prefer horror movies when with friends
  example: preference (friends, horror, 0.8)

HOW MUCH
score

WHICH STATE
Context Specification

WHICH TUPLES
Database Predicates

# Topic

Given a set of context-dependent preferences

find the tuples with the highest scores

# Motivation

query → select relevant preferences → sort tuples → return top-k results

online query processing time

<u>full pre-computation</u>

pre-compute a ranking for each context state → query → select ranking

offline computation

online query processing time

# Motivation

query → select relevant preferences → sort tuples → return top-k results

online query processing time

group preferences → pre-compute a ranking for each group → query → select ranking

offline computation

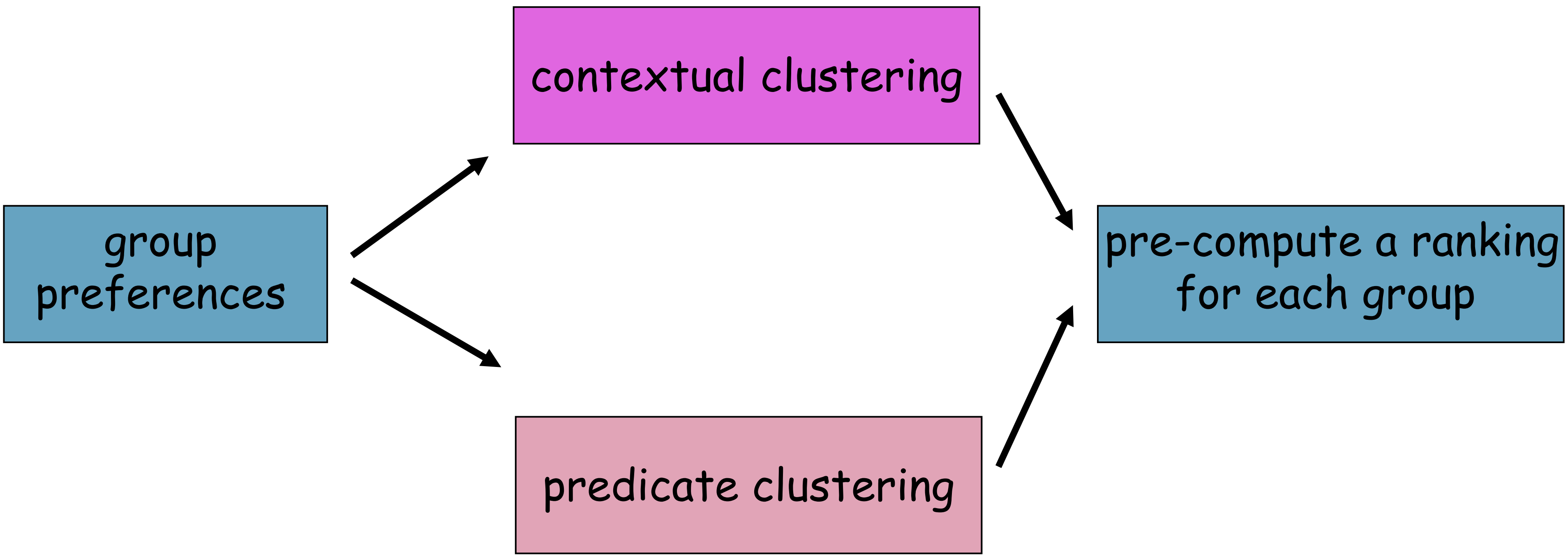
online query processing time

pre-compute a ranking for each context state → query → select ranking

offline computation

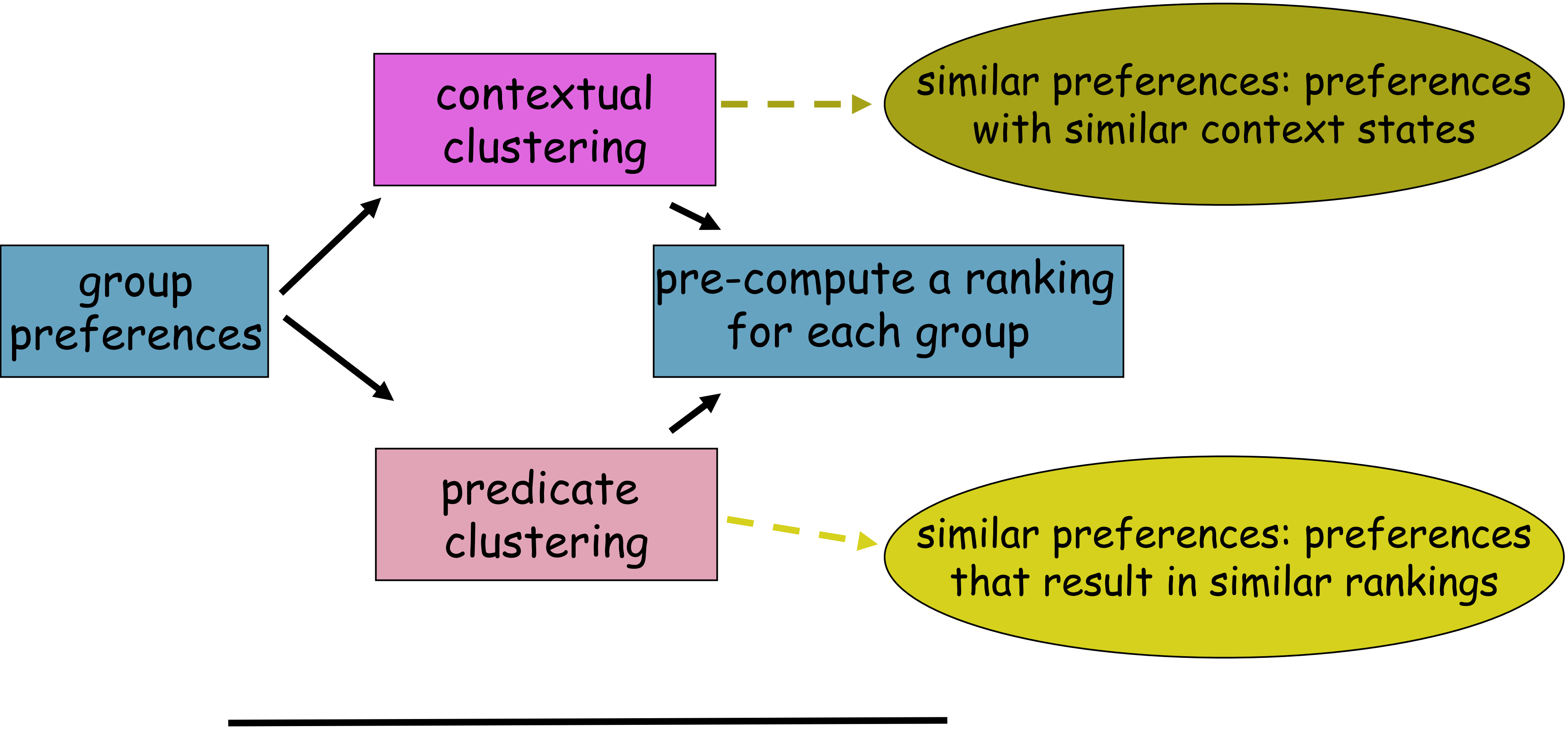
online query processing time

# Overview




offline computation

# Overview




offline computation

# Example

Movies database

| Title | Year | Director | Genre | Language | Duration |
|---|---|---|---|---|---|

Context parameters: user, accompanying people, time period, mood

Examples: I would like to watch thrillers when with friends
I enjoy seeing cartoons when with kids during holidays

# Outline

- Modeling Context
- Contextual Preferences
- Grouping Preferences
  - Contextual Clustering
  - Predicate Clustering
- Evaluation
- Summary

# Outline

- Modeling Context
- Contextual Preferences
- Grouping Preferences
  - Contextual Clustering
  - Predicate Clustering
- Evaluation
- Summary

# Modeling Context

Context is modeled through a finite set of special-purpose attributes, called context parameters ($C_i$)

Two types of context parameters:

- Simple: involves a single context attribute
  - Examples: accompanying people, time period, mood
- Composite: consists of a set of single context attributes
  - Examples: user consists of id, age, gender

Each application X has a context environment $CE_X$ which is a set of n context parameters $\{C_1, C_2, \ldots, C_n\}$

(Movies example): CE = {user, accompanying people, time period, mood}

A context state corresponds to an assignment of values to context parameters

(Movies example): cs = ((id1, youth, male), family, holidays, good)

# Modeling Context

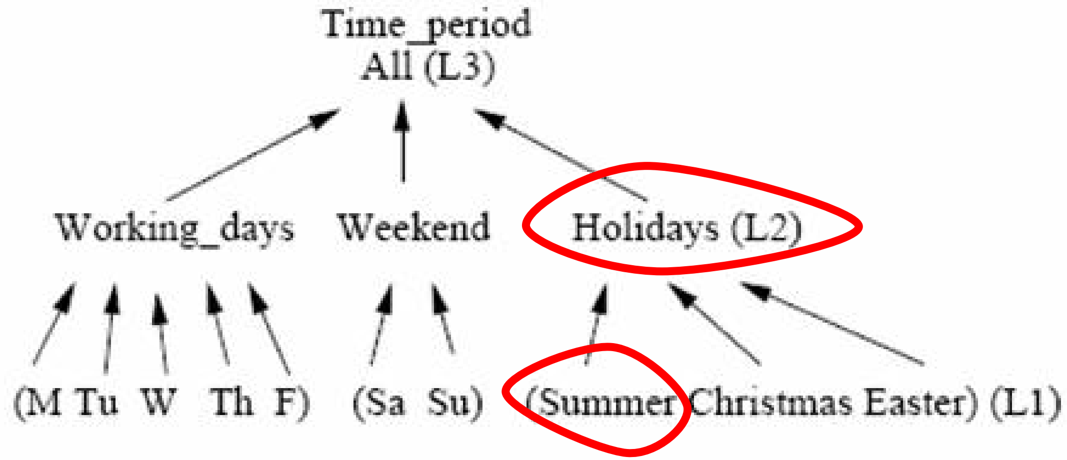
We model context attributes as multidimensional hierarchical attributes

- Each context attribute participates in an associated hierarchy of levels of aggregated data, i.e. it can be expressed with different levels of detail

This allows users to express preferences at various levels of detail

Example: a user can denote different preferences for summer than for holidays, in general

If there is no value for a context attribute, the value All is assumed



# Outline

- Modeling Context
- Contextual Preferences
- Grouping Preferences
  - Contextual Clustering
  - Predicate Clustering
- Evaluation
- Summary

# Contextual Preferences

A contextual preference is a triple (context state, predicate, score), where a predicate specifies conditions on the values of the database attributes

context

predicate

Preference example:

- ((($id_1$, youth, male), friends, holidays, good), (genre = comedy), 0.9)

score

Note: a preference may not depend on all context attributes

- (((All, youth, All), All, holidays, All), (genre = comedy), 0.9)

# No Related Preference

In a given set of preferences, there may be no related preference for a tuple under a context state

- These tuples are assigned a default score of 0
  - We consider preferences expressed by users to be indicators of positive interest

An unrated tuple is less important than any other tuple for which the user has expressed some interest

# More than one Preference

In some cases, there may be more than one preference applicable to a specific database tuple, under the same context

- To compute the score of a tuple at a given context state, we consider only the most specific predicates
- If more than one most specific predicate, the score of a tuple is the maximum score among the relative preferences

# More than one Preference

Database instance

| | Title | Year | Director | Genre | Language | Duration |
|---|---|---|---|---|---|---|
| $t_1$ | Casablanca | 1942 | Curtiz | Drama | English | 102 |
| $t_2$ | Psycho | 1960 | Hitchcock | Horror | English | 109 |
| $t_3$ | Schindler's List | 1993 | Spielberg | Drama | English | 195 |

Example preferences:

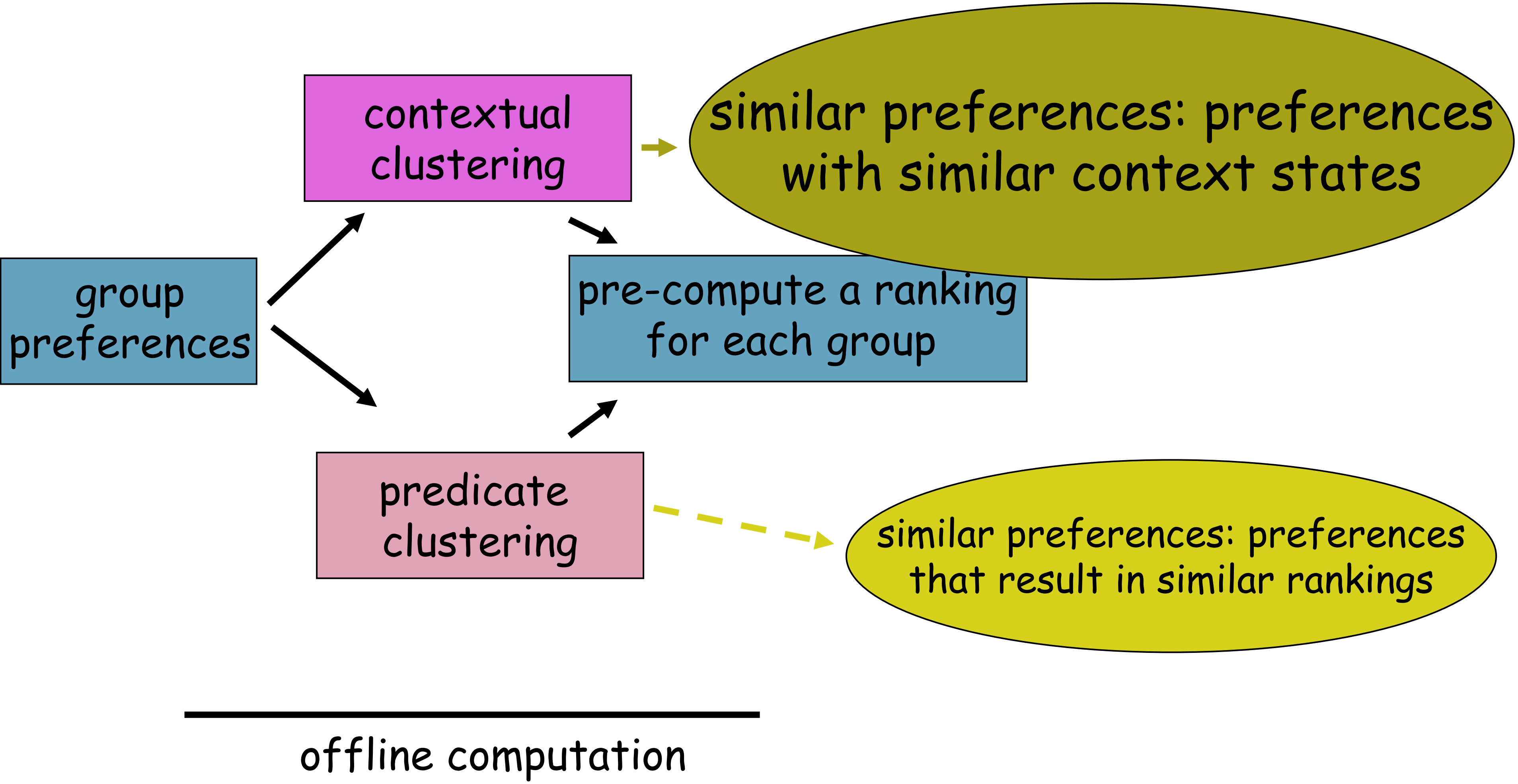
$p_1$ = ((friends), genre = horror, 0.8)
$p_2$ = ((friends), director = Hitscock, 0.7)
$p_3$ = ((alone), genre = drama, 0.9)
$p_4$ = ((alone), (genre = drama and director = Spielberg), 0.5)

- Under context friends, both $p_1$ and $p_2$ are applicable to $t_2$
  - No predicate subsumes the other and the score for $t_2$ is the maximum of the two scores, namely 0.8
- Under context alone, both $p_3$ and $p_4$ are applicable to $t_3$
  - The predicate of $p_4$ subsumes the predicate of $p_3$, and so, $t_3$ has score 0.5

# Outline

- Modeling Context
- Contextual Preferences
- Grouping Preferences
  - Contextual Clustering
  - Predicate Clustering
- Evaluation
- Summary

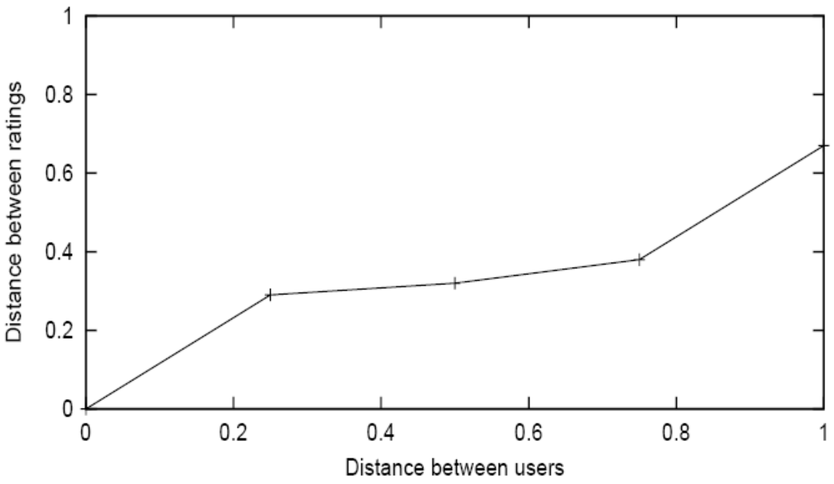
# Contextual Clustering



# Why Contextual Clustering?

The idea of contextual clustering is based on the premise that preferences for similar context states produce similar scores

- We used a real dataset of movie ratings to show that the distance between ratings increases as the distance between users increases




- For users, there is information available of the form (user-id, sex, age, occupation) that we use as our context environment
- We constructed simple predicates that involve the genre of the movies by averaging the rates assigned by each user to movies of each genre

The distance between ratings increases as the distance between users (i.e. context) increases

We need to define distances between context states

How?

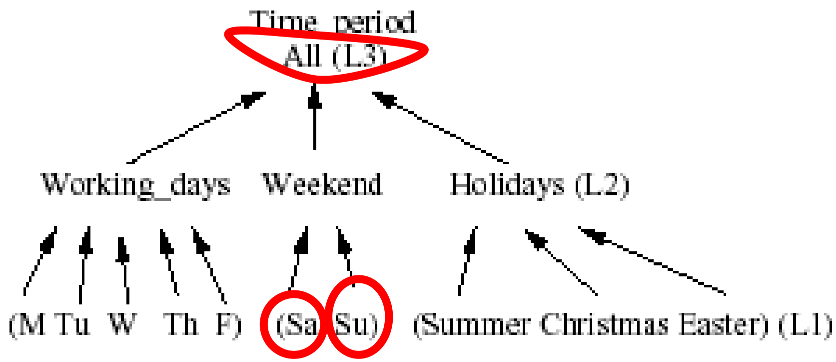
# Outline

- Modeling Context
- Contextual Preferences
- Grouping Preferences
  - Contextual Clustering
    - Similarity between Context Values
    - Similarity between Context States
  - Predicate Clustering
- Evaluation
- Summary

# Similarity between Context Values

Find the length of the minimum path that connects them in their hierarchy

(path distance)




This method may not be accurate, when applied to attributes with large domains and many hierarchy levels, e.g. smaller path lengths for less similar values

- Time period hierarchy: Saturday, Sunday has the same path distance with Saturday, All, it would probably make sense for Saturday to be more similar to Sunday than All

Following related research on defining semantic similarity between terms

- We take into account both their path distance ($dist_P(c_1, c_2)$) and the depth of the hierarchy levels ($dist_D(c_1, c_2)$) that the two values belong to

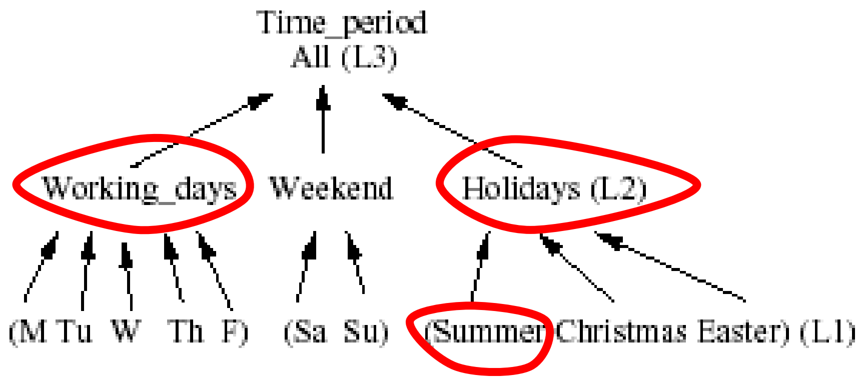
# Overall Value Distance

The overall value distance between two context values $c_1$, $c_2$ is computed as:

$$dist_V(c_1, c_2) = dist_P(c_1, c_2) \times dist_D(c_1, c_2)$$

Simple examples:

- Assume the values working days and summer. Their path distance is 0.95, their depth distance is 1 and so, their overall value distance is 0.95
- Given now, the values holidays and summer their value distance is 0.39,
- Therefore, the value summer is more similar to holidays than to working days (in both examples, $\alpha = \beta = 1$)



# State Distance

The state distance between two context states $cs^1 = (c_1^1, \ldots, c_n^1)$ and $cs^2 = (c_1^2, \ldots, c_n^2)$ is defined as: $dist_S(cs_1, cs_2) = \sum_{i=1}^{n} w_i \times dist_V(c_i^1, c_i^2)$ , where each $w_i$ is a context parameter specific weight

Each weight takes a value according to the cardinality of its related context parameter domain

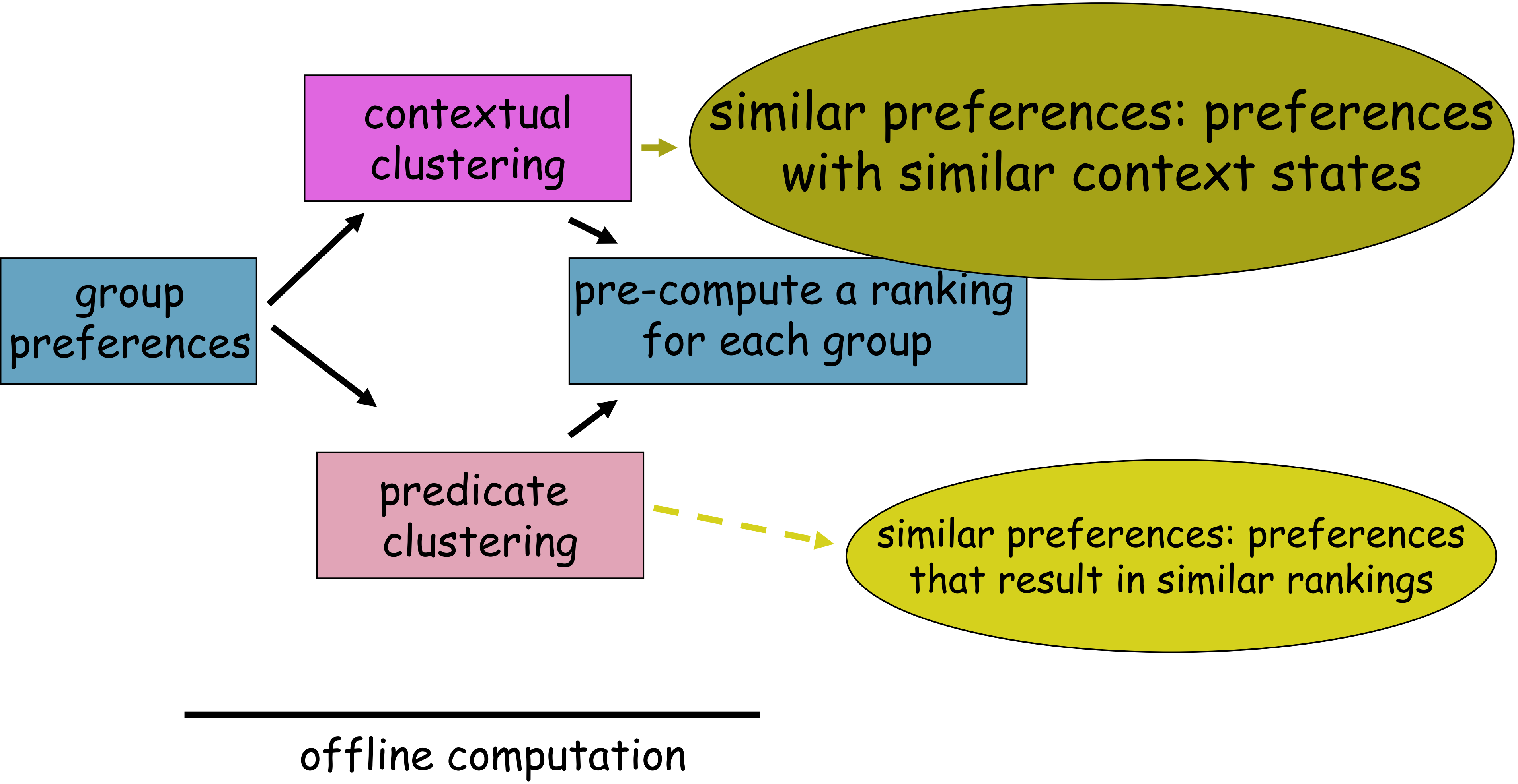
- We consider a higher degree of similarity among values that belong to a large domain

# Contextual Clustering

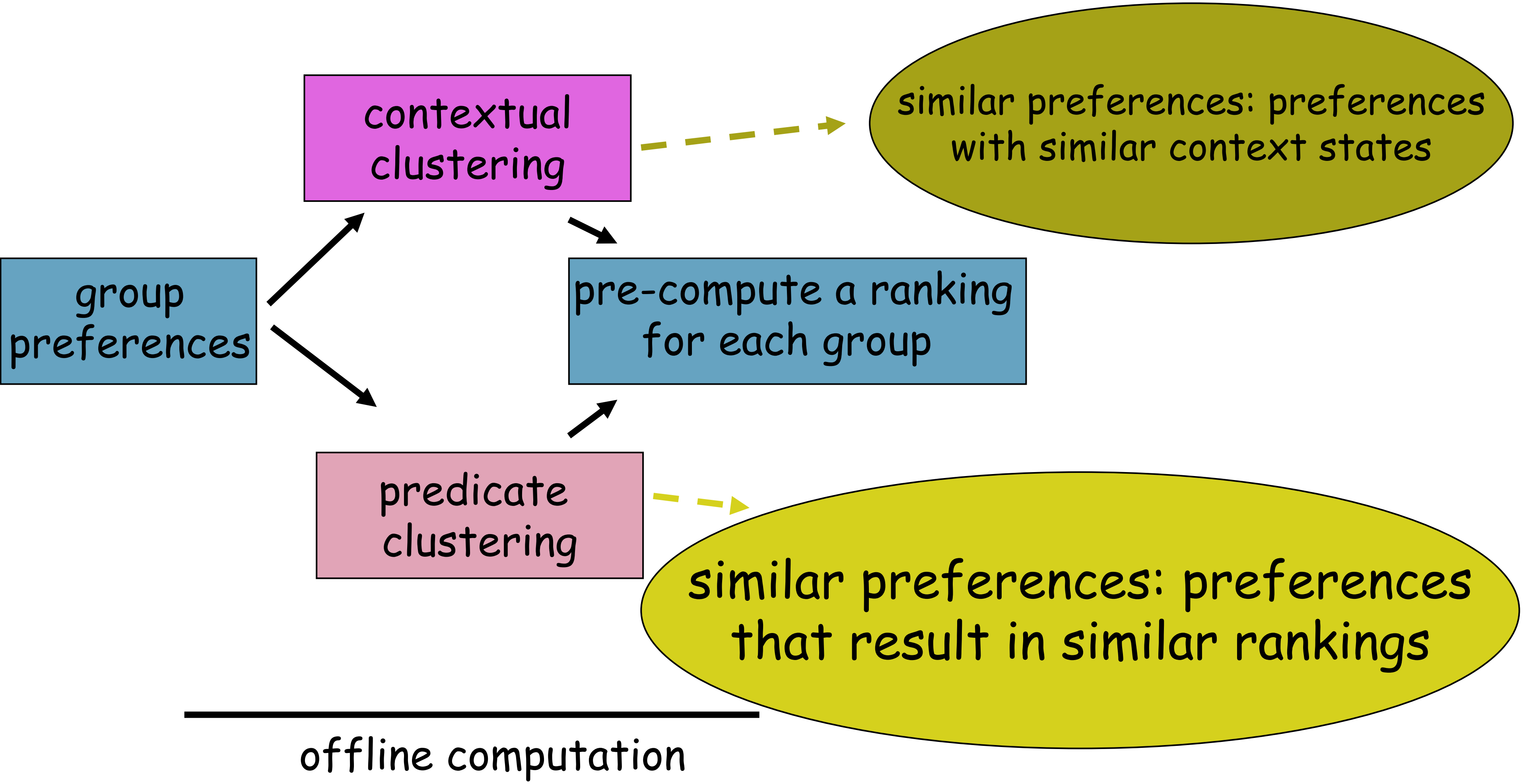
To group preferences with similar context states, we use a hierarchical clustering method that follows a bottom-up strategy

- Initially, each context state is placed in its own cluster
- At each step, merges the two clusters with the smallest distance
  - The distance is defined as the maximum distance between any two states that belong to these clusters
- The algorithm terminates when the closest two clusters have distance greater than $d_{cl}$
  - $d_{cl}$ is an input parameter
- Finally, for each produced cluster, we select as representative the state in the cluster that has the smallest total distance from all the states of its cluster

# Contextual Clustering



# Predicate Clustering



# Predicate Clustering

Predicate clustering aims at grouping together preferences that produce similar scores for most database tuples, i.e. groups together preferences that have similar predicates and scores

To do this, we introduce a bitmap representation of preferences through a matrix whose size depends on the desired precision of the resulting scoring

# Predicate Matrix

First step: create a bitmap matrix for each context state

- One column for each preference predicate
- One row for each score

Preference examples (Movies database):

- $p_1$ = (friends, genre = horror, 0.8)
- $p_2$ = (friends, director = Hitscock, 0.7)
- $p_3$ = (friends, director = Spielberg, 0.65)

If two matrices of two states are the same, then all tuples have the same scores for these states

friends

| | genre = horror | director = Hitscock | director = Spielberg |
|---|---|---|---|
| 0.8 | 1 | 0 | 0 |
| 0.7 | 0 | 0 | 1 |
| 0.65 | 0 | 1 | 0 |

Matrices can be very large, and so, we define approximations

# Predicate Representation

Preference examples:

$p_1$ = (friends, genre = horror, 0.8)

$p_2$ = (friends, director = Hitscock, 0.7)

$p_3$ = (friends, director = Spielberg, 0.65)

friends

| | genre = horror | director = Hitscock | director = Spielberg |
|---|---|---|---|
| 0.8 | 1 | 0 | 0 |
| 0.7 | 0 | 0 | 1 |
| 0.65 | 0 | 1 | 0 |

friends

| | genre = horror | director = Hitscock | director = Spielberg |
|---|---|---|---|
| 0.8 | 1 | 0 | 0 |

friends

| | genre = horror | director = Hitscock | director = Spielberg |
|---|---|---|---|
| 0.6 | 1 | 1 | 1 |

# Overall Predicate Representation

Preference examples:

$p_1$ = (friends, genre = horror, 0.8)

$p_2$ = (friends, director = Hitscock, 0.7)

$p_3$ = (friends, director = Spielberg, 0.65)

friends

| | genre = horror | director = Hitscock | director = Spielberg |
|---|---|---|---|
| 0.8 | 1 | 0 | 0 |

| | genre = horror | director = Hitscock | director = Spielberg |
|---|---|---|---|
| 0.6 | 1 | 1 | 1 |

friends

| | genre = horror | director = Hitscock | director = Spielberg |
|---|---|---|---|
| 0.8 | 1 | 0 | 0 |
| 0.6 | 1 | 1 | 1 |

The number of bits that two predicate matrices differ at is an indication of the number of tuples that they rank differently

# Distance between Predicate Matrices

Preference examples:

$p_1$ = (friends, genre = horror, 0.8)

$p_2$ = (friends, director = Hitscock, 0.7)

$p_3$ = (friends, director = Spielberg, 0.65)

friends

| | genre = horror | director = Hitscock | director = Spielberg |
|---|---|---|---|
| 0.8 | 1 | 0 | 0 |
| 0.6 | 1 | 1 | 1 |

$p_4$ = (alone, genre = horror, 0.7)

$p_5$ = (alone, director = Spielberg, 0.6)

alone

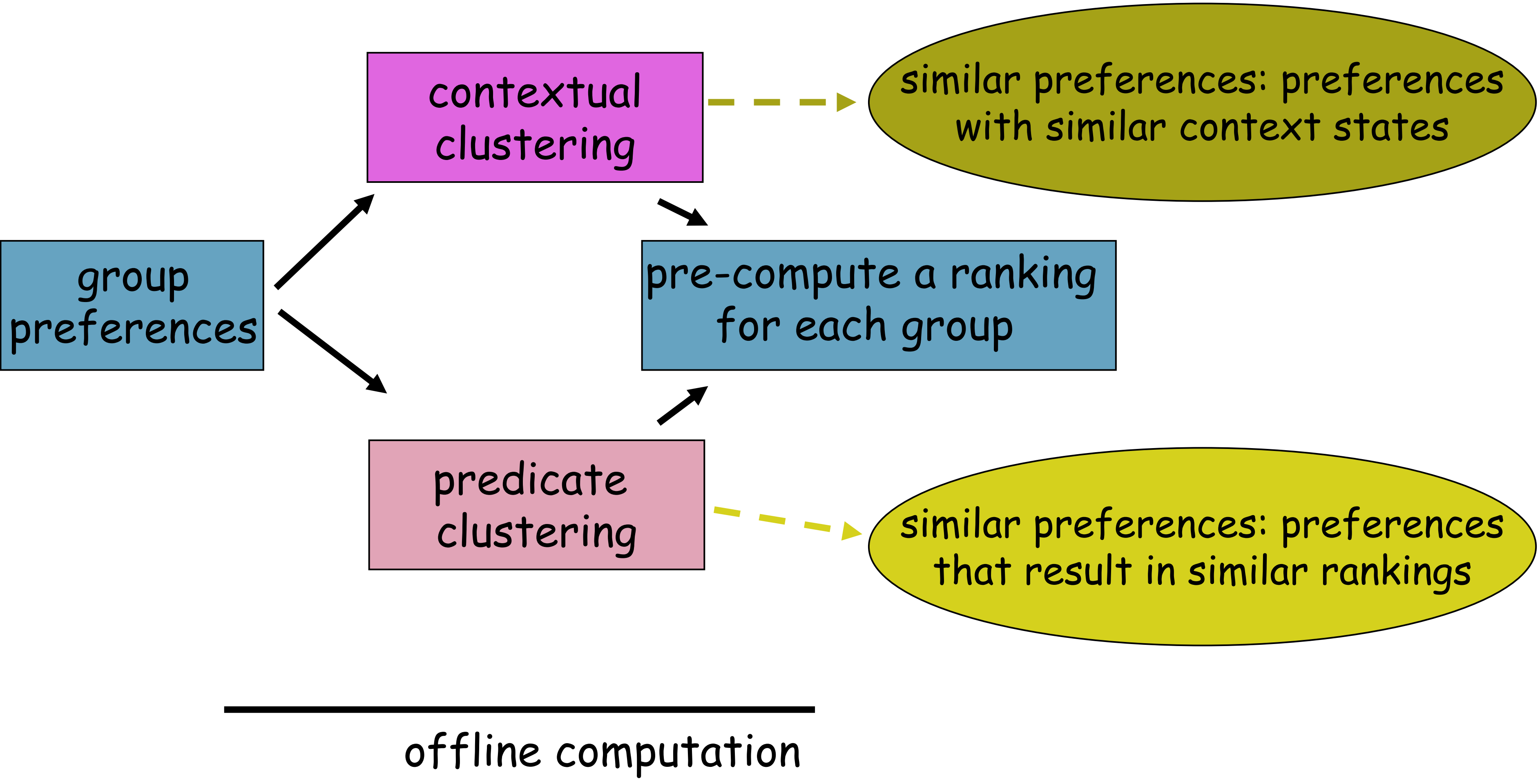
| | genre = horror | director = Hitscock | director = Spielberg |
|---|---|---|---|
| 0.8 | 0 | 0 | 0 |
| 0.6 | 1 | 0 | 1 |

# Predicate Clustering

We create clusters of preferences that result in similar scorings of tuples using distances among predicate matrices

- We use the previous algorithm with a simple modification on how to merge two clusters
- Initially, the preferences with a specific context state are placed in a cluster
- At each step, we merge the two clusters with the smallest distance
- The distance between two clusters is defined as the maximum distance between any two predicate representation matrices of context states that belong to these clusters

# Grouping Similar Preferences



# Grouping Similar Preferences

Two different ways:

- Contextual clustering:
  - To compute distances we exploit the hierarchical nature of context attributes
- Predicate clustering:
  - To compute distances uses similar predicates and scores

# Aggregate Scores

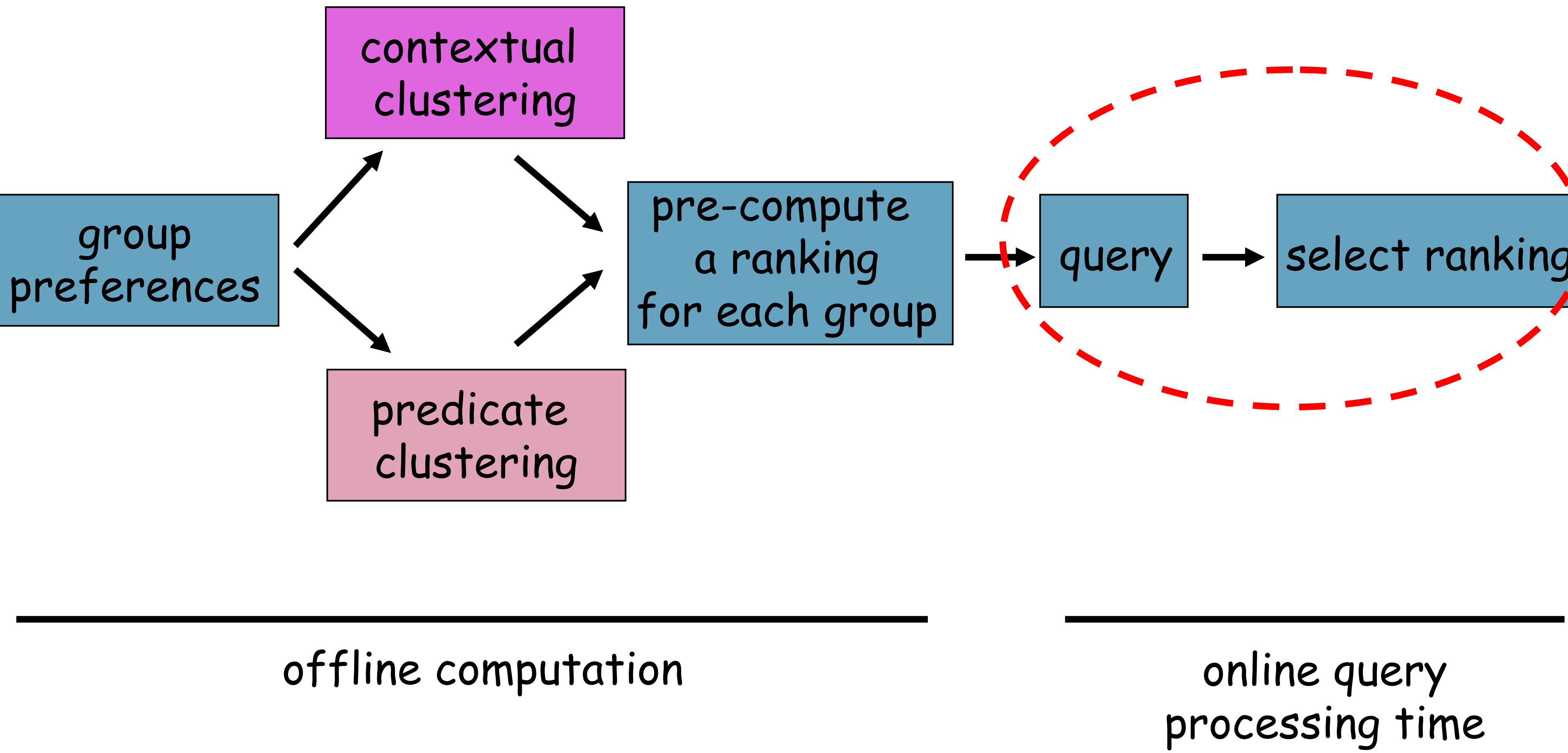
Having created the clusters of preferences, we compute for each of them an aggregate score for each tuple specified in any of its preferences

This score is no less than the score computed using any of the context states belonging to the cluster

For each produced cluster $cl_i$, we maintain a relation table $cl_i$Scores(tuple_id, score)

- We store in decreasing order only the nonzero scores of tuples

# Online Phase



# Outline

- Modeling Context
- Contextual Preferences
- Grouping Preferences
  - Contextual Clustering
  - Predicate Clustering
- Evaluation
- Summary

# Quality of Results

We evaluate the quality of the returned top-k results for a query

- Results(d-max) is the set of the top-k tuples computed using clustering
- Results(opt) is the set of top-k tuples computed using the context states that are most similar to the query without pre-computation

We compare these two sets using the Jaccard coefficient defined as:

$$\frac{|\,\mathrm{Re}\,sults(d-\max)\cap \mathrm{Re}\,sults(opt)\,|}{|\,\mathrm{Re}\,sults(d-\max)\cup \mathrm{Re}\,sults(opt)\,|}$$

The higher its value, the more similar the two top-k tuple sets

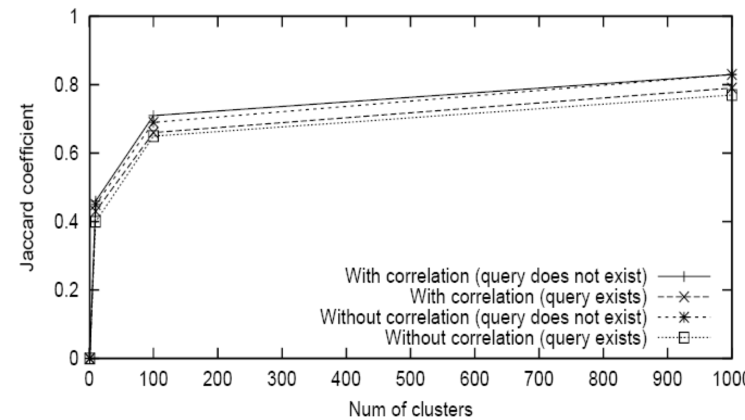
We consider two kinds of queries:

- Queries whose context state is included in the preferences
- Queries whose context state is not in the preferences, thus, a similar one is used

# Quality of Results – Synthetic Preferences

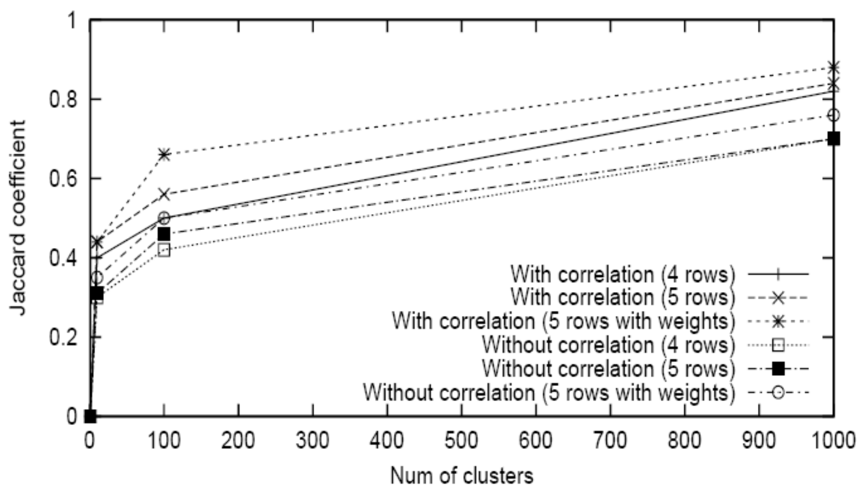
Results of the contextual clustering approach (with and without correlation)

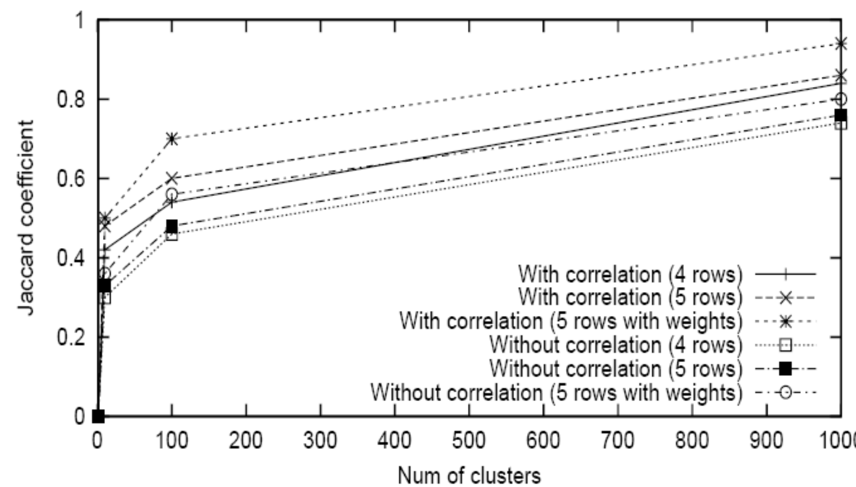
- When the query states do not exist in the preferences, the Jaccard coefficient increases on average by 5%




Results of the predicate clustering approach, using predicate matrices with 4, 5 and 5 rows with weights, when query states exist in the preferences (left) or not (right)

- The Jaccard coefficient increases at around 10 to 15% for correlated preferences, and on average 5% when a query does not exist in the preferences

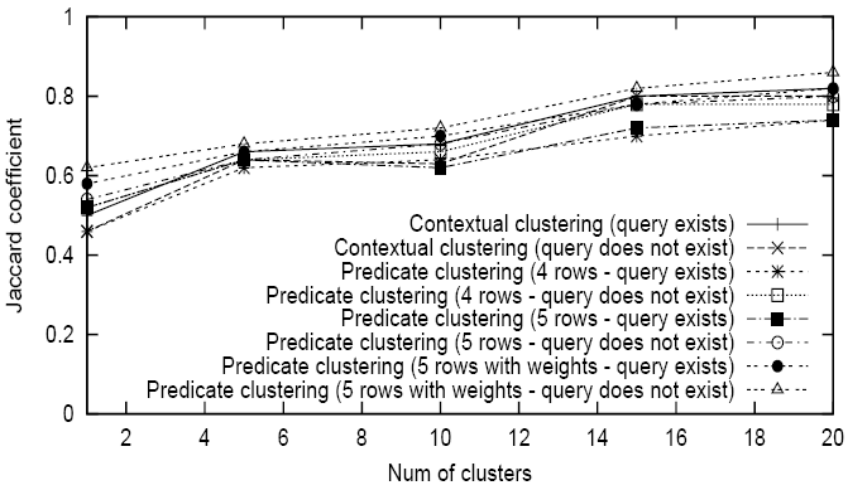


# Quality of Results - Real Preferences

Results for both clustering approaches

- The Jaccard coefficient takes larger values because of the high degree of similarity among user preferences
- Again, if a query state does not exist in the preferences the results are better, at around 17%



# Outline

- Modeling Context
- Contextual Preferences
- Grouping Preferences
  - Contextual Clustering
  - Predicate Clustering
- Evaluation
- Summary

# Summary

We address the problem of finding interesting data items based on contextual preferences that assign interest scores to pieces of data based on context

To do it efficiently, we propose pre-processing steps:

- We construct clusters of similar preferences
  - Preferences that have either the same or similar context states
  - Preferences that result in similar scores for all tuples

# Ongoing Work

- Preferential Keyword Search in Relational Databases
- Preferential Search in Publish/Subscribe

# Preferences in Keyword Search

- Why keyword search?
- How?

example: q = {drama, L. Neeson}

**Movies**

| idm | title | genre | year | director |
|---|---|---|---|---|
| m1 | The Good Shepherd | thriller | 2007 | R. De Niro |
| m2 | Twelve Monkeys | thriller | 1996 | T. Gilliam |
| m3 | Seven | thriller | 1996 | D. Fincher |
| m4 | Schindler's List | drama | 1993 | S. Spielberg |
| m5 | Pulp Fiction | drama | 1994 | Q. Tarantino |

**Play**

| idm | ida |
|---|---|
| m1 | a1 |
| m2 | a2 |
| m3 | a2 |
| m4 | a3 |
| m5 | a4 |

**Actors**

| ida | name | gender | dob |
|---|---|---|---|
| a1 | A. Jolie | female | 1975 |
| a2 | B. Pitt | male | 1963 |
| a3 | L. Neeson | male | 1952 |
| a4 | S. L. Jackson | male | 1948 |

# Preferences in Keyword Search

Rank results based on preferences

example: I prefer thrillers with A. Jolie directed by R. De Niro than those directed by D. Liman

This is expressed through contextual preferences

example: ({thriller , A. Jolie}, R. De Niro ≻ D. Liman)

Context

Choice

To compute the results of a keyword query, we use the preferences having context equal to the query

Issues: Context Relaxation
Diversity of Results
Overlap of Results

# Ongoing Work

- Preferential Keyword Search in Relational Databases
- Preferential Search in Publish/Subscribe

# Preferential Publish/Subscribe

- In current Publish/Subscribe systems, all subscriptions are considered equally important
- To express priorities, we introduce preferential subscriptions

  preferential subscription = (subscription, score)
- Based on preferential subscriptions, propagate to users only the notifications that are the most interesting to them (top-k)
- Each notification is associated with an expiration time: notifications for old events will eventually die away and let new ones be delivered to users

PrefSiena http://www.cs.uoi.gr/~mdrosou/PrefSIENA

Preliminary results in “Preferential Publish/Subscribe”, PersDB 2008, in conjunction with VLDB 2008, to appear

# Thank You

http://dmod.cs.uoi.gr

# Related Work

- Modeling and Storing Context-Aware Preferences, ADBIS' 06, K. Stefanidis, E. Pitoura, P. Vassiliadis
  - Preferences include a single context parameter
  - Interest scores of preferences involving more parameters, computed by a simple weighted sum
- Adding Context to Preferences, ICDE'07, K. Stefanidis, E. Pitoura, P. Vassiliadis
  - Contextual preferences involve more than one context parameter
- Situated Preferences and Preference Repositories for Personalized Database Applications, ER' 04, S. Holland, W. Kiessling
  - Situated preferences: situations (i.e., context states) are uniquely linked through an N:M relationship with qualitative preferences
    - Our model is compatible with this approach and further supports context resolution
- A Context-Aware Preference Model for Database Querying in an Ambient Intelligent Environment, DEXA' 06, A. van Bunningen, L. Feng, P. Apers
  - A knowledge-based context-aware query preference model
- Context-Sensitive Ranking, SIGMOD' 06, R. Agrawal, R. Rantzau, E. Terzi
  - Ranking database results based on contextual preferences

# Handling Updates

Pre-computing results increases the efficiency of queries but introduces the overhead of maintaining the results in the presence of updates

Handling insertions and deletions of:

- Database tuples
  - When a tuple is added (deleted), we need to add (delete) its entries in all scoring tables
    - Clustering is not affected
- Contextual preferences
  - Add (delete) a preference with a context state that already exists
  - Add (delete) a preference with a new context state

# Handling Updates

Profile updates

- Add (delete) a preference with a context state that already exists
  - Contextual clustering
    - The clustering itself is not affected
    - Update the scores of the cluster of all tuples affected
  - Predicate clustering
    - The clustering may be affected and preferences must be moved
    - Update the scores of the relative clusters
- Add (delete) a preference with a new context state
  - Contextual clustering
    - Find an appropriate cluster for the new state and update the scores
  - Predicate clustering
    - Compute the predicate matrix from the new state, enter the state in the appropriate cluster and update the scores

# Value Distances: Path & Depth

Path Distance: The $f_p$ function is a monotonically increasing function that increases as the path length becomes larger

The path distance $dist_P(c_1, c_2)$ between two context values $c_1$, $c_2$ is:

- 1, if $c_1$, $c_2$ are values of the lowest hierarchy level and their least common ancestor ($lca(c_1, c_2)$) is the root of the corresponding hierarchy
- is computed through the $f_p$ function $1 - e^{-(\alpha \times \rho)}$, where $\alpha > 0$ is a constant and $\rho$ is the minimum path length connecting them in the associated hierarchy

Depth Distance: The $f_d$ function is a monotonically increasing function of the depth of the lowest common ancestor

- Takes into account the minimum distance between their lowest common ancestor and the root value

The depth distance $dist_D(c_1, c_2)$ between two context values $c_1$, $c_2$ is:

- 0, if $c_1 = c_2$
- 1, if $lca(c_1, c_2)$ is the root value of the corresponding hierarchy
- is computed through the $f_d$ function $1 - e^{-(\beta / \gamma)}$, where $\beta > 0$ is a constant and $\gamma$ is the minimum path length between $lca(c_1, c_2)$ value and the root value of the corresponding hierarchy

# Distance between Predicate Matrices

The distance between two predicate matrices of two context states $cs_1$, $cs_2$ is computed as:

$$dist(cs_1, cs_2) = \frac{\sum_{i=1}^{b} dist_V(BV(cs_1, s_i), BV(cs_2, s_i))}{b}$$

where $b$ is the num of rows and $BV(cs_i, s_j)$ the relative row to score $s_i$

The distance between two bitmap vectors $BV_1$, $BV_2$ is computed using the Jaccard coefficient that ignores the negative matches:

$$dist_V(BV_1, BV_2) = \frac{diff}{diff + pos}$$

where *diff* is the num of bits that the two vectors differ at, and *pos* the num of 1 for both vectors