Star Shaped HIN-Clustering

1.0

Generated by Doxygen 1.7.3

Mon Apr 11 2011 15:16:39

Contents

Chapter 1

Class Index

1.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 3

Class Documentation

3.1 BerryLatticeAlgos Class Reference

Author: Faris Alqadah.

`#include "berry.h"`

Inheritance diagram for BerryLatticeAlgos:

Public Member Functions

- void Star_N_Concepts (RelationGraph *g, int lrnrContext)

  Inteface for Berry-based algorithms that compute the n-concepts or n-clusters from a star shaped HIN.

- vector< NCluster * > * UpperNeighbors (NCluster *c, RelationGraph *g, int s, int t)

  Computes the upper neighbors of a concept using the "Cover" algorithm in Berry et al.

- vector< NCluster * > * LowerNeighbors (NCluster *c, RelationGraph *g, int s, int t)

  Computes the lower neighbors of a concept using the "Cover" algorithm in Berry et al.

Private Member Functions

- void Enum_NConcepts_Berry (NCluster *a, RelationGraph *g, IOSet *marked, int s, int t)

Computes n-clusters utilizing the algorithm described in "A local approach to Concept generation" berry et al. as a basis.

- vector< IOSet * > * MaxMod_Partition (Context *ctx, NCluster *c, int s, int t)

  Computes the maxmods of the sub-context ctx(A-c(s),c(t)) where domain s takes the place of attributes (see berry et al.)

- list< IOSet * > * NonDominating_MaxMods (Context *ctx, NCluster *c, int s, int t, vector< IOSet * > *maxmods, vector< IOSet * > *primes, vector< IOSet * > *domInfo)

  Returns the set of non-dominating maxmods given a set of maxmods.

- void RemoveMarked (list< IOSet * > *ndMaxMods, IOSet *marked)

  Sets the flag to true for each maxmod which is non-domninating.

- NCluster * MakeMatch (NCluster *lrnrConcept, RelationGraph *g, int s, int t)

  Attempts to find mathcing semi-concepts to lrnrConcept(s,t) in the rest of g.

3.1.1 Detailed Description

Author: Faris Alqadah. This is a derived class of LatticeAlgos and implements Lattice based algorithms based on the paper "A local approach to Concept Generation" by Berry et al. Essentialy, n-clusters as described in the disseration "Mining multi-domain information networks" are mined by generalizing the algorithm presented by Berry et al. to a star shaped network. In addition, functions for computing the upper and lower neighbors of a given concept in a single context are also provided.

Definition at line 15 of file berry.h.

3.1.2 Member Function Documentation

3.1.2.1 void BerryLatticeAlgos::Enum_NConcepts_Berry (NCluster * *a*, RelationGraph * *g*, IOSet * *marked*, int *s*, int *t*) [private]

Computes n-clusters utilizing the algorithm described in "A local approach to Concept generation" berry et al. as a basis.

Performs the berry depth first search version

Parameters

a	the current ncluster which contains the current concept for this search level (see berry et al.)
marked	corresponds to the marked set (see berry et al.)
g	the relation graph, should be star shaped
s	the "source id" of the domain for which the concept a(s,t) is the current concept

t	the "target id" of the domain for which the concept a(s,t) is the current concept

Definition at line 192 of file berry.cpp.

3.1.2.2 vector< NCluster * > * BerryLatticeAlgos::LowerNeighbors (NCluster * *c*, RelationGraph * *g*, int *s*, int *t*)

Computes the lower neighbors of a concept using the "Cover" algorithm in Berry et al.

Parameters

c	the n-cluster containg the concept as c(s,t)
g	the relation graph for which (s,t) should be a context
s	the "source id" of the domain for which the concept a(s,t) is the current concept, the lower neighbors will be sub-sets of domain s
t	the "target id" of the domain for which the concept a(s,t) is the current concept, the lower neighbors will be super-sets of domain t

Definition at line 317 of file berry.cpp.

3.1.2.3 NCluster * BerryLatticeAlgos::MakeMatch (NCluster * *lrnrConcept*, RelationGraph * *g*, int *s*, int *t*) [private]

Attempts to find mathcing semi-concepts to lrnrConcept(s,t) in the rest of g.

Assumes g is a star-shaped HIN. Then tries to find matching semi-concepts to lrnrConcept(s,t) in g

Parameters

lrnrConcept	the learner concept that we are trying match is lrnrConcept(s,t)
g	the star-shaped relation graph
s	the domain id of the articulation node and concept lrnrConcept(s,t)
t	the domain id of the other domain that makes up the learner concept

Definition at line 267 of file berry.cpp.

3.1.2.4 vector< IOSet * > * BerryLatticeAlgos::MaxMod_Partition (Context * *ctx*, NCluster * *c*, int *s*, int *t*) [private]

Computes the maxmods of the sub-context ctx(A-c(s),c(t)) where domain s takes the place of attributes (see berry et al.)

This is a generalization of the Maxmod-Partition algorithm described in berry et al.

Parameters

ctx	the full context which corresponds to g(s,t)
c	contains the concept c(s,t)
s	the "source id" of the domain for which the concept c(s,t) is the current concept
t	the "target id" of the domain for which the concept c(s,t) is the current concept

Definition at line 100 of file berry.cpp.

3.1.2.5 list< IOSet * > * BerryLatticeAlgos::NonDominating_MaxMods (Context * *ctx*, NCluster * *c*, int *s*, int *t*, vector< IOSet * > * *maxmods*, vector< IOSet * > * *primes*, vector< IOSet * > * *domInfo*) [private]

Returns the set of non-dominating maxmods given a set of maxmods.

In addtion the primes of each maxmod is computed, while we keep track of what maxmods dominate each non-dominating maxmod

Parameters

ctx	The full context for which the non-dominating maxmods will be computed
c	the n-cluster that contains the concept with which the maxmods were generated
maxmods	list of maxmods
primes	this vector should be empty, and the routine will store the primes of the maxmods in here
domInfo	this vector should be empty, the route will store the index to which maxmods dominate every maxmod here

Definition at line 145 of file berry.cpp.

3.1.2.6 void BerryLatticeAlgos::Star_N_Concepts (RelationGraph * *g*, int *lrnrContext*)

Inteface for Berry-based algorithms that compute the n-concepts or n-clusters from a star shaped HIN.

Parameters

g	pointer to relation graph for which computation will take place. This must be a star-shaped relation graph
lrnrContext	the id of the context that will server as the learner (See "An effective algorithm for 3-clustering" by Alqadah et al.)

The following external variables from LatticeAlgosExternals should be set:

See also

enumerationMode

qualityMode
ovlpMode
pruneMode
ovlpThresh
topKK

Definition at line 4 of file berry.cpp.

3.1.2.7 vector< NCluster * > * BerryLatticeAlgos::UpperNeighbors (NCluster * *c*, RelationGraph * *g*, int *s*, int *t*)

Computes the upper neighbors of a concept using the "Cover" algorithm in Berry et al.

Parameters

c	the n-cluster containg the concept as c(s,t)
g	the relation graph for which (s,t) should be a context
s	the "source id" of the domain for which the concept a(s,t) is the current concept, the upper neighbors will be super-sets of domain s
t	the "target id" of the domain for which the concept a(s,t) is the current concept, the upper neighbors will be sub-sets of domain t

Definition at line 289 of file berry.cpp.

The documentation for this class was generated from the following files:

- headers/nclusters/berry.h
- source/nclusters/berry.cpp

3.2 Context Class Reference

`#include "Context.h"`

Public Member Functions

- Context (int num1, int num2)

  Default constructor that sets the number of objects in the first and second domain.

- Context (Context &a)

  Copy constructor.

- Context (NCluster *d1, NCluster *d2)

  Constructor that takes in the actual full domains.

- ~Context ()

  Descturor.

- IOSet * GetSet (int domain, int setNum)

  Returns an object-set from the context.

- IOSet * GetLabels (int domain)

  Returns an IOSet with all the labels or object-ids of the specifed domain.

- int GetId ()

  Return the id of the context.

- pair< int, int > GetDomainIds ()

  Returns an interger pair corresponding to the ids of the domains.

- void SetNameMap (int dId, NameMap *nm)

  Assign a name map to one of the domains.

- NameMap * GetNameMap (int dId)

  Returns a pointer to the name map associated with one of the domains.

- void SetId (int)

  Set the id of the context.

- string GetName ()

  Return the name of the context.

- void SetName (string &)

  Set the name of the context.

- void SetDomainId (int setNum, int id)

  Set the domain id of either set1 or set2.

- int GetDomainId (int setNum)

  Returns the id of the selected set, (eithier 0 or 1)

- void PrintAsMatrix ()

  Print the context to stdout as a binary matrix.

- void PrintAsMatrix (ofstream &)

  Print the context to ofstream as a binary matrix.

- Context * GetSubContext (IOSet *a, IOSet *b)

  Resturns a sub-context of the original context.

- void PrintAsFIMI ()

  Print the context in FIMI style to stdout.

- void PrintAsFIMI (ofstream &)

  Print the context in FIMI style to ofstream.

- int GetNumSets (int domainId)

  Return the number of objects for the specifed domain.

- int GetNumOnes ()

  Return the number of ones or relations between objects in domain1 and domain2.

- double GetDensity ()

  Return the number of ones / |domain1|∗|domain2|.

Private Attributes

- NCluster ∗ domain1

  represent first domain

- NCluster ∗ domain2

  represent second domain

- int id

  id of the context

- string name

  name of the context

- NameMap ∗ nameMap1

  name map associated with domain 1

- NameMap ∗ nameMap2

  name map associated with domain 2

3.2.1 Detailed Description

Author: Faris Alqadah Class for representing a context as described in Formal Concept Analysis

Represent a context from Formal Concept Analysis as two n-clusters in FIMI form. The two sets in the relationship are denoted by domain ids. By default set 1 domain id = 0 and set 2 domain id = 1.

Definition at line 18 of file Context.h.

3.2.2 Constructor & Destructor Documentation

3.2.2.1 Context::Context (int *num1*, int *num2*)

Default constructor that sets the number of objects in the first and second domain.

Parameters

num1	the number of objects in the first domain
num2	the number of objects in the second domain

Definition at line 4 of file Context.cpp.

3.2.2.2 Context::Context (Context & *a*)

Copy constructor.

Parameters

a	another context to be deep copied

Definition at line 12 of file Context.cpp.

3.2.2.3 Context::Context (NCluster * *d1*, NCluster * *d2*)

Constructor that takes in the actual full domains.

Parameters

d1	n-cluster representing the "rows" or domain1 in FIMI form
d2	n-cluster representing the "columns" or domain2 in FIMI form

Definition at line 18 of file Context.cpp.

3.2.3 Member Function Documentation

3.2.3.1 IOSet * Context::GetLabels (int *domain*)

Returns an IOSet with all the labels or object-ids of the specifed domain.

Parameters

domain	the id of the domain from which the object-ids will be returned

Definition at line 46 of file Context.cpp.

3.2.3.2 NameMap * Context::GetNameMap (int *dId*)

Returns a pointer to the name map associated with one of the domains.

Parameters

dId	the id of the domain for which the name map will be returned

Definition at line 169 of file Context.cpp.

3.2.3.3 IOSet * Context::GetSet (int *domain*, int *setNum*)

Returns an object-set from the context.

Parameters

domain	the id of the domain from which the object-set will be returned
setNum	the object number in domain for which the object-set or prime will be returned

Definition at line 35 of file Context.cpp.

3.2.3.4 Context * Context::GetSubContext (IOSet * *a*, IOSet * *b*)

Resturns a sub-context of the original context.

Constructs a sub context based on the objects in the paramaters a and b

Parameters

a	set of objects from domain1 that will form the "rows" or first domain of the sub-context
b	set of objects from domain2 that will form the "columns" or second domain of the sub-context

Definition at line 111 of file Context.cpp.

3.2.3.5 void Context::SetDomainId (int *setNum*, int *id*)

Set the domain id of either set1 or set2.

Parameters

setNum	the set to which the id will be assigned, this should be eithier 0 or 1
id	will be assigned to be the domainId of the selected set

Definition at line 64 of file Context.cpp.

3.2.3.6 void Context::SetNameMap (int *dId,* NameMap * *nm*)

Assign a name map to one of the domains.

Parameters

dId	the id of the domain to assign the name map to
nm	pointer to the name map to be assigned to the domain

Definition at line 161 of file Context.cpp.

The documentation for this class was generated from the following files:

- headers/Context.h
- source/Context.cpp

3.3 IOSet Class Reference

Public Member Functions

- IOSet ()

  Default constructor.

- IOSet (int sz)

  Constructor that pre-allocates the size of the IOSet.

- IOSet (IOSet *a)

  Copy constructor.

- ~IOSet ()

  Destructor.

- int Size ()

  Returns the size or number of elements in the IOSet.

- int Id ()

  Returns the id of the IOSet.

- void SetId (int id)

  Set the id of the IOSet.

- void Output ()

  Prints contents of the IOSet as space sperated intergers to stdout.

- void Output (ofstream &f)

  Prints contents of the IOSet as space sperated intergers to ofstream.

- void Output (ofstream &f, NameMap *n)

  Prints contents of the IOSet as space sperated names to ofstream using namemap to map the integers to names.

- void Add (int x)

  Adds integer x to the end of the IOSet, increasing the size of the IOSet.

- void SetSize (int x)

  Assigns the private size variable of the IOSet without actually re-allocationg memory.

- void Resize (int x)

  Resize the IOSet to x, this will physically re-allocate and de-allocate memory unlike SetSize()

- bool Equal (IOSet &)

  Returns true if both IOSets have the same size and the exact contenets in the exact order and false otherwise.

- bool Contains (int)

  Returns true if the IOSet contains the integer specified.

- void DeepCopy (IOSet *)

  Make a deep copy of the input IOSet and assign it to self.

- void Remove (int)

  Remove the element at the specified index.

- void FindRemove (int)

  Find and remove the specified element if it exists.

- void Sort ()

  Sort the elements of the IOSet in asscending order.

- int At (int i)

  Return the ith element.

- void Clear ()

  Remove all elements from the IOSet.

- vector< unsigned int >::iterator GetBegin ()

  Return an iterator to the start of the IOSet.

- vector< unsigned int >::iterator GetEnd ()

  Return an iterator the end of the IOSet.

- void SetMarked (bool)

Set the marked flag.

- bool GetMarked ()

  Returns the value of the marked flag.

- int GetMaxElement ()

  Returns the largest element in the IOSet.

Private Attributes

- int size

  size of the ioset

- int id

  id of the ioset

- vector< unsigned int > d

  vector to hold the data

- bool marked

  has this ioset been marked for whatever reason??

3.3.1 Detailed Description

Definition at line 34 of file IOSet.h.

3.3.2 Member Function Documentation

3.3.2.1 void IOSet::SetSize (int *x*)

Assigns the private size variable of the IOSet without actually re-allocationg memory.

This operation should mainly be used by set operation algorithms where size of the IOSet may not be known a-priori.

Definition at line 42 of file IOSet.cpp.

The documentation for this class was generated from the following files:

- headers/IOSet.h
- source/IOSet.cpp

3.4 LatticeAlgos Class Reference

Inheritance diagram for LatticeAlgos:

Public Member Functions

- LatticeAlgos ()

  Default constructor.

Public Attributes

- int srchLvl

  keeps track of the search level in an enumeration algorithm

- int numConcepts

  keeps track of the total number of concepts or clusters enumerated

- bool dispProgress

  flag to indicate if progress of the algorithm should output to the user (stdout)

- vector< NCluster * > CONCEPTS

  data structure to hold the enumerated clusters in memory during algorithm exectuion

- vector< NameMap * > NAME_MAPS

  vector of name maps to be used to output clusters

- string OUTFILE

  if ENUM_FILE or ENUM_TOPK_FILE is selected then this file is used to output the concepts

- ofstream OUT1

  ofstream used to output to OUTFILE.concepts

- ofstream OUT2

  ofsteram used to output to OUTFILE.concept.names

- int enumerationMode

  Users will set this variable to indicate the enumeration mode.

- int qualityMode

  user will set this variable to indicate the desired qualityMode

- double(∗ qualityFunction)(NCluster ∗, vector< double > &)

  function pointer to a quality measure, interface functions will set this acrroding to qualityMode

- vector< double > params

  store the parameters for a quality function here, see the QualityMeasures.h documentation for specification of these parameters

- int ovlpMode

  user will set this variable to inidicate the desired qualityMode

- double(∗ ovlpFunction)(NCluster ∗, NCluster ∗)

  function pointer to an overlap function computer, interface function will set this according to ovlpMode

- double ovlpThresh

  a threshold value that indicates how much overlap two clusters may have before an algorithm keeps the higher quality cluster

- int topKK

  the number of clusters an algorithm should enumerate if user only wants the top k clusters

- int pruneMode

  user will set this variable to indicate the the desired pruning mode

- vector< int > PRUNE_SIZE_VECTOR

  if PRUNE_SIZE mode is selected this vector should be initialized to the min support of each domain

Static Public Attributes

- static const int ENUM_MEM = 1

  Enumeration mode that specifies to algorithms to mine and store clusters in memory.

- static const int ENUM_FILE = 2

  Enumeration mode that specifies to algorithms to mine and output clusters to a file, this file is specified by OUTFILE.

- static const int ENUM_TOPK_FILE = 3

  Enumeration mode that specifies to algorithms to mine only the top K clusters and output to a file, this file is specified by OUTFILE.

- static const int ENUM_TOPK_MEM = 4

  Enumeration mode that specifies to algorithms to mine only the top K clusters and store in memory.

- static const int AREA = 1

  quality mode that indicates to use the area of a concept as its quality measure

- static const int BETA = 2

  quality mode that indicates to use the beta area of a concept as its quality measure (see "An effective algorithm for mining 3-clusters" by Alqadah et al.)

- static const int AVG_JACCARD = 1

  overlap mode that indicates to use the average jaccard coefficient across all sets of an n-cluster to compute overlapping

- static const int PRUNE_SIZE = 1

  prune mode that indicates pruning will be based on size (support pruning)

3.4.1 Detailed Description

Definition at line 29 of file LatticeAlgos.h.

3.4.2 Member Data Documentation

3.4.2.1 vector<int> LatticeAlgos::PRUNE_SIZE_VECTOR

if PRUNE_SIZE mode is selected this vector should be initialized to the min support of each domain

PRUNE_SIZE_VECTOR[domainId-1] should contain the minumum number of objects a domain with domainId should contian in order to be considered for enumeration. Users must set the values for this vector

Definition at line 122 of file LatticeAlgos.h.

The documentation for this class was generated from the following file:

- headers/LatticeAlgos.h

3.5 NameMap Class Reference

Public Member Functions

- NameMap ()

  Default constructor. No actual map is constructed, since no file is read.

- NameMap (string &file)

  Alternate constructor that passes in the file name and constructs the actual name map.

- NameMap (string, unsigned int n)

  Alternate constructor that passes in the file name and constructs the actual name map but only upto the nth entry.

- string GetFileName ()

  Returns the file name from which the name map was constructed.

- string GetName (unsigned int i)

  returns the string mapped to i

- int GetNumEntries ()

  Returns the number of entries in the map.

- void SetId (int)

  Set the id attribute of the name map.

- int GetId ()

  Returns the id attribue of the name map.

Private Attributes

- vector< string > mapping

  the actual map, maps the index in the vector to the string

- int numEntries

  number of entries

- string fileName

  name of the file from which the map was constructed

- int **id**

3.5.1 Detailed Description

Definition at line 22 of file NameMap.h.

The documentation for this class was generated from the following files:

- headers/NameMap.h
- source/NameMap.cpp

3.6 NCluster Class Reference

Public Member Functions

- NCluster ()

  Default constructor.

- NCluster (unsigned int n)

  Alternate constructor.

- NCluster (unsigned int n, bool allocate)

  Alternate constructor.

- NCluster (unsigned int n, vector< IOSet * > &aa)

  Alternate constructor.

- NCluster (NCluster &)

  Copy constructor.

- ~NCluster ()

  Destructor.

- void DeepCopy (NCluster &)

  Makes a deep copy of the n-cluster.

- void Output ()

  Prints each IOset on a seperate line preceded by the Id of each IOSet to stdout.

- void Output (ofstream &)

  Prints each IOset on a seperate line preceded by the Id of each IOSet to ofstream.

- void Output (ofstream &, vector< NameMap * > &nm)

  Prints each IOset on a seperate line preceded by the Id of each IOSet to ofstream.

- int GetN ()

  Resurns n, the number of sets in the n-cluster.

- IOSet * GetSet (int)

  Returns a pointer to the ith set.

- IOSet * GetSetById (int id)

  Returns a pointer to the set with id, if it exists, assertion is checked.

- void AddSet (IOSet *a)

  Adds the set a to the end of the n-cluster.

- void AssignSet (int i, IOSet *a)

  Assigns a to the ith, deleting the previously defined ith set in the process.

- void AssignSetById (int id, IOSet *a)

  Assigns a to the set with id, if it exists, previoulsy defined set with id is destroyed in the process.

- double GetQuality ()

  Returns the quality attribute.

- void SetQuality (double)

  Set the quality attribute.

- int GetId ()

  Returns the id attribute.

- void SetId (int)

  Set the id attribute.

- bool GetMarked ()

  Return the marked attribute.

- void SetMarked (bool m)

  Set the marked attribute.

- bool ContainsIOSetId (int id)

  Returns true if an IOSet with id=id exists in self, false otherwsie.

- int GetMaxElement ()

  Returns the value of the largest element in all sets of self.

Protected Attributes

- unsigned int n

  degree of the cluster

- vector< IOSet * > sets

  the actual data or sets

- double quality

  quality of the n-cluster

- int id

  id of the n-cluster

- bool marked

  has this n-cluster been marked or flagged for whatever reason??

3.6.1 Detailed Description

Definition at line 13 of file NCluster.h.

3.6.2 Constructor & Destructor Documentation

3.6.2.1 NCluster::NCluster (unsigned int *n*)

Alternate constructor.

Allocates n sets and sets size to n

Parameters

n	number of sets to allocate

Definition at line 9 of file NCluster.cpp.

3.6.2.2 NCluster::NCluster (unsigned int *n*, bool *allocate*)

Alternate constructor.

Makes the n-cluster of size n, but does not allocate the memory

Parameters

n	size of the n-cluster
allocate	this value does not matter, if its true or false, its just to inidcate that no memory should be allocated

Definition at line 27 of file NCluster.cpp.

3.6.2.3 NCluster::NCluster (unsigned int *n*, vector< IOSet * > & *aa*)

Alternate constructor.

Constructor n-cluster of size n and assigin a deep copy of the IOsets from the vector aa to the n-cluster n size of the n-cluster aa a vector of IOSets that will be used to initialize the sets of self by making a deep copy

Definition at line 17 of file NCluster.cpp.

3.6.3 Member Function Documentation

3.6.3.1 void NCluster::Output (ofstream & *out*, vector< NameMap * > & *nm*)

Prints each IOset on a seperate line preceded by the Id of each IOSet to ofstream.

Attempts to match the id of one of the name maps to the id of the sets of self. If the ids match, then then IOSet is output using the name map, otherwise normal printing is performed.

Definition at line 73 of file NCluster.cpp.

The documentation for this class was generated from the following files:

- headers/NCluster.h
- source/NCluster.cpp

3.7 RelationGraph Class Reference

```
#include "RelationGraph.h"
```

Public Member Functions

- RelationGraph ()

  Default constructor.

- ~RelationGraph ()

  Destructor.

- void AddContext (Context *c)

  Adds context c to the network.

- int GetNumNodes ()

  Returns the number of nodes (domains) in the network.

- bool IsEdge (int id1, int id2)

  Returns true if the ids are an edge in the network, false otherwise.

- vector< Context * > * GetContexts (int domain)

  Returns a vector of contexts that contain domain.

- IOSet * GetArtDomains ()

  Returns an IOSet of domain ids which correspond to the articulation nodes of the network.

- Context * GetContext (int ctxId)

  Returns a pointer to the context with the specified ctxId.

- IOSet * GetNeighbors (int domain)

  Reutnrs an IOSet of doamins ids that share an edge with domain.

- IOSet * GetAllContextIds ()

  Reutnrs an IOSet of all the context ids.

- IOSet * GetAllDomainIds ()

  Reutnrs an IOSet of all the domain ids.

- bool IsDomainId (int dId)

  Reutnrs true if dId is a domain id in the network, false otherwise.

- bool IsContextId (int cId)

  Reutnrs true if cId is a context id in the network, false otherwise.

- void Print ()

  Prints the HIN.

- Context * GetContext (int s, int t)

  Returns the context correspoding to the edge (s,t), that is context with domains s and t.

- vector< NameMap * > * GetNameMaps ()

  Returns a vector of name map pointers correspoding to each domain of the HIN.

Private Attributes

- NCluster domainContextMap

  maps domains to contexts

- vector< Context * > contexts

  holds all the contexts

- NCluster domainRelations

  adajcency list of the actual graph

3.7.1 Detailed Description

Author: Faris Alqadah Class for representing a Heterogenous Information Network (HIN).

Represent a Heterogenous Information Network (HIN) as described in data mining literature All edges in the network represent a context, and each node is a domain. The contexts are represented by the context class

See also

Context

Definition at line 21 of file RelationGraph.h.

3.7.2 Member Function Documentation

3.7.2.1 void RelationGraph::AddContext (Context * c)

Adds context c to the network.

Add a context to the network. This method assumes the ids of the domains of the context are well defined, and will use these ids to construct the topplogy of the network.

Parameters

c	the context to add to the network,

Definition at line 10 of file RelationGraph.cpp.

3.7.2.2 vector< NameMap * > * RelationGraph::GetNameMaps ( )

Returns a vector of name map pointers correspoding to each domain of the HIN.

The name maps are not in any specfied order, use the id attribute of each name to figure out correspondce to domains!

See also

NameMap

Definition at line 156 of file RelationGraph.cpp.

3.7.2.3 bool RelationGraph::IsEdge (int *id1*, int *id2*)

Returns true if the ids are an edge in the network, false otherwise.

Returns true if there exists a context in the network that has domains with id1 and id2

Definition at line 58 of file RelationGraph.cpp.

The documentation for this class was generated from the following files:

- headers/RelationGraph.h
- source/RelationGraph.cpp