dcDAGdomainSim

December 19, 2014

dcDAGdomainSim	*Function to calculate pair-wise semantic similarity between domains based on a direct acyclic graph (DAG) with annotated data*

Description

dcDAGdomainSim is supposed to calculate pair-wise semantic similarity between domains based on a direct acyclic graph (DAG) with annotated data. It first calculates semantic similarity between terms and then derives semantic similarity between domains from terms-term semantic similarity. Parallel computing is also supported for Linux or Mac operating systems.

Usage

```
dcDAGdomainSim(g, domains = NULL, method.domain = c("BM.average",
"BM.max",
"BM.complete", "average", "max"), method.term = c("Resnik", "Lin",
"Schlicker", "Jiang", "Pesquita"), force = TRUE, fast = TRUE,
parallel = TRUE, multicores = NULL, verbose = TRUE)
```

Arguments

g	an object of class "igraph" or Onto. It must contain a node attribute called 'annotations' for storing annotation data (see example for howto)
domains	the domains between which pair-wise semantic similarity is calculated. If NULL, all domains annotatable in the input dag will be used for calcluation, which is very prohibitively expensive!
method.domain	the method used for how to derive semantic similarity between domains from semantic similarity between terms. It can be "average" for average similarity between any two terms (one from domain 1, the other from domain 2), "max" for the maximum similarity between any two terms, "BM.average" for best-matching (BM) based average similarity (i.e. for each term of either domain, first calculate maximum similarity to any term in the other domain, then take average of maximum similarity; the final BM-based average similiary is the pre-calculated average between two domains in pair), "BM.max" for BM based maximum similarity (i.e. the same as "BM.average", but the final BM-based maximum similiary is the maximum of the pre-calculated average between two

domains in pair), "BM.complete" for BM-based complete-linkage similarity (inspired by complete-linkage concept: the least of any maximum similarity between a term of one domain and a term of the other domain). When comparing BM-based similarity between domains, "BM.average" and "BM.max" are sensitive to the number of terms invovled; instead, "BM.complete" is much robust in this aspect. By default, it uses "BM.average".

`method.term` the method used to measure semantic similarity between terms. It can be "Resnik" for information content (IC) of most informative common ancestor (MICA) (see http://arxiv.org/pdf/cmp-lg/9511007.pdf), "Lin" for 2*IC at MICA divided by the sum of IC at pairs of terms (see http://webdocs.cs.ualberta.ca/~lindek/papers/sim.pdf), "Schlicker" for weighted version of 'Lin' by the 1-prob(MICA) (see http://www.ncbi.nlm.nih.gov/pubmed/16776819), "Jiang" for 1 - difference between the sum of IC at pairs of terms and 2*IC at MICA (see http://arxiv.org/pdf/cmp-lg/9709008.pdf), "Pesquita" for graph information content similarity related to Tanimoto-Jacard index (ie. summed information content of common ancestors divided by summed information content of all ancestors of term1 and term2 (see http://www.ncbi.nlm.nih.gov/pubmed/18460186))

`force` logical to indicate whether the only most specific terms (for each domain) will be used. By default, it sets to true. It is always advisable to use this since it is computationally fast but without compromising accuracy (considering the fact that true-path-rule has been applied when running `dcDAGannotate`)

`fast` logical to indicate whether a vectorised fast computation is used. By default, it sets to true. It is always advisable to use this vectorised fast computation; since the conventional computation is just used for understanding scripts

`parallel` logical to indicate whether parallel computation with multicores is used. By default, it sets to true, but not necessarily does so. Partly because parallel backends available will be system-specific (now only Linux or Mac OS). Also, it will depend on whether these two packages "foreach" and "doMC" have been installed. It can be installed via: `source("http://bioconductor.org/biocLite.R"); biocLite(c("foreach","doMC"))`. If not yet installed, this option will be disabled

`multicores` an integer to specify how many cores will be registered as the multicore parallel backend to the 'foreach' package. If NULL, it will use a half of cores available in a user's computer. This option only works when parallel computation is enabled

`verbose` logical to indicate whether the messages will be displayed in the screen. By default, it sets to true for display

Value

an object of S4 class `Dnetwork`. It is a weighted and undirect graph, with following slots:

- `nodeInfo`: an object of S4 class, describing information on nodes/domains
- `adjMatrix`: an object of S4 class `AdjData`, containing symmetric adjacency data matrix for pair-wise semantic similarity between domains

Note

For the mode "shortest_paths", the induced subgraph is the most concise, and thus informative for visualisation when there are many nodes in query, while the mode "all_paths" results in the complete subgraph.

See Also

dcRDataLoader, dcDAGannotate, dcConverter, Dnetwork-class

Examples

```
# 1) Semantic similarity between SCOP domain superfamilies (sf)
## 1a) load onto.GOMF (as Onto object)
g <- dcRDataLoader(onto.GOMF)
## 1b) load SCOP superfamilies annotated by GOMF (as Anno object)
Anno <- dcRDataLoader(SCOP.sf2GOMF)
## 1c) prepare for ontology appended with annotation information
dag <- dcDAGannotate(g, annotations=Anno, path.mode="shortest_paths",
verbose=FALSE)
## 1d) calculate pair-wise semantic similarity between 8 randomly chosen domains
alldomains <- unique(unlist(nInfo(dag)$annotations))
domains <- sample(alldomains,8)
dnetwork <- dcDAGdomainSim(g=dag, domains=domains,
method.domain="BM.average", method.term="Resnik", parallel=FALSE,
verbose=TRUE)
dnetwork
## 1e) convert it to an object of class igraph
ig <- dcConverter(dnetwork, from=Dnetwork, to=igraph)
ig
## 1f) visualise the domain network
### extract edge weight (with 2-digit precision)
x <- signif(E(ig)$weight, digits=2)
### rescale into an interval [1,4] as edge width
edge.width <- 1 + (x-min(x))/(max(x)-min(x))*3
### do visualisation
dnet::visNet(g=ig, vertex.shape="sphere", edge.width=edge.width,
edge.label=x, edge.label.cex=0.7)

############################################################
# 2) Semantic similarity between Pfam domains (Pfam)
## 2a) load onto.GOMF (as Onto object)
g <- dcRDataLoader(onto.GOMF)
## 2b) load Pfam domains annotated by GOMF (as Anno object)
Anno <- dcRDataLoader(Pfam2GOMF)
## 2c) prepare for ontology appended with annotation information
dag <- dcDAGannotate(g, annotations=Anno, path.mode="shortest_paths",
verbose=FALSE)
## 2d) calculate pair-wise semantic similarity between 8 randomly chosen domains
alldomains <- unique(unlist(nInfo(dag)$annotations))
domains <- sample(alldomains,8)
dnetwork <- dcDAGdomainSim(g=dag, domains=domains,
method.domain="BM.average", method.term="Resnik", parallel=FALSE,
verbose=TRUE)
dnetwork
## 2e) convert it to an object of class igraph
ig <- dcConverter(dnetwork, from=Dnetwork, to=igraph)
ig
## 2f) visualise the domain network
### extract edge weight (with 2-digit precision)
x <- signif(E(ig)$weight, digits=2)
### rescale into an interval [1,4] as edge width
edge.width <- 1 + (x-min(x))/(max(x)-min(x))*3
```

```
### do visualisation
dnet::visNet(g=ig, vertex.shape="sphere", edge.width=edge.width,
edge.label=x, edge.label.cex=0.7)

############################################################
# 3) Semantic similarity between InterPro domains (InterPro)
## 3a) load onto.GOMF (as Onto object)
g <- dcRDataLoader(onto.GOMF)
## 3b) load InterPro domains annotated by GOMF (as Anno object)
Anno <- dcRDataLoader(InterPro2GOMF)
## 3c) prepare for ontology appended with annotation information
dag <- dcDAGannotate(g, annotations=Anno, path.mode="shortest_paths",
verbose=FALSE)
## 3d) calculate pair-wise semantic similarity between 8 randomly chosen domains
alldomains <- unique(unlist(nInfo(dag)$annotations))
domains <- sample(alldomains,8)
dnetwork <- dcDAGdomainSim(g=dag, domains=domains,
method.domain="BM.average", method.term="Resnik", parallel=FALSE,
verbose=TRUE)
dnetwork
## 3e) convert it to an object of class igraph
ig <- dcConverter(dnetwork, from=Dnetwork, to=igraph)
ig
## 3f) visualise the domain network
### extract edge weight (with 2-digit precision)
x <- signif(E(ig)$weight, digits=2)
### rescale into an interval [1,4] as edge width
edge.width <- 1 + (x-min(x))/(max(x)-min(x))*3
### do visualisation
dnet::visNet(g=ig, vertex.shape="sphere", edge.width=edge.width,
edge.label=x, edge.label.cex=0.7)

############################################################
# 4) Semantic similarity between Rfam RNA families (Rfam)
## 4a) load onto.GOBP (as Onto object)
g <- dcRDataLoader(onto.GOBP)
## 4b) load Rfam families annotated by GOBP (as Anno object)
Anno <- dcRDataLoader(Rfam2GOBP)
## 4c) prepare for ontology appended with annotation information
dag <- dcDAGannotate(g, annotations=Anno, path.mode="shortest_paths",
verbose=FALSE)
## 4d) calculate pair-wise semantic similarity between 8 randomly chosen RNAs
alldomains <- unique(unlist(nInfo(dag)$annotations))
domains <- sample(alldomains,8)
dnetwork <- dcDAGdomainSim(g=dag, domains=domains,
method.domain="BM.average", method.term="Resnik", parallel=FALSE,
verbose=TRUE)
dnetwork
## 4e) convert it to an object of class igraph
ig <- dcConverter(dnetwork, from=Dnetwork, to=igraph)
ig
## 4f) visualise the domain network
### extract edge weight (with 2-digit precision)
x <- signif(E(ig)$weight, digits=2)
### rescale into an interval [1,4] as edge width
edge.width <- 1 + (x-min(x))/(max(x)-min(x))*3
### do visualisation
```

```
dnet::visNet(g=ig, vertex.shape="sphere", edge.width=edge.width,
edge.label=x, edge.label.cex=0.7)

############################################################
# 5) Advanced usage: customised data for ontology and annotations
# 5a) customise ontology
g <-
dcBuildOnto(relations.file="http://dcgor.r-forge.r-project.org/data/onto/igraph_GOMF_edges.txt",
nodes.file="http://dcgor.r-forge.r-project.org/data/onto/igraph_GOMF_nodes.txt",
output.file="ontology.RData")
# 5b) customise Anno
Anno <-
dcBuildAnno(domain_info.file="http://dcgor.r-forge.r-project.org/data/InterPro/InterPro.txt",
term_info.file="http://dcgor.r-forge.r-project.org/data/InterPro/GO.txt",
association.file="http://dcgor.r-forge.r-project.org/data/InterPro/Domain2GOMF.txt",
output.file="annotations.RData")
## 5c) prepare for ontology appended with annotation information
dag <- dcDAGannotate(g, annotations=Anno, path.mode="shortest_paths",
verbose=FALSE)
## 5d) calculate pair-wise semantic similarity between 8 randomly chosen domains
alldomains <- unique(unlist(nInfo(dag)$annotations))
domains <- sample(alldomains,8)
dnetwork <- dcDAGdomainSim(g=dag, domains=domains,
method.domain="BM.average", method.term="Resnik", parallel=FALSE,
verbose=TRUE)
dnetwork
## 5e) convert it to an object of class igraph
ig <- dcConverter(dnetwork, from=Dnetwork, to=igraph)
ig
## 5f) visualise the domain network
### extract edge weight (with 2-digit precision)
x <- signif(E(ig)$weight, digits=2)
### rescale into an interval [1,4] as edge width
edge.width <- 1 + (x-min(x))/(max(x)-min(x))*3
### do visualisation
dnet::visNet(g=ig, vertex.shape="sphere", edge.width=edge.width,
edge.label=x, edge.label.cex=0.7)
```