NetPredictor

Abhik Seal

2015-10-18

Table of Contents

---

1. Introduction

Social and biological systems can be represented by graphs where nodes represent individuals,biological experiments(protein,genes,etc.) web users and so on. Networks allows methods of graph theory to be applied to the task of predicting links. Link prediction predicts missing links in networks or links in future networks, it is also important for mining and analyzing the evolution of networks. Link prediction problem is a long-standing challenge in modern information science, and a lot of algorithms based on Markov chains and statistical models have been proposed by computer science community. The link prediction problem is usually defined in unipartite graphs. The netpredictor package is developed to solve the problem of bipartite link prediction using Random walk with restart(RWR) and network based inference methods(NBI). We plan to integrate varierty of other algorithms in near future. All of the code is developed in R which also provides parallel execution modes.

Consider an undirected, unweighted network $G = (V, E)$, where V is the set of nodes and E is the set of links. For each pair of nodes $\{a,b\} \in$ V we can assign a proximity score by executing the random walk procedure as follows :

- we start a random walker from a.
- At each time step, with the probability $1 - c$, the walker walks to one of the neighbors, b, according to the transition probability matrix $W_{ab} = \frac{S_{ab}}{T}$ where S_{ab} is the adjacency matrix of the network and (S_{ab} equals 1 if node a and b are connected, 0 otherwise) K_a denotes the degree of a.
- With the probability c, the walker goes back to a.
- After many time steps the probability of finding the random walker at node x converges to the steady-state probability, which is our proximity score $S_{a \to x}$.

One of the most widely used ways to solve random walk w1ith restart is the matrix iteration method, iterating the equation (1) until convergence, i.e, until the L2 norm of successive estimates of below our threshold T 10^{-7}.

$$P_{t+1} = (1 - c)W^T P_t + cP_0 \tag{1}$$

---

2. Installation

A stable tested version of from github using the devtools package:

Installing from github

```
install.packages("devtools")
library(devtools)
install_github("abhik1368/netpredicter")
```

3. Examples

Here at first we look at the properties which can be calculated on unipartite graphs.

```r
require(igraph)
require(netpredictor)
g1 <- upgrade_graph(erdos.renyi.game(100, 1/100))
V(g1)$name <- seq(1,100,1)
score_mat <- unetSim(g1,"aa")
head(which(score_mat!=0, arr.ind = T))

## Common neighbors vertex similarity
score_mat <- unetSim(g1,"cn")
head(which(score_mat!=0, arr.ind = T))

## Jaccard Index similarity

score_mat <- unetSim(g1,"jc")

## Dice similarity

score_mat <- unetSim(g1,"dice")

## Katz Index similarity

score_mat <- unetSim(g1,"katz")

## Geodesic distance vertex similarity

score_mat <- unetSim(g1,"dist")

## Cosine vertex similarity/ Salton index

score_mat <- unetSim(g1,"cosine")

## Preferential attachment vertex similarity

score_mat <- unetSim(g1,"pa")

## Local Paths Index
## This function counts the number of two-paths and three-paths between nodes.
```

```
score_lpsim <- unetSim(g1,"lp")

## Hub promoted Index
## This measures assigns higher scores to links adjacent to hubs (high degree nodes). It
## counts common neighbors of two vertices and weigths the result.

score_hpsim <- unetSim(g1,"hpi")

## Similarity measure based on resource allocation process (number of common neighbours
## weighted by the inverse of their degrees)
score_hpsim <- unetSim(g1,"ra")
```

Next we look at the properties which can be calculated on Bipartite graphs.

```
suppressPackageStartupMessages(library(igraph))
suppressPackageStartupMessages(library(netpredictor))
## dataset enzyme is provide in netpredictor package
data(Enzyme)

## Get the Enzyme and compound adjacency matrix
A <- t(enzyme_ADJ)

## degree Centrality of the Bipartite Graph
get.biDegreeCentrality(A,SM=FALSE)

## Compute Graph density of Bipartite Graph
get.biDensity(A)

## Compute betweeness centrality of Bipartite Graph
get.biBetweenessCentrality(A)

## Projects Bipartite Networks into monopartite networks default method is shared
## neighbours.
get.biWeightedProjection(A,weight = TRUE)
```

Next we will use the different methods to predict links. Here we have shown examples based on drug target prediction. With the growing understanding of complex diseases, the focus of drug discovery has shifted away from "one target, one drug" model, to a new "multi-target, multi-drug" model. Predicting potential drug-target interactions from heterogeneous biological data is critical not only for better understanding of the various interactions and biological processes, but also for the development of novel drugs and the improvement of human medicines. To predict polypharmacology people use bayesian methods, SVM and Random Forest models, but in all of those algorithms the methods depends on labelled data to predict unknown links. Network based approaches does not rely on labelled data . Two of the algorihtms implemented in this package Random walk based Restart(RWR) and Network based Inference(NBI) to do it. For performing RWR we used Drug target network which is a bipartite graph in which every links connects drugs to proteins.

```
suppressPackageStartupMessages(require(igraph))
suppressPackageStartupMessages(require(netpredictor))

## We use the enzyme data provided in the netpredictor package. The example
# below shows how to perform random walk with restart
```

```r
data(Enzyme)
## load the adjacency matrix
A <- enzyme_ADJ

## load the chemical similarity matrix calculated from other packages or softwares
S2 = enzyme_Csim

## load the protein similarity matrix

S1 = enzyme_Gsim

## Convert the adjacency matrix to igraph object because biNetwalk function used igraph object
g1 = graph.incidence(A)

## Run the RWR in bipartite network.
pScore <- biNetwalk(g1,s1=S1,s2=S2,normalise="laplace", dataSeed=NULL,
                restart=0.8, parallel=TRUE, multicores=NULL, verbose=T)
```

```
## First, get the adjacency matrix of the input graph (2015-10-18 23:22:40) ...
## Note: using unweighted graph!
## got the transition matrix for RWR
##  do parallel computation using 2 cores ...
## Executing parallel:
## Rescaling steady probability vector (2015-10-18 23:23:02) ...
## Runtime in total is: 22 secs
```

```r
dim(pScore)
```

[1] 478 212

In this example we attempt to use the dataseed file which contains the pairs relations between targets and drugs. This can be useful when one is trying to investigate relations for a specific set of relations . The Drug names and proteins names should be included in the adjacency matrix when one uses the file option to provide dataseed. In the dataseed file the first column contains the proteins names and the second column the drug names. Ouput is a matrix of unique drugs against the number of targets in the adjacency matrix.

```r
suppressPackageStartupMessages(require(igraph))
library(netpredictor)
data(Enzyme)

A <- t(enzyme_ADJ)
g1 <- upgrade_graph(graph.incidence(A,mode = 'all'))
S1 = enzyme_Csim
S2 = enzyme_Gsim
## Read the dataseed file from the user
dataF<- read.csv("seedFile.csv",header=FALSE)
```

```
## Warning in read.table(file = file, header = header, sep = sep, quote =
## quote, : incomplete final line found by readTableHeader on 'seedFile.csv'
```

```
knitr::kable(dataF)
```

V1	V2
hsa2936	D00014
hsa2950	D00014
hsa5033	D00018
hsa5351	D00018

```
Mat <- biNetwalk(g1,s1=S1,s2=S2,normalise="laplace", dataSeed =dataF,
                 restart=0.8,parallel=TRUE, multicores=NULL, verbose=T)
```

```
## First, get the adjacency matrix of the input graph (2015-10-18 23:23:03) ...
## Note: using unweighted graph!
## got the transition matrix for RWR
##  do parallel computation using 2 cores ...
## Executing parallel:
## Rescaling steady probability vector (2015-10-18 23:23:04) ...
## Runtime in total is: 1 secs
```

In this next example we will see how we can plot the significant communities of drugs from the final RWR computed matrix. For community detection we used the walktrap algorithm [13], which places nodes into communities based on neighborhood similarity from short random walks. We also input a list of drugs as vector and retrieve top 10 interactions for each of those drugs. In this package after getting the results one can easily write the results in GML format for visualization in Gephi or cytoscape. It also support export to GEXF format (Gephi specific file format) . Below shows the example of exporting to GML format.

```
suppressPackageStartupMessages(require(igraph))
suppressPackageStartupMessages(require(netpredictor))

A <- enzyme_ADJ
S1 = enzyme_Gsim
S2 = enzyme_Csim
g1 = graph.incidence(A)
Q = biNetwalk(g1,s1=S1,s2=S2,normalise="laplace",dataSeed=NULL,restart=0.8,
              parallel=TRUE,multicores=NULL, verbose=T)
```

```
## First, get the adjacency matrix of the input graph (2015-10-18 23:23:04) ...
## Note: using unweighted graph!
## got the transition matrix for RWR
##  do parallel computation using 2 cores ...
## Executing parallel:
## Rescaling steady probability vector (2015-10-18 23:23:27) ...
## Runtime in total is: 23 secs
```

```
## Get the top results of RWR prediction. This function returns the associations
## of drugs and target names with scores and type interaction whether
## True / predicted interactions.

knitr::kable(head(getTopresults(A,Q,top=10,druglist=NULL)))
```

drug	pnames	score	type
D00014	hsa2936	0.396969136434142	True Interactions
D00014	hsa2950	0.393772891664937	True Interactions
D00014	hsa1719	0.0011479214893255	Predicted Interactions
D00014	hsa55312	0.00104834689722637	Predicted Interactions
D00014	hsa7172	0.0010385344733581	Predicted Interactions
D00014	hsa4128	0.000974699916298979	Predicted Interactions

```
## Get top results of RWR prediction using a list of drug names. One should be careful using
## a drug list it should contain drug names which are in the adjacency matrix.

drugs = c("D00014","D00018", "D00029", "D00036","D00045","D00049")
result <- getTopresults(A,Q,top=10,druglist=drugs)

## Get the results
head(result)
```

```
drug   pnames                score                    type
```

1 D00014 hsa2936 0.396969136434142 True Interactions 2 D00014 hsa2950 0.393772891664937 True Interactions 3 D00014 hsa1719 0.0011479214893255 Predicted Interactions 4 D00014 hsa55312 0.00104834689722637 Predicted Interactions 5 D00014 hsa7172 0.0010385344733581 Predicted Interactions 6 D00014 hsa4128 0.000974699916298979 Predicted Interactions

```
## Save the top results in GML format for visualization in Gephi.
g<-graph.data.frame(result[,1:2],directed=FALSE)

## Set the edge values
g <- set.edge.attribute(g, "weight", value=result[,3])
saveGML(g,"netresult.gml","netresult")

## Get the significance graph
Z = sig.net(data=A,g=g1,Amatrix=Q,num.permutation=100,adjp.cutoff=0.01,
            p.adjust.method="BH",parallel=FALSE)
```

```
## Third, generate the distribution of contact strength based on 100 permutations on nodes respecting ra
## Also, construct the contact graph under the cutoff 1.0e-02 of adjusted-pvalue (2015-10-18 23:23:38).
## Runtime in total is: 3 secs
```

```
## Get the graph for plotting commnuties
g <- Z$cgraph

gp <- get.Communities(g)

## Get members from the first community
print(gp[[1]]$community)
```

IGRAPH clustering walktrap, groups: 7, mod: 0.37 + groups: $1 [1] “D00127” “D00315” “D00330” “D00377” “D00425” “D00566” “D00904” [8] “D00969” “D01325” “D01397” “D01475” “D01513” “D01578” “D01718” [15] “D01811” “D01866”

$2 [1] “D00322” “D00416” “D00567” “D00593” “D00882”

$3 [1] “D00109” “D00217” “D01122” + … omitted several groups/vertices

```
## Total number of communities
length(gp)
```

[1] 7

```
## Plot the communities with 5 columns
plot_Community(gp,cols=5)
```

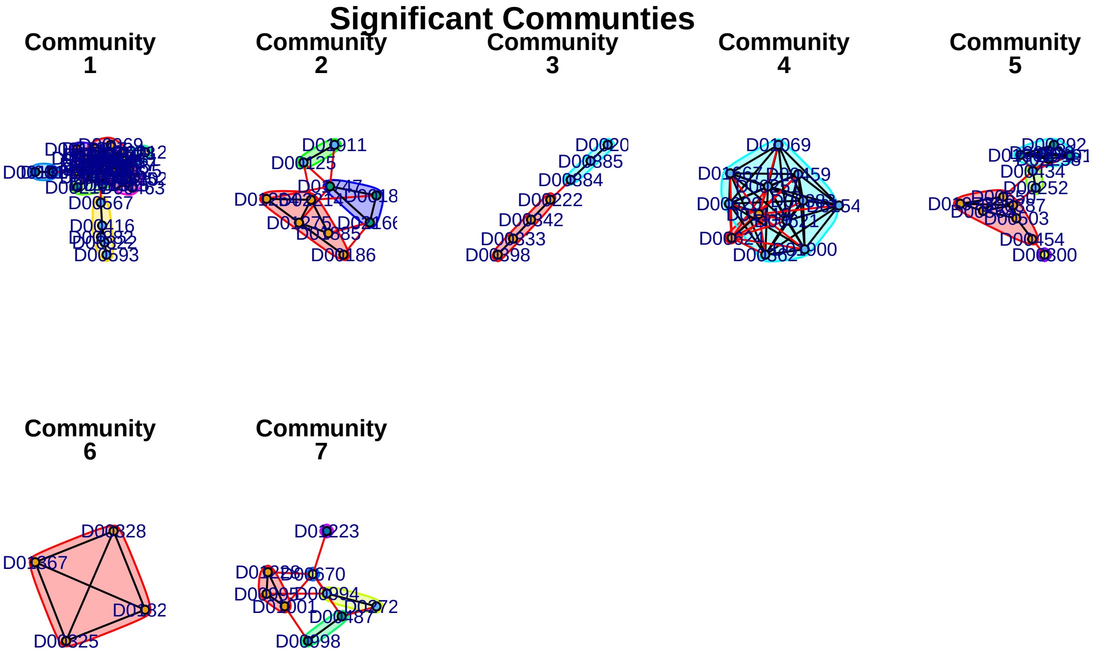

Figure 1:

One can use net.perf(), it samples and removes links from the adjacency matrix and predicts them and calculates area under accumulation curve, AUC, BEDROC (bdr), and Enrichment factor (EF). The area under the receiver operating characteristic (ROC) curve (AUC) is widely used metric for evaluation of predictive models. The advantage of using AUC is that it is bounded, between 0 to 1 with 0.5 corresponding to random prediction. But AUC method has been critized in cheminformatics based virtual screening methods because it is not sensitive to early recognition compounds . The EF tries to solve early recognition problem but it is dependent on the ratio of actives to inactives and the choice of subset X (fraction of active and inactive set) . To try and overcome these limitations numerous other evaluation methods, such as robust initial enhancement (RIE) [10] and Boltzmann-enhanced discrimination of ROC (BEDROC) have been proposed[11]. Sheridan et al. developed an exponential weighted scoring scheme RIE which gives heavier weight in “early recognized” hits. The BEDROC is constructed on top of RIE by, in essence, forcing the RIE to be bounded by 0 and 1, avoiding the dependence on the active/inactive ratio. In the example below we remove 50 links and repredict those links. While repredicting them we calculate performance metrics like AUC,bedroc and enrichment factor. As the number of links (relinks) increases the performance of prediction drops. ‘Calgo’ option uses different algorithms like “nbi” , “rwr” and “netcombo” .

```
data(Enzyme)
A = enzyme_ADJ
S1 = enzyme_Gsim
```

```
S2= enzyme_Csim

## We want to remove the links from the links which has two or more interactions.
m = net.perf(A,S1,S2,relinks = 50,numT=2,Calgo="nbi")
```

```
## Detected (212) drugs & (478) proteins with (1515) interactions...
## Running prediction for (50) links removed using (nbi) ..
## Running NBI Algorithm
## Running computation of the input graph (2015-10-18 23:23:41) ...
## Done computation of the input graph (2015-10-18 23:23:43) ...
## Runtime in total is: 2 secs
```

```
m
```

$aucc [1] 0.8368468

$auc [1] 0.9705882

$auctop [1] 0.5319805

$bdr [1] 0.3733507

$efc [1] 4.62

In 2010 Zhou et al., proposed a recommendation method based on the bipartite network projection technique implementing the concept of resources transfer within the network. The method developed here is from the article DT-Hybrid where they integrated the similarity matrices of drugs and proteins in order to make the prediction with the heatS equation [12]. The example given below one can use both the methods of either using similarity matrices and also simply using heatS equation with the adjacency matrix.

```
data(Enzyme)
A <- t(enzyme_ADJ)
S1 = as.matrix(enzyme_Csim)
S2 = as.matrix(enzyme_Gsim)
g1 = graph.incidence(A)
## Using the similarity matrices
P1 <- nbiNet(A,alpha=0.5, lamda=0.5,  s1=S1, s2=S2,format = "matrix")
```

```
## Running computation of the input graph (2015-10-18 23:23:43) ...
## Done computation of the input graph (2015-10-18 23:23:44) ...
## Runtime in total is: 1 secs
```

```
## Get the significance graph
Z = sig.net(data=A,g=g1,Amatrix=P1,num.permutation=100,adjp.cutoff=0.01,
            p.adjust.method="BH",parallel=FALSE)
```

```
## Third, generate the distribution of contact strength based on 100 permutations on nodes respecting ra
## Also, construct the contact graph under the cutoff 1.0e-02 of adjusted-pvalue (2015-10-18 23:23:50).
## Runtime in total is: 6 secs
```

```
## Get the graph for plotting commnuties
g <- Z$cgraph

gp <- get.Communities(g)
```

```
## Get members from the first community
print(gp[[1]]$community)
```

IGRAPH clustering walktrap, groups: 26, mod: 0.28 + groups: $1 [1] “hsa10188” “hsa1445” “hsa25” “hsa2534” “hsa3716” “hsa3717” [7] “hsa3718” “hsa4145” “hsa55359” “hsa5604” “hsa5605” “hsa5607” [13] “hsa5747” “hsa6416” “hsa657” “hsa660” “hsa6714” “hsa6725” [19] “hsa695” “hsa7297” “hsa7525” “hsa7535” “hsa90” “hsa91”
[25] “hsa93” “hsa94”

$2 [1] “hsa130399” “hsa2241” “hsa3702” “hsa3932” “hsa5606”
[6] “hsa7006”
+ . . . omitted several groups/vertices

```
## Total number of communities
length(gp)
```

[1] 11

```
## Plot the communities with 5 columns
plot_Community(gp,cols=3)
```

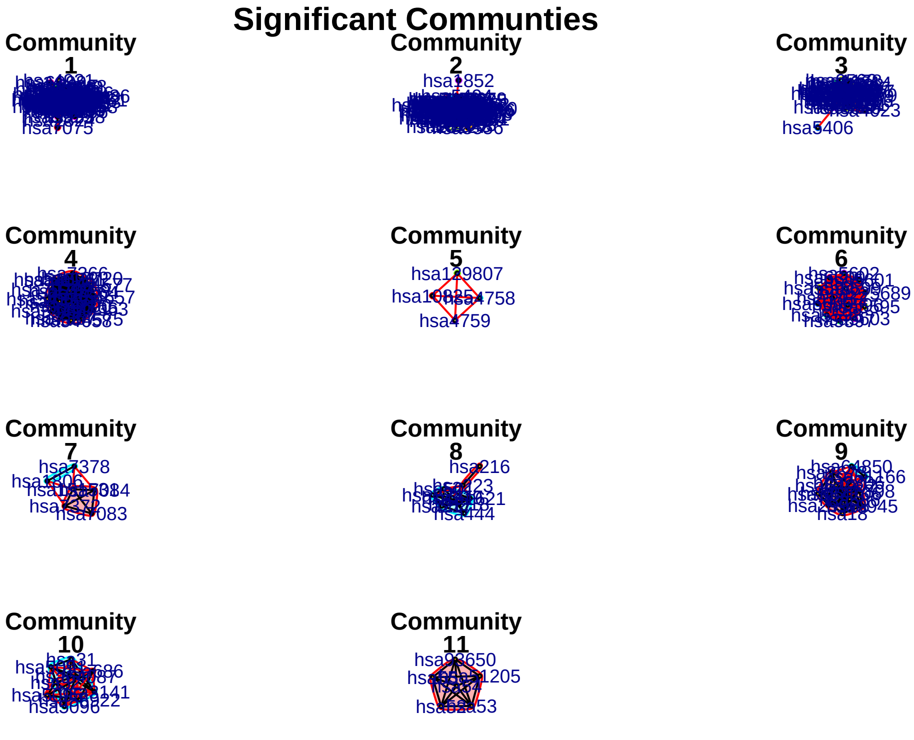

Figure 2:

```
## Using the heatS equation with the adjacency matrix.
P2 <- nbiNet(A,alpha=0.5,lamda=0.5,format="matrix")
```

```
## Running computation of the input graph (2015-10-18 23:23:51) ...
## Done computation of the input graph (2015-10-18 23:23:52) ...
## Runtime in total is: 1 secs
```

We also give users to compute performance metrics using different algorithms to get AUCC, AUC, auctop,bdr and efc so that one can compare the performance using different algorithms.

```
library(netpredictor)
library(igraph)
data(Enzyme)
A <- t(enzyme_ADJ)
S1 = as.matrix(enzyme_Csim)
S2 = as.matrix(enzyme_Gsim)
## Use all the algorithms NBI, RWR and netcombo
m = net.perf(A,S1,S2,relinks = 50,numT=2,Calgo="all")
```

```
## Detected (478) drugs & (212) proteins with (1515) interactions...
## Running prediction for (50) links removed using (all) ..
## Running all the algorithms ...
## First, get the adjacency matrix of the input graph (2015-10-18 23:23:53) ...
## Note: using unweighted graph!
## got the transition matrix for RWR
##  do parallel computation using 2 cores ...
## Executing parallel:
## Rescaling steady probability vector (2015-10-18 23:24:40) ...
## Runtime in total is: 47 secs
##
## Running computation of the input graph (2015-10-18 23:24:40) ...
## Done computation of the input graph (2015-10-18 23:24:41) ...
## Runtime in total is: 1 secs
```

```
tab <- rbind(data.frame(m[[1]]),data.frame(m[[2]]),data.frame(m[[3]]))
knitr::kable(tab)
```

type	score.aucc	score.auc	score.auctop	score.bdr	score.efc
rwr	0.9322963	0.9761905	0.6029874	0.5202676	9.206349
nbi	0.9234052	0.9761905	0.6762691	0.5003086	5.468254
netcombo	0.9366202	0.9761905	0.6025943	0.5202393	9.087302

Above table shows the performance of different algorithms. Next we will look after how do we calculate a significant interaction using network based inference algorithm. We compute the association score between drug a target and we want to found out whether the predicted association score is significant or not . We make 1000 permutations of the association matrix and similarity matrix and compute NBI scores for 1000 random matrices and then we used a normal distribution to calculate p-value . We convert the original compute score to an associated Z-score. Once the Z-score is found the probability that the value could be less the Z-score is found using the pnorm command. Also for a two sided test we need to multiply the result by two. Below gives a idea how we can acheive this . We can create a significant network based on these significant associations found.

```
## Load the data
data(Enzyme)
A <- t(enzyme_ADJ)
S1 = as.matrix(enzyme_Csim)
S2 = as.matrix(enzyme_Gsim)
#g1 = graph.incidence(A)
```

```
## Compute NBI
P1 <- nbiNet(A,alpha=0.5, lamda=0.5,  s1=S1, s2=S2,format = "matrix")

## Create a list where to store the matrices
perm = list()

## Set a random seed
set.seed(12345)

## Compute scores for 1000 permutations where you sample the matrix everytime
for ( i in 1:1000){
    A <- t(enzyme_ADJ)
    A <- A[sample(nrow(A)),sample(ncol(A))]
    S1 = as.matrix(enzyme_Csim)
    S2 = as.matrix(enzyme_Gsim)
    S1 <- S1[sample(nrow(S1)),sample(ncol(S1))]
    S2 <- S2[sample(nrow(S2)),sample(ncol(S2))]
    R1 <- nbiNet(A,alpha=0.5, lamda=0.5,  s1=S1, s2=S2,format = "matrix")
    perm[[i]] <- R1
}

## Get the mean and standard deviation of the matrix

mean_mat <- apply(simplify2array(perm), 1:2, mean)
sd_mat  <-  apply(simplify2array(perm), 1:2, sd)

## Comput the Z-score of matrix
Z <- (P1 - mean_mat)/sd_mat

## Compute the significance score
sigNetwork <- 2*(pnorm(-abs(Z)))

## Get the significant interactions where, P < 0.05
sigNetwork[sigNetwork <  0.05] <- 1
sigNetwork[sigNetwork != 1] <- 0

sum(A) ## Total number of interactions we had earlier
[1] 1515

sum(sigNetwork) ## Total number of interactions after computation.
[1] 2368
```

---

5. Citations

[1] Kohler S, et al. Walking the Interactome for Prioritization of Candidate Disease Genes. American Journal of Human Genetics. 2008;82:949 – 958.

[2] Can, T., Camoglu, O., and Singh, A.K. (2005). Analysis of protein-protein interaction networks using random walks. In BIOKDD '05: Proceedings of the 5th international workshop on Bioinformatics (New York, USA: Association for Computing Machinery). 61–68

[3] Cheng F, et al. Prediction of drug-target interactions and drug repositioning via network-based inference. PLoS Comput. Biol. 2012;8:e1002503.

[4] Zhou T, et al. Solving the apparent diversity-accuracy dilemma of recommender systems. Proc. Natl Acad. Sci. USA 2010;107:4511-4515.

[5] Zhou T, et al. Bipartite network projection and personal recommendation. Phys. Rev. E Stat. Nonlin. Soft Matter Phys. 2007;76:046115.

[6] http://data2quest.blogspot.com/2015/02/link-prediction-using-network-based.html

[7] Vanunu O, Sharan R. Proceedings of the German Conference on Bioinformatics. Germany: GI; 2008. A propagation-based algorithm for inferring gene-disease assocations; pp. 54–63.

[8] Chen X, et al. Drug–target interaction prediction by random walk on the heterogeneous network. Mol. BioSyst 2012;8:1970-1978

[9] Seal A, Ahn Y, Wild DJ . Optimizing drug target interaction prediction based on random walk on heterogeneous networks Journal of Cheminformatics 2015, 7:40.

[10] Truchon et al. Evaluating Virtual Screening Methods: Good and Bad Metrics for the "Early Recognition" Problem. J. Chem. Inf. Model. (2007) 47, 488-508.

[11] Sheridan RP et al. Protocols for bridging the peptide to nonpeptide gap in topological similarity searches. J. Chem. Inf. Comput. Sci. (2001) 41, 1395-1406.

[12] Alaimo S, Pulvirenti A, Giugno R, Ferro A: Drug-target interaction prediction through domain-tuned network-based inference. Bioinformatics 2013, 29(16):2004-2008.