# PROBLEM FORMULATION

Let the prediction method , produce a gene ranking for disease and seed genes , with known data . A given method may require different elements within , such as gene graphs, gene-gene correlation matrices, gene-disease associations, disease to descriptor associations, or others.

## Disease Gene Prediction

A disease gene prediction instance is expected to produce the best possible ranking for a given target gene in , we denote the prediction function as , and the ranking . Naturally, gene is restricted to be associated with in .

From this definition, the performance of two predictions and are comparable through the position of in and -- i.e. prediction is better than if it ranks closer to the first position, formally: . However, the performance of a set of predictions is established through a statistic measure on the individual predictions. In particular, we calculate how many predictions are found among the top *n* predictions:

### Evaluation measure

In this work, we evaluate the performance of a disease prediction method by calculating the fraction of predictions in which the target is found among the top ranked results. We consider the top predictions: targets found among the top 1 and top 10 stand as perfect and very high performance in prediction, top 100 represents a good prediction and top 200 is an acceptable prediction (notice that 200 results represent the top 1%-2% genes in the interactomes used).

Formally, for a test set , let the performance of the prediction be a vector in , where is the number elements in , defined as:

where the indicator function establishes if the target is among the top ranked genes, andis defined as:

### Building test sets

The definition of a disease prediction instance requires the existence of a single target to predict per test. This constraint naturally allows the creation of synthetic leave-one-out test sets. However, some diseases gain multiple gene associations over time. Therefore, disease gene prediction of these diseases must split each gene association in a separate prediction instance. Splitting associations as multiple instances have the unfortunate effect of considering the yet unknown associations as negative results; nonetheless, the effect is negligible due to the amount of real non-targets present in the predictions.

To illustrate, let genes and be new associations of disease , which need to be predicted. Let be the gene ranking obtained for disease from the information held at the moment of the predictions (where represents wrong predictions). When is being evaluated, is regarded as a wrong prediction, but it is not significant, since is ranked higher than . However, when is being evaluated, is regarded as a wrong prediction, producing an average error of order in its ranking. The error is small on its own, and it is further decreased as the test instances are averaged over all the set, and only produces an insignificant lower score in the performance evaluation of a method. Furthermore, as all methods are evaluated with the same measure, the relative performances are not affected.

## Disease Module Prediction

A disease module prediction instance is expected to produce the best possible ranking for a set of target genes , we denote the prediction function as , and the ranking . Naturally, target genes are not seed genes , and none of them are associated with in .

### Normalised AUC

In this work we evaluate an individual prediction considering results only up to the first false positive results of the ROC curve. We can understand the normalisation of the curve, as a “zoom” up to a false positive rate in the curve ( is the amount of non-targets in ), to rescale x-axis a range of . Note that the ROC curve does not necessarily reach 1 in the -axis; it will reach 1 if all targets are found before non-targets within . Furthermore, the random value is not the diagonal of the graph as in the traditional ROC curve; random expectation is as line from to ), and AUC.

Formally, we can define the normalised ROC for false positives on a vector of ranked predictions , with the series of points as varies within the interval . The true positive rate of the top predictions is calculated as usual, as:

While the false positive rate is calculated for non-targets, as:

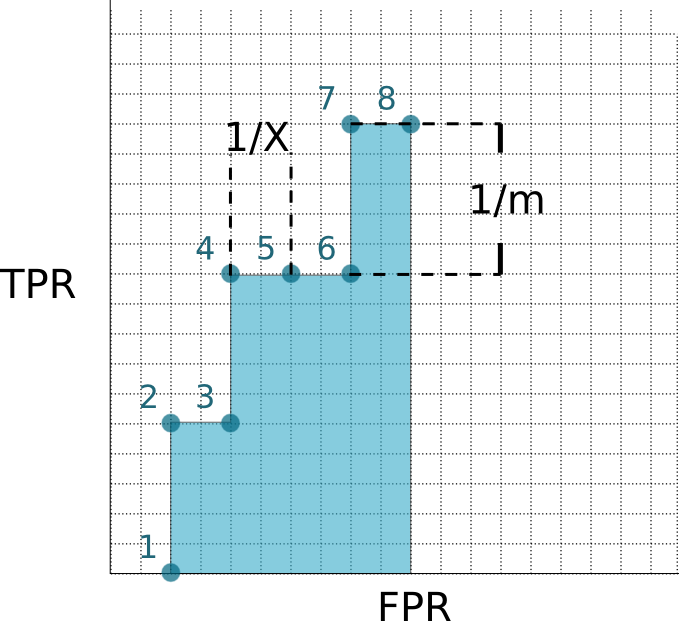


Figure 1: Example ROC curve normalised to X false positives for a discrete prediction sequence. In general, there are m possible right answers, and X wrong answers available. This example shows a prediction in which the first eight predictions are in order: wrong, right, wrong, right, wrong, wrong, right (note that this figure is partial, and the ROC X curve still continues until it reaches 1 in the FPR-axis).

Figure 1 illustrates the curve built as points are added to the prediction. Intuitively, the area under the entire ROC curve is given by calculating the rectangular areas under each different TPR value:

where is the set of correct predictions. Following the example of Figure 1, this would yield the sum .

*Note that while it might seem more natural to calculate the bounded ROC measure using the top predictions instead of the top false predictions (which can have different lengths, as the number of true positive results can vary). This option could yield misleading results as seen in the following example:*

*Pretend we are interested in evaluating the ROC curve for a vector with 5 elements and 2 targets. The ideal ranking would be =[target, target, x, x, x], while =[target, x, x, target, x] would be clearly a worse ranking. The AUC of vector considering only the first two elements would be 0 (there is no step in the axis after two values), while the AUC of vector would be 0.5. Under these conditions, a higher measure does not reflect an improvement in the results. The proposal of keeping the top results until reaching the first false positives gives an AUC of for vector (a perfect prediction), and for vector , which are expected results when comparing both vectors.*

# CARDIGAN (the mathematical formulation)

Formally, the static inputs for Cardigan are an undirected graph serving as the interactome (represented by an adjacency matrix ), the disease similarity matrix , and the known disease gene associations . The dynamic inputs are the disease , or list of diseases when dealing with a disease family , i.e. *the query* and its known genes .

For a given prediction, Cardigan collects all genes associated with any disease, and assigns the highest similarity of any diseases containing the gene and the query in , defined as:

Then, and are converted into the *Query Weight Set* (QWS) used for the diffusion vector , defined as:

Where and are the centre, slope of the sigmoid, and is relative importance of a gene from a disease other than the query. The diffused labels are those which minimise the cost function , i.e. . The cost function is defined as:

The vector is sorted in decreasing order by vector , such that and is the -th largest value in , and is the output of the method.

## Parameters of the method

The method includes several parameters which were tuned for the release version, using diseases which were removed from the every evaluation set. The disease was Acute Lymphoblastic Leukemia (MIM613065)

### Sigmoid

The centre and slope from the sigmoid were tuned so different values in would approach the 0.1, and 0.9 values after the sigmoid is applied. Real values in S range from 0 to 4, therefore and changed in amounts allowing and , such that all combinations of 0.1 steps of were tested. The height was thereafter tested between 0.05 and 1 in intervals of 0.05.

### Diffusion

The Zhou method contains a single parameter to tune, . The parameter was tested from 0.01 to 0.99 in intervals of 0.01.

# Datasets

We compiled some relevant facts about the networks used and disease gene associations, in order to give a sense of possible recall rates for the different graphs in the paper.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| OMIM Release | # of Charted Diseases | # Charted Diseases with 2 or more known genes | # of Uncharted Diseases | # of Different Genes | # of Disease Gene Associations |
| 2013 | 4870 | 293 | 2670 | 4040 | 6303 |
| 2017 | 5992 | 264 | 2388 | 4820 | 7292 |

***Table 1: Relevant counts from the OMIM Databases****. The number of charted diseases includes diseases with 2 or more known genes. The number of uncharted diseases accounts for all diseases with no known molecular basis which have annotated publications (diseases with annotated suspected genes are not included in this count). The number of different genes is the set of all disease genes (some genes belong to multiple diseases). The number of disease gene associations are all unique disease-gene pairs annotated in the OMIM database.*

Notice that while the 2017 OMIM database includes 989 more disease associations than the 2013 release, over 1400 associations are not included in the older version. This difference comes from changes in diseases identifiers between versions, changes in the genes associated to the diseases, and even removed diseases.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Network | Nodes | Edges | Coverage OMIM2013 | Coverage OMIM2017 | Edge Type |
| *HPRD* | 9670 | 39220 | 54% | 54% | Exp. Binary |
| *DiamondNet* | 13460 | 141296 | 65% | 66% | Exp. + Inf. Binary |
| *BioGRID* | 19803 | 279187 | 69% | 71% | Exp. Binary |
| *HIPPIE* | 16552 | 239684 | 71% | 74% | Exp. Weighted |
| *FUNCOUP* | 18113 | 4476818 | 71% | 74% | Exp. + Inf. Weighted |

***Table 2: Characteristics of Protein-Protein interaction networks****. Coverage shows the fraction of the different disease genes from the OMIM database found in the network (see Table 1). Edge type shows the type of evidence (exp. for experimental, inf. for inferred), and whether the edges are binary or weighted.*

Note that even if a gene is found in a network some methods can still fail to predict it. Both ProDiGe and DIAMOnD in practice require that the predicted genes are located in the main connected component of the network[[1]](#footnote-1). However, Cardigan is still able to predict genes within the other components of the network if there is a known disease gene in said component.

# Execution times

Although time is not an essential aspect for disease gene prediction, the run time of the used turn appear to vary considerably. We produced a table with an average prediction time for each algorithm using different networks. *Note that these measurements do not include the initial time to load data into memory for any of the methods.*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Network | Cardigan | DIAMOnD | Prodige 1 | Prodige 4 |
| HPRD | 0.48 | 1.32 | 2.59 | 176 |
| DiamondNet | 0.62 | 3.86 | 4.79 | 291 |
| BioGRID | 0.92 | 7.73 | 9.98 | 433 |
| HIPPIE | 0.75 | - | - | - |
| FUNCOUP | 1.10 | - | - | - |

***Table 3: Average run times in seconds for a single prediction on different interactomes****. The averages were taken from the same test set with over 100 predictions on the same system. Intel XEON 2.6GHz, 32 GB RAM running Debian Jessie.*

Notably, all the presented methods except DIAMOnD produce a ranking for all genes in the network. DIAMOnD is only producing 200 predictions, as the iterative nature of the method allows the procedure to stop when a certain number of results is produced.

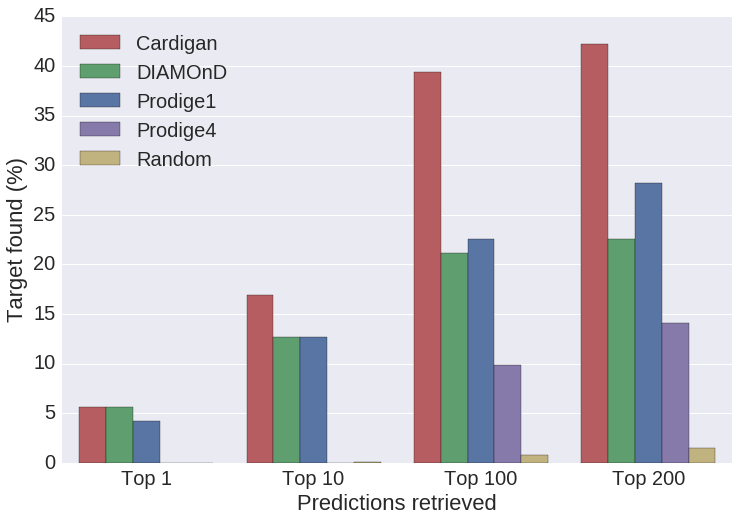
# Other results

We include the results on different combinations of networks and OMIM databases to make an extensive observation of the results presented in the main paper. As the implementations provided by ProDiGe1, ProDiGe4 and DIAMOnD can only run on binary networks, tests are produced for HPRD (in the main paper), DiamondNet and BioGRID.

Since only Cardigan could produce results for *time-lapse uncharted* diseases (ProDiGe4, the only other method that could in principle make predictions for uncharted diseases, is not applicable since its disease kernel does not include any of these diseases), we show the comparison for *time-lapse charted*, *leave-one-out charted* and *leave-one-out uncharted* experiments.

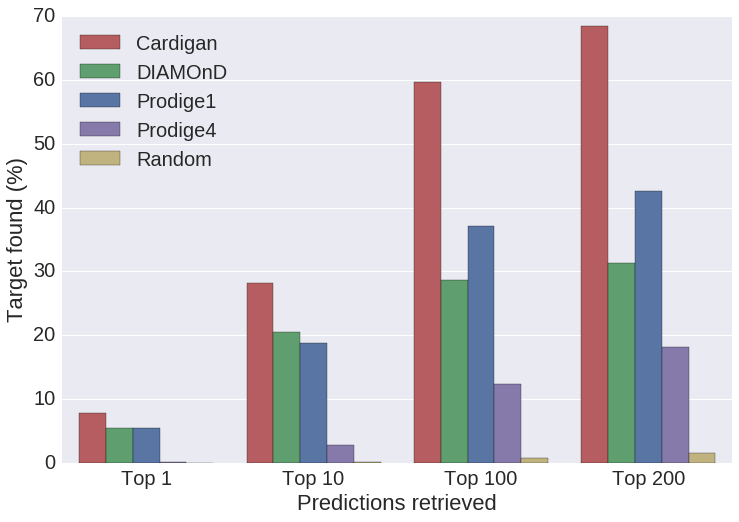
## Results using DiamondNet

**Time Lapse Charted**



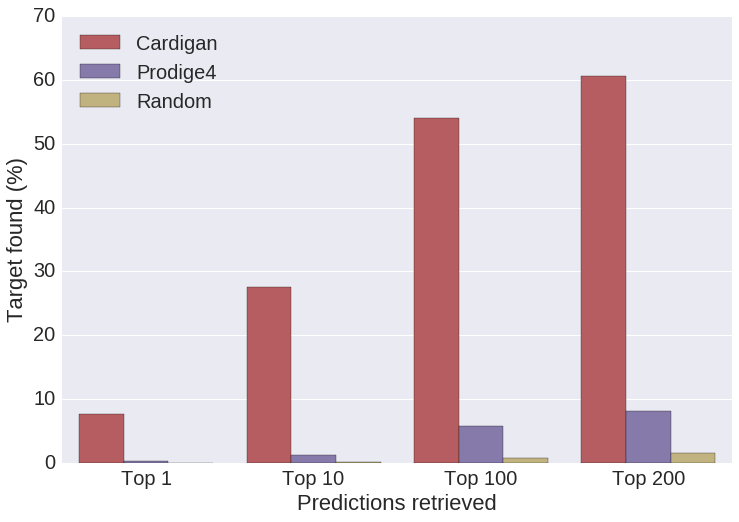
Out of the 1413 disease gene associations which were new in the 2017 version of OMIM, only 95 of them were added to diseases which were already charted in 2013, 71 of those can be predicted using DiamondNet.

**Leave-one-out Charted**



The 2017 OMIM database contains 264 diseases with two or more genes, which result in 970 possible test cases, 875 of them can be predicted using DiamondNet.

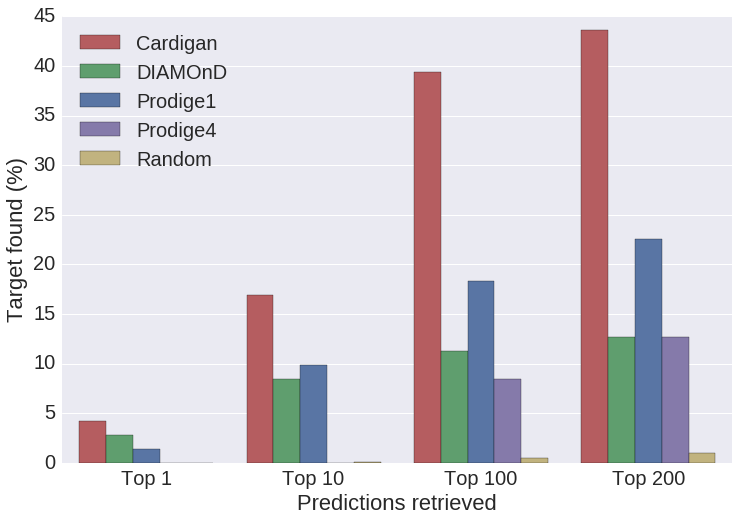
**Leave-one-out Uncharted**



There are 5707 diseases with a single disease gene in the 2017 OMIM database, which result in 5707 possible test cases, 3029 of them can be predicted using DiamondNet.

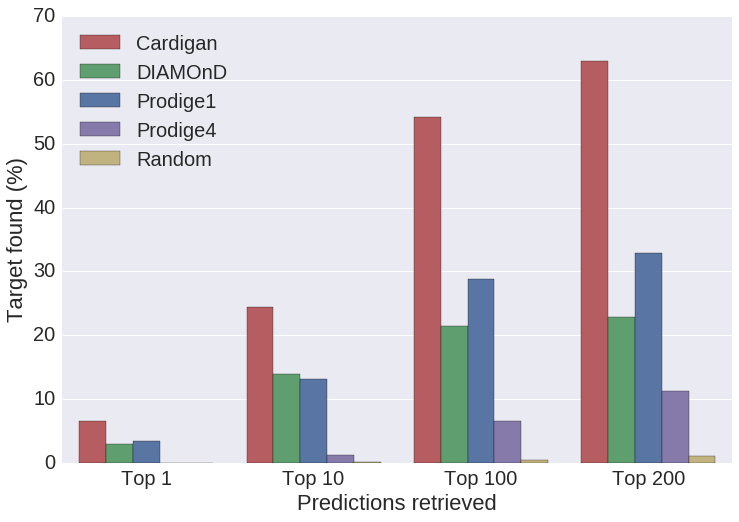
## Results using BioGRID

**Time Lapse Charted**



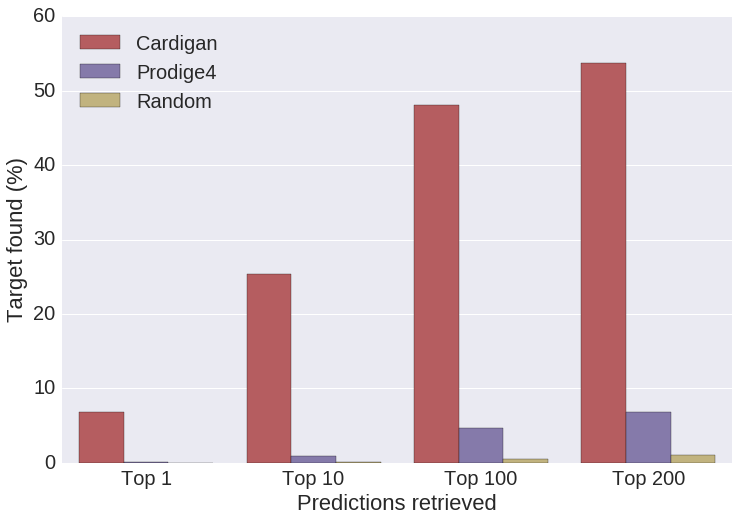
Out of the 1413 disease gene associations which were new in the 2017 version of OMIM, only 95 of them were added to diseases which were already charted in 2013, 71 of those can be predicted using BioGRID.

**Leave-one-out Charted**



The 2017 OMIM database contains 264 diseases with two or more genes, which result in 970 possible test cases, 893 of them can be predicted using BioGRID.

**Leave-one-out Uncharted**



There are 5707 diseases with a single disease gene in the 2017 OMIM database, which result in 5707 possible test cases, 3208 of them can be predicted using BioGRID.

# Running the code

## Dependencies

We provide the code to run Cardigan on Python 2.7, however the code has the following library dependencies:

* NumPy <http://www.numpy.org>
* SciPy <https://www.scipy.org>

Besides that, there are some data dependencies to run the code, however terms of service prevent us from the redistribution of most of the following datasets:

## OMIM database: the disease-gene association file (morbidmap), and the MIM to gene translation file (mim2gene).

## Caniza matrix: the code to generate this matrix is provided by the authors http://www.paccanarolab.org/disease\_similarity/, and we host the 2017 computed matrix used for this paper.

## PPIs: we provide parsers for the original networks used in the paper (HPRD, DiamondNet, BioGRID, HIPPIE, FUNCOUP). We use Entrez as our preferential identifier, when this identifier is not available in the database, we require an extra translation file to obtain the Entrez identifier.

## Installation

Download the desired bundle from our website (paccanarolab.org/cardigan), and decompress it in the target directory. Further instructions can be found in the README file.

## Example

The basic prediction using Cardigan can be done in a single line (see Minimum working example). This prediction uses all the disease-gene associations found in the provided OMIM database.

*Example: Basic usage*

|  |
| --- |
| import Config  import Cardigan  #use the default configuration  config=Config()  cardigan=Cardigan(config)  #predict new ALL genes, starting with the gene known in 2013  out=cardigan.predict([‘613’],[‘613065’],[],0)  print out |

However you can modify the parameters of the predict function to run a synthetic leave-one-out test.

*Example: Synthetic leave-one-out for a charted disease*

|  |
| --- |
| import Cardigan  # load the module using the default configuration  cardigan = Cardigan.Cardigan()  # predict genes for BDPLT16 (MIM: 187800)  # keep gene 3690 as a seed and use 3674 as a target  out = cardigan.predict(['187800'],['3690'],['3674'])  targetPos = cardigan.evaluate()  print targetPos |

Further details can be found in the software README provided in the paper website (paccanarolab.org/cardigan/).

1. While ProDiGe4 in theory could produce results using the entire network, it failed to retrieve any predictions among the top 200 results when the target gene was outside of the main connected component. [↑](#footnote-ref-1)