Predicting Gene Ontology

(GO) terms on protein sequences

Report Exercise 3, group C

Dominik Achten
Michael Kiening
Maximilian Hecht
Peter Hönigschmid
Florian Auer
Manfred Roos

The first task to accomplish this exercise was to implement the basic three methods.

Blast output parsing

We have written a parser that efficiently condenses the information included in the Blast-output-files, such as score, e-value, identities and positives. The result is grouped by the GO-terms found for every specific target, and can be further processed.

```
Target: T09413 (4OT_PSEST)
3e-16   82      49/63   56/63   0       GO:0016853,GO:0019439
2e-15   79      56/63   57/63   0       GO:0016853,GO:0019439
3e-15   79      25/60   38/60   0       GO:0016853,GO:0006725
6e-15   78      20/59   39/59   0       GO:0016853,GO:0006725
1e-14   77      26/60   35/60   0       GO:0016853,GO:0006725
2e-14   76      26/61   40/61   0       GO:0016853,GO:0006725
3e-14   75      31/63   48/63   0       GO:0016853,GO:0019439
7e-14   74      22/59   38/59   0       GO:0016853,GO:0006725
1e-13   74      21/60   36/60   0       GO:0016853,GO:0006725
4e-13   72      25/59   37/59   0       GO:0016853,GO:0006725
1e-12   70      26/60   38/60   0       GO:0016853,GO:0006725
2e-12   69      24/56   37/56   0       GO:0016853,GO:0006725
3e-12   69      19/60   30/60   0       GO:0016853,GO:0006725
1e-11   67      22/56   34/56   0       GO:0016853,GO:0006725
2e-11   66      22/66   38/66   3/66    GO:0016853,GO:0006725
5e-10   61      19/57   35/57   0       GO:0016853,GO:0006725
2e-09   60      22/64   34/64   1/64    GO:0016853,GO:0006725
1e-08   57      20/62   33/62   1/62    GO:0016853,GO:0006725
7e-04   41      10/55   26/55   0       GO:0016853,GO:0006725
0.001   40      14/52   27/52   1/52    GO:0005737,GO:0016862,GO:0006725
0.004   38      13/53   24/53   0       GO:0005737,GO:0016862,GO:0006725
0.023   36      12/52   25/52   1/52    GO:0005737,GO:0016862,GO:0006725
0.056   35      12/50   22/50   0       GO:0016853,GO:0006725
0.062   34      12/59   23/59   0       GO:0016853,GO:0006725
***
```

Example for the parser output

Scoring

In a second step the output is scored according to a specific scoring function:

- Confidence values:

  First of all, we splitted the GO tree into two subtrees. One for the molecular function entities, and one for the biological process ones. This guarantees that there are no connections between the two ontologies.
  The confidence values are calculated totally separate for each of the two trees, resulting in at least two predictions one for the BFO and one for the MFO (more predictions can occur for each ontology if more than one predictions have the highest confidence).

  - Method 3:

    The confidence values calculated for method 3 use all BLAST hits found above the e- value threshold.

    Do the following for every BLAST hit:
    Let GOx be the current GO term of the current BLAST hit.
    - Add one support point to GOx and all of its parents.
    - Assign the maximum positives value to GOx viewn so far (means later hits of GOx just add support points, but do not change the positives score used by this GO term).

    After that, find the GO term with the maximal support (hereinafter max_sup) points excluding the root and the MFO/BFO term.

    To get the confidence score for each GO term, do the following foreach GOx:
    - calculate the proportion from the support of GOx to max_sup and multiply it by the positives score of GOx
    - get the parents of GOx and add the previous calculated score to their current score.

    The final confidence is the maximum of the current score of the GO term, and one.

  - Method 1:

    Method 1 works just as method 3, but using only the first BLAST hit, instead of the whole list.

  - Method 2:

    Method 2 is similar to method 3, but the parents of each GOx get the maximum support of each GOx which is a child of them instead of summing them up. Also the positives score is not taken into account, which means the score for each GO term is just the support of it divided by max_sup.

- Precision/Recall:

For each target precision and recall are calculated seperatly for each of the two Ontologies. Since we were asked to only hand in the most confident prediction, every target can produce up to 2 Precision/Recall values. This restriction turned out to improve precision accuracy while lowering the recall value. At the end, the precision and recall values are summed up and divided by their quantity for the accuracy line in the CAFA output file. Targets with no BLAST hits and thus no prediction, are not taken into account calculating precision and recall values.

In order to get results just in time we decided to divide and conquer: The Database was divided into 100 small sets. Foreach small subset a BLAST-job was queued, all blast results where parsed to our intermediary format and merged. For example Euk_set3.fasta_blast.stat contains all results of psiblast on Euk_set3.fasta parsed by exercise3.pl .
We just spared out weeks to wait for a single big BLAST job and a “process killed by to little RAM” while trying to parse the big output file.

validation and assembly

To make our methods usable for external people, we developed 3 small wrappers, which call the methods in correct order:

```
Wrapper 1: "/mnt/opt/data/pp2_exercise/groups/groupC/CafaWrapper1.pl"
Wrapper 2: "/mnt/opt/data/pp2_exercise/groups/groupC/CafaWrapper2.pl"
Wrapper 3: "/mnt/opt/data/pp2_exercise/groups/groupC/CafaWrapper3.pl"
```

The wrappers can be run with the three CAFA-parameters.

Finally we validated the results produced by our program pipeline. For this task we developed a parser that splits a input database into a certain number of testsets and trainingsets. For our purpose 100 sets each seemed sufficient. We also tried to optimize the three parameters, but we came to the conclusion that the default parameters produced the most reliable output.
Our validation of method 3 resulted in the following values:
Precision: 97%
Recall: 70%
with as little as 0.01 standard deviation. Since the best Blast-hit in almost every case included the best scoring GO-term(s), our more sophisticated methods could hardly perform any better than the trivial method, that only uses the best blast hit against the given database.