Bioinformatics, 31(10), 2015, 1544–1552
doi: 10.1093/bioinformatics/btu851
Advance Access Publication Date: 8 January 2015
Original Paper

OXFORD

Sequence analysis

PANNZER: high-throughput functional annotation of uncharacterized proteins in an error-prone environment

Patrik Koskinen[1,*,†], Petri Törönen[2,†], Jussi Nokso-Koivisto[2] and Liisa Holm[1,2]

[1]Department of Biosciences, University of Helsinki, 00014 Helsinki, Finland and [2]Institute of Biotechnology, University of Helsinki, 00014 Helsinki, Finland

*To whom correspondence should be addressed
Associate Editor: John Hancock

[†]The authors wish it to be known that, in their opinion, the first two authors should be regarded as Joint First Authors.

Received on June 17, 2014; revised on December 11, 2014; accepted on December 24, 2014

Abstract

Motivation: The last decade has seen a remarkable growth in protein databases. This growth comes at a price: a growing number of submitted protein sequences lack functional annotation. Approximately 32% of sequences submitted to the most comprehensive protein database UniProtKB are labelled as 'Unknown protein' or alike. Also the functionally annotated parts are reported to contain 30–40% of errors. Here, we introduce a high-throughput tool for more reliable functional annotation called Protein ANNotation with *Z*-score (PANNZER). PANNZER predicts Gene Ontology (GO) classes and free text descriptions about protein functionality. PANNZER uses weighted *k*-nearest neighbour methods with statistical testing to maximize the reliability of a functional annotation.

Results: Our results in free text description line prediction show that we outperformed all competing methods with a clear margin. In GO prediction we show clear improvement to our older method that performed well in CAFA 2011 challenge.

Availability and implementation: The PANNZER program was developed using the Python programming language (Version 2.6). The stand-alone installation of the PANNZER requires MySQL database for data storage and the BLAST (BLASTALL v.2.2.21) tools for the sequence similarity search. The tutorial, evaluation test sets and results are available on the PANNZER web site. PANNZER is freely available at http://ekhidna.biocenter.helsinki.fi/pannzer.

Contact: patrik.koskinen@helsinki.fi

Supplementary information: Supplementary data are available at *Bioinformatics* online.

1 Introduction

A correctly annotated proteome is the cornerstone of a successful genome research project and therefore accurate and reliable functional annotation tools are needed. However, due to the huge amount of various sequence data and diverse methods used in the functional annotation processes, a large part of these sequences are at risk of being annotated incorrectly (Hadley, 2003; Naumoff *et al.*, 2004; Punta and Ofran, 2008).

The last decade has seen an explosion in the number of genomes being sequenced, and the near future will increase the number far higher. Experimental characterization is not a viable option for

large-scale functional annotation of whole proteomes or large environmental samples. High-throughput experiments using, e.g. mass-spectrometry or RNAi methods yield biased and less specific information about protein function than low-throughput methods (Schnoes *et al.*, 2013). Low-throughput methods are time consuming, complex and expensive and therefore restricted only to small subsets of proteins of interest (Sboner *et al.*, 2011). Annotations are also generated by biocurators by interpretation of experiments from literature. The quality of these literature-based annotations relies heavily on the expertise of biocurators (Brenner, 1999; Schnoes *et al.*, 2009, 2013).

While experimental methods have problems, the computational methods struggle on a whole new level of challenges. The error rate of computationally annotated databases has been increasing rapidly in recent years. A recent study estimates the error level has risen from 5 to 40% within the last decade (Schnoes *et al.*, 2009). In the Gene Ontology (GO) databases the error levels grow even higher: among computationally created GO annotations, the error level has been estimated to be as high as 49% and even within manually curated GO annotations between 28 and 30% (Jones *et al.*, 2007). The increasing error rate in these databases is believed to stem mostly from the propagation of erroneous annotations with usage of poorly performing *in silico* functional annotation tools (Wieser *et al.*, 2004; Gilks *et al.*, 2002, 2005).

We have designed a high-throughput annotation tool called Protein ANNotation with *Z*-score (PANNZER) in order to create more reliable annotations and thereby reduce further error propagation in annotation projects. PANNZER uses the whole sequence similarity neighbourhood and weighted statistical testing in the annotation process in an attempt to maximize the evidence for correct annotation. In doing so, PANNZER prevents function transfer from incorrectly annotated sequences to an uncharacterized sequence.

Here, we evaluate PANNZER in two separate tasks: in the prediction of free text description lines (DE) and in the prediction of GO classes. The description line is a free text sentence about the protein function. Written by biologists, it contains valuable information in human readable format. Therefore, it is surprising how little attention correct DE annotation has gotten in recent years. Some methods do exists, e.g. GeneQuiz (Scharf *et al.*, 1994), PEDANT (Frishman *et al.*, 2001), AutoFACT (Koski *et al.*, 2005) and Blannotator (Kankainen *et al.*, 2012). We introduce a principled metric for the evaluation of description prediction, which allows numerical comparison of description similarities. In the prediction of free text description line, we show a clear improvement to other existing methods.

The GO functional annotation has become the standard tool in computationally based bioinformatics analyses. Due to this, the majority of method development in functional annotation is nowadays focused on GO classes, e.g. GOtcha (Martin *et al.*, 2004), Argot^2 (Falda *et al.*, 2012) and Blast2GO (Götz *et al.*, 2008). A more comprehensive list of GO prediction tools can be found from Radivojac *et al.* (2013). Our results show an improved performance over alternative scoring methods and we also show improvement to our earlier version of PANNZER that was ranked third in Critical Assessment of protein Function Annotation algorithms (CAFA) 2011 challenge (Radivojac *et al.*, 2013).

2 Methods

The PANNZER method predicts protein function using a weighted *k*-nearest neighbours approach with statistical testing.

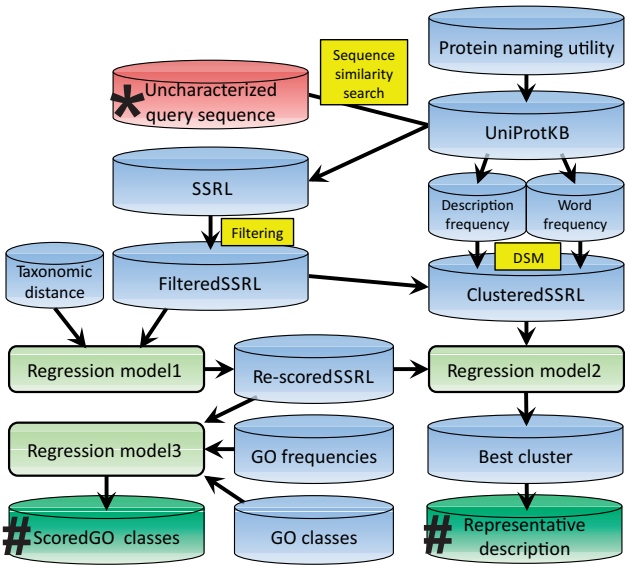

Fig. 1. Schema of the PANNZER methodology. Input for the PANNZER is indicated with asterisk (*) and outputs with hash symbols (#)

The functional annotation can be predicted as description lines or GO classes. PANNZER starts with a sequence search against the sequence database. The resulting Sequence Similarity Result List (*SSRL*) provides the candidate descriptions. The *SSRL* is then partitioned into clusters according to description similarity. The support for the candidate clusters is evaluated using a sophisticated regression model. The description prediction outputs the most representative description. The GO prediction performs an enrichment analysis of GO classes in the *SSRL*. These steps are outlined in Fig. 1 and described in detail below.

2.1. Sequence search and filtering of results

PANNZER starts the analysis with a sequence similarity search against UniProtKB database (Magrane and Consortium, 2011). Here we have used standard BLAST (Altschul *et al.*, 1997) search, but other sequence search methods could be used (e.g. SANS—Koskinen and Holm, 2012).

A large number of locally similar but globally dissimilar sequences in the result list sometimes biases results towards large sequence families. Therefore, we limit the number of sequences taken to the analysis and focus only on the sequences that obtained the strongest results from the sequence scoring and apply pre-set filtering thresholds on alignment coverage, identity percentage, sequence length and informative descriptions. The alignment coverage values are obtained by evaluating the sequence alignment areas covered in the query and target sequence. The information density threshold restricts sequences monitored only to the sequences with informative descriptions and omits uninformative descriptions like 'putative uncharacterized protein'. The selection of informative descriptions is based on Information Density Score (IDS):

$$\mathrm{IDS} = \frac{1}{n}\sum_{i=1}^{n} \mathrm{idf}(w, D)^2, \tag{1}$$

where w is a word in description D and n is the number of words in a description D. idf(w, D) is an Inverse Document Frequency score for a word:

$$\mathrm{idf}(w, D) = \log\left(\frac{|D|}{|\{d \in D : w \in d\}|}\right), \tag{2}$$

where $|D|$ is the total number of descriptions in the corpus (i.e. database), and $|\{d \in D : w \in d\}|$ is the number of descriptions d where the word w occurs. IDS emphasizes the descriptions that contain specific terms over descriptions containing only general terms. The assumption here is that the words conveying specific functional information are rarer in corpus than the general words.

Filtering thresholds are shown in the Supplementary Material. These values were optimized against the training set. The filtered *SSRL* is used in later steps of PANNZER (see Fig. 1).

2.2. Non-linear weighting of taxonomic distances

One information source that is omitted by the standard sequence search is the evolutionary distance between the query and target species. It is intuitive to give more emphasis to sequence matches found from species that have smaller evolutionary distance.

Unfortunately we do not have evolutionary distances available across all the species and therefore we decided to use the distances in the NCBI taxonomic tree as an approximation for the evolutionary distances. Correlation between these distances and description similarities were not linear since according to the general paradigm, paralogs are more likely functionally differentiated. This was corrected with a non-linear similarity function between the descriptions of compared query and target sequence.

More details on this process can be found in the Supplementary Material. The output from this process, non-linear taxonomic distance score, was one of the inputs for the re-scoring of sequence hits.

2.3. Re-scoring sequence hits

In the second step of the PANNZER pipeline we re-score the sequence hits using a sparse regression model that combines various signals from sequence alignment and the signal from the non-linear taxonomic distance score.

All regression models were created in the *R* analysis environment using Lasso and LEAPS packages (Miller, 2012; Robert, 1996). We computed for each found sequence match various values that measure the goodness of the match to the target sequence. These are the BLAST bit score, the BLAST e-value, the sequence identity, the query alignment coverage and the target alignment coverage. In addition, we had the taxonomic distance-based weight score. The regression model was trained against the Description Similarity Measure (DSM) of training data (see Supplementary Material on training). We found that the BLAST bit score or the e-value alone are not the best correlating scores against the functional annotation. The final model, *RegressionModel*1, combines several measures:

$$RegressionModel1 = 0.21 \times \log_{10}(\text{idp}) + 0.26 \times (\text{Cov}_q \times \text{Cov}_t) + 0.40 \times (\text{Cov}_t \times TaxDist_{q,t}), \quad (3)$$

where idp is the alignment identity percentage and Cov is the alignment coverage in query (q) and target (t) sequences. The *TaxDist* (Supplementary Fig. S1) is the non-linear taxonomic distance score between query and target species.

2.4. Description similarity measure

In PANNZER we use the DSM to cluster candidates in the *SSRL*. The DSM is based on the Term Frequency - Inverse Document Frequency (*tfidf*) which is a standard information retrieval and text mining method and is commonly used in clustering related documents in e.g. web search engines. The *tfidf* is a weighting scheme for measuring how important a word is to a document in a corpus. The score upweights words which are frequent in the document and at the same time downweights words which are frequent in the corpus. This kind of an approach helps to control the fact that some words are generally more common than others and therefore possibly have a trivial meaning. For example, the word 'protein' is frequently used in non-related cases in biological databases and is not very informative to a functional description, whereas the word 'hydroxyltransferase' occurs rarely and only in certain descriptions and is therefore more descriptive. *tfidf* is defined as:

$$tfidf(w, d, D) = \text{tf}(w, d) \times \text{idf}(w, D), \quad (4)$$

where $\text{tf}(w, d)$ is the frequency of word w in document d. D is the corpus (i.e. database). To measure how similar two descriptions are to each other, the *tfidf* is used to weight common words in descriptions. In PANNZER the description similarity is calculated by using the *tfidf* weighting in a cosine similarity function. The cosine similarity is a dot product of the *tfidf* scores of common terms between two descriptions. It is calculated using the following equation:

$$\text{DSM} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} A_i^2 \times \sum_{i=1}^{n} B_i^2}}, \quad (5)$$

where A and B are vectors holding *tfidf* weights of common words between two descriptions. Examples of DSM scores between descriptions are presented in Table 1. In the PANNZER methodology the DSM is used to generate *description clusters* of *SSRL* hits by using hierarchical clustering with average linkage.

2.5. Selecting the best cluster

The next step in PANNZER methodology is to select one of the clusters as the best representative for the query sequence. We define a relevance score that is used to separate good representative clusters from the randomly occurring clusters. It would be intuitive to use a single score function for cluster voting. We, however, propose a combination of score functions that is obtained with sparse regression by training the model against the DSM (see Supplementary Material).

The relevance score (Equation 6) uses output from three score functions: GSZ (Toronen *et al.*, 2009), word score (WS) (Andrade and Valencia, 1998; Kankainen *et al.*, 2012) and weighted word score (WWS) as an input (Equations. 8–10).

$$RegressionModel2 = 0.59 + 0.53 \times \log_{\text{trunk}}(\text{WS}) + 0.02 \times \text{WWS} + 0.0004 \times \text{GSZ}_{\text{cluster}}, \quad (6)$$

where

$$\log_{\text{trunk}}(\text{WS}) = \begin{cases} \log(\text{WS}) + 1, & \text{if WS} \geq 1 \\ 1, & \text{if WS} < 1 \end{cases} \quad (7)$$

Each score function in Equation 6 was tested for preprocessing before regression with simple function (X, X^2, $\sqrt{X}$, $\log_{\text{trunk}}(X)$, etc.). These improved the performance of regression model (see Supplementary Material). GSZ, WS and WWS are defined below.

The $\text{GSZ}_{\text{cluster}}$ evaluates the overlap of two sets of sequences: (A) the *SSRL* (i.e. the BLAST result list) and (B) the set of sequences in the database that have one of the clustered descriptions (functionally related sequences). GSZ analyses overlap in a weighted manner and takes the sum of regression score values (Equation 3) for sequences that are in the cluster and creates a Z-score normalization with the mean and STD estimates discussed in the Supplementary Material

Table 1. Examples of DSM scores between different descriptions

Description 1	DSM	Description 2
AMY-1-associating protein expressed in testis 1	0.97	AMY-1-associating protein expressed in testis 1-like
Putative aryl-alcohol dehydrogenase AAD15	0.82	Similar to aryl-alcohol dehydrogenase
Acetoacetyl-CoA synthetase	0.71	Acetoacetyl-coenzyme A synthetase
Biotin carboxylase, chloroplastic	0.64	Acetyl-CoA carboxylase, biotin carboxylase
Xanthoxin dehydrogenase	0.56	Alcohol dehydrogenase
Benzoate–CoA ligase, peroxisomal	0.44	3-methylmercaptopropionyl-CoA ligase
Ethylene-responsive transcription factor ABR1	0.35	Wound-responsive AP2 like factor 1
Agamous-like MADS-box protein AGL18	0.24	Putative MADS domain transcription factor GGM9
Adrenodoxin-like protein, mitochondrial	0.15	Probable YAH1-Ferredoxin of the mitochondrial matrix
Protein arginine *N*-methyltransferase	0.05	Putative uncharacterized protein

and in earlier publication (Toronen *et al.*, 2009). This emphasizes clusters of descriptions that are common in *SSRL* but rare in the background.

For WS we first calculate WS_w for every unique word w that occurs in *SSRL* descriptions d:

$$\mathrm{WS}_w = \frac{\sum_{m \in S_w} \mathrm{Bit}_m}{\sum_{m \in \mathrm{SSRL}} \mathrm{Bit}_i}, \tag{8}$$

where S_w is the subset of *SSRL* sequences that have the word w in the description and Bit is the BLAST bit score. The WS score used in Equation 6 is an average of WS_w scores over the description d. This score function emphasises descriptions containing words that are frequently seen in the descriptions in *SSRL*. This kind of a word scoring scheme has proved to perform well in previous studies (Andrade and Valencia, 1998; Kankainen *et al.*, 2012).

The WWS_w for word w is derived from WS_w, but is weighted by Jaccard Similarity Coefficient JSC_w:

$$\mathrm{JSC}_w = \frac{|A \cap B|}{|A \cup B|}, \tag{9}$$

where A is the *SSRL* and B is the set of descriptions where the word w occurs in the whole database D. Note that $A \cap B$ is the subset of descriptions where the word w occurs in *SSRL*. WWS_w is:

$$\mathrm{WWS}_w = \mathrm{JSC}_w \times \mathrm{WS}_w \tag{10}$$

This function emphasises words that are frequent in the *SSRL* and are not frequent in the background. The WWS score used in Equation 6 is a average of WWS_w scores over the description d.

The selected score functions differed in the way how they analyse the *SSRL*. Two score functions, GSZ and the WWS in Equation 6, do standard statistical testing against the background (i.e. whole database). Only WS does not consider background. Score functions can also be split into two other groups: GSZ uses a group of sequences with similar descriptions (description cluster) as input, whereas WS and WWS define first a score for each single word observed in descriptions and then combine the score of single words to generate the signal of whole description.

Having selected the best cluster according to Equation 6, the PANNZER method selects one of the descriptions as a representative for the whole cluster. The representative description is the most frequent description in the cluster.

2.6. Scoring GO classes

A second form of annotation is the prediction of GO classes. Here we use a method that resembles the description prediction above.

First, the direct GO annotations as well as parent classes are collected from UniProt-GOA database for all the sequences in the *SSRL*. We collected different variables for *SSRL* and each GO class. These variables include the count of GO class members in the *SSRL*, the size of GO class in the whole database, the sum of *RegressionModel*1 scores within GO class members in *SSRL*, size of *SSRL*, size of whole database, etc. These in turn are used to define a various score values for each of GO classes.

Earlier similar works (Falda *et al.*, 2012; Martin *et al.*, 2004; Vinayagam *et al.*, 2004) have selected a single score function to estimate the score value for each GO class. We decided to use a weighted sum of score functions also here. Again the motivation is to emphasize the common signal in different score functions and lessen the different noise signals. The weighted sum was obtained using sparse regression. We optimized the regression against *weightedLin* similarity (see GO semantic distances below). *WeightedLin* was used to test how similar the predicted GO class was to nearest correct GO class. In the final regression model, we excluded all terms that had negative correlation with predicted variable from the model. These terms create non-linear signal that causes non-monotonic behaviour in the model and they constitute a small subset of total signal. The training and evaluation datasets are described later. The regression model that we obtained was:

$$\begin{aligned} \mathit{RegressionModel3} = {} & 0.36 + 0.01 \times \log(\mathit{JSC}) + 0.02 \times \log_{\mathrm{mod}}(\mathit{GSZ}) \\ & + (-3.67 \times 10^{-5}) \times \sqrt{|\mathit{GO}|} - 0.007 \\ & \times \sqrt{|\mathit{SSRL}|}, \end{aligned} \tag{11}$$

where

$$\log_{\mathrm{mod}}(x) = \begin{cases} \mathrm{sign}(x) \times \log(\mathrm{abs}(x)) + 1 & , \text{if } \mathrm{abs}(x) > 1 \\ x, & , \text{if } \mathrm{abs}(x) \leq 1 \end{cases} \tag{12}$$

and $|X|$ is the size of set X. *JSC* is the Jaccard Similarity Coefficient between the GO class hits in *SSRL* and GO class occurrences in the whole UniProtKB database and *GSZ* is a weighted version of hypergeometric Z-score. It is calculated for GO class members in the *SSRL* using the scores from *RegressionModel*1 as weights. GSZ is explained in the Supplementary Material and in previous publications (Toronen *et al.*, 2009). In addition, Equation 11 uses various simple functions, like squareroot, log and modified log ($\log_{\mathrm{mod}}$) to improve the regression performance (see Supplementary Material for details).

2.7. GO semantic distances

The optimization of regression models for GO classes required a similarity measure that estimates how close two GO classes,

predicted and correct, are in the GO tree. The GO similarity methods give the strongest similarity when predicted and correct GO class are exactly the same and weakening similarity when two classes move away from each other in the GO structure (Pesquita *et al.*, 2009). We used two previously published GO semantic similarity measures, Lin-score (Lin, 1998) and weighted Lin-score (Schlicker *et al.*, 2006) for regression model training and evaluation. We also used two modified versions of Lin-score and weighted Lin-score, called Path Lin-score and weighted Path Lin-score. We also used a simple Jaccard similarity (Equation 9) that compared parental classes of two tested GO classes. GO semantic distances are discussed more in the Supplementary Material.

2.8. Training and evaluation datasets for description prediction

In the PANNZER project we used training datasets to find parameters for the sparse regression model and evaluation test sets to compare prediction accuracy. The test sets used in training and evaluation are separate and the training data were never used in evaluation. Sequences were selected from UniProtKB/SwissProt (downloaded November 14, 2012) to test and evaluation datasets so that: (i) they have high-quality annotation, (ii) they do not have considerable sequence similarity with each other and (iii) they do not have strong similarity in description line with each other. More details on this process are shown in the Supplementary Material. The evaluation dataset was further divided into the eukaryote and prokaryote datasets. Viruses, environmental samples and unclear taxonomic cases were excluded from the analysis. These evaluation datasets allow the analysis of methods at different parts of evolutionary tree. Our final evaluation datasets were 2954 prokaryotic sequences and 5115 eukaryotic sequences.

2.9. Training and evaluation dataset for GO prediction

GO regression training and evaluation sets were created using the same rules as description test sets above. We first selected only the sequences that had a manually curated GO annotations (i.e. GO annotations with non-IEA evidence code). Second, we excluded sequences that had annotations only in very large GO classes. Large GO classes are the ones that have many members in the GOA database. Third, the sequences were filtered for mutual sequence similarities starting from randomly selected sequence. The final training set included 8003 sequences and final evaluation set had 80 027. More detailed description is represented in Supplementary Material. Methods were also evaluated without the GO annotations with ISS evidence code. However, qualitatively the differences between the methods stayed the same. The GO evaluation only considered GO annotations that were observed in the annotations of sequences in the BLAST results. Our scoring used only these GO annotations as True Positive set. Although this alters recall values, it does not affect the ordering of the methods.

2.10. Evaluation databases

Sequence similarity searches were conducted against a UniProtKB (downloaded November 14, 2012) database from which the evaluation and training sets were removed. This imitates the situation with novel sequence that cannot be found from database. This also ensures that there is no circular logic in the test, where simple matching of the query with itself in the database would always lead to optimal end result. This was called the NOSELF evaluation database. This was also used in the GO evaluation task.

With well-annotated sequences, like we had in our test sets, there is always the risk that annotations have already propagated to other sequence neighbours creating an annotation cloud around the query sequence. To ensure that we can test also sequences that would not have exactly matching description in their sequence neighbourhood we modified our evaluation database so that we selected first the sequence neighbourhood for each query sequence and removed every sequence from the neighbourhood that were at least 90% identical to query sequence and has an identical description annotation. This database version was called NOCLOUD. More details on the process can be found in the Supplementary Material.

2.11. Comparison to other description prediction methods

In the description prediction we used two different evaluation test sets: prokaryote and eukaryote. The eukaryote test set was found to be more difficult to functionally annotate correctly than the prokaryote set. We did the prediction for the prokaryote test set with BLAST-based methods: Best BLAST Hit (*BBH*), Best Informative BLAST Hit (*BIBH*), Blannotator (Kankainen *et al.*, 2012) and PANNZER using NOSELF and NOCLOUD databases. We also made the prediction using RAST (Rapid Annotation using Subsystem Technology) server (Overbeek *et al.*, 2013) that uses FIGfam database (Meyer *et al.*, 2009) in functional annotation. It is noteworthy that the PANNZER tool is the only method from these that does statistical testing in prediction.

The *BIBH* was derived by going through *SSRL* in best first order and removing following words from descriptions: 'hypothetical', 'uncharacterized', 'putative', 'contig', 'predicted', 'probable', 'fragment', 'genome', 'protein', 'chromosome', 'possible', 'similar', 'homolog', 'conserved', 'homologous', 'complete', 'shotgun', 'cdna' and 'family'. If after word removal there were no words left in description, the description was skipped and the procedure was repeated to the next description in *SSRL* until an informative description is found.

RAST and Blannotator are designed for prokaryotic annotation. Therefore for the eukaryote dataset were predicted by using only BBH, BIBH and PANNZER using NOSELF and NOCLOUD databases.

2.12. Comparison to other GO prediction methods

We show two different types of evaluations. One is the comparison with GO semantic similarities. This shows the performance of the generated regression models: How good they are at estimating the distance of GO classes observed in the *SSRL* from the correct GO classes. The other evaluation was based on classifier evaluation using Receiver Operating Characteristics (ROC) curves and Precision-Recall (PR) curves. These were also used to calculate Area Under Curve (AUC) values from ROC curves and maximum *F*-measure values from PR curves.

Evaluation of GO classifications has one significant problem: Extremely varying GO class sizes. This results in a situation where naive prediction that simply ranks GO classes in their size order performs really well (Radivojac *et al.*, 2013). This problem can be corrected by (*a*) evaluating each GO class separately and combining them, (*b*) excluding largest GO classes and/or (*c*) weighting each GO class prediction with its Information Content. Information Content is negative log of probability of the class ($IC = -\log(p_{class})$). It is used extensively in GO distances (Pesquita *et al.*, 2009) and it was proposed for PR and ROC curves in the CAFA challenge. We used combination of *b* and *c* in our analysis. We excluded GO classes that were larger than 1/3 of the whole data from the analysis and we weighted the remaining classes with IC

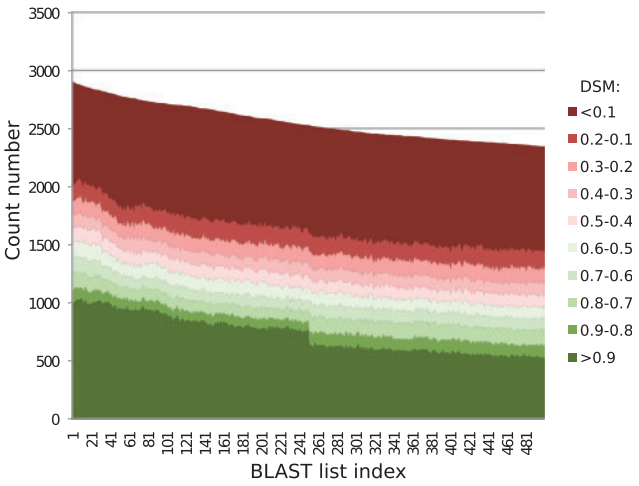

Fig. 2. Distribution of correct and incorrect descriptions in prokaryote BLAST result list. The figure is based on 2954 BLAST results from prokaryote evaluation set. It is noteworthy how evenly correct (DSM $\geq$ 0.7) and incorrect (DSM $\leq$ 0.3) descriptions are located throughout the result list. The ratio between correct and incorrect descriptions is approximately same from the first index to the last index of the list

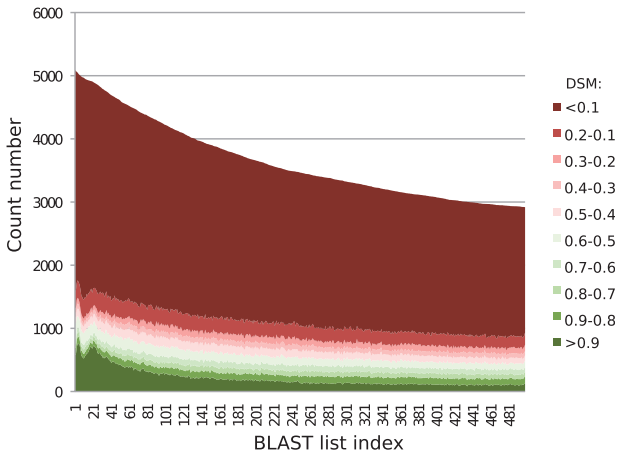

Fig. 3. Distribution of correct and incorrect descriptions in eukaryote BLAST result list. The figure is based on 5115 BLAST results from the eukaryote evaluation set. Figure shows how correct (DSM $\geq$ 0.7) descriptions are small minority throughout the whole result list. As with the prokaryote dataset the ratio between correct and incorrect descriptions stays almost the same from the first index to the last index of the result list

in the result analysis (Supplementary Material explains this task more in detail).

3 Results

The evaluation of the PANNZER method performance was conducted using description prediction and also prediction of GO classes. For the description prediction and the GO prediction we used evaluation test sets described in Section 2. We did the GO evaluation to estimate the performance improvement of our latest version of PANNZER against the PANNZER version that participated in the CAFA 2011 challenge.

3.1. Description prediction

Textual annotation, or descriptions of protein functions in biological databases, provides a concise source of knowledge about protein function and subcellular location. The computational evaluation of textual annotations has been considered to be too difficult due to usage of free text in descriptions. Here we present one approach to perform computational evaluation of free text annotations.

In this study we used the DSM (Equation 5) to calculate similarity between descriptions. The description pairs are divided into bins according to the DSM. In following results 'correct' (DSM $>$ 0.7) includes the very similar to original annotations and 'incorrect' (DSM $<$ 0.3) completely different descriptions. The bins between 'correct' and 'incorrect' (DSM $0.7-0.3$) are intermediate bins where we cannot say for sure if the annotation means the same as original or not. Table 1 shows examples at various DSM levels. We also show how these different categories distribute in the BLAST results (Figs 2 and 3).

3.2. Description prediction of prokaryotes

The first functional annotation prediction with BLAST-based methods was done against the NOSELF database and the second prediction was done against the NOCLOUD database. The results indicate that the PANNZER method is able to find remarkably more hits that fall into category 'correct' than any other competing method (Table 2). In NOSELF category PANNZER is able predict 5–24% more correct functional annotations than competing methods and in NOCLOUD category the improvement is between 6 and 16%. The PANNZER method also proved to be much faster than Blannotator. An average run time for a single query was 1.3 s with PANNZER and 39 s with Blannotator, making PANNZER about 30 times faster. It is notable that RAST annotation was done against FIGfam database where the self hits and propagation cloud was not removed. Therefore the RAST results are not directly comparable to other methods.

3.3. Description prediction of eukaryotes

Using BLAST-based methods the eukaryotic test set proved to be more difficult to annotate correctly than prokaryotic test set. Especially with the eukaryotes the PANNZER method outperforms BLAST-based methods clearly (Table 3). In case of eukaryotes against the NOSELF the PANNZER method predicts 37% more 'correct' annotations than BBH and 12% more than BIBH. In 'incorrect' annotations there are 44% less hits than with BBH and 14% less than with BIBH. When compared with NOCLOUD the differences grow even higher in favour of PANNZER.

3.4. GO prediction

A previous version of PANNZER method participated in CAFA 2011 challenge that provided an independent large-scale blind testing of GO prediction methods. PANNZER was ranked in top three over 54 competing methods (Radivojac *et al.*, 2013). The differences in the prediction accuracy was very small between best performing methods; Jones-UCL (Cozzetto *et al.*, 2013), Argot^2 (Falda *et al.*, 2012) and PANNZER. Since the Jones-UCL and the Argot^2 have no publicly available stand-alone version, we were not able to compare new PANNZER against these methods directly using our NOSELF database. Therefore we decided to only evaluate our latest version of PANNZER against the version that participated in the CAFA 2011 challenge.

We compare how results from new and old methods correlate with various GO semantic distances. The overall improvement of the PANNZER performance is between 28 and 47%, depending on which semantic similarity measure was used (Table 4).

Table 2. Description prediction results with Prokaryote dataset

	RAST[a]	BBH	BIBH	Blannotator	PANNZER
NOSELF					
Correct (%)	32	43.8	47	51	**56**
Incorrect (%)	45	39	35	32	**29**
t test*	1.12×10^{-94}	1.91×10^{-36}	6.71×10^{-22}	0.002	
NOCLOUD					
Correct (%)	32	31	33	42	**48**
Incorrect (%)	45	48	44	37	**35**
t test*	7.62×10^{-27}	2.62×10^{-52}	8.46×10^{-35}	0.005	

The highest correct and lowest incorrect predictions are shown in bold.
**P*-value from pairwise Student's *t* test (2-tailed) against PANNZER.
[a]Please note that the RAST was done against the FigFam database, not NOSELF or NOCLOUD.

Table 3. Description prediction results with Eukaryote dataset

	BBH	BIBH	PANNZER
NOSELF			
Correct (%)	15	40	**52**
Incorrect (%)	81	51	**37**
t test*	$<10^{-300}$[a]	3.52×10^{-115}	
NOCLOUD			
Correct (%)	8	27	**48**
Incorrect (%)	87	62	**40**
t test*	$<10^{-300}$[a]	8.45×10^{-226}	

The highest correct and lowest incorrect predictions are shown in bold.
**P*-value from paired Student's *t* test (2-tailed) against the PANNZER.
[a]Value is too small causing number underflow.

Table 4. Correlations to GO semantic similarities

	PANNZER (new)	PANNZER (old)	Improvement of correlation (%)
WeightedLin	0.38	0.26	47
Lin	0.38	0.26	47
WeightedPathLin	0.29	0.23	26
PathLin	0.29	0.23	26
Jaccard	0.28	0.22	28

The improvement of the PANNZER method compared with version that participated in CAFA 2011 challenge are shown.

In addition, we show GO analysis with three separate score functions: hypergeometric *P*-value, Jaccard and GSZ. This compares how different enrichment scoring functions rank GO classes for class prediction. Jaccard and GSZ were inputs to our PANNZER model training, whereas hypergeometric *P*-value has been used before in the GO prediction (Götz *et al.*, 2008). We also include two reference methods that were also used in CAFA competition: Best BLAST and Naive prediction. Best BLAST selects the maximum Bit score reported for GO class members. Naive prediction reports simply GO classes in their size order starting from largest class. These test whether our regression outperforms single score functions and reference methods from CAFA competition.

We also evaluated the GO-prediction using PR and ROC-curves. Here the GO evaluation dataset is divided into Biological Process, Cellular Component and Molecular Function GO categories. The latest version of PANNZER outperforms the old version in prediction accuracy especially in BP (Figs 4 and 5). Difference between PANNZER versions is clearer from Table 5. In addition the individual score functions show weaker performance, although the difference to GSZ is quite small. Reference methods show clearly weaker performance than other methods. Especially Best BLAST is even weaker than naive prediction. This again underlines the fact that straight analysis of *SSRL* without any summarization of hits is sub-optimal annotation method.

4. Discussion

As the amount of newly submitted sequences grow rapidly in public databases, and the functional annotation is critical step before studying these sequences, we need more reliable methods for *in silico* functional annotation. The PANNZER method outperforms competing methods in functional annotation prediction accuracy and brings novel statistical testings to the analysis.

In particular, the *k*-nearest neighbour clustering with statistical testings bring major advantages over traditionally used nearest neighbour method (e.g. Best BLAST Hit). Our results in description prediction show that the use of the nearest neighbour does not bring any advantage in functional annotation. It is remarkable how evenly 'correct' and 'incorrect' description hits are distributed over the BLAST result lists. In the standard BLAST (using default parameters) against the NOSELF database, the 'correct' hit count does not rise above 'incorrect' count in any index of the result list, including the best hit (Figs 2 and 3). It seems that the probability of having correct annotation from the best hit is no different to any other hit in a BLAST result list.

Free text description is the most comprehensive way to describe functionality of a protein and is required for every protein sequence that is submitted to a public sequence database. The current release of UniProtKB contains more than 1.5 million unique descriptions about protein functions and GO annotations that contain today 40 000 non-obsolete live terms. Despite a large fraction of synonymous descriptions, the difference is considerable. GO annotation suffers of biased usage of large and general GO terms which explains the unexpectedly good performance of the Naive GO prediction method (Figs 4 and 5). According to our results Naive method outperforms the Best BLAST Hit method clearly. This highlights the fact that closest neighbour-based methods should be avoided.

Surprisingly description prediction has obtained recently very little attention in the bioinformatics community. This could be because the free text annotation is seen as an ill-defined problem without effective evaluation metrics and difficulties in handling synonyms and homonyms. To alleviate these shortcomings to some extent, we propose DSM as a new standard in description evaluation.

Since descriptions and GOs are used in different contexts, both annotations are needed. DE annotations are used by the biologists and

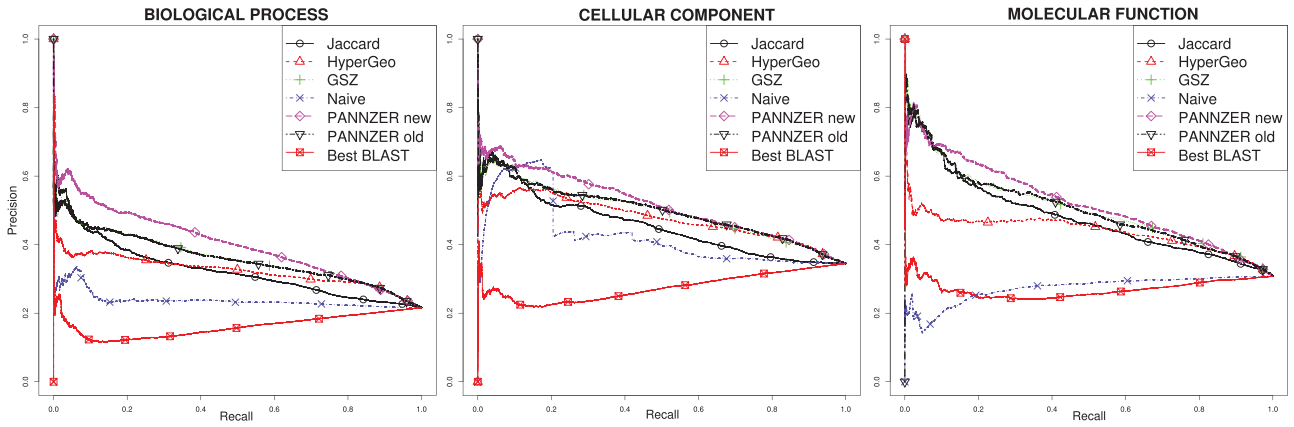

Fig. 4. PR curves. The performance of the two PANNZER regression models, three individual statistics (Jaccard, hypergeometric and GSZ) and two reference methods (Naive and best BLAST score) is shown. New PANNZER model outperforms other methods especially in Biological Process. Note also how new model is better at the beginning parts of curves. See text for more details

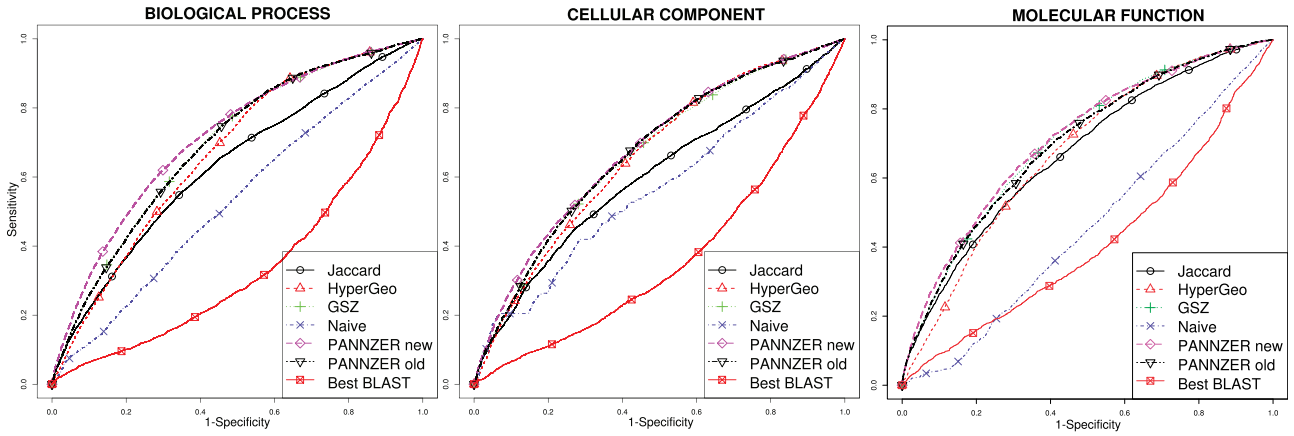

Fig. 5. ROC curves for the compared methods. Methods are the same as in Fig. 4. Results confirm the earlier results

Table 5. Results for different methods with AUC and *F*-measure

	Jaccard	HyperGeom	GSZ	PANNZER (new)	PANNZER (old)	Best BLAST	Naive
BP							
AUC	0.631	0.669	0.691	**0.714**	0.692	0.339	0.532
F-max	0.397	0.428	0.440	**0.460**	0.439	0.355	0.356
MF							
AUC	0.672	0.675	0.706	**0.713**	0.700	0.412	0.457
F-max	0.515	0.531	0.546	**0.549**	0.537	0.471	0.471
CC							
AUC	0.599	0.659	0.666	**0.678**	0.671	0.362	0.562
F-max	0.514	0.555	0.551	**0.557**	0.556	0.513	0.514

AUC scores were calculated from ROC-curves. New PANNZER (bold) is best method in each test.

other non-computationally related researchers, whereas GO terms are frequently used in computational functional analysis and have become the standard in, e.g. enrichment analysis. PANNZER is the only tool to our knowledge that does both types of prediction.

Acknowledgements

We would like to thank Petri Auvinen for critical reading of the manuscript and Teija Ojala for the artistic eye. We would also like to thank the anonymous reviewers for their constructive comments.

Funding

This work was supported by Biocenter Finland, University of Helsinki and Institute of Biotechnology.

Conflict of Interest: none declared.

References

Altschul,S.F. *et al.* (1997) Gapped blast and psi-blast: a new generation of protein database search programs. *Nucleic Acids Res.*, **25**, 3389–3402.

Andrade,M.A. and Valencia,A. (1998) Automatic extraction of keywords from scientific text: application to the knowledge domain of protein families. *Bioinformatics*, **14**, 600–607.

Brenner, S.E. (1999) Errors in genome annotation. *Trends Genet.: TIG*, **15**, 132–133.

Cozzetto,D. *et al*. (2013) Protein function prediction by massive integration of evolutionary analyses and multiple data sources. *BMC Bioinformatics*, **14**(Suppl. 3), S1.

Falda,M. *et al*. (2012) Argot2: a large scale function prediction tool relying on semantic similarity of weighted gene ontology terms. *BMC Bioinformatics*, **13**(Suppl. 4), S14.

Frishman,D. *et al*. (2001) Functional and structural genomics using pedant. *Bioinformatics*, **17**, 44–57.

Gilks,W.R. *et al*. (2002) Modeling the percolation of annotation errors in a database of protein sequences. *Bioinformatics*, **18**, 1641–1649.

Gilks,W.R. *et al*. (2005) Percolation of annotation errors through hierarchically structured protein sequence databases. *Math. Biosci.*, **193**, 223–234.

Götz,S. *et al*. (2008) High-throughput functional annotation and data mining with the blast2go suite. *Nucleic Acids Res.*, **36**, 3420–3435.

Hadley,C. (2003) Righting the wrongs. *EMBO Rep.*, **4**, 829–831.

Jones,C.E. *et al*. (2007) Estimating the annotation error rate of curated go database sequence annotations. *BMC Bioinformatics*, **8**, 170.

Kankainen,M. *et al*. (2012) Blannotator: enhanced homology-based function prediction of bacterial proteins. *BMC Bioinformatics*, **13**, 33.

Koski,L.B. *et al*. (2005) Autofact: an automatic functional annotation and classification tool. *BMC Bioinformatics*, **6**, 151.

Koskinen,J.P. and Holm,L. (2012) Sans: high-throughput retrieval of protein sequences allowing 50 mismatches. *Bioinformatics*, **28**, i438–i443.

Lin,D. (1998) An information-theoretic definition of similarity. In: *International Conference on Machine Learning (ICML)*, **Vol. 98**, pp. 296–304.

Magrane,M. and Consortium,U. (2011) UniProt knowledgebase: a hub of integrated protein data. *Database (Oxford)*, **2011**, bar009.

Martin,D.M. *et al*. (2004) Gotcha: a new method for prediction of protein function assessed by the annotation of seven genomes. *BMC Bioinformatics*, **5**, 178.

Meyer,F. *et al*. (2009) Figfams: yet another set of protein families. *Nucleic Acids Res.*, **37**, 6643–6654.

Miller,A. (2012) *Subset Selection in Regression*. CRC Press, Boca Raton, Florida, New York.

Naumoff,D.G. *et al*. (2004) Retrieving sequences of enzymes experimentally characterized but erroneously annotated: the case of the putrescine carbamoyltransferase. *BMC Genomics*, **5**, 52.

Overbeek,R. *et al*. (2013) The seed and the rapid annotation of microbial genomes using subsystems technology (rast). *Nucleic Acids Res.*, **42**, D206–D214.

Pesquita,C. *et al*. (2009) Semantic similarity in biomedical ontologies. *PLoS Comput. Biol.*, **5**, e1000443.

Punta,M. and Ofran,Y. (2008) The rough guide to in silico function prediction, or how to use sequence and structure information to predict protein function. *PLoS Comput. Biol.*, **4**, e1000160.

Radivojac,P. *et al*. (2013) A large-scale evaluation of computational protein function prediction. *Nat. Methods*, **10**, 221–227.

Robert,T. (1996) Regression shrinkage and selection via the lasso. *J. R. Stat. Soc.*, **58**, 267–288.

Sboner,A. *et al*. (2011) The real cost of sequencing: higher than you think! *Genome Biol.*, **12**, 125.

Scharf,M. *et al*. (1994) Genequiz: a workbench for sequence analysis. In: *Intelligent Systems for Molecular Biology (ISMB)*, **Vol. 2**, pp. 348–353.

Schlicker,A. *et al*. (2006) A new measure for functional similarity of gene products based on gene ontology. *BMC Bioinformatics*, **7**, 302.

Schnoes,A.M. *et al*. (2009) Annotation error in public databases: misannotation of molecular function in enzyme superfamilies. *PLoS Comput. Biol.*, **5**, e1000605.

Schnoes,A.M. *et al*. (2013) Biases in the experimental annotations of protein function and their effect on our understanding of protein function space. *PLoS Comput. Biol.*, **9**, e1003063.

Toronen,P. *et al*. (2009) Robust extraction of functional signals from gene set analysis using a generalized threshold free scoring function. *BMC Bioinformatics*, **10**, 307.

Vinayagam,A. *et al*. (2004) Applying support vector machines for gene ontology based gene function prediction. *BMC Bioinformatics*, **5**, 116.

Wieser,D. *et al*. (2004) Filtering erroneous protein annotation. *Bioinformatics*, **20**(Suppl. 1), i342–i347.