# Massive Automatic Functional Annotation MAFA

José Nelson Perez-Castillo[1], Cristian Alejandro Rojas-Quintero[2], Nelson Enrique Vera-Parra[3]

[1]GICOGE Research Group - Director Center for Scientific Research and Development, Distrital University Francisco José de Caldas, Bogotá D.C., Colombia
nelsonp@udistrital.edu.co
[2]GICOGE Research Group - Teacher / Researcher, Distrital University Francisco José de Caldas, Bogotá D.C., Colombia
neverap@udistrital.edu.co
[3]GICOGE Research Group - Student, Distrital University Francisco José de Caldas, Bogotá D.C., Colombia
carojasq@correo.udistrital.edu.co

**Abstract.** Functional annotation represents a means to investigate and classify genes and transcripts according to their function within a given organism. The annotation process for unknown sequences involves the use and integration of various tools that deal with the following tasks: local-alignment search for comparing unknown sequences with known-sequence databases (e.g. Swissprot, Uniprot, Refseq, among others), and proper association between sequences and the ontology that describes the functionality of such sequences (thus allowing categorization and statistical analysis of the corresponding associations).

This paper illustrates the use of Massive Automatic Functional Annotation (MAFA), which is an open-source bioinformatics tool that allows automation, unification and optimization of functional annotation processes when dealing with large volumes of sequences. MAFA includes tools for categorization and statistical analysis of associations between sequences and their corresponding ontology. After assessing the performance of MAFA with a set of data taken from Diploria-Strigosa transcriptome (using an 8-core computer, namely E7450 @ 2.40GHZ with 256GB RAM), processing rates of 4.8 seconds per sequence (Uniprot) and 80.2 seconds per sequence (Non-redundant) were found together with particular RAM usage patterns that depend on the database being processed (1GB for Uniprot database and 9GB for Non-redundant database). **Aviability:** https://github.com/alejo0317/MAFA

**Keywords:** Annotator, Functional annotation, High Throughput Sequencing, Gene ontology.

## 1 Introduction

Biological-sequence decoding plays an essential role in almost all research branches of Biology. For various decades, sequencing processes were conducted using the Sanger method (including the human genome project, where this method was crucial). However, the cost of the method and its limitations in terms of performance, scalability, speed and resolution have led to a migration trend towards using new procedures in the last 5 years, namely the so called “next generation sequencing” [1-2]. These new technologies allow having lower-cost, more-efficient sequencing, which leads to an exponential growth in the volumes of sequenced data.

Optimization of the sequencing process would be worthless without the development and optimization of suitable computing tools capable of analyzing such large sequenced-data volumes. In this context, one of the main needs of genomic-transcriptomic data mining is functional annotation. As a process, functional annotation consists of two stages, namely a search for known similar sequences (through alignment) and the association of such sequences to functional categories. The type of tools that are commonly used to carry out functional annotation processes are the following: BLAST - Basic Local Alignment Search Tool [3-5] (for finding sequences through alignment) and GO - Gene Ontology [6] (which provides controlled-term vocabulary to describe particular genes and the gene-product attributes within a particular organism).

When working with small sets of sequences, the functional-annotation process can be carried out as follows: the search for known similar sequences is performed over the BLAST server, offered by NCBI (National Center for Biotechnology Information); and similar-sequence associations to the GO terms is conducted on a sequence-to-sequence basis, using an online tool called AmiGO [7]. Each of the associations is categorized, one by one, by surfing (spanning) the trees available in the GO server. Inconveniences occur when dealing with large sets of sequences because the BLAST server (offered by NCBI) is limited to a small number of sequences [8] and also because the one-by-one task processing requires a long time to finish.

The present paper consists of four sections. Firstly, a general description of FAMA is presented, including its functionalities; then, an architectural description is provided, with a module-by-module explanation (namely modules LBS, GOAS, GOAN and DBA); finally, a performance assessment is conducted and analyzed.

## 2 General Description

MAFA is an open-source bio-informatics tool that has been optimized to carry out functional annotation processes over large numbers of nucleotide sequences (genomes and transcriptomes). Moreover, MAFA includes additional tools to perform categorization and statistical analysis of the corresponding sequence-ontology associations.

MAFA is intended to operate by using commands that are regarded as extremely simple (almost intuitive) for biologists.

## 3 Architecture

MAFA consists of 4 modules that constitute a work flow. In order to run and integrate the modules, it is necessary to use additional tools that apply to all modules. Figure 1 shows the 4 modules together with the work flow and the cross-module applicable tools.

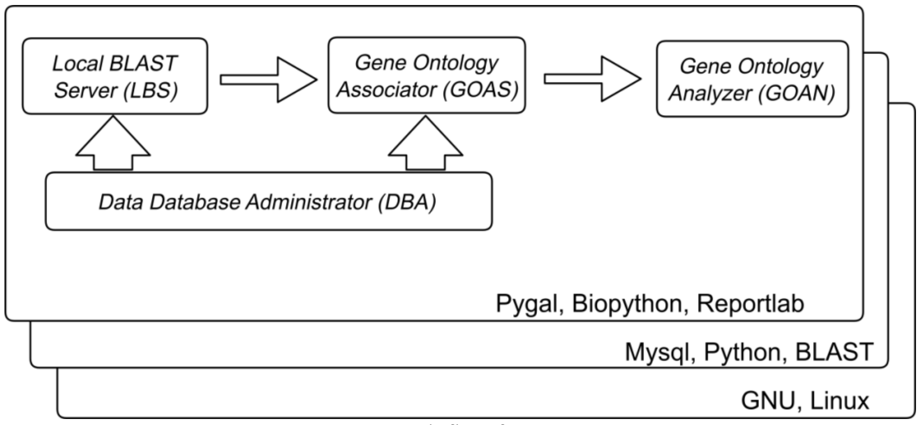
Local BLAST Server (LBS)
Gene Ontology Associator (GOAS)
Gene Ontology Analyzer (GOAN)
Data Database Administrator (DBA)
Pygal, Biopython, Reportlab
Mysql, Python, BLAST
GNU, Linux

**Fig. 1.** Work flow for MAFA

### 3.1 Cross-module Software Components

- MySQL: A relational and multi-thread, multi-user data-base management system, also freeware (open).
- GNU/Linux: An open-source operating system that is suitable for servers and also for running bio-informatics tools.
- Biopython [9]: A freeware project with various modules available intended to facilitate manipulation of bio-informatics data.
- Pygal: Freeware libraries that assist the production of graphical materials for the representation of information.
- BLAST (Basic Search Alignment Tool): A tool intended to find local regions of similarity through sequence alignment.
- Reportlab: A freeware Pyton-based library that facilitates the creation of PDF-format files.

### 3.2 *Local BLAST Server* Module

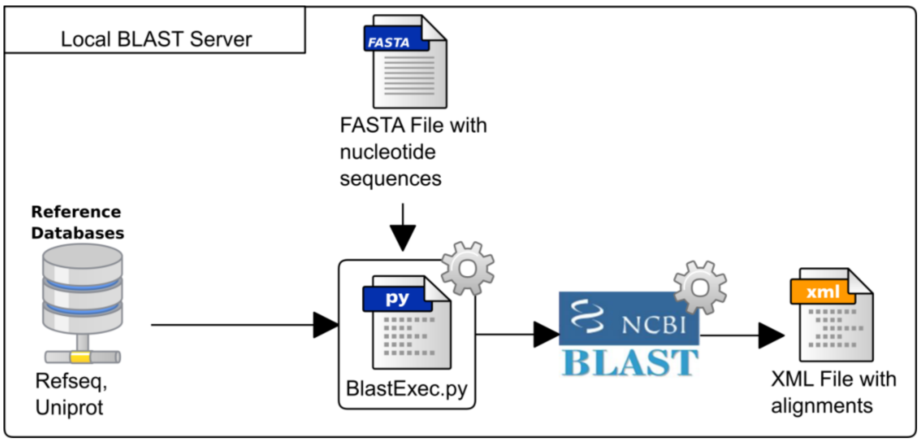

**Fig. 2.** LBS-module Diagram

This module is in charge of running BLAST (Nucleotides vs Amino-acids) and also of storing the corresponding output using the XML format. The scripts involved in this module are the following:

– *BlastExec.py*. Inputs: a FASTA file with sequences of nucleotides and a database for comparison purposes. Process: the process orders the system to run blastx using various cores. Output: XML file containing the alignments of the sequences.

### 3.3 GO Associator

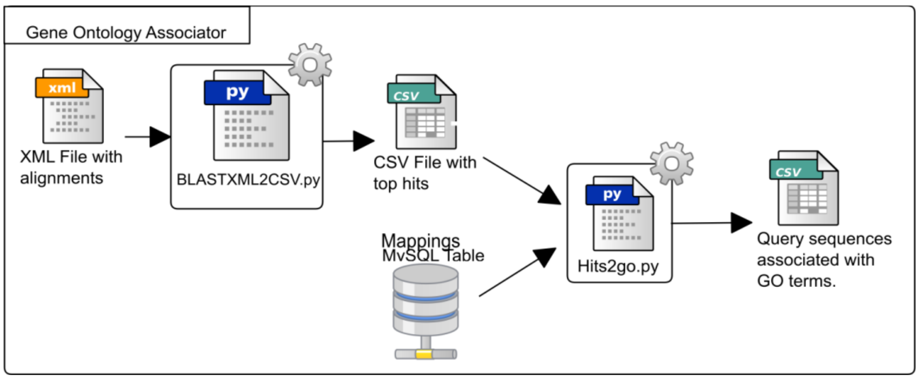

**Fig. 3.** GOAS-module Diagram

This module establishes the existing associations between the best hits, obtained from BLAST, and the terms from Gene Ontology. These associations are made by means of mapping tables between sequence identifiers and GO terms. This module involves the following scripts:

- *BLASTXML2CSV.py*: Input: an XML file containing the alignment of the corresponding sequences. Process: the process selects the best alignment per sequence (top hit) and also writes the new file in CSV format. Output: CSV file containing the best alignments (top hit).
- *Hits2go.py*: Inputs: a CSV file containing the best alignments (top hits) together with a table that holds all mapping between terms and identifiers. Process: the process makes an association between sequence identifiers and GO terms. Output: CSV file containing the associations between GO terms and sequence identifiers.

## 3.4 GO Analyzer

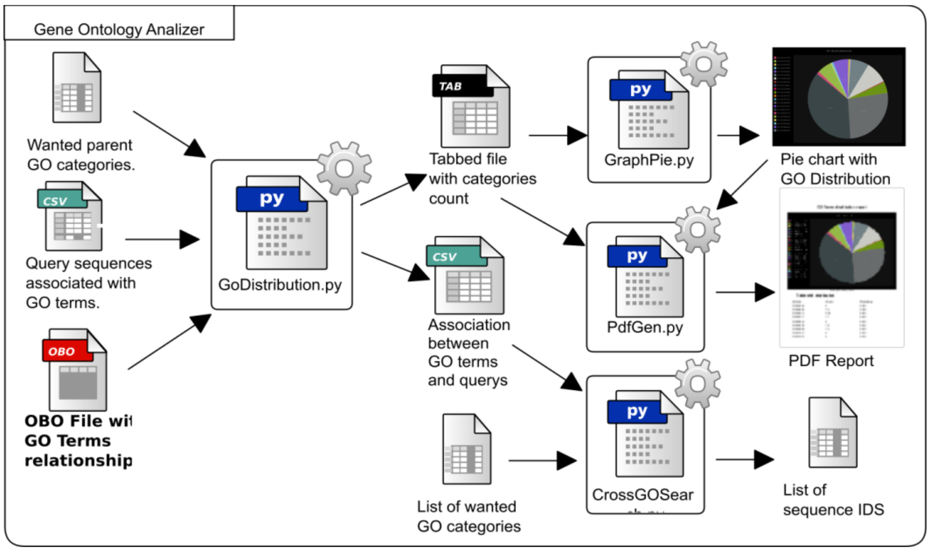
Gene Ontology Analizer
Wanted parent GO categories.
CSV
Query sequences associated with GO terms.
OBO
OBO File wit GO Terms relationship
py
GoDistribution.py
TAB
Tabbed file with categories count
CSV
Association between GO terms and querys
List of wanted GO categories
py
GraphPie.py
py
PdfGen.py
py
CrossGOSear
Pie chart with GO Distribution
PDF Report
List of sequence IDS

**Fig. 4.** GOAN-module Diagram

This module categorizes the GO terms according to user's interests. The module also counts how many times particular input sequences appear into the per-user categories and produces a complete report of the results. Additionally, the module is capable of finding sequences that belong to more than one GO category. The scripts involved in this module are as follows:

- *GoDistribution.py:* Inputs: a CSV file containing the associations between GO categories and sequence identifiers, a file that contains a list of more abstract terms intended to be associated to the more specific terms; finally, an OBO file that contains the relations between GO terms. Process: the process associates the desired GO categories (desired by users) to the more specific terms; it also counts how many times input sequences appear per desired GO category. Outputs: a CSV file containing associations between GO categories and sequence identifiers, a tabulated file that contains the various counts per each GO term that was to be found.

– *GraphPie.py:* Input: a tabulated file including the various counts of each GO category that was to be found. Process: the process produces a circular graph that illustrates the distribution of the categories. Output: a PNG-format image that represents the data.

– *CrossGOSearch.py:* Input: a CSV file containing the relation between GO categories and sequence identifiers and also a list of GO categories that are to be found in shared (common) sequences. Process: the process is a filter of all the sequences that appear in various GO categories at the same time. Output: a list of sequence identifiers.

– *PdfGen.py*: Inputs: a tabulated file containing the various counts per GO category that was to be found, a PNG-format image that represents data. Process: the process produces a PDF-format report that contains the analysis results. Output: PDF file containing a report about the analysis.

## 3.5 Database Administrator

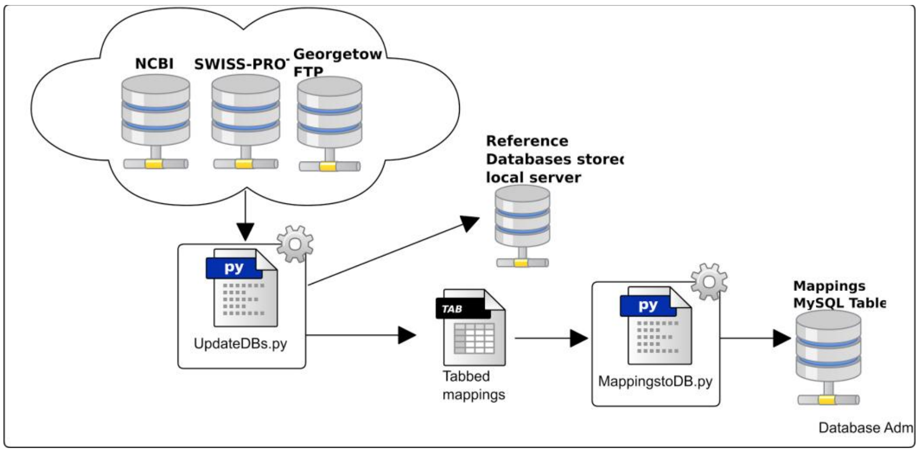

**Fig. 5.** DBA-module Diagram

This module carries out updating tasks over the databases of both sequences and mapping so that the databases are available in the local server. The elements involved in this module are as follows:

– *UpdateDBs.py:* Processes: a first process connects to servers NCBI and Swisprot in order to download the databases, another process generates the indices of downloaded files for BLAST, a third process connects to the FTP at Georgetown University in order to download a file that maps the various types of identifiers onto GO terms. Output: FASTA-format sequence files together with their corresponding BLAST indices, a file that maps GO terms onto sequence identifiers.

– *MappingstoDB.py:* Input: a file that maps GO terms onto sequence identifiers. Process: the process stores the corresponding mapping in a MySQL table so as to provide quick access. Output: a MySQL table filled with the corresponding mapping.

## 4 Methodology and Assessment

### 4.1 Data set

– Organism: Diploria Strigosa.

– Type of Sequences: Transcriptomics.

– Number of Sequences: 500, 1000, 2000, 4000.

– Format: FASTA.

– Database to perform the search: RefSeq Non-Redundant [10], Uniprot [11].

– Expected Value: 1e-3.

This transcriptome was selected because it is a representative and typical example of the data normally required to annotate by the researchers from Evolutionary Immunology and Immunogenetics Group from Genetics Institute of Colombia National University which are researching for immune response of coral organisms and disappearance of reefs (MAFA was developed in the framework of this research).

### 4.2 Metrics

– Performance: Processing time and RAM usage.

– Functionality: Aligned sequences, annotated sequences.

### 4.3 Configuration

– Processing Cores: 8 of 24 from a Xeon E7450 @ 2.40GHz

– Available RAM: 256GB

## 5 Results and Analysis

### 5.1 Performance Results

**Table 1.** MAFA Performance Analysis Results

| | | | | Module | | | | |
|---|---|---|---|---|---|---|---|---|
| Databases | # of sequences | | | LBS | GOAS | GOAN | TOTAL | |
| | Original | BLAST hits | Annotated with GO | Time (S) | Time (S) | Time (S) | RAM (MB) | Time (S) |
| Uniprot | 500 | 180 | 170 | 3460 | 0 | 2 | 1050 | 3462 |
| | 1000 | 384 | 367 | 4497 | 2 | 17 | 1050 | 4516 |
| | 2000 | 729 | 689 | 8678 | 23 | 67 | 1050 | 8768 |
| | 4000 | 1585 | 1513 | 19067 | 67 | 125 | 1050 | 19259 |
| Refseq | 500 | 191 | 176 | 35976 | 3 | 8 | 9134 | 35987 |
| | 1000 | 406 | 376 | 86780 | 34 | 23 | 9134 | 86837 |
| | 2000 | 759 | 706 | 190670 | 89 | 178 | 9134 | 190937 |
| | 4000 | 1672 | 1572 | 287808 | 140 | 201 | 9134 | 288149 |

Table 1 indicates that the module that requires longer processing times is Local Blast Server. Additionally, it can be observed that the relation between processing time and the number of sequences is almost linear, reaching database-dependent rates of 4.8 seconds per processed sequences (for Uniprot) and 80.3 seconds per processed sequence (for Non-redundant).

Regarding RAM usage, there is direct dependency on the database in use; on the other hand, there is no dependency on the number of sequences to be processed. For Uniprot, RAM usage is approximately 1GB; for Non-redundant, RAM usage is 9GB.

## 6 Conclusions

MAFA is a tool that allows functional annotation and further annotation classification provided there are some given term-specific categories of Gene Ontology. MAFA's main functions include the following: the generation of structured-data outputs that advertise the amount of sequences associated to each GO term, and the establishment of relations between the target term identifiers of Gene Ontology and the identifiers of the given sequences. Additionally, MAFA generates easy-to-interpret graphs for users as well as complete PDF reports containing the results from the corresponding analysis. It is also possible to conduct search processes in order to find sequences that are simultaneously associated to various categories or GO terms.

Regarding performance of the tool (MAFA), a linear behavior was observed when analyzing processing time and the number of sequences. In this respect, database-dependent rates (using the 8 cores of a Xeon E7450 processor and 256GB RAM) were found to be 4.8 seconds per sequence for Uniprot and 80.2 seconds per sequence for Non-redundant. Additionally, it was observed that RAM usage patterns are independent of the number of sequences to be processed and only depend on the reference database in use.

## Acknowledgements

Work done in collaboration with High Performance Computational Center (CECAD) - Distrital University Francisco José de Caldas, Bogotá D.C., Colombia (http://cecad.udistrital.edu.co) and Evolutionary Immunology and Immunogenetics Group (http://www.genetica.unal.edu.co/gie/) - Genetics Institute - National University (IGUN), Colombia, (http://www.genetica.unal.edu.co).

## References

1. Metzker, M.: Sequencing technologies – the next generation. Nature Reviews Genetics 11, 31–46 (2010)
2. Martin, J., Wang, Z.: Next-generation transcriptome assembly. Nature Reviews Genetics 12, 671–682 (2011)
3. Altschul, S.F., Gish, W., Miller, W., Myers, E.W., Lipman, D.J.: Basic local alignment search tool. J. Mol. Pub. Med. Biol. 215, 403–410 (1990)
4. Madden, T.L., Tatus
5. ov, R.L., Zhang, J.: Applications of network BLAST server. Meth. Enzymol. 266, 131–141 (1996)
6. Camacho, C., Coulouris, G., Avagyan, V., Ma, N., Papadopoulos, J., Bealer, K., Madden, T.L.: BLAST+: architecture and applications. BMC Bioinformatics 10, 421 (2008)
7. Ashburner M., Ball C.A., Blake J.A., Botstein D, Butler H, et al.: Gene ontology: tool for the unification of biology. The Gene Ontology Consortium. Nat Genet 25, 25–2 (2000)
8. Carbon, S., et al: AmiGO: online access to ontology and annotation data. Bioinformatics 25.2: 288-289. (2009)
9. Parra, N.E, Pérez, J. N, Rojas, C.A: "Presentation and Evaluation of ABMS (Automatic Blast for Massive Sequencing). Advances in Computational Biology. Springer International Publishing, 199-204.( 2014)
10. Cock, P. J., Antao, T., Chang, J. T., Chapman, B. A., Cox, C. J., Dalke, A., ... & de Hoon, M. J.: Biopython: freely available Python tools for computational molecular biology and bioinformatics. Bioinformatics, 25(11), 1422-1423. (2009)
11. Pruitt, K.D., Tatusova, T., Maglott, D.R.: NCBI reference sequences (RefSeq): a curated non-redundant sequence database of genomes, transcripts and proteins. Nucleic Acids Research 35(suppl. 1), D61–D65 (2007)
12. Bairoch, A., Apweiler, R., Wu, C. H., Barker, W. C., Boeckmann, B., Ferro, S., ... & Yeh, L. S. L.: The universal protein resource (UniProt). Nucleic acids research, 33(suppl 1), D154-D159. (2005)