Tutorial for EDAMmapper

05.07.2017

Getting the EDAMmapper

The latest version of EDAMmapper is available at:
https://github.com/edamontology/edammap/releases

From there, we need those two files:
https://github.com/edamontology/edammap/releases/download/v0.1.1/edammap-0.1.1.jar
https://github.com/edamontology/edammap/releases/download/v0.1.1/processor-0.1.1.jar

Using the EDAMmapper

Input

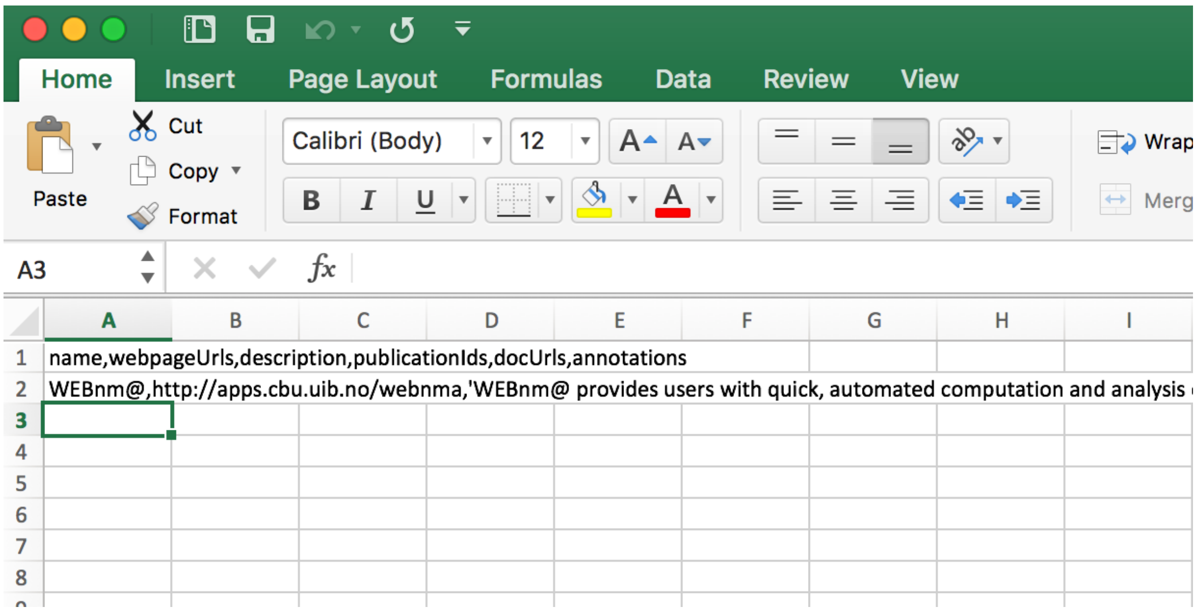
Basically, we need to give at least two files as an input. One of them is EDAM ontology file and the other is a file containing information about the tools what we want to annotate.

Ontology
The latest EDAM ontology (currently, EDAM_1.17.owl) is available at:
http://edamontology.org/page

csv file
The second input file is a csv file containing information about tools what we want to annotate. More specifically, it should contain tool's name, homepage, description, keywords (words describing the tool), publication ID-s (PMID, PMCID or doi), documentation url and annotations). Header is following:

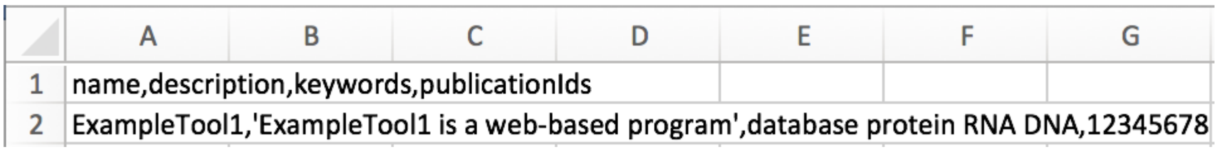
name,webpageUrls,description,publicationIds,docUrls,annotations

For example, it may look like this (example.csv):



```
name,webpageUrls,description,publicationIds,docUrls,annotations
WEBnm@,http://apps.cbu.uib.no/webnma,'WEBnm@ provides users with quick, automated computation and analysis
```

Please note that different keywords should be separated by space not comma and description needs to be between ‘’ if it contains comma. Adding annotations to csv file is important only if we want to generate benchmark file (e.g. to tune different parameters). This is further explained in benchmark section. In addition, it is not necessary to fill in all these fields. For example, following csv input is also functional:



```
name,description,keywords,publicationIds
ExampleTool1,'ExampleTool1 is a web-based program',database protein RNA DNA,12345678
```

(There is also an example of input file (example.csv) as well as its output files (example_results.html and example_benchmark.html)).

The simplest way to run this program is:

```
java -jar edammap-0.1.1.jar -e EDAM_1.17.owl -q example.csv -t generic
```

<u>Database</u>

If we need to analyse same inputs several times (e.g. to test different parameters) or we want to calculate IDF file (more about it in IDF file section), then we need the contents of publications and web pages each time. In order to save time, we are using MapDB library which allows to store information in an on-disk database. The unique key for each map entry will be an URL or a publication ID. As for the content, in case of webpages and docs, the value stored will be the string representing the plain text content of a website, and in case of publications, the stored value will be the serialized publication structure we have filled in during fetching. If a database file is specified to the automatic mapper, then contents of web pages and publications are read from there. If some are not present there and fetching is not disabled, they will be fetched from the web and put to the database. The database can also be filled beforehand, as described in here:

```
java -jar processor-0.1.1.jar --init-database new.db

java    -jar    processor-0.1.1.jar    --add-all-missing-from-file    new.db
```

```
biotools.xml biotools
```

(There is also a database file – db.db)

IDF (inverse document frequency weights) file
The purpose of this file is to raise the importance of more meaningful words. For example, a large number of bioinformatics articles contain a word “sequence”. However, this word is definitely not the most important word for describing all these articles. Instead, more meaningful words for describing a certain article/tool tend to occur less frequently. In this file, each word has a specific value (weight) which describes how common a term is in a collection of documents and in a sense, this value indicates how much information each word provides. We can construct it with following command:

```
java -jar processor-0.1.1.jar --make-query-idf biotools.xml biotools db.db
biotools.idf
```

(There is also already made IDF file - biotools.idf)

Biotools content
For constructing aforementioned file and database, we need to get the content from bio.tools Registry (https://bio.tools/). The latest version can be downloaded from: https://bio.tools/api/tool. However, this can also be done with attached python script (get_all_pages_from_biotools_xml.py) which we can run on a command line with following command:

```
python3 get_all_pages_from_biotools_xml.py
```

This program downloads all the content in biotools in xml format. However, content for every page is in a separate xml file. To combine all the xml files in a folder we can use another attached python file (combine_xml.py) with following command:

```
python combine_xml.py *.xml > biotools_combined.xml
```

This program outputs combined xml file from all xml files in a given folder. Yet, newly generated file is not suitable for EDAMmapper (a problem which arose due to changes in biotools xml format). Firstly, this needs to be converted to different format. A file transform.xslt will do that:

```
xsltproc transform.xslt biotools_combined.xml >
biotools_combined_formatted.xml
```

Running the EDAMmapper

Now, we can run the programm like this:

```
java -jar edammap-0.1.1.jar -e EDAM_1.17.owl -q example.csv -t generic -d
db.db --query-idf biotools.idf -b topic operation data format -m 5 -r
results.html -k benchmark.html
```

Some more common parameters and their meaning:

`"--fetching-disabled"` - do not download information from internet for those articles and webpages that are not in db.db
`"-b topic operation data format"` - use all ontology branches
`"-m 5"` - give 5 best suggestions for each branch
`"--obsolete"` - give also obsolete terms
`"-r results.html -k benchmark.html"` - output files

In the case of bio.tools we can add following:

```
--disable-query-idf-branches data format
--good-score-data 0.77
--good-score-format 0.70
--bad-score-data 0.65
--bad-score-format 0.63
```

In order to avoid entering command line parameters we can use configuration file:
https://raw.githubusercontent.com/edamontology/edammap/master/edammapper/options.conf

We just need to remove (#) in front of parameters that we want to change from default and also remove "--" from the starting row as well as from the following row. After that we can run program like this:

```
java -jar edammap-0.1.1.jar @options.conf
```

Output

Results file

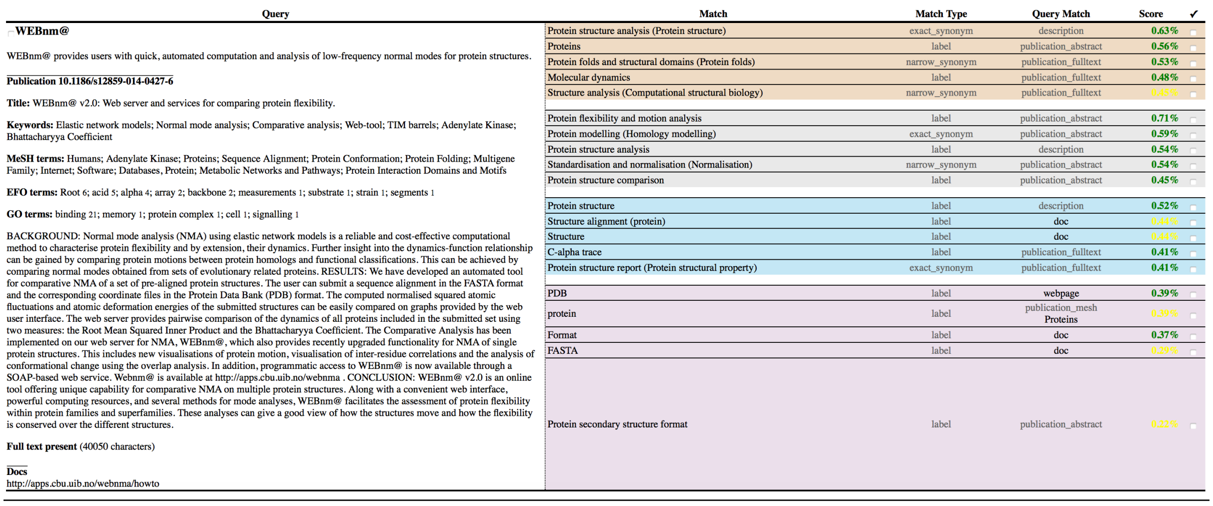
Results file can be outputted in two different formats: plain text and HTML (figure 1). The plain text is more suitable as an input for other tools and HTML is better for displaying the results in more intuitive way. The example HTML is shown in figure 1.

On the left hand side, we can see the query content. First, the tool name (WEBnm@) followed by a short description. It has one publication, (doi: 10.1186/s12859-014-0427-6), attached to it. We can see the publication title, MeSH, EFO, GO terms and abstract. The full text is not presented, only its character count is printed. The tool has also one documentation URL attached, which is also brought out. The homepage can be accessed by clicking on the name (WEBnm@). Also, clicking on the Publication header goes to the corresponding journal article and clicking on terms goes to the individual term web pages. For EFO and GO terms, their count (how many times they were found in the full-text) is also brought out.

On the right hand side, we can see found matches, grouped by branch and ordered by score. From top to bottom, branches are topic, operation, data and format. For each branch, we can see the top 3 matches. First, we have the matched concept label, followed by the best matched concept part, followed by the best matched query part. In case the best matched
concept part is not a label (i.e. the preferred name), the content of the concept part is brought out in parentheses after the label. Such as the narrow synonym "Protein structure" for label "Protein structure analysis". Clicking on the matched concept label takes to the corresponding EDAM URI. If the matched label text is stricken through, this means the concept is obsolete

(we didn't use obsolete concepts in matching). Scores are in the last column and colours are set by the score limits given as parameters and mean the following: green for good match, yellow for mediocre match and red for bad match.

If we want to upload a tool to bio.tools, then we should neglect those annotations which are marked as obsolete. This option is only there to help manual annotation. To omit obsolete terms we just need to avoid giving following parameter `--obsolete`.



Query

☐ **WEBnm@**

WEBnm@ provides users with quick, automated computation and analysis of low-frequency normal modes for protein structures.

Publication 10.1186/s12859-014-0427-6

Title: WEBnm@ v2.0: Web server and services for comparing protein flexibility.

Keywords: Elastic network models; Normal mode analysis; Comparative analysis; Web-tool; TIM barrels; Adenylate Kinase; Bhattacharyya Coefficient

MeSH terms: Humans; Adenylate Kinase; Proteins; Sequence Alignment; Protein Conformation; Protein Folding; Multigene Family; Internet; Software; Databases, Protein; Metabolic Networks and Pathways; Protein Interaction Domains and Motifs

EFO terms: Root 6; acid 5; alpha 4; array 2; backbone 2; measurements 1; substrate 1; strain 1; segments 1

GO terms: binding 21; memory 1; protein complex 1; cell 1; signalling 1

BACKGROUND: Normal mode analysis (NMA) using elastic network models is a reliable and cost-effective computational method to characterise protein flexibility and by extension, their dynamics. Further insight into the dynamics-function relationship can be gained by comparing protein motions between protein homologs and functional classifications. This can be achieved by comparing normal modes obtained from sets of evolutionary related proteins. RESULTS: We have developed an automated tool for comparative NMA of a set of pre-aligned protein structures. The user can submit a sequence alignment in the FASTA format and the corresponding coordinate files in the Protein Data Bank (PDB) format. The computed normalised squared atomic fluctuations and atomic deformation energies of the submitted structures can be easily compared on graphs provided by the web user interface. The web server provides pairwise comparison of the dynamics of all proteins included in the submitted set using two measures: the Root Mean Squared Inner Product and the Bhattacharyya Coefficient. The Comparative Analysis has been implemented on our web server for NMA, WEBnm@, which also provides recently upgraded functionality for NMA of single protein structures. This includes new visualisations of protein motion, visualisation of inter-residue correlations and the analysis of conformational change using the overlap analysis. In addition, programmatic access to WEBnm@ is now available through a SOAP-based web service. Webnm@ is available at http://apps.cbu.uib.no/webnma . CONCLUSION: WEBnm@ v2.0 is an online tool offering unique capability for comparative NMA on multiple protein structures. Along with a convenient web interface, powerful computing resources, and several methods for mode analyses, WEBnm@ facilitates the assessment of protein flexibility within protein families and superfamilies. These analyses can give a good view of how the structures move and how the flexibility is conserved over the different structures.

Full text present (40050 characters)

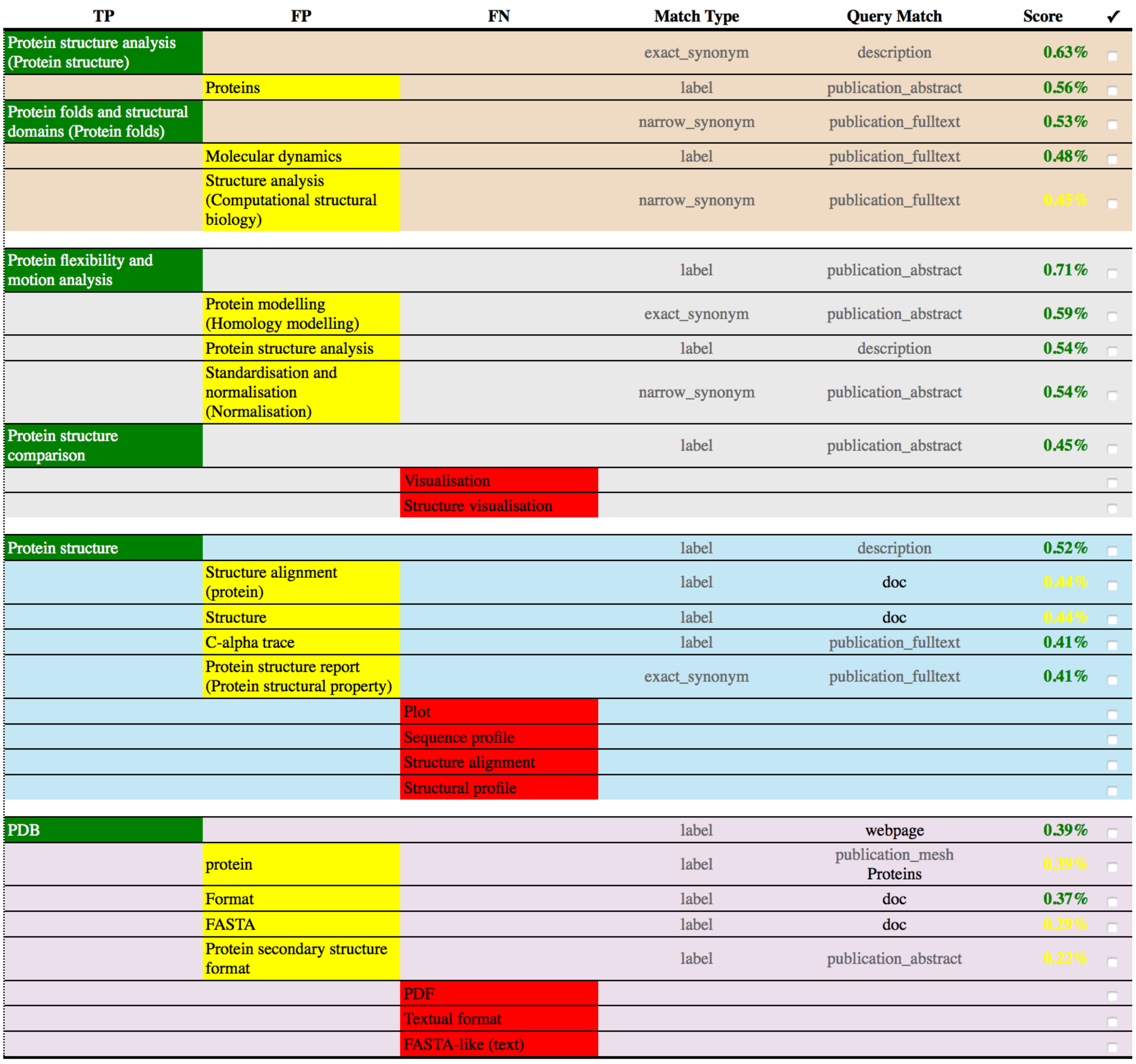
Docs
http://apps.cbu.uib.no/webnma/howto

Match	Match Type	Query Match	Score	✓
Protein structure analysis (Protein structure)	exact_synonym	description	0.63%	☐
Proteins	label	publication_abstract	0.56%	☐
Protein folds and structural domains (Protein folds)	narrow_synonym	publication_fulltext	0.53%	☐
Molecular dynamics	label	publication_fulltext	0.48%	☐
Structure analysis (Computational structural biology)	narrow_synonym	publication_fulltext	0.45%	☐
Protein flexibility and motion analysis	label	publication_abstract	0.71%	☐
Protein modelling (Homology modelling)	exact_synonym	publication_abstract	0.59%	☐
Protein structure analysis	label	description	0.54%	☐
Standardisation and normalisation (Normalisation)	narrow_synonym	publication_abstract	0.54%	☐
Protein structure comparison	label	publication_abstract	0.45%	☐
Protein structure	label	description	0.52%	☐
Structure alignment (protein)	label	doc	0.44%	☐
Structure	label	doc	0.44%	☐
C-alpha trace	label	publication_fulltext	0.41%	☐
Protein structure report (Protein structural property)	exact_synonym	publication_fulltext	0.41%	☐
PDB	label	webpage	0.39%	☐
protein	label	publication_mesh Proteins	0.39%	☐
Format	label	doc	0.37%	☐
FASTA	label	doc	0.29%	☐
Protein secondary structure format	label	publication_abstract	0.22%	☐

Figure 1. Results in HTML (example_results.html)

Benchmark file

As mentioned above, we may want to evaluate how well program works with certain parameters and compare obtained annotations to those which were added manually. For this purpose, EDAMmapper can generate benchmark file (figure 2). In this file, we can see true positives, false positives and false negatives. True positives are represented in green and these are terms which were added by both - the automatic mapper and manual annotation. Yellow marks false positives which are terms that were found by the automatic mapper but were not used to annotate the tool. False negatives, represented in red, are terms that were used in manual annotation but what EDAMmapper failed to find.



TP	FP	FN	Match Type	Query Match	Score	✓
Protein structure analysis (Protein structure)			exact_synonym	description	0.63%	☐
	Proteins		label	publication_abstract	0.56%	☐
Protein folds and structural domains (Protein folds)			narrow_synonym	publication_fulltext	0.53%	☐
	Molecular dynamics		label	publication_fulltext	0.48%	☐
	Structure analysis (Computational structural biology)		narrow_synonym	publication_fulltext	0.45%	☐
Protein flexibility and motion analysis			label	publication_abstract	0.71%	☐
	Protein modelling (Homology modelling)		exact_synonym	publication_abstract	0.59%	☐
	Protein structure analysis		label	description	0.54%	☐
	Standardisation and normalisation (Normalisation)		narrow_synonym	publication_abstract	0.54%	☐
Protein structure comparison			label	publication_abstract	0.45%	☐
		Visualisation				☐
		Structure visualisation				☐
Protein structure			label	description	0.52%	☐
	Structure alignment (protein)		label	doc	0.44%	☐
	Structure		label	doc	0.44%	☐
	C-alpha trace		label	publication_fulltext	0.41%	☐
	Protein structure report (Protein structural property)		exact_synonym	publication_fulltext	0.41%	☐
		Plot				☐
		Sequence profile				☐
		Structure alignment				☐
		Structural profile				☐
PDB			label	webpage	0.39%	☐
	protein		label	publication_mesh Proteins	0.39%	☐
	Format		label	doc	0.37%	☐
	FASTA		label	doc	0.29%	☐
	Protein secondary structure format		label	publication_abstract	0.22%	☐
		PDF				☐
		Textual format				☐
		FASTA-like (text)				☐

Figure 2. Benchmark (example_benchmark.html)

Extracting relevant information from the results

When we have generated results html file, we may also like to extract relevant information from it and and upload tools to bio.tools. To achieve this, we can use python program parsing_results_html.py. The idea of this script is to find description and annotations from html document. However, a lot more information is needed for uploading, therefore this program should be combined with others to add all the necessary information. Unfortunately, this program is also very slow, because it fetches EDAM ontology terms from webpages. Command for using this program is following:

```
python3 parsing_results_html.py –t tool_type –l file_location –o output_file.json
```

This program outputs a json file ready to be added to bio.tools. Please note that tool type, file location and output file name are given as arguments (flag –t for tool type and –l for location and –o for output file). -t argument can be omitted and then default value for tool type ("Web application") is used. To save time in uploading the tools (not to upload one at a time) we can use a program bulkpost with following command:

```
python bulkpost.py list.json
```

This programs uploads the list of tool in json format to bio.tools and also outputs two files, one containing those tools which failed to upload and the other that contains tool names with reasons why uploading failed (for example, they might have been added already).

For further reading there is also a theses about EDAMmapper ("Automatic mapping of free texts to bioinformatics ontology terms.pdf").