# ENGR 102 Programming Practice

**Mini Project 2** *(Last update: March 25, 16:50)*

*March 21, 2019*

(*Due on April 4, 2019*)

Proteins are annotated with functionalities based on different annotations techniques (sequence similarity to another protein, experimental validation, etc.). These functionalities are represented as terms (e.g., GO:00000219) from Gene Ontology. In this mini project, you are going to implement a tool that will predict new functionalities for proteins based on the currently assigned functionalities for that protein. To this end, you will take advantage of the collaborative filtering skills that you learned in the class. Please read on for more details.

## How should it look like?

The graphical user interface of your tool should look like as the following one.

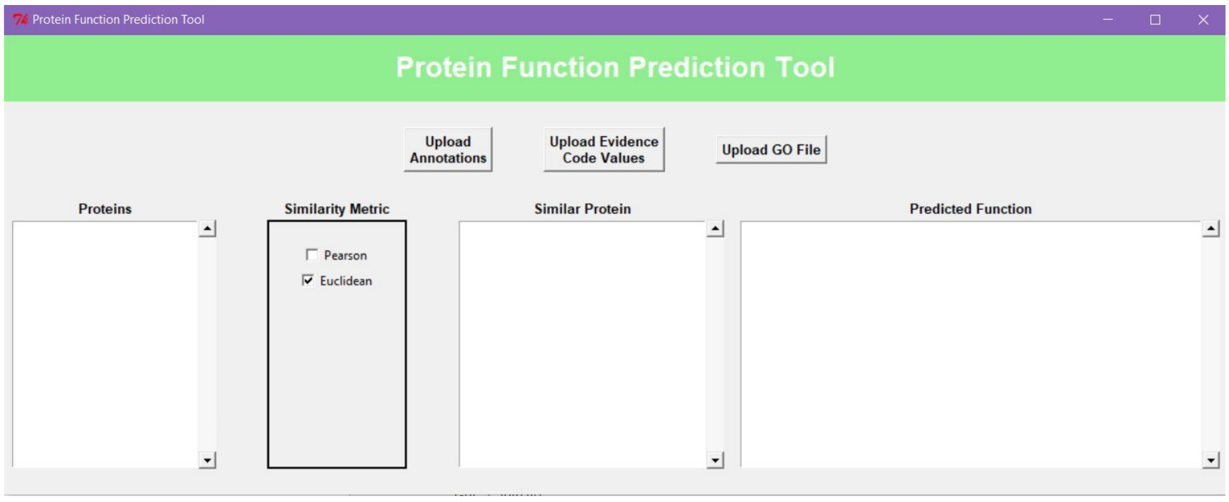


Figure 1

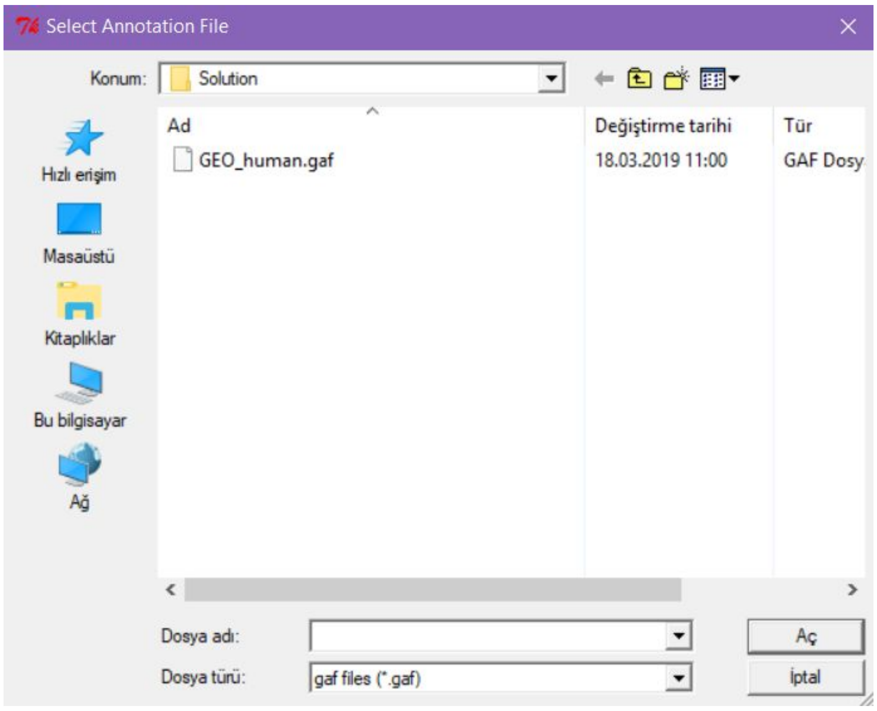
## How should it work?

Please see the animated gif posted on LMS to see a demo of how your application should work. Detailed instructions are provided below.

- In the beginning, the user will first upload the Annotation Data, Evidence Code Values Data, and Gene Ontology (GO) data using the three upload buttons at the top, respectively. The formats of the input files are as follows:
  - Annotation Data: In the annotation file (GEO_human.gaf), each line specifies an annotation of a protein with a functionality (represented by a GO term). Each protein may have multiple annotations each of which is stored on a separate line. You need to extract protein id, protein name, functionality annotation (i.e., a GO term), and evidence code which are located in columns 2, 3, 4, and 6, respectively (columns are tab separated). Protein ids are unique, but names may be shared by different proteins.
  - Evidence Code Values Data: Evidence code for an annotation specifies how the annotation was derived. Annotation evidences have different reliability scores. In the evidence code values file (ecv.txt), for each evidence code that you extracted in the

above step, there is a corresponding numerical value in a tab-separated manner. You will be using these values as a kind of rating for the annotation.

- GO Data: In the GO data file (go.obo), names corresponding to each GO id stored. First, the id of a GO term is provided, then on the next line, its name is included. You will need GO term names to display the predicted functionalities.

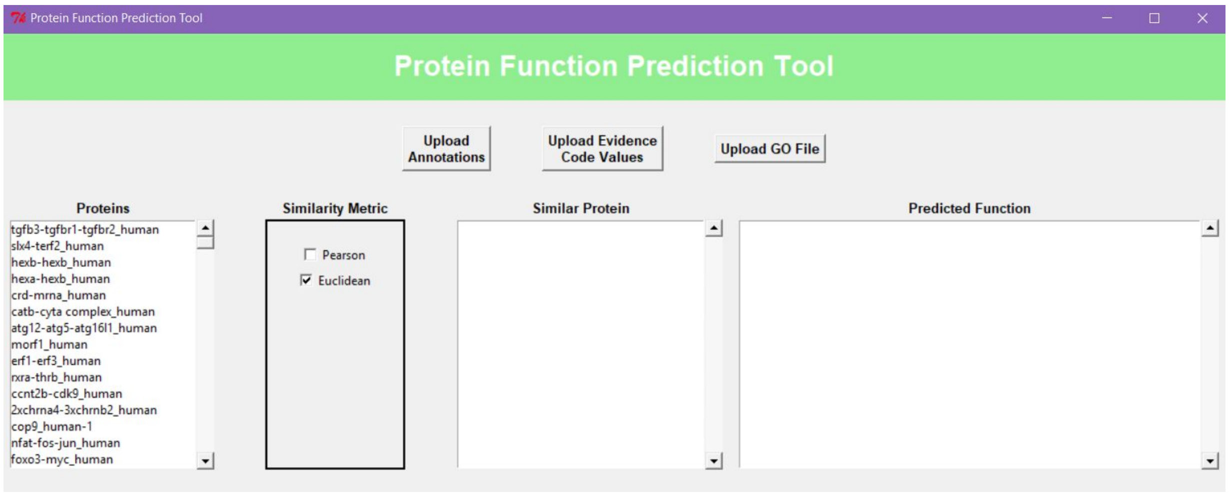


Konum: Solution

| Ad | Değiştirme tarihi | Tür |
|---|---|---|
| GEO_human.gaf | 18.03.2019 11:00 | GAF Dosy |

Dosya türü: gaf files (*.gaf)

Figure 2



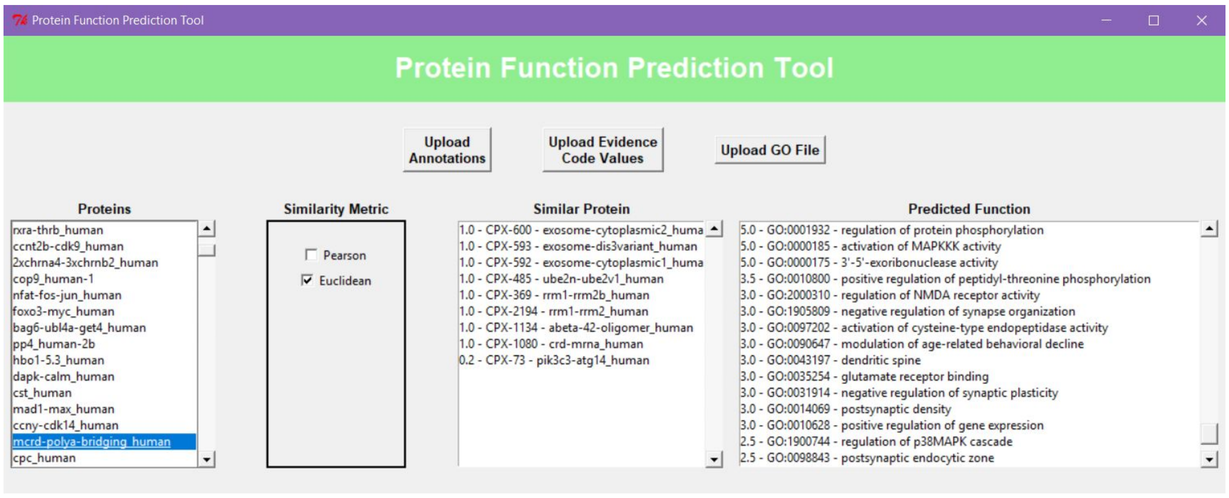
Similarity Metric: Pearson [✓ Euclidean]

Proteins:
tgfb3-tgfbr1-tgfbr2_human
slx4-terf2_human
hexb-hexb_human
hexa-hexb_human
crd-mrna_human
catb-cyta complex_human
atg12-atg5-atg16l1_human
morf1_human
erf1-erf3_human
rxra-thrb_human
ccnt2b-cdk9_human
2xchrna4-3xchrnb2_human
cop9_human-1
nfat-fos-jun_human
foxo3-myc_human

Figure 3

- Once the annotation data is uploaded, on the left-hand side, protein names for each protein from the annotation data file should be displayed. (please see Figure 3). A **Listbox** widget should be used to display protein names.

- When the user uploads the remaining files; you must parse the files, and store the extracted data under proper classes and data structures (please see the implementation notes for some hints).
- Once all the files have been uploaded, the user should be able to select a protein from the listbox containing protein names and select a similarity metric via checkbuttons on the right-hand side (Euclidean must be selected by default). Here, instead of checkbuttons, if you like, you may also use the radio button widget, which is more proper for these kinds of selections.
- After selecting a protein and a similarity metric, proteins that are similar to the currently selected protein should be automatically listed in the **listbox** in the middle, and the predicted functions for the selected protein also should automatically be displayed in the **listbox** at the rightmost part (see Figure 4). Please use the following format to display the computed data in the corresponding listboxes:
  - Similar Proteins: score - protein id - protein name (sorted by the score from larger to smaller). Please do not show proteins with similarity 0 or less. Also, please only show one digit after the dot in similarity values (consider using the built-in round function).
  - Predicted Function : score - annotation term - function name (sorted by score).
- For finding similar proteins and predicted functions, you should use the appropriate methods in recommendations.py (available on LMS).
- All listboxes must contain **scrollbars** that allow scrolling along the y-axis.


Protein Function Prediction Tool
Protein Function Prediction Tool
Upload Annotations
Upload Evidence Code Values
Upload GO File
Proteins
Similarity Metric
Similar Protein
Predicted Function

Proteins:

rxra-thrb_human
ccnt2b-cdk9_human
2xchrna4-3xchrnb2_human
cop9_human-1
nfat-fos-jun_human
foxo3-myc_human
bag6-ubl4a-get4_human
pp4_human-2b
hbo1-5.3_human
dapk-calm_human
cst_human
mad1-max_human
ccny-cdk14_human
mcrd-polya-bridging_human
cpc_human

Similarity Metric:

☐ Pearson
☑ Euclidean

Similar Protein:

```
1.0 - CPX-600 - exosome-cytoplasmic2_huma
1.0 - CPX-593 - exosome-dis3variant_human
1.0 - CPX-592 - exosome-cytoplasmic1_huma
1.0 - CPX-485 - ube2n-ube2v1_human
1.0 - CPX-369 - rrm1-rrm2b_human
1.0 - CPX-2194 - rrm1-rrm2_human
1.0 - CPX-1134 - abeta-42-oligomer_human
1.0 - CPX-1080 - crd-mrna_human
0.2 - CPX-73 - pik3c3-atg14_human
```

Predicted Function:

```
5.0 - GO:0001932 - regulation of protein phosphorylation
5.0 - GO:0000185 - activation of MAPKKK activity
5.0 - GO:0000175 - 3'-5'-exoribonuclease activity
3.5 - GO:0010800 - positive regulation of peptidyl-threonine phosphorylation
3.0 - GO:2000310 - regulation of NMDA receptor activity
3.0 - GO:1905809 - negative regulation of synapse organization
3.0 - GO:0097202 - activation of cysteine-type endopeptidase activity
3.0 - GO:0090647 - modulation of age-related behavioral decline
3.0 - GO:0043197 - dendritic spine
3.0 - GO:0035254 - glutamate receptor binding
3.0 - GO:0031914 - negative regulation of synaptic plasticity
3.0 - GO:0014069 - postsynaptic density
3.0 - GO:0010628 - positive regulation of gene expression
2.5 - GO:1900744 - regulation of p38MAPK cascade
2.5 - GO:0098843 - postsynaptic endocytic zone
```

Figure 4

**Implementation Notes:**

- Please make sure that you create and use the following classes at least. You may add more if you like:

i. A class that represents a functionality (i.e., a GO term). It will have each GO term's id and name as its attributes.

ii. A class that represents an evidence code with attributes evidence code acronym and its numeric value.

iii. A class that represents an annotation of a protein by a functionality. It will have at least two attributes: functionality [type: a GO object] and evidence code [type: an Evidence Code object].

iv. A class that represents a protein. It will have each protein's id, name, and annotations [type: a dictionary of annotation objects - key: GO id from the annotation object's GO term, value: Annotation object] as its attributes.

v. A class that represents your application's data center which keeps at least one attribute: proteins [type: a dictionary of all protein objects - key: Protein id, value: Protein object] (optionally, you may have another attribute: evidence codes [type: a dictionary of all evidence code objects - key: evidence code acronym, value: Evidence Code object]). It should have at least three methods:

   - one method that reads and parses the data included in the GO file (go.obo).
   - one method that reads and parses the data included in the annotation file (GEO_human.gaf).
   - one method that reads and parses the data included in the ecv.txt file.

vi. A class that represents the GUI. This class should have an object of your data center class as one of its attributes. For any data needs and file parsing, GUI will use this object as the gateway.

- In .gaf file, occasionally, there may be annotations with the same GO term but different evidence code acronyms for the some proteins. In such cases, please consider the first record in the file, and ignore the others.
- To establish an analogy to what we have learned in the class, you may consider proteins as critics, functionalities as movies, and evidence codes as ratings. You will import recommendations.py that we built in the class, and call proper functions from it at different parts. Please do not copy and paste code from recommendations.py into your project file. Moreover, please do not submit recommendations.py with your project file.
- You may treat the input files as regular text files, although their extensions may be different.
- For more information about Scrollbar and its use with a Listbox widget in an example, please see the following reference:

  https://www.tutorialspoint.com/python/tk_scrollbar.htm
- The following link contains useful example of using tkFileDialog. https://pythonspot.com/tk-file-dialogs/

**Warnings:**

- You **CANNOT** use place for geometry, only grid and pack are allowed.
- Do not talk to your classmates on project topics when you are implementing your projects. Do not show or email your code to others. If you need help, talk to your TAs or myself, not to your classmates. If somebody asks you for help, explain them the lecture slides, but do not explain any project related topic or solution. Any similarity in your source codes will have serious consequences for both parties.
- Carefully read the project document, and pay special attention to sentences that involve “should”, “should not”, “do not”, and other underlined/bold font statements.
- If you use code from a resource (web site, book, etc.), make sure that you reference those resource at the top of your source code file in the form of comments. You should give details of which part of your code is from what resource. Failing to do so may result in plagiarism investigation. Last but not least, you need to understand code pieces that you may get some other resources. This is one of the goals of the mini projects.
- Even if you work as a group of two students, each member of the team should know every line of the code well. Hence, it is important to understand all the details in your submitted code.

**How and when do I submit my project?**

- Projects may be done individually or as a small group of two students (doing it individually is **strongly** recommended for the best learning experience). If you are doing it as a group, only **one** of the members should submit the project. File name will tell us group members (Please see the next item for file naming details).

- Submit your own code in a single Python file. Name it with your and your partner’s first and last names. As an example, if your team members are Deniz Barış and Ahmet Çalışkan, then name your code file as deniz_baris_ahmet_caliskan.py (Do not use any Turkish characters in the file name). If you are doing the project alone, then name it with your name and last name similar to the above naming scheme.
    - Those who do not follow the above naming conventions will **get** 10% **off** of their project grade.
- Submit it online on LMS by 17:00 on April 4, 2019.

  **Late Submission Policy:**

    - -10%: Submissions between 17:01 – 18:00 on the due date
    - -20%: Submissions between 18:01 – midnight (00:00) on the due date
    - -30%: Submissions after which are up-to 24 hours late.
    - -50%: Submissions which are up-to 48 hours late.
    - Submission more than 48 hours late will not be accepted.

**Grading Criteria?**

| GUI Design (25) | Loading proteins data and showing them properly in the first listbox (25) | Computing and showing similar proteins for each selected protein properly (25) | Computing and showing predicted functionalities for each selected protein properly (25) |
|---|---|---|---|

From your overall grade, we will deduct points by the specified percentage for the following items:

- Inappropriate/cryptic variable names (-10%)
- Classes and objects are not used properly (-30%)
- Insufficient commenting (-10%).
- Inappropriate file naming (-10%)

**Have further questions?:**

- If you need help with anything, please use the office hours of your TAs and the instructor to get help. **Do not walk in randomly (especially on the last day) into your TAs' or the instructor's offices. Make an appointment first. This is important. Your TAs have other responsibilities. Please respect their personal schedules.**