**PROJECT SPECIFICATION**

Zhang Weiwei and Di Silvestro Irene

22nd April 2022

**AIM:** given two datasets named disease\_evidences.tsv and gene\_evidences.tsv derived from DisGeNET COVID-19 data collection as input, write a program able to read the two files, compute operations on them exploiting classes, Object-Oriented programming and Python packages (Pandas and Flask) and present the user the relevant outcomes on a web application.

**WORKFLOW**

* After reading the given file and having understood the instructions to build our web application, first we focus on how to define our normal classes, one per each operation and then, when possible, we try to link some classes together by implementing a couple of abstract classes serving as template from which we can generate subclasses to perform similar operations in a faster and more efficient way. Then, we move to the design of the CRC cards basing on the classes we are going to implement in part 2.

Below, there are the examples of an abstract and a normal class:

Immagine che contiene tavolo

Descrizione generata automaticamente

Immagine che contiene testo

Descrizione generata automaticamente

* Then, we build the complete UML diagram by connecting the abstract classes with their realizations (through blue dashed lines) and normal ones with their associations (through dark straight lines). All the classes are characterized by their respective attributes and methods.

Foto

The next step consists of the implementation of three parts: [part 1] with the two dataset readers using pandas and the registry listing the different types of operations specified in [part 2] with their relative classes. While [part 3] contains the codes to build the web-based user interface and its Flask application. Such a UI provides a list of choices, where each choice enables an analytical objective.

REGISTRY CLASS OF [PART 1]

* It defines the readers of the two datasets, the list of operations needed to be performed and the list of links of each web page, one per operation.
* If provides a function for each operation calling the result from the imported file of [part 2]. This is necessary for the implementation of [part 3].

Grouping all functions of [part 1] in a unique Registry class that taking as parameters the two DataFrames facilitates the performance of the operations and the calling of functions in [part 2].

CLASSES OF [PART 2]

**Class Metadata:**

- Input: the two datasets  
 - Output: list of 4 integers  
 - Functions: return the number of rows/columns of the two DataFrames  
 - Technique(s): built-in function .shape() of Pandas

**Class Semantics:**  
 - Input: the two datasets  
 - Output: a list of strings  
 - Functions: return the labels of the columns of the two DataFrames  
 - Technique(s): built-in function .columns() of Pandas

**Class Record(ABC) 🡪 abstract**

**Class** **Genes(Record):**  
 - Input: gene\_evidences.tsv  
 - Output: an integer and a list of the geneid  
 - Functions: return the number of genes and all genes in an ascending order  
 - Technique(s): built-in function .groupby() of Pandas

**Class** **Disease(Record):**  
 - Input: disease\_evidences.tsv  
 - Output: an integer and a list of the diseaseid  
 - Functions: return the number of diseases + all diseases in an ascending order  
 - Technique(s): *same as Genes*

**Class Sentences(ABC) 🡪 abstract**

**Class** **GeneCovid(Sentences):**  
 - Input: gene\_evidences.tsv  
 - Output: list of strings  
 - Functions: return all sentences associating COVID with genes   
 - Technique(s): usage of [‘column\_label’] to extract the columns of data of interest and index them to look for evidence of covid exploiting the "<span class='disease covid cdisease'" string inside sentences.

**Class** **DiseaseCovid(Sentences):**  
 - Input: disease\_evidences.tsv  
 - Output: list of strings  
 - Functions: return all sentences associating COVID with diseases   
 - Technique(s) / extra: *same as GeneSentences*

**Class Top10:**  
 - Input: the two datasets  
 - Output: list of tuples((geneid, diseaseid): frequency)  
 - Functions: return the top 10 distinct disease/gene associations

- Technique(s): exhaustive search exploiting the pmid values

**Class** **Associated(ABC) 🡪 abstract**

**Class** **AssociatedDiseases(Associated):**  
 - Input: the two datasets  
 - Output: list of strings  
 - Functions: provide disease list given gene symbol/ID  
 - Technique(s): usage of .loc[] and the pmid values

**Class** **AssociatedGenes(Associated):**  
 - Input: both datasets  
 - Output: list of strings  
 - Functions: provide gene list given disease name/ID  
 - Technique(s): *same as AssociatedDiseases*

WEB APPLICATION [PART 3]

Using Flask and importing the file of [part 1], we create a home page with the list of provided operations and for each of them, bound to a class defined in [part 2], we build a web page using a distinct HTML file to return the final result.

🡪 self.\_\_links in [part 1] allows to have short links for their relative webpages. Each one corresponds to their relative operation.

🡪 Implementing clickable “Homepage”, “Go back” and “Continue” buttons allows to facilitate webpage navigation.

EFFICIENCY

From operation 1 to 6, we can observe a high efficiency, as the computing time is very short taking only few seconds. Points 8 and 9 take few more seconds because of the double for loops iterating on large DataFrames. While point 7 take few hours to compute the output due to the exhaustive search approach performed on huge datasets. In this case, we have preferred the optimality of the solution at the expense of the efficiency.

WORKING APPROACH

- logical division of the different parts of the program to build them one after another in order to facilitate their connection increasing the final working efficiency

- reasoning together to understand the general path to follow and then working on the same parts either simultaneously or separately.

- the other parts of the project (such as PNGs, CRC Cards, further comments etc.) can be found in the same folder, with the code and the README.md file).