

This repo releases the code used to construct and execute patient-centric queries by ECO method. Given the annotated patient-centric query first construct the query graph using the code presented in “QUERY\_GRAPH\_EXTRACTION” directory and then construct and execute the focused expanded queries using the code presented in “ECO” directory.

#### A. QUERY GRAPH CONSTRUCTION

Construct the Query Graph for a given patient query given:

- Patient query's entities (forum entities, title entities, post entities) (Data/Queries/Queries.tsv).
- Medical Knowledge graph (ECO/QUERY\_GRAPH\_EXTRACTION/weighted\_KG\_reduced.tsv).

How to run the code:

```
python3 compute_subgraph.py $PATIENT_GRAPH_TYPE
$SUBGRAPH_METHOD $EXTRACTION_METHOD $PATIENT_INPUT_FILE.json
$KNOWLEDGE_BASE_FILE_ADDRESS $WEIGHTING_STATUS
$OUT_PUT_FORMAT $OUT_PUT_FILE_ADDRESS
```

Arguments:

- PATIENT\_GRAPH\_TYPE
  1. default: create the query graph by considering the subforum entity + title entities + post entities and the relations between them.
  2. with\_forum\_neighbors: same as default but add also neighbours of the subform entity.

- SUBGRAPH\_METHOD
  1. forum\_component: compute the connected component of the patient graph that contains the subforum entity.
  2. shortest\_path: make the patient graph connected using shortest path via the forum's component to other components. (it could be wighted or unweighted algorithm based on \$WEIGHTING\_STATUS)

- EXTRACTION\_METHOD
  1. NONE: It will returns the result of the subgraph method itself.
  2. steinter\_tree: It will returns an approximate steiner tree that contains subforum + title entities.

- PATIENT\_INPUT\_FILE.json
  - Address to a json file that contains entities from the subforum, title and post. An example is available in the patient\_inputs directory.

- KNOWLEDGE\_BASE\_FILE\_ADDRESS
  - Address to a text file that should be formatted like a dataframe that each row shows an edge and it should contains at least these 3 columns:
    - u: the first vertex of the edge.
    - v: the second vertex of the edge.
    - weight: the weight of the vertex (It should be a distance

metric.)

- WEIGHTING\_STATUS

- A boolean factor that can be 1 or 0. It shows whether we should assume the graph as a weighted graph or not (1=True and 2=False).

- OUT\_PUT\_FORMAT

- 1. edge\_list: Write the result as an edge\_list of the result subgraph.
  - 2. node\_list: Write the result as the list of nodes of the result subgraph.

- 3. gexf: Write the colored result subgraph as a gexf format, so it can be plotted in networkx and gephi.

- 4. all: Write all the results together.

- OUT\_PUT\_FILE\_ADDRESS: Address of the file that the result should be written there.

#### IMPORTANT NOTE:

Before running anything run the function in utils/write\_edge\_and\_names with your input knowledge graph. This step will hash the edge\_weight dictionary and make the other functions faster (they will read a pickle file named weights.txt for weights). Then run an algorithm.

One Example to Run:

***python3 compute\_subgraph.py default shortest\_path NONE patient\_inputs/0.json weighted\_KG\_reduced.tsv 0 edge\_list list.txt***

#### B. GENERATE AND EXECUTE FOCUSED EXPANDED QUERIES USING ECO METHOD

Construct and execute the focused expanded queries given:

- Dataframe of experimental queries containing queries title, posts, subforms together with their extracted medical entities. (ECO/Patient\_Posts\_With\_Entities.tsv).

- Mapping file of medical term with medical entities code. (ECO/entity\_to\_name.txt)

- Dataframe of the medical knowledge graph. (ECO/weighted\_KG\_reduced.tsv)

- Query Graph given as edge list and node list for each experimental query. (ECO/patient\_graph).

How to run the code:

***python ECO.py \$QUERY\_ID \$RESULT\_FILENAME***

Arguments:

- QUERY\_ID: An integer within the range of the ids given to the experimental queries picked.

- RESULT\_FILENAME: Filename of a json file where the results returned as a response to query number QUERY\_ID by ECO method.

One Example to Run:

***python ECO.py 0 query\_0\_output.json***