QUERYING A KNOWLEDGE GRAPH PROJECT DOCUMENTATION

PROJECT DESCRIPTION :

The project gives a possible solution approach to retrieving relevant texts when queried in natural language. The project uses a Natural language Processing techniques combined with graph based approach using python libraries.

FEATURES:

The project, when queried in natural language, is able to retrieve all the texts which are already stored in the code that have some common keywords with the question.

OVERVIEW:

The code consists of majorly 5 parts which are as follows :-

1. Extracting entities from the given text
2. Extracting relations from the given text
3. Storing the subjects, objects and relations in the form of triples
4. Creating and plotting (optional) the graph using the triples
5. Converting query to triple and searching the triple in the graph

CODE EXPLANATION FOR EACH PART :-

1. Extracting entities

   Entities are extracted using POS (parts of speech tagging) feature of the Spacy library in python. POS tagging recognise each word in the sentence as a noun, verb, compound etc.
   So nsubj is the main subject of the sentence, dobj is the main object and ROOT is the main verb of the sentence. The subject and object are obtained for each sentence and stored in a list. The prefix and modifiers are also included in the subject as they are a part of subject and object too in natural language and this the subject isn't treated just as one word.

   eg. The 10 year old child is watching television.
   Here, the subject will be 10 year old boy, not just boy.

2. Extracting relation

   Same ideology is used to extract the relations i.e. extracting the ROOT part of the sentence and including the adjacent helping verbs to create the full predicate.This is done by pattern matching, which first finds the ROOT, then checks if it is followed by any preposition ('prep') or an agent word. If yes, then it is added to the ROOT word. The list is created for each sentence's predicate.

   eg. The 10 year old child is watching television.
   Here, the predicate will be is watching, not just watching.

3. Storing triples

   After tokenising the text into sentences and extracting subject, object and predicate for each sentence in the form of list, the triple can be either stored in the form of dataframes, lists or dictionary.

4. Creating and plotting the graph

   For this part, networkx library and matpotlib library is used. Firstly, a unidirectional graph is created. Logically, from subject to object. Each node of the graph will contribute as an entity and each edge between two nodes will be the relationship between the node. The relation will be directed from the subject to the object. Nodes and edges are added to the graph based on the triples list that we formed. The nodes are given some 'ids'.

   NOTE : Node id is equal to the sentence number +1 if its the subject of the sentence or it is equal to -(sentence number+1) if it is the object of the node. This is done for purpose of convenience.

   Plotting of graph is straight forward using the matpotlib library (optional)

5. Querying the graph

   The same approach is applied to the question asked, where it is parsed and converted to triple format. Now either of the subject, object or the predicate is searched in the graph and all the matches are stored in the form of node id.
   Finally all the sentences that have their index equal to the node id -1 are given as we stored the node ids accordingly. Finally, we have the corresponding sentences that are related to our search.

EXTENSION TO THE PROJECT :-

Following are some extensions and improvements that can be made to the project :-

1. Using library pyPDF2 for conversion of pdf documents to text format.
2. Using better parsing techniques such as Stanford coreNLP library as they include properties such as coreference which are .
3. Using better graph libraries such as RDFlib and Neo4j for embedding of more semantically rich data and better knowledge graph and even data retrieval using cipher or sparql queries.
4. Using existing frameworks for conversion of complex natural language queries to structured queries in order to query the graph because the given project solution is not able to solve complex question problem.