Report for each Code

Our FYP revolves around big data where we scrapes the data and show it to user based on certain keywords and categories selected by the user.
For the time being, we can achieve from scraping the data to creating a knowledge base, so we'll be demonstrating each step we achieved along our way.

1) Web Scraping

Our main goal was to scrape the data in the internet. We have used wikipedia-api for this task. For that, we have to submit the topic name and let it do its job.
For this, we have used the given libraries.

➔ Wikipedia API

This is responsible for searching our topic. In the code, we will first search our topic, if we found that relevant pages for such topic exist, then we would apply further operation, else it will print its non-existence.

➔ Pandas

Pandas is used for data analysis and its manipulation. In our code, we have used it for tabular data of each list that has to be printed. Our list consists of all pages with relevant text, categories and topic. This list will be added to source that can be printed later in the code.

➔ Concurrent.futures

This is the best part of our code that runs our task in parallel making process execution faster. From our code, we have assigned 5 threads for our task where we are searching for pages for each links. We would get pages link via wikipedia-api, then we would search if the pages exists against each link. When this process finds relevant pages, it'll appends data in source list, and will update the progress bar.

➔ tqdm

This library handles printing of progress bars. When the program executes and finds relavant links, it searches out for pages from each links where we observe the progress of finding as shown in output below.

2) Get_entityPairs.py

Getting entity pairs of each pages.

For this task, we have selected the first page of our topic, that is most relevant,
As spacy and neuralcoref only works well with spacy 2.1.0, neuralcoref 4.0, Python 3.7.0, so we have to install it as per requirements,
in order for it to work.
Also the downside of spacy 2.1.0, we have to add filter spans function that only works with spacy above 2.1.4.

After all requirements, we would create pairs for first page, let's see how it works.
To begin with, we would use regex to replace next line and numbers, as we don't need it for the nlp.
Then we would use neuralcoref to replace each pronoun with the main person (resolving co-reference clusters), after that,
we would create unwanted tokens that would include particle, determiners, etc. Then, we would resolve each entity if it's not in unwanted token or stop words.
We may confirm each entity along with its type after the refinement part. After refinement, the program would tokenize each entity, and at the last, we would use pandas Data frame for printing the data accordingly.

3) Draw_kg.py

By using NLP library SpaCy, we extracted entity pairs from large amount of text. The spaCy library comes particularly handy when dealing with big amounts of text. It may be used to create data extraction and natural language understanding systems, as well as to pre-process text for deep learning.
The dependencies can be mapped in a directed graph representation:

- Words are the nodes.
- The grammatical relationships are the edges.

The get_entity_pairs function defines entity pairs as entities/noun chunks with subject — object dependencies connected by a root verb.
from_pandas_edgelist(df, source='source', target='target', edge_attr=None, create_using=None, edge_key=None). This function returns a graph from Pandas DataFrame containing an edge list.
The Pandas DataFrame contain at least two columns of node names and zero or more columns of edge attributes.
layout = nx.spring_layout(k_graph, k=0.15, iterations=20)
This function creates a graph where k controls the distance between the nodes and iterations is the number of times simulated annealing is run. If we don’t use this fuction then the nodes will be overlapping each other which would be very difficult to understand.
The next function nx.draw_networkx draws the graph with Matplotlib with options for node positions, labeling, titles, and many other drawing features means we can set the property of each element according to our choice. The network_edge_label creates a connection between vertices and then we show it up.