Project 2 Information Extraction Project

Teammates: Caiwu Chen (cc4786) & Zixiang Qu (zq2231)
Course: COMS 6111 Advanced Database Systems
Github: https://github.com/6111ADS/lab2

Setup and Running

1. Linux environment
2. Python 3.10
3. Setup requirements as described in the prompt:
    a. The Google Custom Search API to search the web. You used this API for Project 1. In your new `dbproj` virtual environment, run:
    ```
    pip3 install --upgrade google-api-python-client
    ```
    b. The Beautiful Soup toolkit to extract the actual plain text from a given webpage, and ignore HTML tags, links, images, and all other content that would interfere with the information extraction process. To install the toolkit, run:
    ```
    pip3 install beautifulsoup4
    ```
    c. The spaCy library to process and annotate text through linguistic analysis (e.g., split sentences from paragraphs, tokenize text, detect entities). You will need to use Python 3.10 or newer (see above), and follow these steps on your Google Cloud VM:
    ```
    sudo apt-get update
    pip3 install -U pip setuptools wheel
    pip3 install -U spacy
    python3 -m spacy download en_core_web_lg
    ```
    d. The SpanBERT classifier to extract the following four types of relations from text documents:
        i. **Schools_Attended** (internal name: `per:schools_attended`)
        ii. **Work_For** (internal name: `per:employee_of`)
        iii. **Live_In** (internal name: `per:cities_of_residence`)
        iv. **Top_Member_Employees** (internal name: `org:top_members/employees`)
    e. We have implemented the scripts for downloading and running the pre-trained **SpanBERT** classifier for the purpose of this project:
        i. `git clone https://github.com/Shreyas200188/SpanBERT`
        ii. `cd SpanBERT`

iii. `pip3 install -r requirements.txt`

iv. `bash download_finetuned.sh`

f. The Google Gemini API to extract the above relations from text documents by exploiting large language models, as a state-of-the-art alternative to **SpanBERT**:
`pip install -q -U google-generativeai`

4. If you had issue on downloaded SpanBERT's requirement package, you can manually download pytorch package. After you generate a pre-trained SpanBERT successfully, you should drag four files into the main directory:

   spanbert.py, spacy_help_functions.py,

   These two files are already in the compresesd file we submitted. You are welcome to overwrite them. They are the same. You may try if the original file in the compressed file works.

   pytorch_pretrained_bert, pretrained_spanbert are also essential to use. But these two dir are too big, you have to **drag them down to the dir**.

Package

As we setup before, this project use Google Custom Search API, Beautiful Soup toolkit, spaCy library, SpanBERT classifier, Google Gemini API, Python's requests package, and support optional features (e.g., saving logs or data to the cloud)

`pip install -r requirements.txt`

This command will generate the required packages for your environment.

Description

1. README.pdf: description of the project
2. config.py: the python file is the backup of the google api key, engineer key, and Gemini key, but never be called.
3. Google_search.py: the file borrowed from project 1 which uses Google search API to get 10 results but passes a parameter to skip seen urls.
4. Webpage_retriever.py: text clean process. Using beautiful soup to retrieve the content of the URL and do some preprocessing such as getting plain text, removing control characters, and spaces duplicating new lines and backslashing
5. relation_extractor.py: containing function of relation extraction by either spanbert or Gemini. They both use spaCy to do the annotation for the text and then for Spanbert, it predicts the relationship between subject and object, and we filter out the relationships that are not our target to too low confidence. For Gemini, we send sentences that have a potential relationship to Gemini with clear prompts and let Gemini decide. We will process Gemini's response to add our result.

6. Project2.py: running by this kind of command: `python3 project2.py [-spanbert|-gemini] <google api key> <google engine id> <google gemini api key> <r> <t> <q> <k>` where
    a. [-spanbert|-gemini] is either -spanbert or -gemini, to indicate which relation extraction method we are requesting>
    b. <google api key> is your Google Custom Search Engine JSON API Key (see above)
    c. <google engine id> is your Google Custom Search Engine ID (see above)
    d. <google gemini api key> is your Google Gemini API key (see above)
    e. <r> is an integer between 1 and 4, indicating the relation to extract: 1 is for Schools_Attended, 2 is for Work_For, 3 is for Live_In, and 4 is for Top_Member_Employees
    f. <t> is a real number between 0 and 1, indicating the "extraction confidence threshold," which is the minimum extraction confidence that we request for the tuples in the output; t is ignored if we are specifying -gemini
    g. <q> is a "seed query," which is a list of words in double quotes corresponding to a plausible tuple for the relation to extract (e.g., "bill gates microsoft" for relation Work_For)
    h. <k> is an integer greater than 0, indicating the number of tuples that we request in the output
7. spanbert.txt: result of using SpanBERT for r=2, t=0.7, q=[bill gates microsoft], and k=10
8. gemini.txt: using Google's Gemini API for r=2, t=0, q=[bill gates microsoft], and k=10
9. Pretrained_spanbert: a dir containing pretrained spanbert mode (originate from SpanBERT, this file is too big, so it will not in the compressed file)
10. pytorch_pretrained_bert : a dir containing pytorch pretrained spanbert mode (originate from SpanBERT, this file is too big, so it will not in the compressed file)
11. Spacy_help_functions.py: helper function to get annotation (originate from SpanBERT)
12. spanbert.py: spanbert model function (originate from SpanBERT)

The workflow pipeline takes the given parameters and decides which model will be used. They both use the query to do the search and retrieve text from the website to do the relation extraction. For each model, the algorithm is explained in the relation_extractor.py. It will continue running until it finds an adequate amount of relationships or all queries have been explored. It will not contain duplicate questions. For SpanBERT, it starts with the highest confidence; for Gemini, it begins with the query order appended to the list.

Step

1. Finish the setup environment, and make sure SpanBERT is working correctly. You can test by running an example, which will be included in the spanBERT after you clone successfully.
2. Drag four files into the root directory.
3. Make sure you have all the required packages installed.
4. Follow the order of parameter taking in the project2.py in the Description above.

Result

Available results in spanbert.txt and Gemini.txt. Feel free to test.