Project 2

Iterative Set Expansion (ISE)

Note

Project code: `project_2.py`

Use the helper functions provided by TA in `spacy_help_functions.py`

Authors

- Evan Ting-I Lu (tl3098)
- Pitchapa Chantanapongvanij (pc2806)

Files

1. `project_2.py` (main program)
2. `README`
3. `transcript.txt`

How to run our program (including setting up)

1. Create a new confda environment and install python 3.6

```
conda create --name 6111_ADB_proj_2 python=3.6
```

2. Activate the conda env

```
conda activate 6111_ADB_proj_2
```

3. Verify Python version

`python --version` (should see `Python 3.6.13 :: Anaconda, Inc.`)

4. Install packages using pip

```
python -m pip install google-api-python-client beautifulsoup4 spacy
```

5. Download "en_core_web_lg" with spacy

```
python3 -m spacy download en_core_web_lg
```

6. Clone our project from GitHub (our repo is set to private so please inform us if need access)

```
git clone https://github.com/evantilu/SP22-6111-ADB-Projects.git
```

7. `cd` to project folder

```
cd SP22-6111-ADB-Projects/project_2/
```

8. Install more required packages for SpanBERT

```
python -m pip install -r requirements.txt
```

9. Download fine-tuned SpanBERT model

```
bash download_finetuned.sh
```

10. Manually install missing packages. In our case there were error during step. 6 so need this step to make the project run

```
python -m pip install boto3 scipy
```

Now the program is ready to run

11. Issue `python project_2.py -h` to check for options

12. Issue commands with our API key and Search engine ID

- API Key: `AIzaSyBBi7WOxVf_3waDwBkM_roXIMaceDCsVog`
- Engine ID: `91b445df8028cd711`

> Example: `python project_2.py AIzaSyBBi7WOxVf_3waDwBkM_roXIMaceDCsVog 91b445df8028cd711 2 0.7 "bill gates microsoft" 10`

Internal design of our project

Our program can be broken into 1 main, and 4 function definitions (initiate_query, extract_plain_text, truncate_plain_text, and update_query).

`Main()`: Follows the direction of the project 2 description such that it first takes in input parameters r, t, q, and k from the user. While the number of extracted tuples is less than k (number of tuple user requested in the output) we will call initiate_query() which starts the query using Google JSON API credentials on the given query string q (initially q is provided by the user). Then we iterate through each webpage in the dictionary that is returned by initiate_query(). For each webpage we use Beautiful Soup to extract the plain text, each time making sure that the URL has not been previously seen. To extract plain text from the webpage we call extract_plain_text() and make sure to timeout the process if it tastes more than 20 seconds. Following this, we call truncate_plain_text() to reduce the plain text to 20000 characters. Having done this we then feed the plain text into the SpacyBERT process in order to split the text into sentences and for named entity recognition for each of the sentences.

With the extracted relations returned from extract_relations() stored in 'relations', we iterate through each key (tuple of subject, relation type, object). We make sure that the relation type is the same type as the target relation r (provided by the user). Then we only append to return set X if the tuple does not already exist or the confidence level is higher than the one already in X. After iterating through all the relations, if k results have not been reached query q is updated by calling update_query(). Finally, when k results have been extracted we break and return results in X to users.

`initiate_query()`: Starts a query using Google JSON API with the current query value and returns the top 10 query result in its raw format which is a Python dictionary.

`extract_plain_text()`: Use Beautiful Soup to extract plain text from the given URL. Also removes unwanted elements and Unicode chars from the text.

`truncate_plain_text()`: Truncated the plain text into 20000 characters.

`update_query()`: Uses subject and object attributes in each tuple in X to update the query q accordingly (the top tuple with the highest score that has not been previously appended to query q will be added).

Detailed description of how we did step 3

For step 3 we kept a variable called prev_url to track previously seen URLs and make sure that we only process new URLs each time. Also, for the Beautiful Soup plain text extraction, we set a 20 seconds timeout. Then we truncate the plain text to 20000 characters and feed the text into the SpacyBERT process. We apply the spacy model to extract entities of interests from raw text. As instructed, we only feed the entity pairs to the next step that will possibly give us our interest of relations. SpanBERT will then load pre-train model and extract relation. We then check whether or not the extracted relation is of the type that we target. For each returned relation key we made sure that the tuple is unique and that the confidence score is higher than ones already in X.

Credentials

- Google Custom Search Engine JSON API Key

> AIzaSyBBi7WOxVf_3waDwBkM_roXIMaceDCsVog

- Engine ID:

> 91b445df8028cd711