Project 1: Semantic Word Comparator

Lucia Camenisch

November 9, 2022

1 List and description of project files

The project contains the following files:

1. `ref-sentences.txt`
2. `stopwords.py`
3. `functions.py`
4. `main_function.py`
5. `tests.py`

1.1 `ref-sentences.txt`

This is the given text file which will serve as our text source for the project. The file will be read by our program, it will not be modified.

1.2 `stopwords.py`

This is the second given file. It contains a list of stopwords (words with no intrinsic semantic meaning which will be discarded from our results). This file won't be modified as well. It will be imported in `functions.py`.

1.3 `functions.py`

This file contains our four handmade functions which will act as the building blocks of our final program.

1.3.1 `read_reference_text`

This function takes a text file input and creates a list. Each element of this list is a sentence of the text file and the sentence itself is a list which contains each word of the sentence as individual strings. Punctuation is removed and all words are lowercase.

1.3.2 `make_word_vector`

This function takes a `word` and a `text` and will return how many times a word that is in the same sentence as the chosen `word` inside our `text` is present. This result will be given as a dictionary, where keys are words in the same sentences as `word` and the values are integers, representing how many times we found the key word.

1.3.3 `scalar_product`

This function computes the scalar product between two dictionaries by multiplying values which have the same keys in the two dictionaries together and then summing all of these products.

1.3.4 `sim_word_vec`

This function computes the cosine similarity between two dictionaries (called vectors).

1.4 main_function.py

This is the main program function. It will use the four building blocks created previously and wrap everything together from start to end.

This function takes a text file and a list of words (word_list). It returns a dictionary which for each word from the list (key) gives the closest word from the list semantically, as well as the cosine similarity between the two words. The choice of the closest word and the computation of the cosine similarity are based on the text file.

1. We use read_reference_text to process ref-sentences.txt. This will be our text.
2. We create a dictionary called words_vectors. The keys of words_vectors are the words of word_list. The value of each key is the dictionary created by make_word_vector, using text.
3. for each word in word_list, we execute the following steps:
   (a) We create a dictionary called sim_dic. The keys of this dictionary are the words from word_list except our word.
   (b) The value associated to each key in sim_dic is the cosine similarity between our word and the key. This is computed using sim_word_vec (scalar_product is called directly inside sim_word_vec).
   (c) we select the key-value pair inside sim_dic which has the biggest cosine similarity.
   (d) This key-value pair is stored as a dictionary with one element.
4. We create a dictionary which contains each word from our word_list as keys and the single key-value pair dictionaries created at the previous step as values of each word. This final dictionary is called similarities and it is the result returned by our main_function.

1.5 `tests.py`

This file tests our Semantic Word Comparator on the example sample (from canada to train) and then on the requested sample (from agriculture to web). We display the output of `tests.py`:

File - tests

```
"/Users/lucia/Documents/Travail/Business Analytics Master/Algorithmics and data management/Project-1/venv/bin/python" /Users/lucia/Documents/Travail/Business Analytics Master/Algorithmics and data management/Project-1/tests.py
{'canada': {'switzerland': 0.45939402815457103},
 'car': {'industry': 0.43555381285188766},
 'conflict': {'industry': 0.6017917385344789},
 'disaster': {'conflict': 0.5366651487769895},
 'flood': {'disaster': 0.37853661383543374},
 'germany': {'switzerland': 0.5106055211661624},
 'industry': {'conflict': 0.6017917385344789},
 'rail': {'road': 0.7854022187912391},
 'road': {'rail': 0.7854022187912391},
 'switzerland': {'germany': 0.5106055211661624},
 'technology': {'industry': 0.5755225096775411},
 'train': {'road': 0.462887400455656}}
{'agriculture': {'industry': 0.6667323464738124},
 'anchovy': {'fish': 0.3858237607603034},
 'china': {'mexico': 0.6357788221166607},
 'cod': {'fish': 0.4538295033485589},
 'communication': {'industry': 0.7298598624422352},
 'fish': {'industry': 0.4713384412176382},
 'fishery': {'industry': 0.44940171739187956},
 'france': {'italy': 0.7265663050409344},
 'industry': {'communication': 0.7298598624422352},
 'internet': {'communication': 0.5787280129745741},
 'italy': {'france': 0.7265663050409344},
 'labour': {'industry': 0.577365382209493},
 'mexico': {'china': 0.6357788221166607},
 'spain': {'france': 0.660737138568571},
 'transport': {'industry': 0.7190298531914845},
 'tuna': {'fish': 0.40811602938873387},
 'web': {'internet': 0.337350760743591}}

Process finished with exit code 0
```

Page 1 of 1

2 Collaboration

For this project, I had some discussions and exchanges of ideas with Francisco Arrieta.