Cosine similarity - Comment

Hannes Guth

November 19, 2022

Comment on the functionality

The program consists of a main function and 6 other functions. In order to calculate for each word from a given list of words the most similar word, based on the cosine similarity function and a given source text, each of the functions will be called at least once.
The first step is to read the source text and process it in a way that the result can be used in the further functions. This is achieved by calling *get_and_prepare()*. This function extracts the lines of the text file and saves each line as a list of words in its respective place in an array. It also makes all the words lowercase.
As later on in the program, it will be useful to have these lists as sets, the function *make_set(ar)* takes this array of lists and converts the lists to sets, so that a second array ar_set is created.

Now, the databasis for calculating the cosine similarity will be created.

For that, the function *get_numbers(ar_set, ar, wlist)* is used. It gets both created arrays and the wordlist wlist as inputs. In order to store word appearances in the surrounding of each word of wlist, a dictionary, called "word" is created. In this dictionary, every word in wlist becomes a key. The corresponding value to each word/key is another dictionary, containing all words that appear in the surrounding of the word from wlist and the number of their appearances. To fill this inner dictionary, the function *counter(array, w, word_dict)* is called for every word in wlist and every line/position in the arrays ar and ar_set. This function updates the dictionary of the current word for each line and returns an updated version of the dictionary.

Since the databasis for the actual similarity calculation is done, the calculation can begin.

To find the similarity of each word with each other word, the function *similarity_matrix(words, wlist)* creates a cosine-similarity matrix for every word combination. Each position in the similarity matrix is calculated as follows:

$$sim(v_i, v_j) = \frac{v_i * v_j}{\sqrt{(v_i * v_i)(v_j * v_j)}}$$

The result of $\mathrm{v_i}$ * $\mathrm{v_j}$ is denoted in the respective entry in mix matrix, $\mathrm{v_i}$ * $\mathrm{v_i}$ is denoted in its respective entry in one_matrix and so is $\mathrm{v_j}$ * $\mathrm{v_j}$. Multiplying $\mathrm{v_i}$s or $\mathrm{v_j}$s with each other means to multiply their respective dictionaries. Having these values, the similarity matrix can be filled up, using the formula from above. The function *similarity_matrix(words, wlist)* therefore returns a complete matrix with cosine-similarity values.

Finally, the function *show_similar_words(wlist, sim_matrix)* is called. It takes the wordlist wlist and the previously created similarity matrix as inputs. For each word in word lists, this function finds the word with the biggest similarity and returns this word as well as the corresponding similarity value.

I did not collaborate with anybody in this project.

Outcome

In the following, the outcome of the program is presented.

```
canada    ->    switzerland ,  0.47208657916738866
disaster    ->    conflict ,  0.5311945187926544
flood    ->    disaster ,  0.36892781634805605
car    ->    industry ,  0.4318116350745541
road    ->    rail ,  0.7810656193744746
train    ->    road ,  0.4682414286066809
rail    ->    road ,  0.7810656193744746
germany    ->    switzerland ,  0.507524413740697
switzerland    ->    germany ,  0.507524413740697
technology    ->    industry ,  0.5721864549827032
industry    ->    conflict ,  0.592661165460134
conflict    ->    industry ,  0.592661165460134
```