Algorithmics and Data Management
Project #1: Semantic Word Comparator

Due: 2022-11-21

Note on Collaboration: You may work with other students. However, each student should write up and hand in his or her project separately. Be sure to indicate with whom you have worked in the comments of your submission.

Problem statement

The goal of the project is to implement a Python program that can compares words based on their meaning (semantics). According to the distributional hypothesis, the meaning of a word w is given by the set of words that occur in the same context as w.

For this project we will consider that the context of a word is the sentences in which it occurs.Therefore, given a (large) reference text T, the semantic vector of a word w is the dictionary $\{w_1 : f_1, w_2 : f_2, ..., w_k : f_k\}$ where the w_i's are the words that occur at least once in the same sentence as w and each f_i indicates the number of times w_i appears in the same sentence as w. In addition, the w_i's must

- not belong to a given set of stop words (is, and, or, again, ...)
- be different from w
- be at least 3 character long

The semantic similarity of two words x and y is then defined as the cosine similarity (see the lecture notes) of the semantic vectors of x and y.

Specifications of the program

Given

- a large reference text file T that contains one sentence per line,
- a set of stop words S,
- a list of words $W = [w_1, ..., w_n]$,

the program must

1. use T and S to compute the semantic vector of each word of W

2. for each word w_i in W output the word w_j ($w_j \neq w_i$) of W that has the most similar meaning and it similarity value

Sample output

```
Reference file: ref-sentences.txt
Words to compare: ['canada', 'disaster', 'flood', 'car', 'road',
'train', 'rail', 'germany', 'switzerland', 'technology',
'industry', 'conflict']

Most similar words:
canada ->  switzerland, 0.45957109273164287
disaster ->  conflict, 0.5367919016891342
flood ->  disaster, 0.37827942031379597
car ->  industry, 0.4352137682277799
road ->  rail, 0.7841804676988314
train ->  road, 0.465289360430555
rail ->  road, 0.7841804676988314
germany ->  switzerland, 0.5106208094608132
switzerland ->  germany, 0.5106208094608132
technology ->  industry, 0.5753363429461995
industry ->  conflict, 0.6014611520513141
conflict ->  industry, 0.6014611520513141
```

Architecture of the program

This is a suggestion for organizing your program. It is not mandatory.

The program can be decomposed into

1. a function

   ```
   read_reference_text(filename: str) -> list[list[str]]
   ```

   that reads a reference text file that has one sentence per line and creates a list of sentences where each sentence is a list of lowercase strings (the words)

2. a function

   ```
   make_word_vector(w: str, txt: list[list[str]]) -> dict[str, int]
   ```

   that creates the semantic vector of a word w using a reference list of sentences

3. a function

   ```
   sim_word_vec(v1: dict[str, int], v2: dict[str, int]) -> float
   ```

   that computes the cosine similarity of two semantic vectors

4. a main function that

(a) reads the reference file

(b) creates the semantic vectors for the words in W

(c) for each word in W finds the word with the highest similarity and prints it

TODO

Write your program in Python and test it on the provided reference file and the word list `['spain', 'anchovy', 'france', 'internet', 'china', 'mexico', 'fish', 'industry', 'agriculture', 'fishery', 'tuna', 'transport', 'italy', 'web', 'communication', 'labour', 'fish', 'cod']`

Please submit

- the source code of the program. Each function definition must have a "docstring" describing the parameters and the pre and post conditions of this function
- the output it produces for the provided reference file and test word list
- a brief comment file (in pdf) that explains how your program works and with whom you collaborated

Practical information

- The reference file is `ref-sentences.txt`. It contains one sentence par line
- To read a file line by line use the following code :

```
f = open(filname)  # filename is a string containing the file name;
                   # if the file is in the same folder as the program
                   # the filename is sufficient, otherwise specify
                   # the full file access path
for line in f:
    ## process the line ##
    . . .
f.close()
```

- To extract the words of a string (line) you can use

```
import re     # put it at the beginning of your program
...
re.split("[ ,;:\'\"\?!]+", the_string)
```

- The English stop words are in the file `stopwords.py` (you can copy it directly into your program)