Final assignment

Status: Completed
Your identity is visible during marking.

Deadline: 16 Januari 2019

Retake final assignment

See the PDF with instructions Listen , and download the archive with materials.

Comment 1

Here is an example call and (abbreviated!) output for the `split_data`-function.

```
>>> import solution_gjb as s
>>> test_and_reference = s.split_data()
>>> test_and_reference[0]
[('adventure', ['Dan', 'Morgan', 'told', 'himself', ...]),
 ('adventure', ['Gavin', 'paused', 'wearily', '.', ...]),
 ...]
>>> test_and_reference[1]
[('fiction', ['the', 'bishop', 'looked', 'at', ...]),
 ('humor', ['i', 'realized', 'that', 'Hamlet', ...]),
 ...]
>>>
```

Note 1: What I call 'tokens' in the description refers to *words* and *punctuation symbols*.
Note 2: The items in the test collection always contain 1 label and 1 document.
Note 3: The list holding the reference collections contains tuples pairing a category label to the concatenation of all documents of that category that were not selected as test documents.

Comment 2

With regard to the functions that calculate the top-n profiles:

```
  >>> # Only the 10 most freq words/tokens in this example as a `dummy
value' so that you
  ... # can copy/paste and test your own code.
  ...
  >>> top_10_in_brown = {'the':0.06025790739171472,
',':0.050236308896375446,  # 1 2
                        '.':0.042495986882444936,
'of':0.03135743270708031,  # 3 4
                        'and':0.02484774266443448,
'to':0.022526851717889894, # 5 6
                        'a':0.019920047675147608,
'in':0.018375083534850394, # 7 8
                        'that':0.009123383557585654,
'is':0.008705709305610097} # 9 10
  >>> from nltk.corpus import brown
  >>> example_doc = brown.words()[:2000]
  >>> example_doc[13:21]  # just for inspection
  ['election', 'produced', '``', 'no', 'evidence', "''", 'that', 'any']
  >>> import solution_gjb as s
  >>> s.topn_profile(example_doc,['the',',','.','of','and'])
  {',': 0.0345, '.': 0.0365, 'the': 0.0585, 'of': 0.0305, 'and': 0.017}
  >>> # Note the rel freqs of these 5 words are all slightly lower in
example_doc
  ... # than in top_10_in_brown. We should see negative z-values...
  ...
  >>> s.ztopn_profile(example_doc,['the',',','.','of','and'])
  {',': -3.22..., '.': -1.32..., 'the': -0.33..., 'of': -0.22..., 'and':
-2.25...}
  >>>
```

Hemanth Kumar Battula , 11 Nov 2018 22:46

File name: Text_Classification.py (11,6 KB)

Status set to: To be marked

Hemanth Kumar Battula , 11 Nov 2018 22:47

File name: Program Report.rtf Listen (128,1 KB)

Status set to: To be marked

Jacobo Rouces Gonzalez , 30 Nov 2018 17:24

Status set to: Completed

Grade set to: VG

Comment: - Parts A,B,C,D and VG produce correct results. The accuracies and recalls using ztopn are slightly different to the results from our implementation but consistent, so it is probably due to a minor difference in implementation.

- Note that the expression "accuracy for rank 5" is incorrect. It is "recall at rank 5", or at rank 1 for that matter. It happens that recall at rank one is mathematicaly equivalent to accuracy, but accuracy is not dependent on any rank.

- The most significant issue is that your implementation is terribly slow. It took around 10 min in my machine while my own implementation takes around 10 seconds! This is a toy example but for "big data" stuff in the real world it would become impossible. The cause seems to be that in the get_xxx_profile_accuracy functions, for every test item, you repeat the creation of profiles for all the reference items. This is unnecessary, you can first create each profile only once and then do all the possible cosine distances between them.

We don't discount points from performance issues, but we consider that these are important for you to know in the future.