T9 - Artificial Intelligence

T-AIA-901

Travel Order Resolver

Bootstrap

1.2

You are about to build an automated tool that sorts political speeches, kind of a political compass.

More specifically it takes a bunch of speeches in french language and classifies them in two clusters, based on the vocabulary they use. The underlying question is: *"Does this clustering make sense, according to what we know about the authors?"*

GETTING THE DATA SETS

Firstly, you need to recover the texts from *discours.zip*. You should be able to produce a **Bunch** object with several attributes, including:

- a **filenames** attribute that contains *"Alain_Madelin_702.txt*, *Arlette_Laguiller_32.txt"*, …
- a **data** attribute that contains the texts themselves

scikit_learn documentation might help you with data loading and text processing

To make sure everything works all right, display a dictionary that associates all file titles with the first 40 characters of the file. Your result should look like this:

Terminal

```
~/T-AIA-901>
'Arlette_Laguiller_299.txt': 'Travailleuses, travailleurs, camarades et amis,
Pour commencer ce meeting, je', ...
```

From raw text to vector

As far as pictures are concerned, the typical mapping is to code every vertex with a RGB value. When it comes to text documents, the simplest way is to count (for every possible word in the language) how many times this word appears in the text.

Next step is to apply a tf-idf transformation, to replace this raw count with relative frequencies.

```
  (0,  0)	0.0
  (0,  1)	0.0
  (0,  2)	0.0
  (0,  3)	0.0
  (0,  4)	0.0
  (0,  5)	0.0
  (0,  6)	0.0
  (0,  7)	0.57501852
  (0,  8)	0.34113499
  (0,  9)	0.0
  (0, 10)	0.0
  (0, 11)	0.0
  (0, 12)	0.0
  (0, 13)	0.47152341
  (0, 14)	0.57501852
  (0, 15)	0.0
  (1,  0)	0.0
  (1,  1)	0.0
  (1,  2)	0.0
  (1,  3)	0.40824829
  (1,  4)	0.40824829
  (1,  5)	0.0
  (1,  6)	0.0
  (1,  7)	0.0
  (1,  8)	0.0
  (1,  9)	0.0
  (1, 10)	0.0
  (1, 11)	0.0
  (1, 12)	0.40824829
  (1, 13)	0.40824829
  (1, 14)	0.40824829
  (1, 15)	0.0
  (2,  0)	0.0
  (2,  1)	0.490779
  (2,  2)	0.490779
  (2,  3)	0.0
  (2,  4)	0.0
  (2,  5)	0.0
  (2,  6)	0.0
  (2,  7)	0.0
  (2,  8)	0.0
  (2,  9)	0.0
  (2, 10)	0.0
  (2, 11)	0.0
  (2, 12)	0.490779
  (2, 13)	0.4024458
  (2, 14)	0.0
  (2, 15)	0.3397724
  (3,  0)	0.26396385
  (3,  1)	0.0
  (3,  2)	0.0
  (3,  3)	0.0
  (3,  4)	0.0
  (3,  5)	0.0
  (3,  6)	0.52792769
  (3,  7)	0.0
  (3,  8)	0.15659897
  (3,  9)	0.0
  (3, 10)	0.0
  (3, 11)	0.0
  (3, 12)	0.0
  (3, 13)	0.0
  (3, 14)	0.0
  (3, 15)	0.0
  (4,  0)	0.0
  (4,  1)	0.59430676
  (4,  2)	0.0
  (4,  3)	0.0
  (4,  4)	0.0
  (4,  5)	0.0
  (4,  6)	0.0
  (4,  7)	0.0
  (4,  8)	0.35257792
  (4,  9)	0.59430676
  (4, 10)	0.0
  (4, 11)	0.0
  (4, 12)	0.0
  (4, 13)	0.0
  (4, 14)	0.0
  (4, 15)	0.41144595
  (5,  0)	0.0
  (5,  1)	0.0
  (5,  2)	0.0
  (5,  3)	0.0
  (5,  4)	0.0
  (5,  5)	0.0
  (5,  6)	0.0
  (5,  7)	0.0
  (5,  8)	0.26990725
  (5,  9)	0.0
  (5, 10)	0.0
  (5, 11)	0.0
  (5, 12)	0.0
  (5, 13)	0.0
  (5, 14)	0.9099135
  (5, 15)	0.0
  (6,  0)	0.31497221
```

Anyway, this builds a huge, scarce matrix whose number of columns is the size of the vocabulary.
You're expected to produce a (9w,216z0) sparse matrix, for there are 9w texts and 216z0 different words.

w and z represent two different digits

Doing this has you lose any information about positions of words in the text. This might not be the smartest thing to do for some applications.

Unsupervised learning

If you don't want to feed our algorithm with *a priori* information about the political opinion of the orator, but instead, you want to associate texts freely, based on the sole vocabulary. Hence, you need to do some *unsupervised learning*.

Let's start with a simple basic classifier you may already encounter somewhere, namely *k-means*. Fix $k = 2$ and see what comes out of the box.

Notice that most politicians have all their speeches that belong to one class, which makes sense!

Find the dates when they were candidates, then make a hypothesis on how the algorithm divides speeches in two categories here?

Be careful that k-means is based on strong hypotheses (Voronoi-based division of the space), especially when you have only two classes. You can probably do much better!

Now, play around with other algorithms, tune some parameters, investigate metrics, observe results.

Text detection can be improved a lot by removing *stopwords*.
Similarly, look for *lemmatization* to help you with *inflection*.

Supervised learning

Now, if you want to challenge some algorithms to identify one specific author, you can divide your sample in 3 categories:

- a folder containing a part of the texts from one politician
- a folder containing an equal part of the texts from other candidates
- a folder containing all the remaining texts from all the orators

Train a **Naive Bayes** algorithm using the first folders, then check predictions quality with the third folder.

How much learning an algorithm do you need to train? Find out an ideal training/testing ratio!

Naive Bayes is a very common, though simple, algorithm in NLP. Take time to understand the maths behind it.
Other algorithms, such as Topic Modeling, are much more difficult to understand. So, best to master this one before jumping to more sophisticated ones.