Tim auf der Landwehr
t.auf.der.landwehr@student.rug.nl
S2548682
Last revision: December 8, 2013

Learning from Data
Assignment 4

Summary

This is the assignment 4 in the course Learning from Data at the University of Groningen.

Exercise 1: arff-file generation for reuters-allcats.csv

The arff file generation script is provided in `reuters_cluster_arff_gen.py`. It is supposed to be used in the following manner. If you run in the first time, the parameter `--features` needs to be used to generate the features file.

```
 Reuters-ARFF-Generator v1.0 by Tim auf der Landwehr

 Usage: reuters_cluster_arff_gen.py
  Loads features from [reuters.features.json] and dialectology from
      [reuters-allcats.csv]
   and writes arff output to [reuters-allcats.arff]

 --features
  Generates features to [reuters.features.json].
 --in file
  Change name of the annotated input file.
 --out file
  Change name of the arff output file.
```

Exercise 2: Running K-means

Runnig K-means on the arff file generated in excercise 1 by the command

```
java -cp weka.jar weka.clusterers.SimpleKMeans -c 1 -N 7 -t reuters-allcats.arff
```

leads to the following results

```
Clustered Instances

0         3 (  0%)
1         1 (  0%)
2         1 (  0%)
3       465 ( 11%)
4         2 (  0%)
5      2356 ( 58%)
6      1251 ( 31%)


Class attribute: @@class@@
Classes to Clusters:

    0    1    2    3    4    5    6  <-- assigned to cluster
    3    1    1  349    2 1293  280 | Neg-
    0    0    0   70    0  509  211 | Pos-acq
    0    0    0    5    0   29    1 | Pos-coffee
    0    0    0   39    0  490  751 | Pos-earn
    0    0    0    2    0   29    3 | Pos-gold
    0    0    0    0    0    0    4 | Pos-heat
    0    0    0    0    0    6    1 | Pos-housing
```

```
Cluster 0 <-- No class
Cluster 1 <-- No class
Cluster 2 <-- No class
Cluster 3 <-- Pos-acq
Cluster 4 <-- No class
Cluster 5 <-- Neg-
Cluster 6 <-- Pos-earn

Incorrectly clustered instances : 1965.0 48.1736 %
```

This result is actually not very satisfactory, but it was the best I could reach.

Exercise 3: Lexical classification of dutch dialects

I solved this task using all uincode letters from `\u0020` to `\u03E8`. Therefore the features are totally independent from the input. This range was defined by me with the help of the script `check_unicode_in_dialectology.py`, which checks all the characters in an given file against a certain range of unicode characters and tries to find those that do not occur in the range. Using this on several of the dialectology files, I figured that the defined range is sufficient to match all characters that are used in these files.

To be able to judge the accuracy of this features, I annoted the file `035.txt` using the script `annotate_dialectology.py`, that simply adds the classes 'devil', 'lucifer' and 'no_class' to the each line depending on whether it starts with *d*, *l* or something else. This divides the data quite well into the two classes and sets 'no_class' for those line which do not contain any data (but '|').

This annotated file can be processed with the script `dialectology_arff_gen.py`, which generates an WEKA-conform arff file from the annotated input file.

The arff-file can be processed in weka using

```
java -cp weka-3-6-10/weka.jar weka.clusterers.SimpleKMeans -c 1 -N 3 -t
    assignment4/035.arff
```

That leads to the following output

```
Clustered Instances

0       21 (  6%)
1      231 ( 64%)
2      110 ( 30%)

Class attribute: @@class@@
Classes to Clusters:

   0   1   2  <-- assigned to cluster
   1 225   0 | devil
  20   0 109 | lucifer
   0   6   1 | no_class

Cluster 0 <-- No class
Cluster 1 <-- devil
Cluster 2 <-- lucifer

Incorrectly clustered instances : 28.0 7.7348 %
```

This is a quite good result, but I should mention, that a simple classifier that only uses the first letter of the line would work better in this case. But this apporach will do better clustering other words.