Homework 7:
Unsupervised Learning: Kmeans

Benjamin Roth, Marina Sedinkina
Symbolische Programmiersprache

Due: Thursday December 13, 2018, 16:00

In this exercise you will:

- implement kmeans clustering

Exercise 1: Kmeans [7 points]

On the course homepage, you can find the file `courses.txt`, containing several LMU courses of studies per line. Download it into the `data/` folder of your project. Take a look at `hw07_kmeans/kmeans.py`. In this exercise you will have to implement some methods to perform the clustering. This homework will be graded using unit tests by running: `python3 -m unittest -v hw07_kmeans/test_kmeans.py`

1. Implement the Reader class method `get_lines(self)`. This method should return the list of courses from the file courses.txt with listed courses, i.e. one course per line.

2. Implement the Reader class method `normalized_word(self, word)`. This method should normalize the word by making it lower case and deleting punctuation marks from it. `Hint`: you can use set(string.punctuation)

3. Implement the Reader class method `get_vocabulary(self)`. This method should return vocabulary: the list of unique normalized words from file, sorted alphabetically. `Note:` words in vocabulary should be normalized, use `normalized_word(self, word)` to do this.

4. Implement the Reader class method `vectorspaced(self,course)`. This method should represent each course by one-hot vector: vector filled with 0s, except for a 1 at the position associated with the word in vocabulary. `Note`: the length of vector should be equal to the vocabulary size

5. Implement the KMeans class method `distance(self,x,y)`. This method should calculate Euclidean distance between two given vectors.

6. Implement the KMeans class method `vector_mean(self,vectors)`. This method should calculate the mean of the vectors. `Hint`: you can use **numpy** library

7. Implement the KMeans class method `classify(self,input)`. This method should assign the cluster to the given vector. To achieve this, first, calculate Euclidean distances between input vector and the means, and second, return the mean index closest to the input.

8. Once you have implemented all missing functionality, have a look at `train(self,vectors)` method where you call KMeans implemented functions. Then, you can have a look at `run_kmeans.py` to see how to use Kmeans in practice. Run the code with:

   ```
   python3 -m hw07_kmeans.run_kmeans
   ```