Homework 9:
Unsupervised Learning: K-means

Robert Litschko*
Symbolische Programmiersprache

Due: Thursday December 22, 2022, 12:00 (noon)

In this exercise you will:

- Implement k-means clustering

Exercise 1: Kmeans [6 points]

In GitLab, you can find the file `courses.txt`, containing several LMU courses of study. Take a look at `hw09_kmeans/kmeans.py`. In this exercise you will have to implement some methods to perform the clustering of these LMU courses.

This homework will be graded using unit tests by running:

```
python3 -m unittest -v hw09_kmeans/test_kmeans.py
```

Note: Some tests may require more than one of the methods to be implemented before they can pass.

1. Implement the Reader class method `get_lines(self)`. This method should return a list with the courses of the file courses.txt (in the order given in the file). **[1 point]**

2. Implement the Reader class method `normalize_word(self, word)`. This method should normalize the word by making it lower case and deleting punctuation marks from it. `Hint`: you can use string.punctuation to access a string of punctuation characters; (have a look at the constructor of the class). **[1 point]**

3. Implement the Reader class method `get_vocabulary(self)`. This method should return the vocabulary: the list of unique, normalized words from the file, sorted alphabetically. `Note:` words in the vocabulary should be normalized, use `normalized_word(self, word)` to do this. **[1 point]**

*Credit: Exercises are based on previous iterations from Katerina Kalouli.

4. Implement the KMeans class method `distance(self,x,y)`. This method should calculate the euclidean distance between two given vectors. **[1 point]**

5. Implement the KMeans class method `vector_mean(self,vectors)`. This method should calculate the mean of the vectors. `Hint`: you can use the off-the-shelf method of the **numpy** library to find the vectors' means. **[1 point]**

6. Implement the KMeans class method `classify(self,input)`. This method should assign the cluster to the given vector. To achieve this, first, calculate the euclidean distances between the input vector and the means, and second, return the mean index closest to the input. **[1 point]**

7. Once you have implemented all of the missing functionality, have a look at the `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 hw09_kmeans.run_kmeans`