IST 2018/2019

Processamento e Recuperação de Informação

Lab 03: Text Classification and Clustering

In the following exercises, we will use the *20 Newsgroup* dataset. This is a well known document collection, frequently used to evaluate text classifiers. You can find more information at `http://qwone.com/~jason/20Newsgroups/`.

To solve the proposed problems, we will use the *scikit-learn* machine learning library for Python[1]. The goal is to experiment with simple approaches for text classification and clustering, reusing the implementations from *scikit-learn*.

Whereas in previous lab classes you implemented mechanisms for reading textual documents and representing them according to the vector space model, in this case you will re-use existing methods for performing these operations. However, keep in mind that you should also be able to implement by yourself all the involved algorithms.

Background

The scikit-learn library already provides access to the 20 Newsgroups dataset. You can use the following code to obtain it:

```
from sklearn.datasets import fetch_20newsgroups
train = fetch_20newsgroups(subset='train')
test = fetch_20newsgroups(subset='test')
```

This already provides a standard split into training and test sets.

You can see the first 10 documents in the dataset through `train.data[:10]` and the classes of those documents through `train.target[:10]`. You will notice that the classes are represented as numbers. To see the class names you can use: `train.target_names`.

The actual data is in text format. You need to transform it into numeric weight vectors (i.e., according to the vector space model, and using binary, term frequency, or TF-IDF weights). The scikit-learn library provides methods for this:

```
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer = TfidfVectorizer( use_idf=False )
trainvec = vectorizer.fit_transform(train.data)
testvec = vectorizer.transform(test.data)
```

Once you do this to all data, you can fit a classifier on the training data and test it on the testing data.

[1] `http://scikit-learn.org/stable/`. All present code excerpts are adapted from the documentation.

```
from sklearn.naive_bayes import MultinomialNB
classifier = MultinomialNB()
classifier.fit(trainvec, train.target)
classes = classifier.predict(testvec)
```

The scikit-learn library also provides classes to evaluate classification results. The standard evaluation metrics, used in information retrieval and in text classification settings, will be detailed later in the course. The simplest of these metrics is perhaps the classification accuracy, which corresponds to the percentage of test instances that were assigned to the correct class:

```
from sklearn import metrics
print metrics.accuracy_score(test.target, classes)
print metrics.classification_report(test.target, classes)
```

Document clustering can be achieved similarly with the scikit-learn library, for instance by using the K-Means algorithm:

```
from sklearn.cluster import MiniBatchKMeans
cluster = MiniBatchKMeans(20)
cluster.fit(trainvec)
```

To check the clusters in which each document was placed, you can use `cluster.labels_`.

1

Implement a classifier for the 20 Newsgroups collection and measure its performance. Use the implementation of Multinomial Naïve Bayes classifiers, available from scikit-learn.

2

Try to improve the classification by:

(a) Removing stopwords;

(b) Removing very rare words (e.g. words that occur less than 3 times)

(c) Removing very frequent words (e.g. words that occur in more than 90% of the documents)[2]

(d) Use different classification algorithms, such as:

[2]Note that these text preprocessing operations can be done using the *Vectorizer* classes provided by scikit-learn.

- A nearest neighbour classifier (i.e., `sklearn.neighbors.KNeighborsClassifier`)
- The Perceptron algorithm (i.e., `sklearn.linear_model.Perceptron`)
- Support Vector Machines (i.e., `sklearn.svm.LinearSVC`)

Measure the performance of each of these variations.

3

Implement a clustering of the full 20 Newsgroup collection, using the implementation of K-Means clustering available from scikit-learn.

4 Pen and Paper Exercise

4.1

Consider the following six textual documents, each associated to one of three possible classes.

ID	Document	Class
D1	the movie is nothing but great	Positive
D2	mixed feelings about the movie	Neutral
D3	not so great	Negative
D4	great fantastic movie	Positive
D5	good movie overall	Positive
D6	overall the movie is terrible	Negative

- Estimate the parameters of a binary naïve Bayes model required for classifying the document

  great movie overall

  State which would be the most likely class for the given document, presenting all involved calculations.

  Use maximum likelihood estimation without considering any smoothing technique.

- Estimate the parameters of a Perceptron classifier based on the first 3 training instances, discriminating the positive instances from all other instances (i.e., the negative and the neutral). Start with an all-zero parameter vector, consider binary representations for the documents, and consider a single iteration over the training instances.

4.2

Consider the following collection of 4 text documents.

number	text document
1	shipment of gold damaged in fire
2	delivery of silver arrived in silver truck
3	shipment of silver arrived in truck
4	truck damaged in fire

Based on the *binary* document representations for the documents above, state which would be the result of applying *one iteration of the k-means algorithm* over the documents.

When simulating the execution of one iteration of the algorithm, consider $k = 2$, and consider that the two clusters are initialized with centroids equaling the representations for documents 2 and 3.

Two document representations $\mathbf{a}$ and $\mathbf{b}$ can be compared through the Manhattan distance, given by:

$$\text{dist}(\mathbf{a}, \mathbf{b}) = \|\mathbf{a} - \mathbf{b}\|_1 = \sum_{i=1}^{n} |a_i - b_i|$$