## Exercice 1

### 1a) Initial classifier

```
import nltk
import random
import numpy as np
import scipy as sp
import sklearn
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.linear_model import LogisticRegression

from nltk.corpus import movie_reviews

raw_movie_docs = [(movie_reviews.raw(fileid), category) for
                    category in movie_reviews.categories() for fileid
in
                        movie_reviews.fileids(category)]

random.seed(2000)
random.shuffle(raw_movie_docs)
movie_test = raw_movie_docs[:200]
movie_dev = raw_movie_docs[200:]

dev_test_data = movie_dev[:1600]
train_data = movie_dev[1600:]

train_texts = []
train_target = []

dev_test_texts = []
dev_test_target = []

for item in train_data:
    (reviews, labels) = item
    train_texts.append(reviews)
    train_target.append(labels)

for item in dev_test_data:
    (reviews, labels) = item
    dev_test_texts.append(reviews)
    dev_test_target.append(labels)


from sklearn.feature_extraction.text import CountVectorizer

v = CountVectorizer()
v.fit(train_texts)
```

```
CountVectorizer()

train_vectors = v.transform(train_texts)
dev_test_vectors = v.transform(dev_test_texts)

train_vectors

<200x14344 sparse matrix of type '<class 'numpy.int64'>'
     with 68224 stored elements in Compressed Sparse Row format>

clf = MultinomialNB()
clf.fit(train_vectors, train_target)

MultinomialNB()

dev_test_texts[14]
clf.predict(dev_test_vectors[14])

array(['pos'], dtype='<U3')

clf.predict(dev_test_vectors)

array(['neg', 'pos', 'pos', ..., 'neg', 'pos', 'pos'], dtype='<U3')

clf.score(dev_test_vectors, dev_test_target)

0.768125
```

### 1b) Parameters of the vectorizer

```
def calculateScore(binary,ngramRange):
    v = CountVectorizer(binary=binary,ngram_range=ngramRange)
    v.fit(train_texts)
    train_vectors = v.transform(train_texts)
    dev_test_vectors = v.transform(dev_test_texts)
    clf = MultinomialNB()
    clf.fit(train_vectors, train_target)
    return(clf.score(dev_test_vectors, dev_test_target))

calculateScore(False,(1,1))

0.768125

calculateScore(True,(1,1))

0.7625

calculateScore(False,(1,2))

0.734375

calculateScore(True,(1,2))

0.784375
```

```
calculateScore(False,(1,3))
```

```
0.7225
```

```
calculateScore(True,(1,3))
```

```
0.774375
```

## Exercice 2

### 2a)