Assignement 1

In [ ]:

```
import nltk
from nltk.corpus import webtext
nltk.download('webtext')
```

```
[nltk_data] Downloading package webtext to
[nltk_data]     /home/frederico/nltk_data...
[nltk_data]   Package webtext is already up-to-date!
```

Out[ ]:

```
True
```

In [ ]:

```
forum = webtext.raw('firefox.txt')
print(forum[:100])
```

```
Cookie Manager: "Don't allow sites that set removed cookies to set future
cookies" should stay check
```

In [ ]:

```
import re

http_urls = re.findall('http[s]?://(?:[a-zA-Z]|[0-9]|[$-_@.&+]|[!*\(\),]|(?:%[0-9a-fA-F][0-9a-fA-F]))+', forum)
www_urls = re.findall('www\.(?:[a-zA-Z]|[0-9]|[$-_@.&+]|[!*\(\),]|(?:%[0-9a-fA-F][0-9a-fA-F]))+', forum)
#regex = '(?:https?:\/\/)?(?:www\.)?[-a-zA-Z0-9@:%._\+~#=]{1,256}\.[a-zA-Z0-9()]{1,6}'
#urls = re.findall(regex, forum)
urls = http_urls + www_urls
shortcuts = re.findall('(?:ctrl|shift|alt)+\s*\+\s*[a-z+\-.,/]', forum, flags=re.IGNORECASE)
print(urls)
print(shortcuts)
```

```
['http://www.scripting.com/misc/msswitchad', 'http://www.watch.impress.co.
jp', 'http://bugzilla.mozilla.org', 'http://www.http://mozilla.org', 'htt
p://www.peterre.com', 'http://texturizer.net/firebird', 'http://foo', 'htt
p://http://', 'http://james', 'http://www.lexis.com', 'http://www.woolwort
h.de', 'http://ftp.mozilla.org/pub/mozilla.org/firebird/nightly', 'http://
http://', 'http://extensionroom.mozdev.org/more-info', 'http://www.odeon.c
o.uk/odeon', 'http://www.cctvusa.com', 'http://www.trenitalia.com/home/i
t', 'https://www.fortify.net', 'http://irc-galleria.net', 'http://http',
'http://www.mozilla.org/products', 'http://labs.google.com/cgi-bin', 'htt
p://www.timbressuisses.ch', 'https://www.eposasp.com/ebpp', 'www.scriptin
g.com/misc/msswitchad', 'www.foo.com', 'www.localhost.net.au', 'www.watch.
impress.co.jp', 'www.*.com', 'www.aol.com', 'www.php.net', 'www.fnac.fr',
'www.http://mozilla.org', 'www.hvv.de', 'www.petetownshend.co.uk', 'www.go
ogle.com', 'www.wamu.com)', 'www.excite.com', 'www.peterre.com', 'www.logi
tech.com', 'www.mozilla.org', 'www.xy.com', 'www.blogger.com', 'www.pcpits
top.com', 'www.mozilla.org', 'www.zoneedit.com', 'www.libpr0n.com', 'www.u
s.army.mil', 'www.linuxmail.org', 'www.debian.org', 'www.lexis.com', 'www.
lexis.com', 'www.m-w.com', 'www.woolworth.de', 'www.file.com', 'www.altern
ate.de)', 'www.microsoft.com', 'www.odeon.co.uk/odeon', 'www.cctvusa.com',
'www.mp3.de', 'www.domain', 'www.microsoft.com', 'www.trenitalia.com/home/
it', 'www.rmvplus.de', 'www.fortify.net', 'www.microsoft.com', "www.uboot.
com'", 'www.microsoft.com', 'www.mozilla.org/products', 'www.lycos.co.uk',
'www.calciomercato.com', 'www.odeon.co.uk/odeon', 'www.atozwebtools.com',
'www.X.com', 'www.timbressuisses.ch', 'www.vipernetworks.com', 'www.eposas
p.com/ebpp', 'www.yahoo.com', 'www.intellicast.com', 'www.w3c.org']
['Ctrl+M', 'ctrl+t', 'Ctrl+M', 'ALT+F', 'ctrl+d', 'Ctrl+C', 'ctrl+C', 'CTR
L+c', 'Ctrl+E', 'Ctrl+B', 'ctrl + e', 'Ctrl+E', 'CTRL+F', 'Ctrl+T', 'Ctrl+
S', 'Ctrl+S', 'Alt+H', 'ctrl+d', 'ctrl+t', 'Alt + D', 'shift+l', 'ctrl +
s', 'CTRL + m', 'ctrl+e', 'alt+e', 'ctrl+e', 'ctrl+e', 'Ctrl+-', 'Ctrl++',
'ctrl+e', 'ALT + L', 'Ctrl+L', 'Shift+C', 'ctrl+e', 'Ctrl+Q', 'Ctrl+W', 'a
lt+b', 'Ctrl+L', 'alt+f', 'Ctrl+L', 'alt+s', 'shift+s', 'shift+s', 'alt+
s', 'Ctrl+E', 'ALT+D', 'Alt+D', 'Alt+d', 'Ctrl+E', 'CTRL+E', 'Shift+G', 'c
trl+p', 'Alt+E', 'ctrl +\r\nA', 'ALT+d', 'Ctrl+S', 'Alt+C', 'ctrl++', 'Shi
ft +F', 'Alt+F', 'Alt+d', 'CTRL+K', 'Ctrl+W', 'Ctrl+W', 'Shift+E', 'Alt+
E', 'Ctrl+M', 'Ctrl+K', 'Ctrl+T', 'CTRL+E', 'alt+e', 'alt+e', 'Ctrl+W', 'C
trl + V', 'Shift + V', 'ctrl+T', 'Ctrl+P', 'Alt+E', 'Ctrl + u', 'Shift+C',
'Alt+E', 'Ctrl+T', 'CTRL+Y', 'CTRL+L', 'Ctrl+K', 'Ctrl+K', 'Alt+D', 'Ctrl
+ B', 'Alt + S', 'Ctrl+A', 'Ctrl+F', 'Ctrl+T', 'Ctrl+E', 'Ctrl+W', 'Ctrl+
t', 'Ctrl+x', 'Alt+f', 'Ctrl+ C', 'ctrl+K', 'ctrl+K', 'Alt+D', 'Ctrl+E',
'Ctrl+K', 'Ctrl+S', 'Ctrl+m', 'Ctrl+E', 'Ctrl+w', 'Shift+c', 'ctrl+p', 'Al
t+f', 'ctrl++', 'SHIFT + T', 'Alt+E', 'Ctrl+P', 'Ctrl+K', 'Shift+D', 'Alt+
D', 'shift+d', 'ctrl+f', 'Ctrl+W', 'ctrl+t', 'Alt+L', 'Ctrl+m', 'ALT+F',
'Ctrl+B', 'Ctrl+B', 'CTRL+B', 'CTRL+B', 'CTRL+I', 'Ctrl+B', 'Ctrl + K', 'C
trl+S', 'CTRL+P', 'Alt+f', 'Ctrl+R', 'CTRL+Y', 'CTRL+E', 'Ctrl+E', 'Ctrl+
E', 'Ctrl+E', 'CTRL+T', 'CTRL+A', 'Ctrl+T', 'Ctrl+S', 'Ctrl + B', 'Ctrl+
E', 'ctrl+e', 'Ctrl+F', 'Ctrl+F', 'CTRL + F', 'CTRL + F']
```

Assignement 2

In [ ]:

```
import nltk
from nltk.corpus import gutenberg
nltk.download('gutenberg')
```

```
[nltk_data] Downloading package gutenberg to
[nltk_data]     /home/frederico/nltk_data...
[nltk_data]   Package gutenberg is already up-to-date!
```

Out[ ]:

```
True
```

In [ ]:

```
hamlet = gutenberg.raw('shakespeare-hamlet.txt')
print(hamlet[:100])
```

```
[The Tragedie of Hamlet by William Shakespeare 1599]


Actus Primus. Scoena Prima.

Enter Barnardo a
```

3 classes tagger

In [ ]:

```
import nltk
from nltk.tag.stanford import StanfordNERTagger

PATH_TO_JAR = '/home/frederico/Desktop/DS/stanford-ner-2020-11-17/stanford-ner.jar'
PATH_TO_MODEL = '/home/frederico/Desktop/DS/stanford-ner-2020-11-17/classifiers/englis
h.all.3class.distsim.crf.ser.gz'
NER = StanfordNERTagger(model_filename=PATH_TO_MODEL,path_to_jar=PATH_TO_JAR, encoding
='utf-8')
```

In [ ]:

```
words = nltk.wordpunct_tokenize(hamlet)
tagged = NER.tag(words)
people = []

for (word,label) in tagged:
    if label == 'PERSON':
        people.append(word)
people = list(set(people))
print(people[:10])
print(len(people))
```

```
['Voltemand', 'Horatio', 'Sallets', 'Carbuncles', 'Controuersie', 'Vnckl
e', 'Scullion', 'Fox', 'Throate', 'Pesant']
498
```

4 classes tagger

In [ ]:

```
PATH_TO_JAR = '/home/frederico/Desktop/DS/stanford-ner-2020-11-17/stanford-ner.jar'
PATH_TO_MODEL = '/home/frederico/Desktop/DS/stanford-ner-2020-11-17/classifiers/englis
h.conll.4class.distsim.crf.ser.gz'
NER = StanfordNERTagger(model_filename=PATH_TO_MODEL,path_to_jar=PATH_TO_JAR, encoding
='utf-8')

tagged = NER.tag(words)
people = []

for (word,label) in tagged:
    if label == 'PERSON':
        people.append(word)
people = list(set(people))
print(people[:10])
print(len(people))
```

```
['Voltemand', 'Horatio', 'Sallets', 'Carbuncles', 'Controuersie', 'Vnckl
e', 'Scullion', 'Fox', 'Throate', 'Pesant']
498
```

7 classes tagger

In [ ]:

```
PATH_TO_JAR = '/home/frederico/Desktop/DS/stanford-ner-2020-11-17/stanford-ner.jar'
PATH_TO_MODEL = '/home/frederico/Desktop/DS/stanford-ner-2020-11-17/classifiers/englis
h.muc.7class.distsim.crf.ser.gz'
NER = StanfordNERTagger(model_filename=PATH_TO_MODEL,path_to_jar=PATH_TO_JAR, encoding
='utf-8')

tagged = NER.tag(words)
people = []

for (word,label) in tagged:
    if label == 'PERSON':
        people.append(word)
people = list(set(people))
print(people[:10])
print(len(people))
```

```
['leaue', 'Sonne', 'Horatio', 'Keepes', 'Barnardo', 'Gertrude', 'Business
e', 'Scullion', 'Leuies', 'Scul']
226
```

Assignement 3

In [ ]:

```
import nltk
from nltk.corpus import reuters
nltk.download('reuters')
```

```
[nltk_data] Downloading package reuters to
[nltk_data]     /home/frederico/nltk_data...
[nltk_data]   Package reuters is already up-to-date!
```

Out[ ]:

```
True
```

In [ ]:

```
rts = reuters.raw('test/14826')
print(rts[:100])
```

```
ASIAN EXPORTERS FEAR DAMAGE FROM U.S.-JAPAN RIFT
  Mounting trade friction between the
  U.S. And Ja
```

In [ ]:

```
print(reuters.categories())
```

```
['acq', 'alum', 'barley', 'bop', 'carcass', 'castor-oil', 'cocoa', 'coconu
t', 'coconut-oil', 'coffee', 'copper', 'copra-cake', 'corn', 'cotton', 'co
tton-oil', 'cpi', 'cpu', 'crude', 'dfl', 'dlr', 'dmk', 'earn', 'fuel', 'ga
s', 'gnp', 'gold', 'grain', 'groundnut', 'groundnut-oil', 'heat', 'hog',
'housing', 'income', 'instal-debt', 'interest', 'ipi', 'iron-steel', 'je
t', 'jobs', 'l-cattle', 'lead', 'lei', 'lin-oil', 'livestock', 'lumber',
'meal-feed', 'money-fx', 'money-supply', 'naphtha', 'nat-gas', 'nickel',
'nkr', 'nzdlr', 'oat', 'oilseed', 'orange', 'palladium', 'palm-oil', 'palm
kernel', 'pet-chem', 'platinum', 'potato', 'propane', 'rand', 'rape-oil',
'rapeseed', 'reserves', 'retail', 'rice', 'rubber', 'rye', 'ship', 'silve
r', 'sorghum', 'soy-meal', 'soy-oil', 'soybean', 'strategic-metal', 'suga
r', 'sun-meal', 'sun-oil', 'sunseed', 'tea', 'tin', 'trade', 'veg-oil', 'w
heat', 'wpi', 'yen', 'zinc']
```

In [ ]:

```
documents = [(list(reuters.words(fileid)), category)
             for category in reuters.categories()
             for fileid in reuters.fileids(category)]

# first 10 words of 1 document
print(documents[0][0][:10])
# category of 1 document
print(documents[0][1])

import random
random.shuffle(documents)
```

```
['SUMITOMO', 'BANK', 'AIMS', 'AT', 'QUICK', 'RECOVERY', 'FROM', 'MERGER',
'Sumitomo', 'Bank']
acq
```

Most common words among all words

In [ ]:

```
all_words = nltk.FreqDist(w.lower() for w in reuters.words())
# 5 most common words
print(all_words.most_common(5))
word_features = list(all_words)[:2000]

# check if the document has the most common words among it's words
def document_features(document):
    document_words = set(document)
    features = {}
    for word in word_features:
        features['contains({})'.format(word)] = (word in document_words)
    return features
```

```
[('.', 94687), (',', 72360), ('the', 69277), ('of', 36779), ('to', 36400)]
```

First extractor

In [ ]:

```
featuresets = [(document_features(d), c) for (d,c) in documents]
print(len(featuresets))
train_set, test_set = featuresets[5000:], featuresets[:200]
classifier = nltk.NaiveBayesClassifier.train(train_set)
```

In [ ]:

```
# example
print(classifier.classify(document_features(rts)))
print(reuters.categories('test/14826'))

classifier.show_most_informative_features(5)
```

```
gas
['trade']
Most Informative Features
          contains(palm) = True           palm-o : earn   =   1614.4 : 1.0
        contains(rubber) = True           rubber : earn   =   1433.8 : 1.0
          contains(zinc) = True             zinc : earn   =   1396.7 : 1.0
      contains(supplies) = True           propan : earn   =   1372.8 : 1.0
        contains(coffee) = True           coffee : earn   =   1355.1 : 1.0
```

In [ ]:

```
print(nltk.classify.accuracy(classifier, test_set))
```

```
0.54
```

Second extractor

Most common words among the document

In [ ]:

```
def alternative_document_features(document):
    document_words = nltk.FreqDist(w.lower() for w in document)
    word_features = list(document_words)[:2000]
    return dict([(word, True) for word in word_features])
```

In [ ]:

```
featuresets = [(alternative_document_features(d), c) for (d,c) in documents]
print(len(featuresets))
print(featuresets[0])
train_set, test_set = featuresets[5000:], featuresets[:200]
classifier = nltk.NaiveBayesClassifier.train(train_set)
```

```
13328
({'the': True, ',': True, 'stock': True, '.': True, '-': True, 'dividend':
True, 'on': True, 'hydraulic': True, 'said': True, 'split': True, 'of': Tr
ue, 'will': True, 'april': True, 'share': True, '3': True, 'its': True,
'a': True, 'common': True, 'quarterly': True, 'cash': True, 'to': True, 'b
e': True, 'cts': True, 'per': True, 'for': True, 'it': True, '50': True,
"'": True, 's': True, 'payable': True, 'stockholders': True, 'record': Tru
e, 'that': True, 'outstanding': True, 'company': True, '&': True, 'lt': Tr
ue, ';': True, 'thc': True, '>': True, 'splits': True, '2': True, 'hikes':
True, 'co': True, 'board': True, 'approved': True, 'three': True, 'two': T
rue, 'and': True, 'increased': True, 'occur': True, 'through': True, 'pc
t': True, 'distribution': True, '30': True, '15': True, 'is': True, 'pai
d': True, 'pre': True, 'shares': True, 'are': True, 'currently': True, '5
4': True, '75': True, 'up': True, 'from': True, '52': True, 'represent': T
rue, '36': True, 'after': True}, 'earn')
```

In [ ]:

```
# example
print(classifier.classify(alternative_document_features(rts)))
print(reuters.categories('test/14826'))

classifier.show_most_informative_features(5)
```

```
dfl
['trade']
Most Informative Features
                  coffee = True           coffee : earn   =   1622.3 : 1.0
                 propane = True           propan : earn   =   1468.8 : 1.0
                    oats = True              oat : earn   =   1428.0 : 1.0
               argentine = True           lin-oi : earn   =   1360.0 : 1.0
                minister = True            nzdlr : earn   =   1360.0 : 1.0
```

In [ ]:

```
print(nltk.classify.accuracy(classifier, test_set))
```

```
0.005
```

Discussion of results

Given the difference in results between both extractors, we can conclude that the first method of extracting information, that is, extracting the most common words across all texts and then check which of them are present for a given file, has a very supperior accuracy to the normal bag of words approach (54% accuracy vs 0.5% accuracy). One possible reason for this might be that having a basis of possible words (the most common across all files) allows for a easier training of the model and for an easier recognition of the class of the file since we can focus on a smaller and more focused set of words.