Feature Engineering

The next step is to create features from the raw text so we can train the machine learning models. The steps followed are:

1. **Text Cleaning and Preparation**: cleaning of special characters, downcasing, punctuation signs. possessive pronouns and stop words removal and lemmatization.
2. **Label coding**: creation of a dictionary to map each category to a code.
3. **Train-test split**: to test the models on unseen data.
4. **Text representation**: use of TF-IDF scores to represent text.

In [1]:
```
import pickle
import pandas as pd
import re
import nltk
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.feature_selection import chi2
import numpy as np
```

First of all we'll load the dataset:

In [2]:
```
path_df = "/home/lnc/0. Latest News Classifier/02. Exploratory Data Analysis/News

with open(path_df, 'rb') as data:
    df = pickle.load(data)
```

In [3]:
```
df.head()
```

Out[3]:

	File_Name	Content	Category	Complete_Filename	id	News_length
0	001.txt	Ad sales boost Time Warner profit\r\n\r\nQuart...	business	001.txt-business	1	2569
1	002.txt	Dollar gains on Greenspan speech\r\n\r\nThe do...	business	002.txt-business	1	2257
2	003.txt	Yukos unit buyer faces loan claim\r\n\r\nThe o...	business	003.txt-business	1	1557
3	004.txt	High fuel prices hit BA's profits\r\n\r\nBriti...	business	004.txt-business	1	2421
4	005.txt	Pernod takeover talk lifts Domecq\r\n\r\nShare...	business	005.txt-business	1	1575

And visualize one sample news content:

In [4]:
```python
df.loc[1]['Content']
```

Out[4]:
```
'Dollar gains on Greenspan speech\r\n\r\nThe dollar has hit its highest level a
gainst the euro in almost three months after the Federal Reserve head said the
US trade deficit is set to stabilise.\r\n\r\nAnd Alan Greenspan highlighted the
US government\'s willingness to curb spending and rising household savings as f
actors which may help to reduce it. In late trading in New York, the dollar rea
ched $1.2871 against the euro, from $1.2974 on Thursday. Market concerns about
the deficit has hit the greenback in recent months. On Friday, Federal Reserve
chairman Mr Greenspan\'s speech in London ahead of the meeting of G7 finance mi
nisters sent the dollar higher after it had earlier tumbled on the back of wors
e-than-expected US jobs data. "I think the chairman\'s taking a much more sangu
ine view on the current account deficit than he\'s taken for some time," said R
obert Sinche, head of currency strategy at Bank of America in New York. "He\'s
taking a longer-term view, laying out a set of conditions under which the curre
nt account deficit can improve this year and next."\r\n\r\nWorries about the de
ficit concerns about China do, however, remain. China\'s currency remains pegge
d to the dollar and the US currency\'s sharp falls in recent months have theref
ore made Chinese export prices highly competitive. But calls for a shift in Bei
jing\'s policy have fallen on deaf ears, despite recent comments in a major Chi
nese newspaper that the "time is ripe" for a loosening of the peg. The G7 meeti
ng is thought unlikely to produce any meaningful movement in Chinese policy. In
the meantime, the US Federal Reserve\'s decision on 2 February to boost interes
t rates by a quarter of a point - the sixth such move in as many months - has o
pened up a differential with European rates. The half-point window, some believ
e, could be enough to keep US assets looking more attractive, and could help pr
op up the dollar. The recent falls have partly been the result of big budget de
ficits, as well as the US\'s yawning current account gap, both of which need to
be funded by the buying of US bonds and assets by foreign firms and government
s. The White House will announce its budget on Monday, and many commentators be
lieve the deficit will remain at close to half a trillion dollars.'
```

1. Text cleaning and preparation

1.1. Special character cleaning

We can see the following special characters:

- `\r`
- `\n`
- `\` before possessive pronouns (`government's = government\'s`)
- `\` before possessive pronouns 2 (`Yukos' = Yukos\'`)
- `"` when quoting text

In [5]:
```python
# \r and \n
df['Content_Parsed_1'] = df['Content'].str.replace("\r", " ")
df['Content_Parsed_1'] = df['Content_Parsed_1'].str.replace("\n", " ")
df['Content_Parsed_1'] = df['Content_Parsed_1'].str.replace("    ", " ")
```

Regarding 3rd and 4th bullet, although it seems there is a special character, it won't affect us since it is not a *real* character:

In [6]:
```
text = "Mr Greenspan\'s"
text
```

Out[6]:
```
"Mr Greenspan's"
```

In [7]:
```
# " when quoting text
df['Content_Parsed_1'] = df['Content_Parsed_1'].str.replace('"', '')
```

1.2. Upcase/downcase

We'll downcase the texts because we want, for example, `Football` and `football` to be the same word.

In [8]:
```
# Lowercasing the text
df['Content_Parsed_2'] = df['Content_Parsed_1'].str.lower()
```

1.3. Punctuation signs

Punctuation signs won't have any predicting power, so we'll just get rid of them.

In [9]:
```
punctuation_signs = list("?:!.,;")
df['Content_Parsed_3'] = df['Content_Parsed_2']

for punct_sign in punctuation_signs:
    df['Content_Parsed_3'] = df['Content_Parsed_3'].str.replace(punct_sign, '')
```

By doing this we are messing up with some numbers, but it's no problem since we aren't expecting any predicting power from them.

1.4. Possessive pronouns

We'll also remove possessive pronoun terminations:

In [10]:
```
df['Content_Parsed_4'] = df['Content_Parsed_3'].str.replace("'s", "")
```

1.5. Stemming and Lemmatization

Since stemming can produce output words that don't exist, we'll only use a lemmatization process at this moment. Lemmatization takes into consideration the morphological analysis of the words and returns words that do exist, so it will be more useful for us.

In [2]:
```python
# Downloading punkt and wordnet from NLTK
nltk.download('punkt')
print("------------------------------------------------------------")
nltk.download('wordnet')
```

```
[nltk_data] Downloading package punkt to /home/lnc/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.

------------------------------------------------------------

[nltk_data] Downloading package wordnet to /home/lnc/nltk_data...
[nltk_data]   Unzipping corpora/wordnet.zip.
```

Out[2]:
```
True
```

In [12]:
```python
# Saving the lemmatizer into an object
wordnet_lemmatizer = WordNetLemmatizer()
```

In order to lemmatize, we have to iterate through every word:

In [13]:
```python
nrows = len(df)
lemmatized_text_list = []

for row in range(0, nrows):

    # Create an empty list containing lemmatized words
    lemmatized_list = []

    # Save the text and its words into an object
    text = df.loc[row]['Content_Parsed_4']
    text_words = text.split(" ")

    # Iterate through every word to lemmatize
    for word in text_words:
        lemmatized_list.append(wordnet_lemmatizer.lemmatize(word, pos="v"))

    # Join the list
    lemmatized_text = " ".join(lemmatized_list)

    # Append to the list containing the texts
    lemmatized_text_list.append(lemmatized_text)
```

In [14]:
```python
df['Content_Parsed_5'] = lemmatized_text_list
```

Although lemmatization doesn't work perfectly in all cases (as can be seen in the example below), it can be useful.

1.6. Stop words

In [3]:
```python
# Downloading the stop words list
nltk.download('stopwords')
```

```
[nltk_data] Downloading package stopwords to /home/lnc/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
```

Out[3]:
```
True
```

In [16]:
```python
# Loading the stop words in english
stop_words = list(stopwords.words('english'))
```

In [17]:
```python
stop_words[0:10]
```

Out[17]:
```
['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're"]
```

To remove the stop words, we'll handle a regular expression only detecting whole words, as seen in the following example:

In [18]:
```python
example = "me eating a meal"
word = "me"

# The regular expression is:
regex = r"\b" + word + r"\b"  # we need to build it like that to work properly

re.sub(regex, "StopWord", example)
```

Out[18]:
```
'StopWord eating a meal'
```

We can now loop through all the stop words:

In [19]:
```python
df['Content_Parsed_6'] = df['Content_Parsed_5']

for stop_word in stop_words:

    regex_stopword = r"\b" + stop_word + r"\b"
    df['Content_Parsed_6'] = df['Content_Parsed_6'].str.replace(regex_stopword, '
```

We have some dobule/triple spaces between words because of the replacements. However, it's not a problem because we'll tokenize by the spaces later.

As an example, we'll show an original news article and its modifications throughout the process:

In [20]:
```python
df.loc[5]['Content']
```

Out[20]:
```
'Japan narrowly escapes recession\r\n\r\nJapan\'s economy teetered on the brink
of a technical recession in the three months to September, figures show.\r\n\r
\nRevised figures indicated growth of just 0.1% - and a similar-sized contracti
on in the previous quarter. On an annual basis, the data suggests annual growth
of just 0.2%, suggesting a much more hesitant recovery than had previously been
thought. A common technical definition of a recession is two successive quarter
s of negative growth.\r\n\r\nThe government was keen to play down the worrying
implications of the data. "I maintain the view that Japan\'s economy remains in
a minor adjustment phase in an upward climb, and we will monitor developments c
arefully," said economy minister Heizo Takenaka. But in the face of the strengt
hening yen making exports less competitive and indications of weakening economi
c conditions ahead, observers were less sanguine. "It\'s painting a picture of
a recovery... much patchier than previously thought," said Paul Sheard, economi
st at Lehman Brothers in Tokyo. Improvements in the job market apparently have
yet to feed through to domestic demand, with private consumption up just 0.2% i
n the third quarter.'
```

1. Special character cleaning

In [21]:
```python
df.loc[5]['Content_Parsed_1']
```

Out[21]:
```
"Japan narrowly escapes recession Japan's economy teetered on the brink of a te
chnical recession in the three months to September, figures show. Revised figur
es indicated growth of just 0.1% - and a similar-sized contraction in the previ
ous quarter. On an annual basis, the data suggests annual growth of just 0.2%,
suggesting a much more hesitant recovery than had previously been thought. A co
mmon technical definition of a recession is two successive quarters of negative
growth. The government was keen to play down the worrying implications of the d
ata. I maintain the view that Japan's economy remains in a minor adjustment pha
se in an upward climb, and we will monitor developments carefully, said economy
minister Heizo Takenaka. But in the face of the strengthening yen making export
s less competitive and indications of weakening economic conditions ahead, obse
rvers were less sanguine. It's painting a picture of a recovery... much patchie
r than previously thought, said Paul Sheard, economist at Lehman Brothers in To
kyo. Improvements in the job market apparently have yet to feed through to dome
stic demand, with private consumption up just 0.2% in the third quarter."
```

2. Upcase/downcase

In [22]:
```
df.loc[5]['Content_Parsed_2']
```

Out[22]:
```
"japan narrowly escapes recession japan's economy teetered on the brink of a te
chnical recession in the three months to september, figures show. revised figur
es indicated growth of just 0.1% - and a similar-sized contraction in the previ
ous quarter. on an annual basis, the data suggests annual growth of just 0.2%,
suggesting a much more hesitant recovery than had previously been thought. a co
mmon technical definition of a recession is two successive quarters of negative
growth. the government was keen to play down the worrying implications of the d
ata. i maintain the view that japan's economy remains in a minor adjustment pha
se in an upward climb, and we will monitor developments carefully, said economy
minister heizo takenaka. but in the face of the strengthening yen making export
s less competitive and indications of weakening economic conditions ahead, obse
rvers were less sanguine. it's painting a picture of a recovery... much patchie
r than previously thought, said paul sheard, economist at lehman brothers in to
kyo. improvements in the job market apparently have yet to feed through to dome
stic demand, with private consumption up just 0.2% in the third quarter."
```

3. Punctuation signs

In [23]:
```
df.loc[5]['Content_Parsed_3']
```

Out[23]:
```
"japan narrowly escapes recession japan's economy teetered on the brink of a te
chnical recession in the three months to september figures show revised figures
indicated growth of just 01% - and a similar-sized contraction in the previous
quarter on an annual basis the data suggests annual growth of just 02% suggesti
ng a much more hesitant recovery than had previously been thought a common tech
nical definition of a recession is two successive quarters of negative growth t
he government was keen to play down the worrying implications of the data i mai
ntain the view that japan's economy remains in a minor adjustment phase in an u
pward climb and we will monitor developments carefully said economy minister he
izo takenaka but in the face of the strengthening yen making exports less compe
titive and indications of weakening economic conditions ahead observers were le
ss sanguine it's painting a picture of a recovery much patchier than previously
thought said paul sheard economist at lehman brothers in tokyo improvements in
the job market apparently have yet to feed through to domestic demand with priv
ate consumption up just 02% in the third quarter"
```

4. Possessive pronouns

In [24]:
```
df.loc[5]['Content_Parsed_4']
```

Out[24]:
```
'japan narrowly escapes recession japan economy teetered on the brink of a tech
nical recession in the three months to september figures show revised figures i
ndicated growth of just 01% - and a similar-sized contraction in the previous q
uarter on an annual basis the data suggests annual growth of just 02% suggestin
g a much more hesitant recovery than had previously been thought a common techn
ical definition of a recession is two successive quarters of negative growth th
e government was keen to play down the worrying implications of the data i main
tain the view that japan economy remains in a minor adjustment phase in an upwa
rd climb and we will monitor developments carefully said economy minister heizo
takenaka but in the face of the strengthening yen making exports less competiti
ve and indications of weakening economic conditions ahead observers were less s
anguine it painting a picture of a recovery much patchier than previously thoug
ht said paul sheard economist at lehman brothers in tokyo improvements in the j
ob market apparently have yet to feed through to domestic demand with private c
onsumption up just 02% in the third quarter'
```

5. Stemming and Lemmatization

In [25]:
```
df.loc[5]['Content_Parsed_5']
```

Out[25]:
```
'japan narrowly escape recession japan economy teeter on the brink of a technic
al recession in the three months to september figure show revise figure indicat
e growth of just 01% - and a similar-sized contraction in the previous quarter
on an annual basis the data suggest annual growth of just 02% suggest a much mo
re hesitant recovery than have previously be think a common technical definitio
n of a recession be two successive quarter of negative growth the government be
keen to play down the worry implications of the data i maintain the view that j
apan economy remain in a minor adjustment phase in an upward climb and we will
monitor developments carefully say economy minister heizo takenaka but in the f
ace of the strengthen yen make export less competitive and indications of weake
n economic condition ahead observers be less sanguine it paint a picture of a r
ecovery much patchier than previously think say paul sheard economist at lehman
brothers in tokyo improvements in the job market apparently have yet to fee thr
ough to domestic demand with private consumption up just 02% in the third quart
er'
```

6. Stop words

```
In [26]: df.loc[5]['Content_Parsed_6']
```

```
Out[26]: 'japan narrowly escape recession japan economy teeter   brink   technical reces
sion   three months  september figure show revise figure indicate growth   01%
-  similar-sized contraction   previous quarter   annual basis  data suggest a
nnual growth   02% suggest  much  hesitant recovery   previously  think  common
technical definition   recession  two successive quarter  negative growth  gove
rnment  keen  play   worry implications   data  maintain  view  japan economy r
emain   minor adjustment phase   upward climb    monitor developments carefully
say economy minister heizo takenaka    face    strengthen yen make export less c
ompetitive  indications  weaken economic condition ahead observers  less sangui
ne  paint  picture   recovery much patchier  previously think say paul sheard e
conomist  lehman brothers  tokyo improvements   job market apparently  yet  fee
domestic demand  private consumption   02%   third quarter'
```

Finally, we can delete the intermediate columns:

```
In [27]: df.head(1)
```

Out[27]:

	File_Name	Content	Category	Complete_Filename	id	News_length	Content_Parsed_1
0	001.txt	Ad sales boost Time Warner profit\r\n\r\nQuart...	business	001.txt-business	1	2569	Ad sales boost Time Warner profit Quarterly pr...

```
In [28]: list_columns = ["File_Name", "Category", "Complete_Filename", "Content", "Content
         df = df[list_columns]

         df = df.rename(columns={'Content_Parsed_6': 'Content_Parsed'})
```

```
In [29]: df.head()
```

Out[29]:

	File_Name	Category	Complete_Filename	Content	Content_Parsed
0	001.txt	business	001.txt-business	Ad sales boost Time Warner profit\r\n\r\nQuart...	ad sales boost time warner profit quarterly pr...
1	002.txt	business	002.txt-business	Dollar gains on Greenspan speech\r\n\r\nThe do...	dollar gain greenspan speech dollar hit hi...
2	003.txt	business	003.txt-business	Yukos unit buyer faces loan claim\r\n\r\nThe o...	yukos unit buyer face loan claim owners emba...
3	004.txt	business	004.txt-business	High fuel prices hit BA's profits\r\n\r\nBriti...	high fuel price hit ba profit british airways ...
4	005.txt	business	005.txt-business	Pernod takeover talk lifts Domecq\r\n\r\nShare...	pernod takeover talk lift domecq share uk dri...

IMPORTANT:

We need to remember that our model will gather the latest news articles from different newspapers every time we want. For that reason, we not only need to take into account the peculiarities of the training set articles, but also possible ones that are present in the gathered news articles.

For this reason, possible peculiarities have been studied in the *05. News Scraping* folder.

2. Label coding

We'll create a dictionary with the label codification:

In [30]:
```
category_codes = {
    'business': 0,
    'entertainment': 1,
    'politics': 2,
    'sport': 3,
    'tech': 4
}
```

In [31]:
```
# Category mapping
df['Category_Code'] = df['Category']
df = df.replace({'Category_Code':category_codes})
```

In [32]:
```
df.head()
```

Out[32]:

	File_Name	Category	Complete_Filename	Content	Content_Parsed	Category_Code
0	001.txt	business	001.txt-business	Ad sales boost Time Warner profit\r\n\r\nQuart...	ad sales boost time warner profit quarterly pr...	0
1	002.txt	business	002.txt-business	Dollar gains on Greenspan speech\r\n\r\nThe do...	dollar gain greenspan speech dollar hit hi...	0
2	003.txt	business	003.txt-business	Yukos unit buyer faces loan claim\r\n\r\nThe o...	yukos unit buyer face loan claim owners emba...	0
3	004.txt	business	004.txt-business	High fuel prices hit BA's profits\r\n\r\nBriti...	high fuel price hit ba profit british airways ...	0
4	005.txt	business	005.txt-business	Pernod takeover talk lifts Domecq\r\n\r\nShare...	pernod takeover talk lift domecq share uk dri...	0

3. Train - test split

We'll set apart a test set to prove the quality of our models. We'll do Cross Validation in the train set in order to tune the hyperparameters and then test performance on the unseen data of the test set.

In [33]:
```
X_train, X_test, y_train, y_test = train_test_split(df['Content_Parsed'],
                                                    df['Category_Code'],
                                                    test_size=0.15,
                                                    random_state=8)
```

Since we don't have much observations (only 2.225), we'll choose a test set size of 15% of the full dataset.

4. Text representation

We have various options:

- Count Vectors as features
- TF-IDF Vectors as features
- Word Embeddings as features
- Text / NLP based features
- Topic Models as features

We'll use **TF-IDF Vectors** as features.

We have to define the different parameters:

- `ngram_range` : We want to consider both unigrams and bigrams.
- `max_df` : When building the vocabulary ignore terms that have a document frequency strictly higher than the given threshold
- `min_df` : When building the vocabulary ignore terms that have a document frequency strictly lower than the given threshold.
- `max_features` : If not None, build a vocabulary that only consider the top max_features ordered by term frequency across the corpus.

See `TfidfVectorizer?` for further detail.

It needs to be mentioned that we are implicitly scaling our data when representing it as TF-IDF features with the argument `norm` .

In [34]:
```
# Parameter election
ngram_range = (1,2)
min_df = 10
max_df = 1.
max_features = 300
```

We have chosen these values as a first approximation. Since the models that we develop later have a very good predictive power, we'll stick to these values. But it has to be mentioned that different combinations could be tried in order to improve even more the accuracy of the models.

In [35]:
```
tfidf = TfidfVectorizer(encoding='utf-8',
                        ngram_range=ngram_range,
                        stop_words=None,
                        lowercase=False,
                        max_df=max_df,
                        min_df=min_df,
                        max_features=max_features,
                        norm='l2',
                        sublinear_tf=True)

features_train = tfidf.fit_transform(X_train).toarray()
labels_train = y_train
print(features_train.shape)

features_test = tfidf.transform(X_test).toarray()
labels_test = y_test
print(features_test.shape)
```

```
(1891, 300)
(334, 300)
```

Please note that we have fitted and then transformed the training set, but we have **only transformed** the **test set**.

We can use the Chi squared test in order to see what unigrams and bigrams are most correlated with each category:

In [36]:
```python
from sklearn.feature_selection import chi2
import numpy as np

for Product, category_id in sorted(category_codes.items()):
    features_chi2 = chi2(features_train, labels_train == category_id)
    indices = np.argsort(features_chi2[0])
    feature_names = np.array(tfidf.get_feature_names())[indices]
    unigrams = [v for v in feature_names if len(v.split(' ')) == 1]
    bigrams = [v for v in feature_names if len(v.split(' ')) == 2]
    print("# '{}' category:".format(Product))
    print("  . Most correlated unigrams:\n. {}".format('\n. '.join(unigrams[-5:])
    print("  . Most correlated bigrams:\n. {}".format('\n. '.join(bigrams[-2:])))
    print("")
```

```
# 'business' category:
  . Most correlated unigrams:
. market
. price
. economy
. growth
. bank
  . Most correlated bigrams:
. last year
. year old

# 'entertainment' category:
  . Most correlated unigrams:
. tv
. music
. star
. award
. film
  . Most correlated bigrams:
. mr blair
. prime minister

# 'politics' category:
  . Most correlated unigrams:
. minister
. blair
. party
. election
. labour
  . Most correlated bigrams:
. prime minister
. mr blair

# 'sport' category:
  . Most correlated unigrams:
. win
. side
. game
. team
. match
  . Most correlated bigrams:
. say mr
```

```
. year old

# 'tech' category:
  . Most correlated unigrams:
. digital
. technology
. computer
. software
. users
  . Most correlated bigrams:
. year old
. say mr
```

As we can see, the unigrams correspond well to their category. However, bigrams do not. If we get the bigrams in our features:

In [37]:
```
bigrams
```

Out[37]:
```
['tell bbc', 'last year', 'prime minister', 'mr blair', 'year old', 'say mr']
```

We can see there are only six. This means the unigrams have more correlation with the category than the bigrams, and since we're restricting the number of features to the most representative 300, only a few bigrams are being considered.

Let's save the files we'll need in the next steps:

In [38]:
```python
# X_train
with open('Pickles/X_train.pickle', 'wb') as output:
    pickle.dump(X_train, output)

# X_test
with open('Pickles/X_test.pickle', 'wb') as output:
    pickle.dump(X_test, output)

# y_train
with open('Pickles/y_train.pickle', 'wb') as output:
    pickle.dump(y_train, output)

# y_test
with open('Pickles/y_test.pickle', 'wb') as output:
    pickle.dump(y_test, output)

# df
with open('Pickles/df.pickle', 'wb') as output:
    pickle.dump(df, output)

# features_train
with open('Pickles/features_train.pickle', 'wb') as output:
    pickle.dump(features_train, output)

# labels_train
with open('Pickles/labels_train.pickle', 'wb') as output:
    pickle.dump(labels_train, output)

# features_test
with open('Pickles/features_test.pickle', 'wb') as output:
    pickle.dump(features_test, output)

# labels_test
with open('Pickles/labels_test.pickle', 'wb') as output:
    pickle.dump(labels_test, output)

# TF-IDF object
with open('Pickles/tfidf.pickle', 'wb') as output:
    pickle.dump(tfidf, output)
```