Natural Language processing

TP 2 – Bag of words (BOW)

LE Do Thanh Dat, YOU Borachhun

Exercice 1 - BOW

In [1]:
```
#1. Importer les dépendances
import pandas as pd
import nltk
import numpy as np
from nltk.corpus import stopwords
from nltk.tokenize import sent_tokenize as st
from nltk.stem import WordNetLemmatizer as wordnet
import re
```

In [5]:
```
# 2. Importer les données et déclarer quelques variables
## redaing the file
df = pd.read_csv('./data/spam.csv', encoding='ISO-8859-1', usecols=['v1','v2'])
corpus = [] #empty list
wordnet = wordnet() #object instantiation
length = len(df['v2']) #finding total number of rows

df.head(10)
```

Out[5]:

	v1	v2
0	ham	Go until jurong point, crazy.. Available only ...
1	ham	Ok lar... Joking wif u oni...
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...
3	ham	U dun say so early hor... U c already then say...
4	ham	Nah I don't think he goes to usf, he lives aro...
5	spam	FreeMsg Hey there darling it's been 3 week's n...
6	ham	Even my brother is not like to speak with me. ...
7	ham	As per your request 'Melle Melle (Oru Minnamin...
8	spam	WINNER!! As a valued network customer you have...
9	spam	Had your mobile 11 months or more? U R entitle...

In [7]:
```
# 3. Prétraitement des données
for i in range(length):
    #substitute characters at the beginning of the phrase
    rev = re.sub('[^a-zA-z]',' ',df['v2'][i])

    #text to lowercase
    rev = rev.lower()

    #each word of the sentence becomes the element of a list
```

```
    rev = rev.split()

    #lemmatization via list comprehension
    rev = [wordnet.lemmatize(word) for word in rev if word not in stopwords.word

    #from list to string
    rev = ' '.join(rev)

    # from list to string
    corpus.append(rev) #appending to the list
```

In [15]:
```
# 4. Implémenter BOW, avec sklearn
from sklearn.feature_extraction.text import CountVectorizer

#to take max features(columns), 2500
cv = CountVectorizer(max_features=2500)

#converting to array
x = cv.fit_transform(corpus).toarray()

#dependent variable
y = df['v1']
```

Ce code ci-dessus effectue la vectorisation du texte en utilisant l'approche Bag of Words avec un maximum de 2500 caractéristiques (mots) sélectionnées. Les données vectorisées sont stockées dans la variable x, et les étiquettes associées sont stockées dans la variable y.

Le paramètre ***max_features = 2500*** signifie que le vecteur résultant ne contiendra que les 2500 caractéristiques les plus fréquentes dans le corpus. Cela signifie que seules les 2500 caractéristiques les plus fréquentes seront utilisées pour représenter les documents.

L'avantage d'avoir une valeur plus grande pour ***max_features*** est que cela permet de conserver plus de mots dans la représentation vectorielle. Cela peut être bénéfique si le corpus contient un vocabulaire riche et varié, et si les mots moins fréquents contiennent également des informations importantes pour la tâche de classification ou d'analyse ultérieure. Une valeur plus grande de max_features peut potentiellement capturer plus de spécificités dans les documents.

D'un autre côté, il y a des avantages à avoir une valeur plus petite pour ***max_features***. En limitant le nombre de caractéristiques, on réduit la dimension de la représentation vectorielle, ce qui peut être bénéfique en réduction de la complexité computationnelle et de la consommation de mémoire. De plus, en éliminant les mots moins fréquents, on peut réduire le bruit et la variabilité dans les données.

In [16]:
```
# 5. Transformation de la variable cible
## y is a categorical variable so will encode it
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
y = le.fit_transform(y)
```

Exercice 2 - Modélisation

In [17]:
```
# 1. Définition de la base d'apprentissage et test
## split into train and test set
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x,y,test_size=0.2)
```

In [19]:
```
# 2. Apprentissage d'un modèle de classification
## training the model
from sklearn.naive_bayes import MultinomialNB
model = MultinomialNB() # using naive bayes classification algorithm
model.fit(x_train, y_train) #fitting the model
```

Out[19]:
```
▾ MultinomialNB

MultinomialNB()
```

Il existe d'autres modèles que on peut utiliser tels que Régression logistique, Machines à vecteurs de support (SVM), Arbres de décision (Decision Trees), Forêts aléatoires (Random Forests), Réseaux de neurones (Neural Networks).

In [20]:
```
# 3. Prédiction et calcul des performances

## predicting the values
y_pred = model.predict(x_test)

#score of the model
model.score(x_test, y_test)

from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
```

In [21]:
```
# 4. Afficher la performance
from sklearn.metrics import accuracy_score
accuracy_score(y_test, y_pred)
```

Out[21]:
```
0.9811659192825112
```