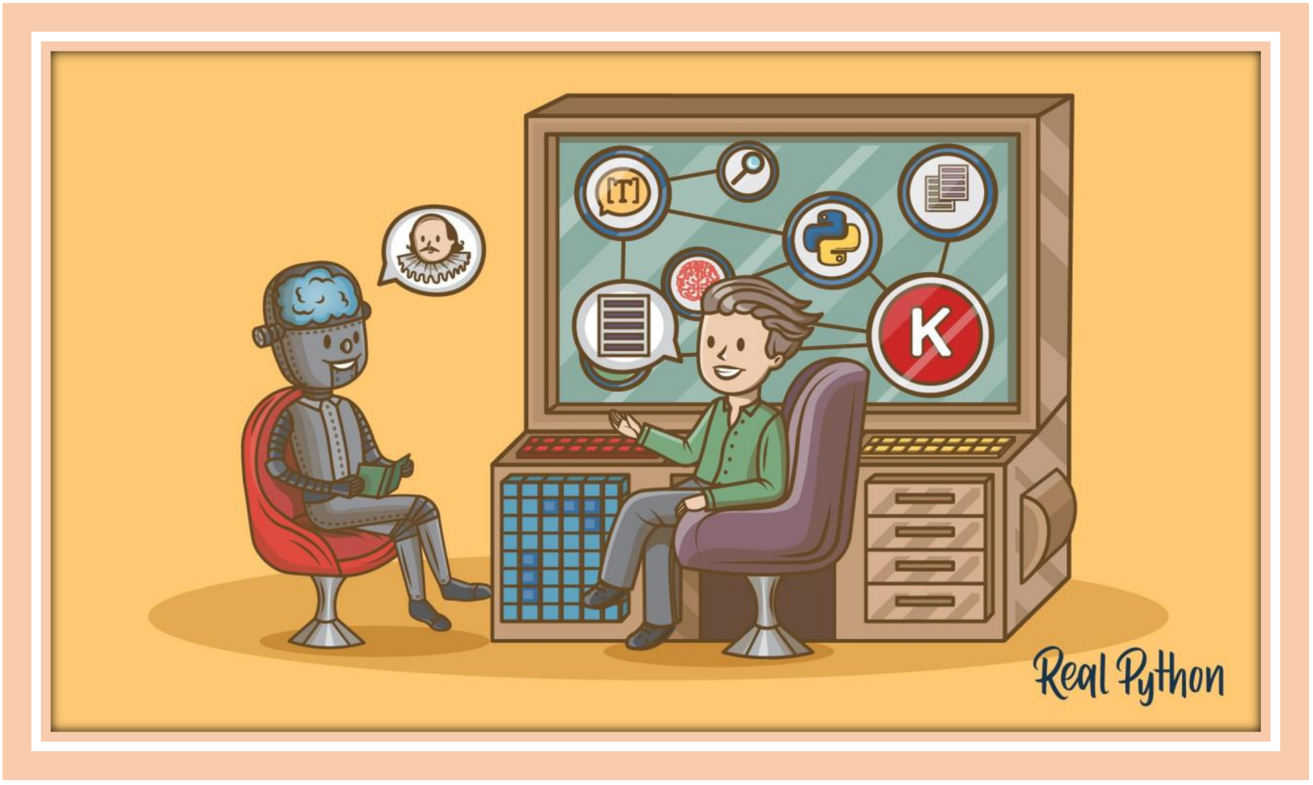
Alexis Ribat
BIA-1

Project
From knowledge extraction to knowledge representation



Abstract

In this project, we will see how to use machine learning in order to extract knowledge from a set of documents using Python and its libraries.

Text classification is the way of assigning a set of predefined data to open-ended. It is used to organize, structure and categorize any kind of text from documents (media, medical studies etc…).

With the growth of the industry, there are more and more data all over the word than can be useful.

Here, we will see (one way) how to extract data from row files and make it readable to the machine.

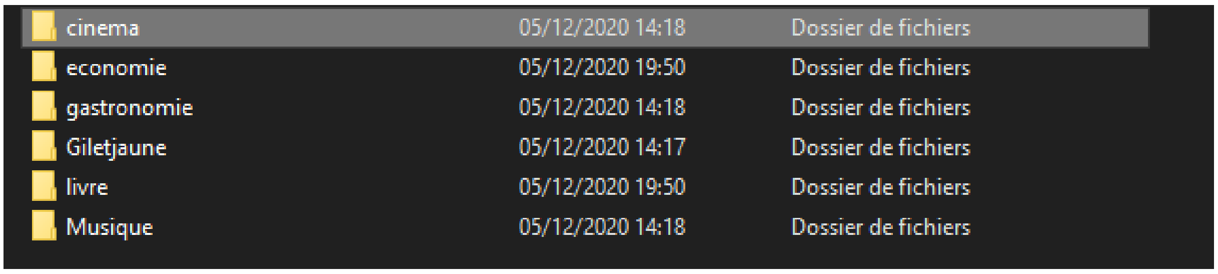
Many algorithms exist in order to extract knowledge from data: Naïve Bayes, SVM Model, Deep Neural Networks, K-nearest Neighbor, etc…

I don’t know (yet) all of that models and how they work, but I’m interested to know more about it. It is also interesting knowing that all of this exists to my futures projects.

Now, let see how I managed to do this project.

Introduction

I chose a dataset of 30 texts divided in 6 themes with 5 texts from articles of each themes: “Ecologie”,” Cinema”,” Giletjaune”, “Gastronomie”,” Livre” & “Musique”.

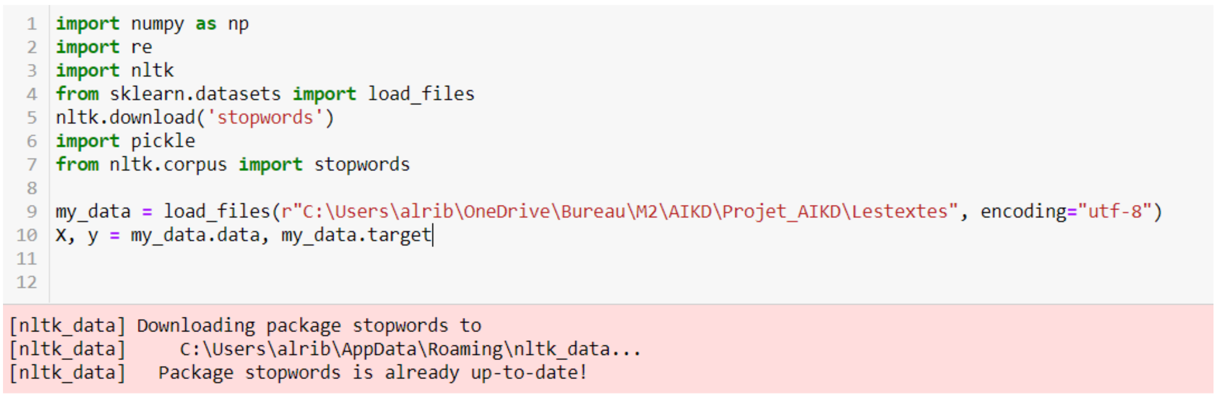


cinema	05/12/2020 14:18	Dossier de fichiers
economie	05/12/2020 19:50	Dossier de fichiers
gastronomie	05/12/2020 14:18	Dossier de fichiers
Giletjaune	05/12/2020 14:17	Dossier de fichiers
livre	05/12/2020 19:50	Dossier de fichiers
Musique	05/12/2020 14:18	Dossier de fichiers

In each document, there are 5 articles (.txt). I had to manage to import them in order to extract knowledge.

Approach

The next step after choosing our dataset is to import them. I used Jupyter Notebook to this part. We can easily import datas with Sklearn :



```
import numpy as np
import re
import nltk
from sklearn.datasets import load_files
nltk.download('stopwords')
import pickle
from nltk.corpus import stopwords

my_data = load_files(r"C:\Users\alrib\OneDrive\Bureau\M2\AIKD\Projet_AIKD\Lestextes", encoding="utf-8")
X, y = my_data.data, my_data.target


```

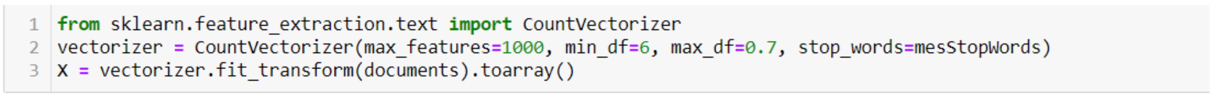
```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\alrib\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

Here, X is the list of all words in all text files in our dataset, stored in y. y is a Numpy array of size 30 (our 30 texts).

The next step is to preprocess the text. In order to do that, we can use the function of the “Re” library to clean the data (extract symbols, lower case, stopwords, etc…).

Now, we have a clean dataset with words than can be used to extract knowledge. But, since Machines cannot understand texts, we must convert words into numbers.

I chose the *Bag od Words* method in order to do that.



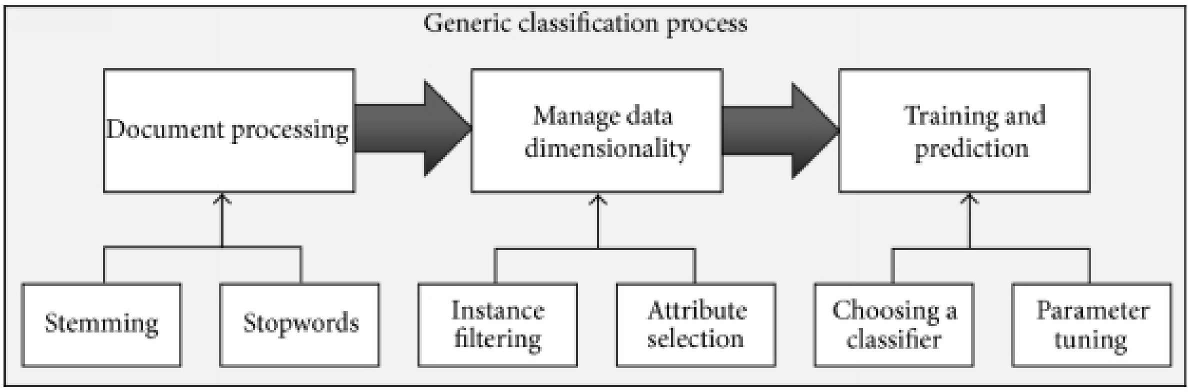
```
from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer(max_features=1000, min_df=6, max_df=0.7, stop_words=mesStopWords)
X = vectorizer.fit_transform(documents).toarray()
```

I used that function CountVectorizer, that contain import parameters like:

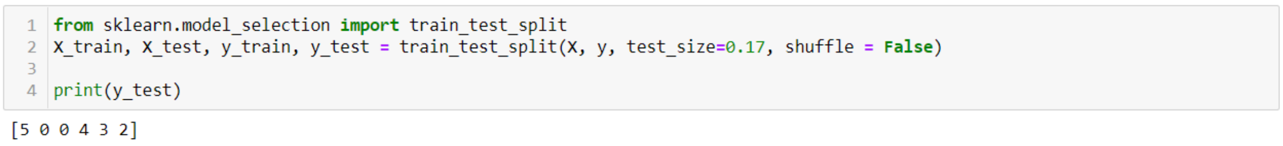
- Max_features: it is used to determine the words that appears most of the time in our dataset
- Min_df: it represents the minimum number of documents that should contain this feature.

- Max_df: it represents the maximum number of documents that should contain this feature in percentage. Here, we want only those words that occur in a maximum of 70% of our dataset.
- Stop_words: It is useful to clean the data and to remove the words that are not relevant (le, la, un, une etc…)

Then, the function fit_transorm converts the text documents into numeric features.



Then, we will train the model. To do so, we will use the `train_test_split` utility from the `sklearn.model_selection` library.



```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.17, shuffle = False)

print(y_test)
```

```
[5 0 0 4 3 2]
```

Train_X → Training Data Predictors
Train_Y → Training Data Target
Test_X → Test Data Predictors
Test_Y → Test Data Target

This script divides data into 17% test set (0.17*30 = 5 texts) and 83% training set.
So, we have 5 texts for the tests and 25 for the training.

Now we can use the model we want to train and test our dataset. I chose to use the Random Forest Algorithm to train my model.

The Random Forest Classifier has different parameters like:

- N_estimators: calculus the number of trees in the forest
- Random_state: Useful to the randomness of the boostrapping of the samples used when building trees.

Conclusion

I learned a lot of things, like using Python libraries and extracting knowledge from texts. I feel like a have a lot to learn, but now I know where to go and what to search if I must do this in the future.

I liked doing this project and if I had more time, I would have test other classification algorithms to compare them.

If I did have to re-do this project, I would better choose English texts than French texts.

In English, there are more interesting function to use, like Lemmetization, which is used to eliminate redundant prefix or suffix of a word and extract the base word. For example, the words ‘cats’ and ‘cat’ could be the same word ‘cat’. I did not find an equivalent for French words yet.

I continue to learn more and I know this will be useful in the future.