From knowledge extraction to knowledge representation

Project

He NI

BI2

20150742

# Abstract

In the previous tutorials, we have seen different tools and techniques that allow : Represent knowledge and enrich it using RDF, RDFS and OWL Extract knowledge from text using Naîve Bayes and SVM In this project, the objective is to build a simple end to end pipeline in order to extract knowledge from a set of documents, represent this knowledge using rdf, query the created rdf graph and eventually use reasoning to enrich the extracted knowledge. Before starting the implementation of your pipeline. You should collect 30 news articles about different topics from different websites.

# **Introduction**

Data cleansing of articles downloaded from the Internet first

Then use nltk to split the text and remove unused words

Analysis of the processed data (NaîveBayes) and create a SVM classifier

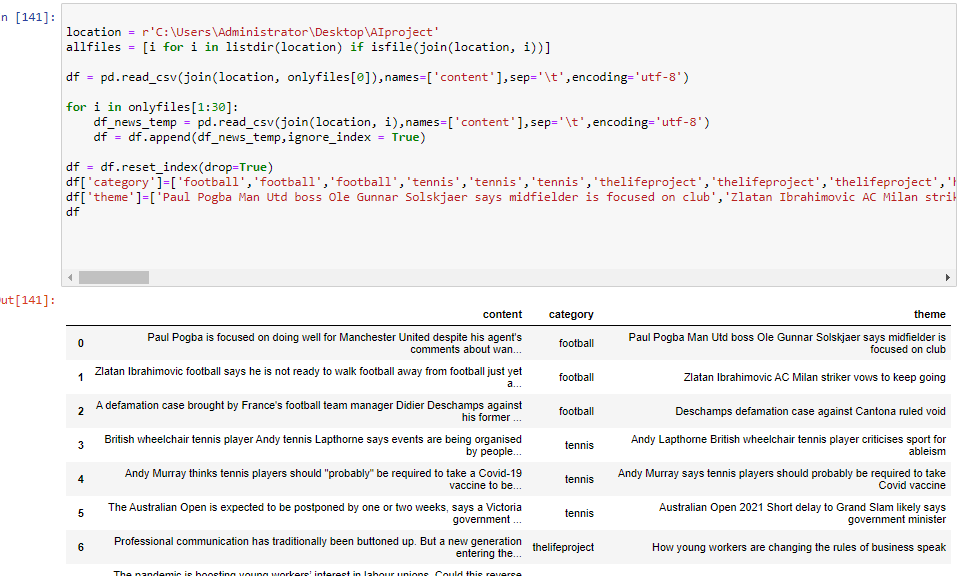
(It is very easy to find the wrong keywords here, and I will indicate the difficulties I encountered in a later section).

Exporting the results of the data from the test set and constructing the rdf to create a triple tuple

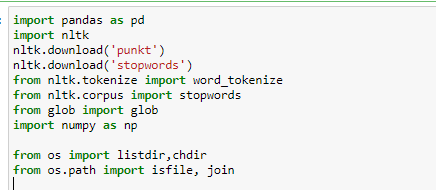
Querying rdf with sparql

# Approach

1. I search in the internet and download 30 articles. And then use python to import all these articles as dataframe. I use function (for) to loop input the articles and name each frame with category and theme



Notes: download(nltk)



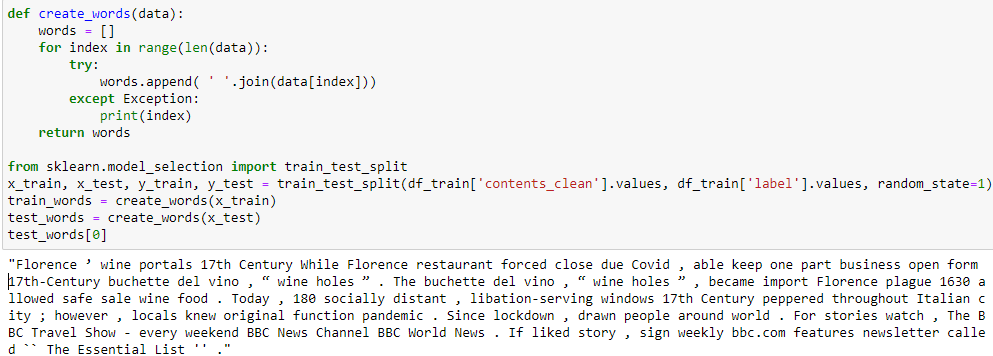
1. Use NLTK to tokenize to separate the article to words with punctuation



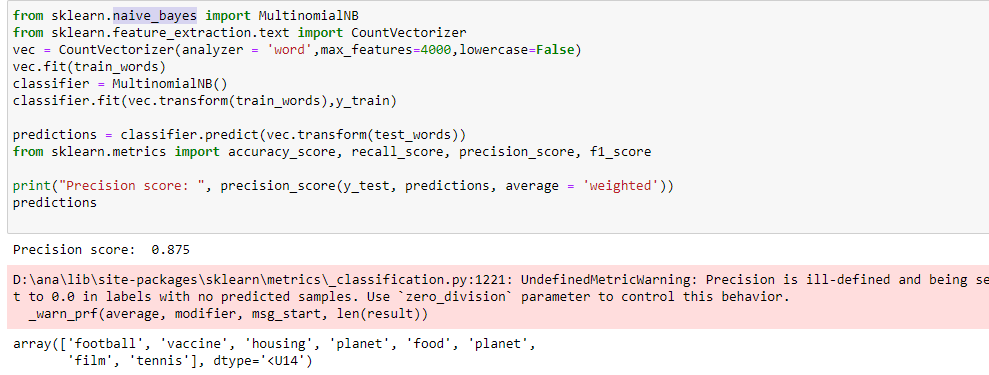
1. Remove the words which is not important in these articles, there are still some unuse words in the contents , I should clean them , but I don’t find the correct base in corpus.



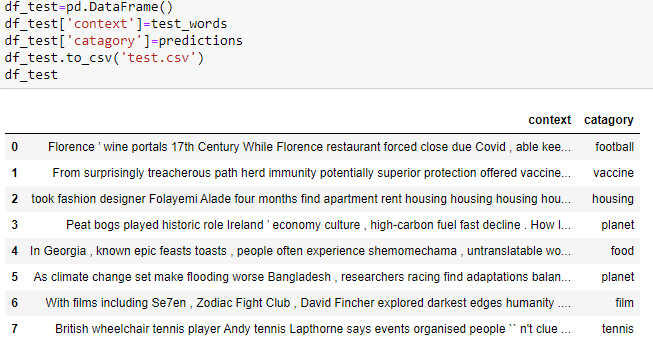
1. Split the train dataset and test dataset using sklearn



1. Use naïve bayes to make a classifier and as you see, I got 87.5% accuracy rate



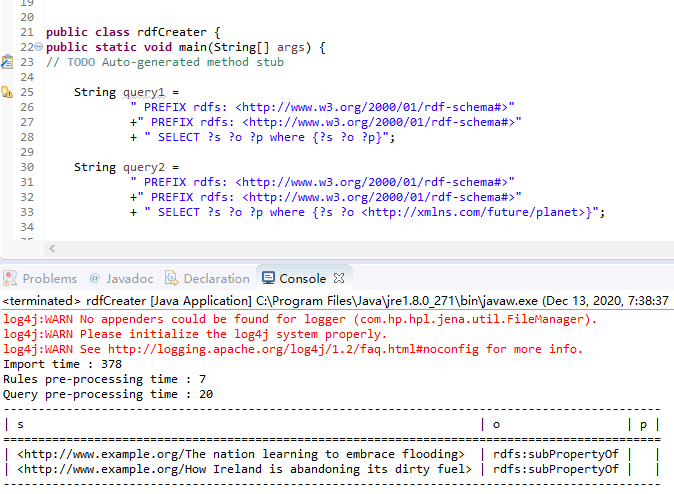
1. Output the result of prediction to a .csv file



1. Use rdflib to create ontology

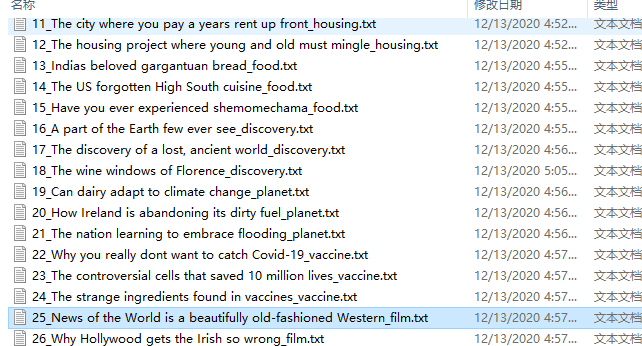


1. Use jena to do the query.



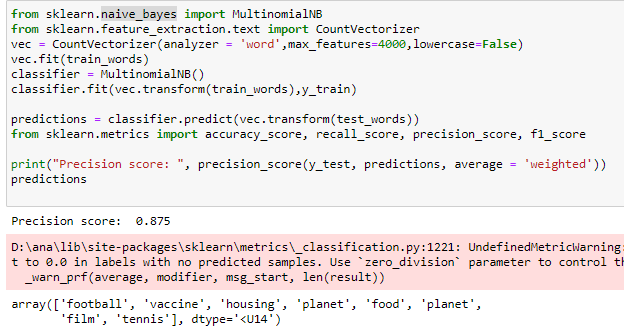
# Experiment and pipeline

### The dataset(s):



There are 30 articles in my datasets and I download all these articles from BBC NEWS, and I separate them with 8 categories which are football, tennis, the life project, housing, food, discovery, planet, vaccine, film, music.

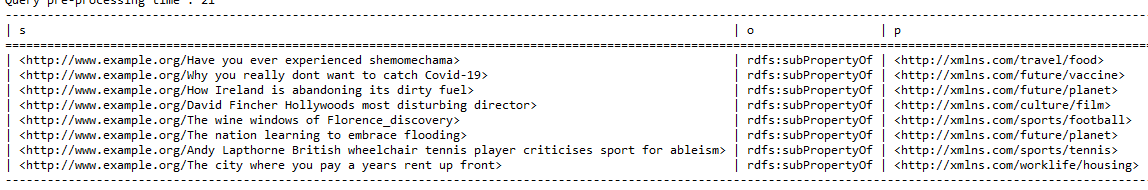
### The machine learning

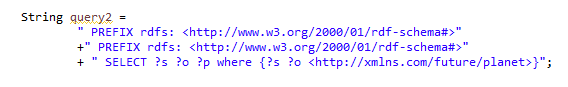


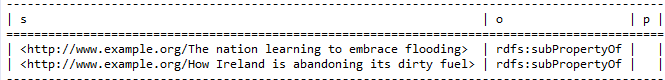
I use naïve bayes to make the classifier, Imagine that text can be divided into a number of categories, firstly text can be labelled by a number of word sets which are independently distributed, find the probability of the i-th word occurring in a given category C text and classify it by the probability of each word occurring.

### The sparql queries









### Conclusion:

1. Articles downloaded from the Internet can be garbled, and if the system does not recognise this, then we need to ETL them.

For example, there can be no carriage returns and no empty paragraphs.

2. Sorting using plain Bayes is quite simple, but it is very difficult to get a high accuracy rate.

For example, if I want to determine whether an article belongs to football, I will look at the probability of the word 'football' appearing and find that it is not as likely as the word 'team' to appear, so it is likely that the article will be classified as tennis.

How to make a keyword more likely is something I don't know yet, but with more time I'm sure I could be more accurate.

3. As my txt file does not extract the author's name, or details such as dates, I am unable to gain further knowledge.