PROJECT 1 – Document Classification

Important Notes:

1. If you do not have time to finish the project during the practical sessions, **you must finish it later**.
2. **Must be used:**
   a) **The same name we propose for the functions and procedures**
   b) **The same messages**

Otherwise, the correction system will not work and you will not pass the evaluation.

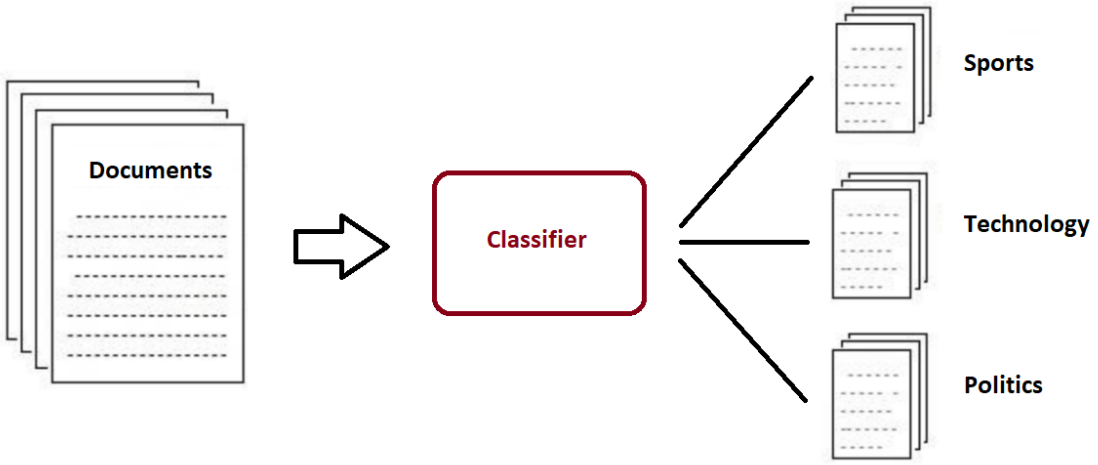
Introduction

Document classification is the process of assigning categories to documents to make them easier to search, sort, analyze, filter, etc. A document contains information related to some specific category (or topics). Assigning one or more classes (categories/topics) to a document makes it extremely beneficial for news sites, blogs, business industries, magazine publishers or anybody dealing with a huge amount of content.

Document classification is seen as a very useful Machine Learning (ML) application in the domain of Natural Language Processing (NLP). In general, there are two types of ML techniques: supervised and unsupervised. A ML model is built using **supervised** methods by creating a **training set**, which contains predefined categories (called **classes**) of documents and every document is tagged with one or more categories (called **labels**). A document **classifier** ML model is then trained on the dataset to perform prediction of a new document's category during the later **test** phase. The classifier may also additionally offer a confidence score to show how confident it is that the classification label is correct, depending on the classification algorithm or strategy utilized.

An example illustration of a Document Classification Model

Intuitively, given that a document is about a particular topic, one would expect some particular frequent words. For example, "goal" and "corner" will often appear in documents about football, "president" and "ministry" will appear in documents about politics, whereas stop words, like "the" and "with" will appear approximately equally in both. Indeed, words in a document can be used as a powerful tool (called **features**) to help in the prediction of a document category.



Goal:
The project consists in developing a program to classify documents in the real world. You will be provided with a **News dataset collection** containing documents in **txt** format of different categories (e.g. business, medical, space, sports, etc.), and also, a **txt** file with the list of keywords per category.

Exercise 1: Implement *DocClass_Welcome()*

Make a procedure called *DocClass_Welcome()*. This procedure has no input parameters and must display the following informational message:

```
Document Classification is typically used in Document Processing for
assigning categories to documents. Thus, it eases their posterior
processing and analysis.

The program "DocumentClassification" classifies an input folder
containing documents in txt format. It will also receive a file
containing information about the topics to classify.

The program will generate the following files:
A txt file containing the category of the documents
A txt file containing the statistics of the classification
```

Exercise 2: Implement *Read_Categories_Keywords()*

The function *Read_Categories_Keywords()* will open the **txt** file and read the categories and the keywords of each category. The input will be the name of the file containing the lists of categories and keywords (e.g. **keywords.txt**), whereas the output will be:

- *LCategories*: list of categories
- *LKeywords*: list (nested list) of keywords in each category

The format of the input file (e.g. **keywords.txt**) will consist of one category per line, with keywords separated by blank space. For example:

```
Category1: KeyWord1A  KeyWord1B  KeyWord1C

Category2: KeyWord2A  KeyWord2B  KeyWord2C  KeyWord2D  KeyWord2E

…

CategoryN: KeyWordNA  KeyWordNB  KeyWordNC  KeyWordND
```

Important: The number of categories and the number of keywords per category are **variable**.

For example, if the list of categories and keywords are:

```
medical: physician infection patient care
space: planet orbit nasa
sports: medal record score team training
```

The output should be:

- `LCategories = ["medical", "space", "sports"]`
- `LKeywords = [["physician", "infection", "patient", "care"],["planet", "orbit", "nasa"],["medal", "record", "score", "team", "training"]]`

Exercise 3: Implement *Document_Reader()*

The function *Document_Reader()* receives, as input, the path, the folder name and the filename as parameters. It opens the file and reads its contents. It returns the list of words *LWords* appearing in the document file.

Important:

- This function is case insensitive, which means that lower and uppercase letters are equivalent (NASA = nasa, Federer = federer).
- Punctuation marks are not considered: Full Stop (.), Question Mark (?), Quotation Marks (“), Comma (,), Hyphen (-), dash (–), Exclamation Mark (!), Colon (:), Semicolon (;), Parentheses (), Brackets [], Ellipsis (…), Slash (/).

For example, if a filename contains the text:
`"Once rocket 12 reached the planet, the NASA-team celebrated a party."`

LWords will be: `["once", "rocket", "12", "reached", "the", "planet", "the", "nasa", "team", "celebrated", "a", "party"].`

Exercise 4: Implement *Count_Frequencies*()

Create a function called *Count_Frequencies()* that takes as input the following parameters:

- *LWords*: list of words in the document.
- *LCategories*: list of categories.
- *LKeywords*: list (nested list) of keywords in each category.

The function will output:

- *LFrequencies*: list (nested list) with the number of occurrences of each keyword per category.

The function will count the number of occurrences of each keyword per category in the given input file and return the list *LFrequencies*. For example, if the list of categories and keywords are:

```
medical: physician infection patient care
space: planet orbit nasa
sports: medal record score team training
```

and the document file contains the text:

```
"Once rocket 12 reached the right distance from the planet, the
NASA-team celebrated a big party at the nasa main building."
```

The output list *LFrequencies* should be: [[0,0,0,0], [1,0,2], [0,0,0,1,0]].

Exercise 5: Implement ***Classify_Doc()*****:**

Create a function called *Classify_Doc()* that takes as input the list of frequencies *LFrequencies* and outputs:

- *LVotes*: list of occurrences in each category.
- *posMaxCat*: position of the maximum value in LVotes.

This function should count the total number of keyword occurrences in each category. Then, it will find the position of the category with the maximum number of occurrences (in other words, the maximum value in *LVotes*). For example, if *LFrequencies* is [[0,0,0,0], [1,0,2], [0,0,0,1,0]], the output will be *LVotes* = [0,3,1] and *posMaxCat* = 1.

Exercise 6: Implement ***main()*****:**

Implement the main program that will make the calls to the corresponding functions and procedures to classify the documents. The sequence is as follows:

1. Call the procedure *DocClass_Welcome()* to inform about the program.
2. Ask the path and name of the file that contains the definition of the categories and keywords. If the file does not exist or contains no data, ask the user again, until a valid file is entered.
3. Read the contents of the file, using the function *Read_Categories_Keywords()*.
4. Ask the path and the name of the folder that contains the document files to be processed. If the folder does not exist or contains no *txt* files, ask the user again, until a valid path and folder name is entered.
5. For each file in the folder, do the following:
    a) If it is a **txt** file, call the functions *Document_Reader()*, *Count_Frequencies()* and *Classify_Doc()*. Write the category of each file in the file named "<folder_name>_DocClassification.txt" with the following format:

```
Filename1: Category3 Keyword3A: X, Keyword3B: Y, …

Filename2: Category1 Keyword1A: X, Keyword1B: Y, …

…

FilenameM: Category5 Keyword5A: X, Keyword5B: Y, …
```

b) If the file has any other format, it should write the message: "Attention: the following non txt file has been detected: <filename>" in the file "<folder_name>_DocClassification.txt" (same as above). For example:

```
news1.txt: Category3 Keyword3A: X, Keyword3B: Y, …

news2.txt: Category1 Keyword1A: X, Keyword1B: Y, …

…

Attention: the following non txt file has been detected: news34.doc

…

newsM.txt: Category5 Keyword5A: X, Keyword5B: Y, …
```

6. Once all files in the folder are classified, compute the percentage (maximum 2 decimals) of **txt** documents of each category (note that for the statistics, the number of non-txt files does not count). Write the result in a file called "<folder_name>_DCStatistics.txt" (for example, if the folder is "MyDocs", the output file will be "MyDocs_DCStatistics.txt"). Also, compute the top 10 keywords in the folder (no matter if they belong to different categories). The format of this file will be:

```
Statistics

Total txt files: N

Category1: XX%

Category2: YY%

…

CategoryN: ZZ%

Top 10 keywords: KW1, KW2, KW3 … KW10
```

Important Remarks:

- The management of possible errors (e.g. file does not exist) is mandatory. For example:

```
try:

    f = open(filename,"r")

except:

    raise FileNotFoundError
```

- Feel free to create more functions if necessary.
- Comments in the code are very welcome.

Hints

Some useful instructions for reading all files in a folder:
The OS module in Python provides functions for interacting with the operating system. If you import this module, then:

- The function **os.listdir()** returns the list of all files and folders in the specified path of the directory.
- You can use **os.path.exists()** to check if files and/or directories exist.

More info at https://www.geeksforgeeks.org/os-module-python-examples/

Delivery

Deadline: October 26th.
The project will be done in pairs, and the delivery will be via *Caronte*. Instructions:

- Upload the file "DocumentClassification.py".
- Only one of the members of the group must do it.
- At the beginning of the Python file, you should write, as comments, the name, surname and NIU of the members of the group.