**Part 1: Text Processing**

Members: Pat Roca (u172939), Marc Vallcorba (u173384), Aran Montero(u173478), Cristian Bassotto (u210426)

Github URL: <https://github.com/u172939/IRWA-2022-u172939-u173478-u173384>

Github URL: <https://github.com/cristianbass01/IRWA-2022-FINAL-PROJECT/tree/main/IRWA-2022-part-1>

1. **Introduction**

This laboratory is the first part of the project we will be working on for the whole trimester. Our objective is to clean the dataset we are given in order to ease the future task of searching through it, this is the topic for our first part of the project: Text Processing. We have been provided with a dataset of tweets, so we will be doing its preprocessing. We have done it through:

* Removing stop words
* Tokenization of text
* Remove punctuation marks and non-alphanumeric
* Stemming
* etc

With this preprocessing we will be able to create a clean and practical dataset which will be easy to use.

1. **Assumptions**

* All tweets are written in the same language (English). Because of this, we did not take into account special characters (ie: á, ê) that would be something typical in other languages.
* The collection of tweets provided is real information.
* Tweets replying to other tweets are not distinguished from normal tweets.
* All the tweets are still available and have not been deleted.

1. **Implementation**

We based our implementation on the knowledge that we acquired during the practice and the theory classes. Therefore, we used some similar approaches to achieve our goals, which will be explained later.

In the first part of the notebook we mounted the drive in order to obtain all data when necessary. After that we find the *Imports* section.

In the *Functions* section we created some functions that are explained below:

* clean(text):
  + Transforms the text into lower case
  + If there is the url of the tweet it is removed
  + Remove non-alphanumeric characters
  + All previous steps are done with functions from the library called *re (Python Regex).*
  + Finally, it returns the cleaned text.
* build\_terms(text):
  + Calls the clean() function in order to clean the input text
  + Tokenize the text to get a list of words
  + Remove stop words
  + Perform stemming to keep only the root of all words
* create\_mapping(filename, verbose):
  + Creates and returns a dictionary containing the mapping {key: value}
  + If verbose= True it returns information about the dictionary, like the length

Regarding the mapping from document to tweet id we have implemented the code to build a dictionary with the following structure: {document\_id: tweet\_id}. We thought this would be an easy and efficient way to obtain the tweet\_it when from the document\_id. In order to build this dictionary we first read the csv file and load it into a pandas DataFrame. Then, we iterate through the DataFrame and store each row in a {key: value} pair as explained before.

After that we discussed the best way to store the tweets. Having in mind that we mapped document ids to tweet ids we thought it would be useful to have easy access to a concrete tweet when having a tweet\_id. In the end, we came up with a good solution: a dictionary of dictionaries. In other words, we decided to create a dictionary with tweet ids as keys and dictionaries of tweet information as values. Then, in the dictionary for each tweet we will store all the information we want to return as the final output. It means that the dictionary referring to one tweet will be a dictionary with the following keys: “Tweet”, “Username”, “Date”, “Hashtags”, “Likes”, “Retweets”, “Url”.

First, we initialize the dictionary. Then, we iterate through the DataFrame. At each iteration we fetch all the information we want to store in the dictionary and at the end we add it in the value space corresponding to the tweet id.

We have considered that it is better not to take away hashtags, as the text that follows the symbol “#”. This way we will be able to analyze the tendencies in the future, taking into account those words. Even though we kept the hashtag words in the full text, we also stored them as a list in the dictionary due to the need of returning them as an output for the query. We have also decided to format the date into a readable string using the function strftime().

When we analyzed the original dataset we found out that all tweets containing an image had the url as a part of the full\_text section, whereas tweets that hadn’t an image attached didn’t. We did some research and found a way of constructing the url of all tweets by concatenating some sections of the tweet information. This is the implementation we used to generate the tweet url for all tweets: 'https://twitter.com/'+row['user']['screen\_name']+'/status/'+str(row['id']).

At the end of our notebook we wrote a code in order to check that we built it correctly by accessing a tweet in the dictionary.