Part 1

For this first part we have to pre-process a document corpus which is a set of tweets.

First we adapt the code snippet provided in class to read the data. And then we start processing the text.

First we identify what are the fields that we are going to need. We know that the final output must return (when present) the following information for each of the selected documents: Tweet | Username | Date | Hashtags | Likes | Retweets | Url (here the “Url” means the tweet link).

From the data and the official API website (<https://developer.twitter.com/en/docs/twitter-api/v1/data-dictionary/object-model/tweet>), we know that the names of the fields that we need are:

* "text"
* "user.name"
* "created\_at"
* "entities.hashtags.text"
* "favorite\_count"
* "retweet\_count"
* "urls"

To start with, we create the function **build\_terms** that pre-processes text as required in the lab andthat you can find documented in the code.

Now we can parse the fields of the JSON that we need for our output. For that we have created all the following functions that access each field of the JSON to extract the information:

* get\_text: we iterate through all entities that the tweet has, for each entity we check the indices sub-field, this indicates where in the full\_text the entity is located, then we replace the entity with a stop word, we chose the # symbol, so later when stopwords are removed the string is fully cleaned
* get\_username
* get\_hashtags
* get\_likes
* get\_retweets
* get\_url: Manually builds the url from the username and the id
* get\_date
* get\_id
* get\_doc: checks in the given CSV the parsing of the tweet id to document number, and formats the string as we want

**Why do we think get\_text is important?**

As the raw json structure of a tweet has a field named "full\_text" which contains the raw text content of the tweet, we do believe that the text presented in that form is not what is optimal to have in our final dataframe. The reason is that in this raw "full\_text" content we find several "entities" along the actual text, that is, it contains artifacts such as Urls, media urls, hashtags.

Some of these types of entities are already in separate columns in our dataframe, so we decided to remove them with the get\_text function to avoid repeating information between columns.

We also wish to tokenize, stem, and process the text of the tweet in other ways later on using the "build terms" function , and the existence of those entities in the string would lead to meaningless words (like fractions of a URL) and thus to a faulty search engine later on, that has biased intra-document frequency calculations for entries. Therefore the actual text alone is what get\_text gets.

We must highlight that the output of this still contains characters like commas and some punctuation signs, that's because we still have not applied our build\_terms functions in the tweet text column in our data frame. In fact, we apply build\_terms later on to the dataframe, because we also want to keep the original text of the tweet (minus its entities) in our dataframe.

**How do we get the tweet URL (get\_url)?**

The json object for a tweet does not include the tweet URL by default, that is why we construct it manually, given the username of the tweet author and the ID of the tweet, which is included in the json. In order for the URL to work, we had to remove emojis from usernames by using “demoji” which is a public package to delete emojis from a string using regex and finally we had to change spaces in usernames into underscores, which is the suitable way to express spaces in URL strings.

Once we can parse the json, we create get\_info, the function that given a tweet, it creates a dictionary, and for each field (this will be the key), adds as value the information.

Next, we discussed how to store the output. For us it is easier to work with pandas DataFrames than with Jsons. That’s why we created “tweet\_df\_dict”, a dictionary containing all the parsed tweets with the relevant information. Then we created a pandas dataframe, called “tweet\_df” from “tweet\_df\_dict”.

Once we have our tweet\_df created, we applied build\_terms to the column “Tweet” and created a new column, “processed\_text” with the text processed.

# Link repository

<https://github.com/judiith1300/IRWA-2022-u161830-u151199-u161685>