# FIT5196 Assessment 1

**Student Name: DISHI JAIN**

**Student ID: 30759307**

Libraries used:

- langid (for verifying the language of a string)
- os (for accessing the files in the operating system)
- re (for regular expression, included in Anaconda Python 3.6)

## 1. Introduction

This assignment focuses on the extracting data from semi structured text files. From the given dataset containing multiple text files, we have to extract "id" , "date" and "text" of a tweet. This can be done using Regular Expression. After extraction of this information, we need to transform this data to an xml format.

More details of this task will be seen in the following sections.

## 2. Libraries importing

In [1]:
```python
#importing the required libraries
import langid
import re
import os
```

## 3. Loading the data

In [2]:
```python
#storing the entire data into a string called data
data = ""

#path where all files are present
Path = "FIT5196_A1/"
filelist = os.listdir(Path)
for i in filelist:
    #to get all files ending with .txt
    if i.endswith(".txt"):
        with open(Path + i, 'r', encoding="utf8") as start_object:
            data = data + start_object.read()

start_values = data.split("\"data\":")
```

As the output of the first step, we will get an object which contains the different files loaded. These individual files can be accessed by indexing this object. Hence every operation can now be performed by indexing this object.

Encoding of utf-8 is used for this task.

## 4. Creating a final dictionary

In this step, we will first iterate through each file using start_values. Inside this we will use the Regular Expression to extract only the ids, dates and text using thee different regular expressions.

Implementation works as follows -

- Iterating through start_values to get one string for one file containing all the data for that file
- Using the regular expression "id":"\d{19}\b to extract the id from the entire string. As we know that an id is a term of 19 digits hence I have used the regular expression in a way that will extract the 19 digits after it sees a term like "id":
  Hence this will help to extract all the ids
- Next the regular expression used to extract the text or the tweet is - text":"[^"]* This regular expression will help us to extract the text part of every tweet. It will work on the string and look for the term "text" . After this it will extract all the characters hence the * in the RE until a " is seen hence the [^"] prt in the RE. This [^"] tells the RE to see a character that is anything but " By ^ we mean a negation. Hence after this all the text part is extracted. Now another list is created which contains the text as is it hence removing the "text" part. This is done using slicing concept. This is done on the step - fd.append(k[7:]) in the below code.
- Next we create another regular expression which will extract the dates of the tweets. The RE used is - created_at":"[^T]* Similar concept like RE for text is used here. This RE will extract all the created date part from the string which starts with the term created_at and ends when the character T is seen. Hence it will take all the character after created_at until the first occurence of T. This is used to extract the dates only from the string. Also each string contains date multiple times. Hence the dates are extracted multiple times by this RE. This is handled using slicing again. First I check if the extracted part's length is greater that 0 or not. Then using slicing date = e[0][13:] I have created a date variable which will contain the date as a string just once.
- Next the zip function is used between the list of ids and the list of texts. Then the text is checked for english using the library langid. If the text is english then only a new dictionary is created that takes the id as its key and the text as its value. Now this dictionary is appended to a list called new_list. Hence we create a list of dictionaries in this step.
- Next we create another dictionary called fcd which contains the date as its key and the list of dictionaries(created above) as its value. If the date or the key is not in the dictionary fcd then the list becomes the value else if the date or key is already in the dictionary then the list is appended to the value of the date or key.

In [3]:
```
fc = []
fd = []
fcd = {}
date = ''
ft = []
new_list = []

#iterating for each text file
for i in start_values:
    fc = []
    fd = []
    new_list = []

    #regular expression to extract the ids including the term "id"
    c = re.findall(r'\"id\":\"\d{19}\b',i)
    for j in c:
        #creating a list fc containing only the ids
        fc = fc + re.findall(r'\d{19}\b',j)

    #regular expression to extract the text part of the string
    d = re.findall(r'text":"[^"]*',i)
    for k in d:

        #using slicing to remove "text:" terms from the extrcated tweets hence ke
        fd.append(k[7:])

    #regular expression to extract the created date including the "created_at" te
    e = re.findall(r'created_at":"[^T]*',i)
    if len(e) > 0:
        #date variable to store the date as a string just once using indexing and
        date = e[0][13:]

    #zip of the lists fc and fd containing ids and texts only
    for i,j in zip(fc,fd):
        new_dicti = {}

        #check for tweet laguage. If it is english then only it is used
        if langid.classify(j)[0] == 'en':

            j = j.replace('&' , "&amp;")
            j = j.replace("<" , "&lt;")
            j = j.replace(">" , "&gt;")
            j = j.replace("'" , "&apos;")
            j = j.replace('"' , "&quot;")

            #creating dictionary that has id as key and text as value
            new_dicti[i] = j
            #appending the dictionary to a new list
            new_list.append(new_dicti)

    if len(e) > 0:

        #creating a final dictionary that has date as id and a list of dictionari
        if date in fcd.keys():
            for k in new_list:
                fcd[date].append(k)
        else:
            fcd[date] = new_list
```

Hence after this step we get a dictionary whose key is a date and value is a list of dictionaries(Where the dictionary has key as id and value as text)

## 5. Creating a string in the desired output format

In this final step, a string is created that will be in the required format as the sample.xml file. This string is called the final_line which will be used to write into the 30759307.xml file. To get this final string we iterate into the final dictionary created above and use its key and values part to create the string. Tags like , , are also used and written into this string along with their closing tags.

The final string is then written onto the 30759307.xml file using open() and write(). The write() takes up only a string value which will be written onto the file. Encoding of utf-8 is used here.

In [4]:
```
#empty string created
final_line = ''
final_line = '<?xml version="1.0" encoding="UTF-8"?>' + '\n<data>'

#iterating inside the final dictionary to get the keys and values
for i in fcd.keys():

    #adding the date into the final_line
    final_line = final_line + '\n<tweets date="' + i + '">'

    #looping in the values part of the final dictionary which is a list
    for j in fcd[i]:

        #getting the id(key) and text(value) of each dictionary inside the list t
        for k,m in j.items():
            final_line = final_line + '\n<tweet id="' + k + '">' + m + '</tweet>'

    #closing the tweets tag to mark end of tweets of a day
    final_line = final_line + '\n</tweets>'

#closing the data tag to mark the end of the file
final_line = final_line + '\n</data>'

#replacing the \\n with \n
final_line = final_line.replace('\\n','\n')



#opening a file 30759307.xml to write in utf8 encoding
f = open("30759307.xml", "w", encoding="utf8")

#writing the final_line to this xml file
f.write(final_line)

#closing the file
f.close()
```

## 6. Summary

This assessment measured the understanding of basic text file processing techniques in the Python programming language. The main outcomes achieved while applying these techniques were:

- In the first part of reading files all the text files were read that were present in the current directory using os library.
- In the next part of this task regular expression has been used to extract the useful information hence only the date, id, text is extracted for the tweet. In this step the main task was to create a dictionary containing keys as dates and values as a list of dictionaries where each dictionary has key as id and value as text.
- Finally the dictionary was iterated and used to create a final string that contains the tags and desired format as in the output file sample.xml. After this the string is written onto a file called 30759307.xml

```
In [ ]:
```