Extracting texts

February 1, 2023

1 Modules and packages importing

The main modules are the following with their utility:

- `nltk` the best known and most used module in the natural language processing community.
- `bs4` or `Beautifulsoup` allows parsing of the html code in order to access more easily to some tags or specific elements in some web page.
- `requests` for reading the link from the web.
- `re` is useful for doing textual processing such as searching for patterns or correcting errors in the extracted texts.
- `numpy` to manipulate matrices and vectors and the mathematical operations between these objects such as linear algebra.
- `matplotlib` for graphic illustrations.
- `datatime` and `timeit` will provide a measure of the amount of time that the extraction will take according to different methods in order to speed up the program.
- the magic command `%matplotlib notebook` allows you to browse the graphics and choose their size and format manually.

[1]:
```
import datetime
from timeit import default_timer as timer
import matplotlib
import matplotlib.pyplot as plt
import requests
import re
from bs4 import *
import numpy as np
import nltk
import sys

%matplotlib notebook
```

1.1 Additional tools to make the displayed results look pretty

`Join_func` will allow to get the sentence (in one piece) from an ordered list of the words in that sentence. It does the opposite of word tokenization does. `color` class allows you to choose the font style for the printed elements

[2]:
```
def join_func(sentence):
    sentence = ' '.join(sentence) # join normally
    sentence = re.sub(" ([,.;\):])", lambda m: m.group(1), sentence) # stick to␣
 ↪left
    sentence = re.sub("([\(]) ", lambda m: m.group(1), sentence) # stick to␣
 ↪right
    sentence = re.sub(" (['']) ", lambda m: m.group(1), sentence) # join both␣
 ↪sides
    return(sentence)

class color:
   PURPLE = '\033[95m'
   BLACK = '\033[1;90m'
   CYAN = '\033[96m'
   DARKCYAN = '\033[36m'
   BLUE = '\033[94m'
   GREEN = '\033[1;92m'
   YELLOW = '\033[93m'
   RED = '\033[1;91m'
   BOLD = '\033[1m'
   UNDERLINE = '\033[4m'
   END = '\033[0m'
   BBCKGRND = '\033[0;100m'
   RBCKGRND = '\033[0;101m'


print(color.BLUE + color.BOLD + 'Hello World !' + color.END )
```

```
Hello World !
```

2 Download the file containing all the titles

We download the file A[1] which is compressed. After decompressing it, we get file B[2] which contains the titles of all wikipedia articles in a given (desired) language. In our case it will be the "*simplewiki*" language which contains the wikipedia articles that have been simplified. Actually, since there are much more articles in standard English "**enwiki**" (which we consider as complex) than those that have been simplified "**simplewiki**", we thus only extract the simple titles in order to reduce the number of texts that will have only one of the two versions. Remember that the goal is to extract for each article both simple and complex versions (i.e. **simplewiki** and **enwiki**). One can find the files in the following link wikimedia dumps*.

[1] simplewiki-20230123-all-titles-in-ns-0(1).gz
[2] simplewiki-20230123-all-titles-in-ns-0(1)
* https://dumps.wikimedia.org/other/pagetitles/20230123/

2.1 Title file reading and pre-processing

There are titles that contain a lot of spaces, after deleting these extra spaces as well as the first and last titles to lighten the file we obtain a list that contains the **simplewiki** titles to be extracted from the web.

[3]:
```
file = "simplewiki-20230123-all-titles-in-ns-0(1)"
#file = "simplewiki-20200420-all-titles"
titl = open(file,"r").readlines()
title = [w[w.find("\t")+1:] for w in titl[4:-5000]]
titles = np.array([str(re.sub('\"',"",w[:-1])) for w in title])
print(titles[:5])
```

```
['Captain_Lou_Albano' 'Crocodile_Dundee' 'Dimebag_Darrell_Abbott'
 'Heute_für_Morgen_Erste_Liga' 'Hylarana_attigua']
```

[4]:
```
print(str("The number of titles is: " + color.BOLD + "{}" + color.END + ".").
  ↪format(len(titles)))
```

```
The number of titles is: 310373.
```

2.1.1 List of links corresponding to each title:

We transform the list of titles into a list of links that redirect to the web page for the article that corresponds to each title

2.1.2 Simple urls

[5]:
```
def simple_wiki(t):
    return "https://simple.wikipedia.org/wiki/" + str(t)

simple_wiki = np.vectorize(simple_wiki)

# s_url are wikipedia pages written on simple english
s_url = np.array(list(map(simple_wiki,titles[:100])))
print(str("Number of effectively extracted simple urls is " + color.BOLD + "{}"␣
  ↪+ color.END + ".").format(len(s_url)),
      "\nFirst three simple urls: ", *s_url[:3], sep="\n")
```

```
Number of effectively extracted simple urls is 100.

First three simple urls:
https://simple.wikipedia.org/wiki/Captain_Lou_Albano
https://simple.wikipedia.org/wiki/Crocodile_Dundee
https://simple.wikipedia.org/wiki/Dimebag_Darrell_Abbott
```

2.1.3 Complex urls

The only difference is the keyword “en” instead of “simple” at the beginning of each url

[6]:
```
def complex_wiki(t):
    return "https://en.wikipedia.org/wiki/" + str(t)

complex_wiki = np.vectorize(complex_wiki)

# c_url are wikipedia pages written on complex or standard english
c_url = np.array(list(map(complex_wiki,titles[:100])))
print(str("Number of effectively extracted complex urls is " + color.BOLD +␣
 ↪"{}" + color.END + ".").format(len(c_url)),
      "\nFirst three complex urls: ", *c_url[:3], sep="\n")
```

```
Number of effectively extracted complex urls is 100.

First three complex urls:
https://en.wikipedia.org/wiki/Captain_Lou_Albano
https://en.wikipedia.org/wiki/Crocodile_Dundee
https://en.wikipedia.org/wiki/Dimebag_Darrell_Abbott
```

2.2 The extraction of the corresponding web page content to each title

We define a function that receives a list of links and returns the content of the page that corresponds to each link (i.e. each title). `srqst()` function for simple urls and `crqst()` for complex ones.

2.2.1 `srqst()`: Requesting simple texts

[7]:
```
def srqst(t):
    lst = str()
    data = requests.get(t)                          # getting the raw web page␣
 ↪content
    soup = BeautifulSoup(data.text,"html.parser")  # parsing the content
    data2 = soup.find_all("p")                      # accessing the desired␣
 ↪content ("p" as paragraph)
    if len(data2) >= 5 :
        for i in data2[:4] :
            if "Pages for" not in i.text :          # This line allow to delete␣
 ↪empty extractions
                lst += i.text
    else :
        for i in data2 :
            if "Pages for" not in i.text :
                lst += i.text
    return lst

vs = np.vectorize(srqst)

ls = np.array(list(map(vs, s_url[:10])))
for i,j in enumerate(ls[:2]):
```

```
    print(str(color.BOLD + "Complex text number ({}):" + color.END).
↪format(i+1), j)
```

Complex text number (1): Louis Vincent Albano (July 29, 1933 - October 14, 2009[1]) was a very famous American professional wrestler in the 1980s. Albano is best known as Captain Lou Albano. He also appeared in Cyndi Lauper's music video "Girls Just Wanna Have Fun" and played Mario on The Super Mario Bros. Super Show!, based on the series of video games.

Complex text number (2):
Crocodile Dundee (stylized as "Crocodile" Dundee in the United States) is a 1986 Australian-American romantic comedy adventure movie. It is set in the Australian Outback and in New York City. It stars Paul Hogan and Linda Kozlowski.[3] It is based on the true life of Rodney Ansell. The film had a budget of under $10 million.[4] They meant to make a commercial Australian film that America would like. Many people in the world liked the movie. It made the second most money of all movies in the United States in 1986. It also was second in money made in the world in 1986.

2.2.2 crqst(): Requesting complex texts

[8]:
```
def crqst(t):
    lst = str()
    data = requests.get(t)
    soup = BeautifulSoup(data.text,"html.parser")
    data2 = soup.find_all("p")
    try :
        if len(data2) >= 4 :
            for i in data2[:4] :
                if "Pages for" not in i.text :
                    lst += i.text
        else :
            for i in data2 :
                if "Pages for" not in i.text :
                    lst += i.text
    except: KeyboardInterrupt
    return lst


vc = np.vectorize(crqst)

lc = np.array(list(map(vc, c_url[:10])))
for i,j in enumerate(lc[:2]):
```

```
    print(str(color.BOLD + "Complex text number ({}):" + color.END).
↪format(i+1), j)
```

```
Complex text number (1):
Louis Vincent Albano[7] (July 29, 1933 - October 14, 2009) was an Italian-
American professional wrestler, manager and actor, who performed under the
ring/stage name "Captain" Lou Albano. He was active as a professional wrestler
from 1953 until 1969 before becoming a manager until 1995.
Over the course of his 42-year career, Albano guided 15 different tag teams and
three singles competitors to championship gold.[7] Albano was one of the
"Triumvirate of Terror", a threesome of nefarious WWF managers which included
The Grand Wizard of Wrestling and Freddie Blassie. The trio was a fixture in the
company for a decade until The Grand Wizard's death in 1983.

Complex text number (2):
Crocodile Dundee (stylized as "Crocodile" Dundee in the U.S.) is a 1986 action
comedy film set in the Australian Outback and in New York City. It stars Paul
Hogan as the weathered Mick Dundee, and American actress Linda Kozlowski as
reporter Sue Charlton.[6] Inspired by the true-life exploits of Rod Ansell, the
film was made on a budget of under $10 million as a deliberate attempt to make a
commercial Australian film that would appeal to a mainstream American audience,
but proved to be a worldwide phenomenon.
Released on 30 April 1986 in Australia, and on 26 September 1986 in the United
States, it was the highest-grossing film of all-time in Australia, the highest-
grossing Australian film worldwide, the second-highest-grossing film in the
United States in 1986, the highest-grossing non-US film at the US box office
ever and the second-highest-grossing film worldwide for the year. There are two
versions of the film: the Australian version, and an international version,
which had much of the Australian slang replaced with more commonly understood
terms, and was slightly shorter. As the first film in the Crocodile Dundee film
series, it was followed by two sequels: Crocodile Dundee II (1988) and Crocodile
Dundee in Los Angeles (2001), although both films failed to match the critical
success of the original.
```

Remark

Note that we could have used the command `try:... except...` to be sure that the program will continue to run even when there are extraction problems, when the page no longer exists or the content is not in the right location.

As we can see, there can be errors in the extracted texts, such as the numbers in square brackets indicating the different definitions or the different texts for the same title. Sometimes there are also punctuation problems, such as the comma (or period) not being followed by a space, which will let the function, which splits the sentence into words, believe that the two words surrounding the comma are one word and therefore will not split them into two. As for the example below :

2.2.3 Cleaning data

[9]:
```
example_sequence = "firstword secondword.thirdword 3.14"
print("Raw example: ", example_sequence)
print(color.RED + color.BOLD+ "Word tokenized raw example: " + color.END, nltk.
 ↪word_tokenize(example_sequence),end="\n\n")

corrected_example_sequence = re.sub(r"(\D)\.(\D)",r"\1. \2", example_sequence)
print("The corrected example: ", corrected_example_sequence)
print(color.GREEN + color.BOLD+"Word tokenized corrected example: " + color.
 ↪END, nltk.word_tokenize(corrected_example_sequence))
```

```
Raw example:  firstword secondword.thirdword 3.14
Word tokenized raw example:  ['firstword',
'secondword.thirdword', '3.14']

The corrected example:  firstword secondword. thirdword 3.14
Word tokenized corrected example:  ['firstword', 'secondword',
'.', 'thirdword', '3.14']
```

Notice that we want to put a space after the dot separating two non digit words, but we want to keep it as it is for numbers. Here is another example with extra spaces and comma problem :

[10]:
```
example2 = "Containing some errors like the following brackets[1], [2][3] or␣
 ↪the coma,between two words 3,5.. "
deleting_brackets = re.sub(r"\[.*?\]","", example2)
correcting_comma = re.sub(r"(\D),(\D)", r"\1, \2", deleting_brackets)
correcting_spaces = re.sub(r"\s+", " ", correcting_comma)
print(correcting_spaces)
```

```
Containing some errors like the following brackets, or the coma, between two
words 3,5..
```

3 Starting extraction

Let n be the number of texts we want to extract. We use `np.random.randint` to select randomly n titles from title list.

[11]:
```
n = 100
indx = np.random.randint(0, len(titles), size=n, dtype='int')
selected_titles = titles[indx]
s_url = np.array(list(map(simple_wiki, selected_titles)))
c_url = np.array(list(map(complex_wiki,selected_titles)))
#print(s_url[:2], "\n", c_url[:10])
print("First two simple urls: ", *s_url[:2], "\nFirst two complex urls: ",␣
 ↪*c_url[:2], sep="\n")
```

```
First two simple urls:
https://simple.wikipedia.org/wiki/A15_autoroute
```

```
https://simple.wikipedia.org/wiki/All_purpose_flour

First two complex urls:
https://en.wikipedia.org/wiki/A15_autoroute
https://en.wikipedia.org/wiki/All_purpose_flour
```

3.1 Getting balanced couples for each text

3.1.1 First filtering

We can remark that most of texts have differents sizes according to the language they are extracted from. In order to have similar sizes between both version **simplewiki** and **enwiki**, we will filter extracted texts to keep only those that have sizes between 100 and 1400 characters in both versions. For the example below, we only use it accross 15 texts to quickly access the results.

We also use `np.vectorize` to the function in order to apply the function to list elements in parallel way using the python built-in function `map()`.

[12]:
```
def srqst(t):
    lst = str()
    data = requests.get(t)
    soup = BeautifulSoup(data.text,"html.parser")
    data2 = soup.find_all("p")
    if len(data2) >= 5 :
        for i in data2[:5] :
            if "Pages for" not in i.text :
                lst += i.text
    else :
        for i in data2 :
            if "Pages for" not in i.text :
                lst += i.text
    return lst

vs = np.vectorize(srqst)
ls = np.array(list(map(vs, s_url[:15])))

def crqst(t):
    lst = str()
    data = requests.get(t)
    soup = BeautifulSoup(data.text,"html.parser")
    data2 = soup.find_all("p")
    try :
        if len(data2) >= 3 :
            for i in data2[:3] :
                if "Pages for" not in i.text :
                    lst += i.text
        else :
            for i in data2 :
                if "Pages for" not in i.text :
```

```
                    lst += i.text
    except: KeyboardInterrupt
    return lst

vc = np.vectorize(crqst)
lc = np.array(list(map(vc, c_url[:15])))

idx1 = np.array(list(map(lambda x : 100<len(x)<1400, ls)))
idx2 = np.array(list(map(lambda x : 100<len(x)<1400, lc)))

print("Simple texts that have a number of characters between 1 and 2 are:\n",␣
 ↪*idx1, "\n\n",
      "Complex texts that have a number of characters between 1 and 2 are:\n",␣
 ↪*idx2)
```

```
Simple texts that have a number of characters between 1 and 2 are:
 True True False False True True True True True False True True True True True

 Complex texts that have a number of characters between 1 and 2 are:
 True False False True True True False True True True False False True True True
```

We can get the intersection between the two versions by multiplying indexes, to get only valid texts in both versions at the same time, and exclude those that have a valid size only in one or any of the two versions.

[13]:
```
idx = idx1 * idx2
print("valid common texts: \n", *idx)
```

```
valid common texts:
 True False False False True True False True True False False False True True
True
```

[14]:
```
idx = idx1 * idx2
small_ls = ls[:15] ; small_lc = lc[:15] ;
small_ls = small_ls[idx] ; small_lc = small_lc[idx]
print(color.BOLD + "Simple valid example:\n" + color.END, *small_ls[:2],␣
 ↪"\n\n", color.BOLD+"Complex valid example:"+ color.END, *small_lc[:2])
```

Simple valid example:

```
 The A15 is a short autoroute (motorway) in France. It goes through the
suburbs of the northwest of Paris. It is about 21 kilometres (13 mi) long.[1]

Sir Richard Starkey MBE (born 7 July 1940), known professionally as Ringo Starr,
is an English actor and musician. He is known as a former member of the Beatles.
He joined the group in 1962 as a replacement for their first drummer Pete Best.
He quickly became well-liked and very popular. He sang lead on some of the
band's songs including "Yellow Submarine", "Act Naturally", "Don't Pass Me By",
and "Octopus's Garden".
```

After the group broke up he became a solo artist; his songs included "It Don't Come Easy", "Photograph" (written with George Harrison), "You're Sixteen" (featuring Paul McCartney and Harry Nilsson), and "Only You (And You Alone)" (with John Lennon and again with Nilsson).
Starr acted in several movies aside from the ones he did with the Beatles, including The Magic Christian (1969), That'll Be The Day (1973), Caveman (1980), and the role of Mr. Conductor on the children's show Shining Time Station, during its first season (1989). (Comedian George Carlin later took over the role). He also narrated the children's show Thomas the Tank Engine and Friends for the first two seasons (1984-1986). He also made a movie with the Beatles called A Hard Day's Night in 1964. He guest starred in the first episode of Courage the Cowardly Dog as an Alien Duck.

Complex valid example: The autoroute A15 is an autoroute in the western suburbs of Paris, France.
The motorway starts at Gennevilliers in Hauts-de-Seine and ends in Cergy-Pontoise in Val d'Oise. The A15 was built to relieve the congested RN14 between Paris and the Cergy-Pontoise new town. The A15 is operated by the Île-de-France Council. The motorway is 22 km (14 mi) in length and has no tolls
The A15 serves Gennevilliers, Argenteuil, Pierrelaye in Eragny, Saint-Ouen-l'Aumône and Pontoise.

Sir Richard Starkey[2] MBE[3] (born 7 July 1940), known professionally as Ringo Starr, is an English musician, singer, songwriter and actor who achieved international fame as the drummer for the Beatles. Starr occasionally sang lead vocals with the group, usually for one song on each album, including "Yellow Submarine" and "With a Little Help from My Friends". He also wrote and sang the Beatles songs "Don't Pass Me By" and "Octopus's Garden", and is credited as a co-writer of four others.

3.1.2 Second filtering

Also, we would like the texts to be more or less the same size. For this we can select only those for which the ratio of lengths between their two versions is between 0.5 and 1.5, so that we don't get a text with 1 paragraph in the simple version but with more than 4 paragraphs for the complex version

[15]:
```
vs = np.vectorize(srqst)
ls = np.array(list(map(vs, s_url[:10])))

vc = np.vectorize(crqst)
lc = np.array(list(map(vc, c_url[:10])))

idx1 = np.array(list(map(lambda x : len(x), ls)))   # calculating the length of␣
 ↪each extracted text
idx2 = np.array(list(map(lambda x : len(x), lc)))
```

```
print(idx1, idx2)
```

```
[ 149 1134   67    0 1277  383 1329  364  305 2047] [472  45  67 467 495 393   6
512 528 742]
```

[16]:
```
idxx = np.array(list(map(lambda x : 0.75 < x < 1.5 , idx1/idx2)))
print(color.BOLD + "Texts that are approximately equal in size :\n"+ color.END,␣
 ↪idxx, "\n")
print(color.BOLD+ "Simple version:\n" + color.END , *ls[idxx], "\n\n",
      color.BOLD+ "Complex version\n"+ color.END, *lc[idxx])
```

Texts that are approximately equal in size :

```
 [False False  True False False  True False False False False]
```

Simple version:

```
 Brunswick County is the name of two counties in the United States:
 Charles  Robert Alexandre des Moulins (1798-1875) was a French botanist who
also studied molluscs. In 1830, he proposed to introduce plants into aquaria so
that the animals in the aquarium could breathe better. He described many species
of snails, for example Pagodulina pagodula, and plants, for example Euphorbia
milii. Towards the end of his life he became opposed to Darwinism.
```

Complex version

```
 Brunswick County is the name of two counties in the United States:
 Charles Des Moulins, full name Charles Robert Alexandre Des Moulins (13 March
1798 - 23 December 1875)[1] was a French naturalist, a botanist and
malacologist.
He was a member of several learned societies, including the American
Philosophical Society, which elected him an international Member in 1861,[2] and
the Société linnéenne de Bordeaux, of which he served as its president in
1826.[3]
```

3.2 Useful cleaning functions to delete empty extractions or not valid texts

[17]:
```
# indeces of empty simple texts
np.where(ls == "Wikipedia does not yet have an article with this name.\n")
```

[17]:
```
(array([], dtype=int64),)
```

[18]:
```
# indeces of empty complex texts
np.where(lc == 'Other reasons this message may be displayed:\n')
```

[18]:
```
(array([1]),)
```

[19]:
```
np.delete(range(10), np.where(lc =='Other reasons this message may be displayed:
 ↪\n'))
```

[19]:
```
array([0, 2, 3, 4, 5, 6, 7, 8, 9])
```

4 Comparing time duration between a `for` loop and `map` function

4.1 First Method using a `for` loop

[20]:
```
n = 50
indx = np.random.randint(0, len(titles), size=n, dtype='int')
selected_titles = titles[indx]
s_url = np.array(list(map(simple_wiki, selected_titles)))
c_url = np.array(list(map(complex_wiki, selected_titles)))

start = timer()
def srqst(t):
    lst = str()
    data = requests.get(t)
    soup = BeautifulSoup(data.text,"html.parser")
    data2 = soup.find_all("p")
    if len(data2) >= 5 :
        for i in data2[:4] :
            if "Pages for" not in i.text :
                lst += i.text
    else :
        for i in data2 :
            if "Pages for" not in i.text :
                lst += i.text
    return lst

#print(srqst(urls[12]), "\n")

def crqst(t):
    lst = str()
    data = requests.get(t)
    soup = BeautifulSoup(data.text,"html.parser")
    data2 = soup.find_all("p")
    try :
        if len(data2) >= 4 :
            for i in data2[:4] :
                if "Pages for" not in i.text :
                    lst += i.text
        else :
            for i in data2 :
```

```
                if "Pages for" not in i.text :
                    lst += i.text
    except: KeyboardInterrupt
    return lst


simpl={}
idx = []
for k,i in enumerate(s_url):
    try:
        T = srqst(i)
        if len(T) > 100:
            simpl[i[34:]]= T
    except AttributeError:
        idx.append(k)


print(color.BOLD + "Simple Versions :\n"+ color.END)
for i,k in enumerate(list(simpl.keys())[:2]):
    print(str(i) + ") ", (k, re.sub("\n", " ", simpl[k])), "\n")


compl={}
idx2 = []
for k,i in enumerate(c_url):
    try:
        T = crqst(i)
        if len(T) > 200:
            compl[i[30:]]= T
    except AttributeError:
        idx2.append(k)


print(color.BOLD + "Complex Versions :\n" + color.END)
for i,k in enumerate(list(compl.keys())[:2]):
    print(str(i) + ") ", (k, re.sub("\n", " ", compl[k])), "\n")


keys_a = set(simpl.keys())
keys_b = set(compl.keys())
intersection = keys_a & keys_b
print(str(color.BOLD + "number of common titles is " + color.RED+"{}"+color.
 ↪END).format(str(len(intersection))), end="\n")

duration = timer() - start
print(str(color.BOLD + color.BLUE + "Duration : " + color.END + "{}s").
 ↪format(duration))
```

Simple Versions :

0) ('Gymnasium_Alpenstrasse_Biel', 'Gymnasium Alpenstrasse Biel / Gymnase de la Rue des Alpes de Bienne is a high school in Biel in Switzerland. There are three local high schools and the Gymnasium Alpenstrasse / Gymnase de la Rue des Alpes is one of them. It is the only bilingual (Swiss-German and French) high school in the Canton of Berne. There are about 550 students and about 90 teachers at this school. This school specializes in three subjects: Economy, Spanish and Applied Mathematics. The students usually attend the Gymnasium for three years and take the final examination at the end of their studies. The diploma, which is called Maturité in French or Matura in German, is the entrance ticket for all universities and colleges in Switzerland. There is also a Commercial Highschool in the same building which is part of the Gymnasium Alpenstrasse Biel / Gymnase de la Rue des Alpe de Bienne. It is more oriented for the business world [1] At the beginning, the Gymnasium Alpenstrasse was a Progymnasium. A Progymnasuim is like the today's secondary school. But the school was just attended by boys. Today it's a High School for boys and girls. This school is a Swiss Olympic Partner School. This means, those athletes from various sports can go their training, also during school. There are many different athletes. Some play hockey, volleyball, tennis or they swim or dance etc. It's not just for sports it's also for music. ')

1) ('Juliana_Koo', 'Juliana Koo (formerly Young; née Yen; September 26, 1905 - May 24, 2017) was a Chinese-American diplomat. She served as first head of the United Nations Protocol Department. She was a supercentenarian and after emigrating to the United States, she became the third wife of diplomat and politician Wellington Koo.[1] Koo died on May 24, 2017 in New York City, aged 111.[2][3] ')

Complex Versions :

0) ('Juliana_Koo', 'Juliana Young Koo, born Yen Yu-yun (Chinese: ; pinyin: Yan Youyun; September 26, 1905 - May 24, 2017), was a Chinese-American diplomat who worked in the UN Protocol Department.[1] Her first husband, Chinese diplomat Clarence Kuangson Young was executed by Japanese armies during World War II. She became the long-term mistress for the diplomat and politician V.K. Wellington Koo, before her husband\'s death. After the war, she moved to the United States, in 1956 Koo divorced his wife and married her. On September 26, 1905, Koo was born into a wealthy family with business and government ties in Tianjin, China as Yen Yu-yün (or Yan Youyun). Her father Yan Zijun\xa0[zh] (1872-1931) and her grandfather Yan Xinhou\xa0[zh] (1838-1907) were both prominent businessmen. She attended Keen School when she was 14.[2] She was one of the first women to graduate from Fudan University.[citation needed] At university, a special car took her to campus and brought her back, since its number was 84, the male students nicknamed her "Miss 84".[3] ')

```
1)  ('Arena_Linköping', "The Linköping Arena is an association football stadium
in Linköping, Sweden. Opened in 2013, the stadium has a capacity of 7,400[1] and
hosted four games at the UEFA Women's Euro 2013 tournament. Following the
tournament, the stadium became home to Linköpings FC women's association
football team. The name was discussed for a while and the project name was Arena
Linköping[2] before at the inauguration it was announced as Linköping Arena.[3]
Coordinates: 58°25 5 N 15°38 57 E\ufeff / \ufeff58.41806°N 15.64917°E\ufeff /
58.41806; 15.64917  ")

number of common titles is 42
Duration : 112.81911817400032s
```

4.2 Second Method using map() function

[21]:
```
start = timer()
def srqst(t):
    lst = str()
    data = requests.get(t)
    soup = BeautifulSoup(data.text,"html.parser")
    data2 = soup.find_all("p")
    if len(data2) >= 5 :
        for i in data2[:5] :
            if "Pages for" not in i.text :
                lst += i.text
    else :
        for i in data2 :
            if "Pages for" not in i.text :
                lst += i.text
    return lst

vs = np.vectorize(srqst)
ls = map(vs, s_url)

def crqst(t):
    lst = str()
    data = requests.get(t)
    soup = BeautifulSoup(data.text,"html.parser")
    data2 = soup.find_all("p")
    try :
        if len(data2) >= 4 :
            for i in data2[:4] :
                if "Pages for" not in i.text :
                    lst += i.text
        else :
            for i in data2 :
                if "Pages for" not in i.text :
                    lst += i.text
```

```
    except: KeyboardInterrupt
    return lst

vc = np.vectorize(crqst)
lc = map(vc, c_url)

duration = timer() - start
print(str("duration before putting results in a list: " + color.BOLD + "{}s" +␣
 ↪color.END).format(duration))

start= timer()
ls = np.array(list(ls))
lc = np.array(list(lc))

# More cleaning
ls = ls[np.delete(range(len(lc)), np.where(lc =='Other reasons this message may␣
 ↪be displayed:\n'))]
lc = lc[np.delete(range(len(lc)), np.where(lc =='Other reasons this message may␣
 ↪be displayed:\n'))]

duration = timer() - start
print(str(color.BOLD + color.BLUE + "Duration: " + color.END + color.BOLD +␣
 ↪"{}s" + color.END).format(duration))
```

```
duration before putting results in a list: 0.000363891000233707s
Duration: 44.541464140000244s
```

Conclusion

We can conclude that the second method that uses the mapping operation `map()` takes less time (around 44 seconds) than the first one (around 112 seconds) that uses the `for` loop to extract 37 common texts (simple and complex) among 50 from wikipedia.

[22]:
```
# First filtering
idx1 = np.array(list(map(lambda x : 100<len(x)<1400, ls)))
idx2 = np.array(list(map(lambda x : 100<len(x)<1400, lc)))
idx = idx1 * idx2

ls = ls[idx] ; lc = lc[idx]

# Second filtering
idx1 = np.array(list(map(lambda x : len(x), ls)))
idx2 = np.array(list(map(lambda x : len(x), lc)))

# Applying the filter
idxx = np.array(list(map(lambda x : 0.75 < x < 1.5 , idx1/idx2)))
ls = ls[idxx] ; lc = lc[idxx]
```

5 Saving extracted data using `pickle` module

We stock data in a pickle file using `pickle.dump`. To read it after, one can use `pickle.load` to read the file in next sessions.

[23]:
```
simple_texts_tosave = ls ; complex_texts_tosave = lc
```

5.1 Saving data

[24]:
```
import pickle
with open('raw_simple_texts','wb') as file:
    pickle.dump(simple_texts_tosave, file)

with open('raw_complex_texts','wb') as file:
    pickle.dump(complex_texts_tosave, file)
```

5.2 Reading data

[25]:
```
with open('raw_simple_texts','rb') as file:
    reading_data = pickle.load(file)
```

[28]:
```
print(*reading_data[:1])
```

```
The Linköping Arena is an association football stadium in the town of Linköping
in Sweden. It was opened in 2013 and has a capacity of 8 500 and hosted four
games during the UEFA Women's Euro 2013 tournament. The stadium became home to
Linköpings FC women's association football team after that tournament. The name
was discussed for a while and the project was named the Arena Linköping[1]
before the name was changed to Linköping Arena.[2]
Coordinates: 58°25 5 N 15°38 57 E / 58.41806°N 15.64917°E / 58.41806;
15.64917
```