Mining_geo_names_from_text

June 26, 2022

1 Mining geo names from text.

In this tutorial I will explain: how to extract geonames form text in polish. We will be connecting to Clarin web services, which provide many brilaint tools for language text processing. More inforamtion about this service can be found, on theirs web site https://clarin-pl.eu/.

First thing that have to be done is istaling lpmn_client pip install -i https://pypi.clarin-pl.eu lpmn_client

[41]:
```
from lpmn_client import download_file, upload_file
from lpmn_client import Task
import os
import zipfile
```

Create file, and copy there some text. For me it will be wikipedia page about my home town Płock.

[61]:
```
example_file_name = 'exampleText.txt'
with open(example_file_name,"r") as f:
    content = f.read()
content
```

[61]:
```
'Płock - miasto w Polsce, na prawach powiatu, położone na Pojezierzu Dobrzyńskim
i w Kotlinie Płockiej, nad Wisłą, w województwie mazowieckim, siedziba
ziemskiego powiatu płockiego; historyczna stolica Mazowsza oraz stolica Polski w
latach 1079-1138; siedziba rzymskokatolickiej kurii diecezji płockiej (1075),
siedziba biskupa naczelnego Kościoła Starokatolickiego Mariawitów, port rzeczny,
rafineria ropy naftowej (1964), szkoły wyższe, teatry, muzea. Polskie miasto-
bohater (1921).'
```

Now it is time send that file to clarin. Let's create function for that. It takes tree parameters: - path_to_file-path to file that we would like to extract geo names from. - downlad_path-where the save the answer from the websevice - lmpn-what pipeline of process are to be performed on the text.

I also created new directory in the project root called 'out'. To save results there.

[62]:
```
def send_task_to_service(path_to_file, download_path,lmpn):
    task = Task(lpmn=lmpn)
    file_id = upload_file(path_to_file)
```

```
    output_file_id = task.run(file_id)
    return download_file(output_file_id, download_path)
```

[63]:
```
path_to_file = example_file_name
download_path ='./out'
lmpn = 'any2txt|wcrft2({"morfeusz2":false})|liner2({"model":"n82"})|geolocation'
send_task_to_service(path_to_file,download_path,lmpn)
```

and now let's look in to our 'out' directory, the if condition is for the hidden files. We don't want to see them.

[64]:
```
def list_contents_of_folder(folder):
    return [f.name for f in os.scandir(folder) if '.' not in f.name[0] ]
```

[65]:
```
downloads_file_contents = list_contents_of_folder(download_path)
downloads_file_contents
```

[65]:
```
['e7ceb3dc-12cb-4d63-a8e6-2cd2b224d290.zip']
```

Ok, so we have something there. Let's unpack it!

[66]:
```
def unzipfile(path_to_zip,output_dir):
    with zipfile.ZipFile(path_to_zip, 'r') as zip_ref:
        zip_ref.extractall(output_dir)
```

[70]:
```
path_to_downloaded_file = os.path.join(download_path,downloads_file_contents[0])
unzipfile(path_to_downloaded_file,download_path)
```

Let's list our downolad direcotry to see it there are any changes

[71]:
```
list_contents_of_folder(download_path)
```

[71]:
```
['exampleText.txt', 'e7ceb3dc-12cb-4d63-a8e6-2cd2b224d290.zip']
```

Ok, so we see that compressed in that strangly named file, there is a file named exacly like those we had sent. Let's look inside. What misturies it holds?

[81]:
```
path_to_extracted_example_file = os.path.join(download_path,path_to_file)
with open(path_to_extracted_example_file,"r") as f:
    content = f.read()
content[0:500]
```

[81]:
```
'<?xml version="1.0" encoding="UTF-8"?>\n<!DOCTYPE chunkList SYSTEM
"ccl.dtd">\n<chunkList>\n <chunk type="p" id="ch1">\n  <sentence id="s1">\n
<tok>\n    <orth>Płock</orth>\n    <lex
disamb="1"><base>Płock</base><ctag>subst:sg:nom:m3</ctag></lex>\n    <ann
chan="nam_org_organization">0</ann>\n    <ann chan="nam_fac_goe">0</ann>\n
```

```
<ann chan="nam_loc_gpe_admin1">0</ann>\n    <ann chan="nam_loc_gpe_city"
head="1">2</ann>\n    <ann chan="nam_loc_country_region">0</ann>\n    <ann
chan="nam_loc_gpe_country">0'
```

As you can see the returned document is xml.

To extract valuable inforamtion we have to parse that document.

[89]:
```
import xml.etree.ElementTree as ET
```

[90]:
```
def parse_xml_for_geo_names(xml_file):
    tree = ET.parse(xml_file)
    root = tree.getroot()
    base_tags = root.findall('./chunk/sentence/tok/prop')
    return list(map(lambda x: x.text,base_tags))
```

[91]:
```
parse_xml_for_geo_names(path_to_extracted_example_file)
```

[91]:
```
['19.7136301805999;52.5353471;administrative;Płock, województwo mazowieckie,
Polska',
 '19.0258159;52.0977181;administrative;Polska',
 '20.0867774;52.380897;river;Wisła, gmina Wyszogród, powiat płocki, województwo
mazowieckie, 43-450, Polska',
 '21.2073404;52.5461934;administrative;województwo mazowieckie, Polska',
 '19.4638889;50.1708333;neighbourhood;Płaczki, Trzebinia, gmina Trzebinia,
powiat chrzanowski, województwo małopolskie, 32-540, Polska',
 '21.2073404;52.5461934;administrative;województwo mazowieckie, Polska',
 '19.0258159;52.0977181;administrative;Polska']
```

Succes, there are bunch of names that have been found in the text. But I already see one mistake. There is “Płaczki, Trzebinia” found as a palce mentioned in the text. Accualy, I am from Płock, and belive me Płock has nothing to do with Trzebinia. So how to repair that?

We have to add additonal step to text procesing pipeline. Namely we need to add Lemmatisation. Unfortunaly clarin do not work well with geolocalization and lemmatisaion. That means, if we provide lmpn like that: ‘any2txt|wcrft2({“morfeusz2”:false})|liner2({“model”:“n82”})|ccl_emo({“lang”:“polish”})|geolocation({“limit”:1}

Clarin won’t be searching for assosiations for base form of the word, but for orginal form. Thats why we need workaround. We will send text to clarin with following lmpn:

any2txt|wcrft2({“morfeusz2”:false})|liner2({“model”:“n82”})|ccl_emo({“lang”:“polish”})

Then we will parse the response, and create new text. In this new every world will be replaced with its base form. And that lemmatized text will be send to clarin once again, to find geolocation.

any2txt|wcrft2({“morfeusz2”:false})|liner2({“model”:“n82”})|geolocation({“limit”:1})

Let’s clear the out file

[94]:
```
for file in list_contents_of_folder(download_path):
    path = os.path.join(download_path,file)
    os.remove(path)
```

[95]:
```
list_contents_of_folder(download_path)
```

[95]:
```
[]
```

[97]:
```
conduct_lammatisation = 'any2txt|wcrft2({"morfeusz2":false})|liner2({"model":
 ↪"n82"})|ccl_emo({"lang":"polish"})'
base_from_zip=send_task_to_service(path_to_file,download_path,conduct_lammatisation)
unzipfile(base_from_zip,download_path)
```

[98]:
```
# extract base form from xml tree and joins them to singe string
def parse_xml_for_base_from_of_word(xml_file):
    tree = ET.parse(xml_file)
    root = tree.getroot()
    base_tags = root.findall('./chunk/sentence/tok/lex/base')
    base_words = list(map(lambda x: x.text,base_tags))
    return ' '.join(base_words)
```

Now this tekst looks like that

[99]:
```
text_in_base_form =␣
 ↪parse_xml_for_base_from_of_word(path_to_extracted_example_file)
text_in_base_form
```

[99]:
```
'Płock - miasto w Polska , na prawo powiat , położyć na pojezierze dobrzyński i
w kotlina płocki , nad Wisła , w województwo mazowiecki , siedziba ziemski
powiat płocki ; historyczny stolica Mazowsze oraz stolica Polska w rok 1079 -
1138 ; siedziba rzymskokatolicki kuria diecezja płocki ( 1075 ) , siedziba
biskup naczelny kościół starokatolicki mariawita , port rzeczny , rafineria ropa
naftowy ( 1964 ) , szkoła wysoki , teatr , muzeum . polski miasto - bohater (
1921 ) .'
```

[136]:
```
base_form_file_path = os.path.join(download_path, "base_form_text.txt")
file = open(base_form_file_path,'w')
file.write(text_in_base_form)
file.close()
```

[138]:
```
geolocation = 'any2txt|wcrft2({"morfeusz2":false})|liner2({"model":
 ↪"n82"})|geolocation({"limit":"1"})'
output_path = os.path.join("geo")
path_to_zip = send_task_to_service(base_form_file_path,output_path,geolocation)
```

[139]:
```
unzipfile(path_to_zip,output_path)
```

[140]:
```
extracted_file_path = os.path.join(output_path,'out%base_form_text.txt')
parse_xml_for_geo_names(extracted_file_path)
```

[140]:
```
['19.7136301805999;52.5353471;administrative;Płock, województwo mazowieckie,
Polska',
 '19.0258159;52.0977181;administrative;Polska',
 '20.0867774;52.380897;river;Wisła, gmina Wyszogród, powiat płocki, województwo
mazowieckie, 43-450, Polska',
 '21.2073404;52.5461934;administrative;województwo mazowieckie, Polska',
 '19.4638889;50.1708333;neighbourhood;Płaczki, Trzebinia, gmina Trzebinia,
powiat chrzanowski, województwo małopolskie, 32-540, Polska',
 '21.2073404;52.5461934;administrative;województwo mazowieckie, Polska']
```

And as can be see that didin't help it this case, but from my expirence it is still worth doing. That addidional step helps to reduce many uncorrect clasiflications. However, not this particular one. This software has its limitations and lots of caution is needed when using it for more responsible applications. It has still very good quality, especilay when copmpared to costs.

[ ]:
```

```