Members group TM3

Selin Acikel 2637714

Tessel Haagen 2826310

Ericka Acosta 2679818

Lab1-Assignment

This notebook describes the assignment for Lab 1.

Points: each exercise is prefixed with the number of points you can obtain for the exercise. However, these points are just an indication of what parts we value more. The assignment itself is assessed as PASS/NO-PASS. In general, we value a critical analysis of the OUTPUT of running code more than just showing that you can create or run the code. So if you succesfully carried out the instructed commands in a notebook you are not done yet. We want you to analyse, understand what the code is doing with language and text. Be critical, think about how to explain what you observe and write this down in the notebook after running the code. It will be stated clearly in the assignment when we expect this from you.

You can make the assignment as a group but make sure that you understand and can carry out the coding yourself as well. You need these skills for your final assignment that is graded individually. Feedback will be given at the group level.

We assume you have worked through the following notebooks:

- **Lab1.1-introduction**
- **Lab1.2-introduction-to-NLTK**
- **Lab1.3-introduction-to-spaCy**

In this assignment, you will process an English text stored in the file (**Lab1-apple-samsung-example.txt**) with both NLTK and spaCy and discuss the similarities and differences.

Who to contact for questions

- Piek Vossen (piek.vossen@vu.nl)

Tip: how to read a file from disk

Let's open the file **Lab1-apple-samsung-example.txt** from disk. It should be located in the same folder as this notebook. The most simple way is to specify the full path to the file, e.g.:

```
path_to_file='/Users/piek/Desktop/t-ONDERWIJS/2021-2022/t-MA-
HLT-introduction-2021/ma-hlt-labs/lab1.toolkits'
```

This may work for me but not for you as it is unlikely that the file has the same path on your machine.

You can locate the file on your disk and adapt the path to file manually. Note that WIndows uses backward slashes while Mac and Linux use forward slashes in the path.

A more advanced method is to use the Path module to find the directory of this notebook. Once we have that, we only need to concatenate the name of the text file to this path. This is how you do this:

In [ ]:
```
from pathlib import Path
```

In [ ]:
```
cur_dir = Path().resolve() # this should provide you with the folder in which th
print('Current directory of this notebook:', cur_dir)

## We can now stick the name of the file to the end of the Path using the *joinp
path_to_file = Path.joinpath(cur_dir, 'Lab1-apple-samsung-example.txt')
print('Path to the text file:', path_to_file)
```

```
Current directory of this notebook: C:\Users\xelfj\OneDrive\Documents\Studie\M TM
1\Blok 1\Introduction to Human Langauge Technology\ma-hlt-labs-master\lab1.toolki
ts
Path to the text file: C:\Users\xelfj\OneDrive\Documents\Studie\M TM 1\Blok 1\Int
roduction to Human Langauge Technology\ma-hlt-labs-master\lab1.toolkits\Lab1-appl
e-samsung-example.txt
```

If you are unsure whether the path is correct, you can check if the file exist on that location:

In [ ]:
```
print('does path exist? ->', Path.exists(path_to_file))
```

```
does path exist? -> True
```

If the output from the code cell above says: **does path exist? -> False**, please check that the file **Lab1-apple-samsung-example.txt** is in the same directory as this notebook. In Jupyter lab you should see it in the file overview panel to the left next to the notebook.

Now we can open the file and access its content. Lets read the complete content and ask for it length using the 'len' function, which will tell us how many characters a string has:

In [ ]:
```
with open(path_to_file, encoding="utf-8") as infile:
    text = infile.read()

print('number of characters', len(text))
```

```
number of characters 1139
```

```
In [ ]: print(text)
```

```
https://www.telegraph.co.uk/technology/apple/9702716/Apple-Samsung-lawsuit-six-mo
re-products-under-scrutiny.html

Documents filed to the San Jose federal court in California on November 23 list s
ix Samsung products running the "Jelly Bean" and "Ice Cream Sandwich" operating s
ystems, which Apple claims infringe its patents.
The six phones and tablets affected are the Galaxy S III, running the new Jelly B
ean system, the Galaxy Tab 8.9 Wifi tablet, the Galaxy Tab 2 10.1, Galaxy Rugby P
ro and Galaxy S III mini.
Apple stated it had “acted quickly and diligently" in order to "determine that th
ese newly released products do infringe many of the same claims already asserted
by Apple."
In August, Samsung lost a US patent case to Apple and was ordered to pay its riva
l $1.05bn (£0.66bn) in damages for copying features of the iPad and iPhone in its
Galaxy range of devices. Samsung, which is the world's top mobile phone maker, is
appealing the ruling.
A similar case in the UK found in Samsung's favour and ordered Apple to publish a
n apology making clear that the South Korean firm had not copied its iPad when de
signing its own devices.
```

We now created a string object with the name 'text' that we can use for the assignment below.

If for some reason, you see weird characters in the text you may have a problem with the character encoding. Computers use different systems to represent scripts. For most languages UTF-8 will work as it has representations for many different characters. In some cases, especially older texts, Latin encodings have been used which works for English and some languages but cannot represent characters in others. For non-Western scripts special encodings have been defined. You never know for sure what encoding a text is in but now-adays most texts are in UTF-8.

What to do if you see weird tokens?

First check if you are really using Python 3.x and not Python 2.x when running the notebook. You can do this using:

```
import platform
print(platform.python_version())
```

If your are running 3.x and still have encoding problems try to open the file as utf-8:

```
with open(path_to_file, encoding=‘utf-8') as infile:
```

Note that when you open a text file in a plain text editor, you never know how it loads the file. The weird characters may still be there or disappear. In some cases, you can try to save the text file again using UTF-8 but this can also corrupt your file. It is wise to make a copy of the file before you try this.

[total points: 3] Exercise 1: NLTK

Extract a list of all the named entity expression from the complete text using NLTK. Note that you do not need to split the text into sentences but you can apply the functions to the complete text.

You result should be a list with only the named entity expressions with their type. For example:

`[(ORGANIZATION San/NNP Jose/NNP), (GPE California/NNP)]`

You need to iterate through the results and store the entities in a result list while skipping the rest.

TIP: Remember that the entities of the ne_chunker are of the type *nltk.tree.tree.Tree* so you may test `if type(element)==nltk.tree.tree.Tree` to keep it in your result list and otherwise to ignore it:

```
nltk_entities = []
for item in list:
      if type(item)==nltk.tree.tree.Tree:
          nltk_entities.append(item)
print(nltk_entities)
```

In [ ]:
```
import nltk
from nltk.chunk import ne_chunk

tokens = nltk.word_tokenize(text)
tokens_pos_tagged = nltk.pos_tag(tokens)
tokens_pos_tagged_and_named_entities = ne_chunk(tokens_pos_tagged)

nltk_entities = [i for i in tokens_pos_tagged_and_named_entities if type(i)==nlt
print(nltk_entities)
```

```
[Tree('ORGANIZATION', [('San', 'NNP'), ('Jose', 'NNP')]), Tree('GPE', [('Californ
ia', 'NNP')]), Tree('ORGANIZATION', [('Samsung', 'NNP')]), Tree('GPE', [('Bean',
'NNP')]), Tree('PERSON', [('Apple', 'NNP')]), Tree('ORGANIZATION', [('Galaxy', 'N
NP')]), Tree('PERSON', [('Jelly', 'NNP'), ('Bean', 'NNP')]), Tree('ORGANIZATION',
[('Galaxy', 'NNP')]), Tree('ORGANIZATION', [('Galaxy', 'NNP')]), Tree('PERSON',
[('Galaxy', 'NNP'), ('Rugby', 'NNP'), ('Pro', 'NNP')]), Tree('PERSON', [('Galax
y', 'NNP'), ('S', 'NNP')]), Tree('PERSON', [('Apple', 'NNP')]), Tree('PERSON',
[('Apple', 'NNP')]), Tree('GPE', [('August', 'NNP')]), Tree('PERSON', [('Samsun
g', 'NNP')]), Tree('GSP', [('US', 'NNP')]), Tree('GPE', [('Apple', 'NNP')]), Tree
('ORGANIZATION', [('iPad', 'NN')]), Tree('ORGANIZATION', [('iPhone', 'NN')]), Tre
e('GPE', [('Galaxy', 'NNP')]), Tree('PERSON', [('Samsung', 'NNP')]), Tree('ORGANI
ZATION', [('UK', 'NNP')]), Tree('GPE', [('Samsung', 'NNP')]), Tree('PERSON', [('A
pple', 'NNP')]), Tree('LOCATION', [('South', 'JJ'), ('Korean', 'JJ')])]
```

[total points: 3] Excercise 2: spaCy

Process the same text using spaCY and extract a list of all the named entity expression according to spaCy.

The output should look as follows:

`[('San Jose', 'GPE'), ('California', 'GPE')]`

Note that you need to combine the text with the label_ of the entities in a tuple. You can combine elements in a tuple by surrounding them with round brackets, for example `(ent.text, ent.label_)` forms a tuple.

In [ ]:
```
import spacy
nlp = spacy.load('en_core_web_sm')

doc = nlp(text)

spacy_entities = [(ent.text, ent.label_) for ent in doc.ents]
print(spacy_entities)
```

```
[('https://www.telegraph.co.uk/technology/apple/9702716/Apple-Samsung-lawsuit-six
-more-products-under-scrutiny.html', 'TIME'), ('San Jose', 'GPE'), ('California',
'GPE'), ('November 23', 'DATE'), ('six', 'CARDINAL'), ('Samsung', 'ORG'), ('the
"Jelly Bean', 'LAW'), ('Apple', 'ORG'), ('six', 'CARDINAL'), ('the Galaxy S III',
'ORG'), ('Jelly Bean', 'ORG'), ('8.9', 'CARDINAL'), ('2 10.1', 'DATE'), ('Galaxy
Rugby Pro', 'ORG'), ('Galaxy S III', 'PERSON'), ('Apple', 'ORG'), ('Apple', 'OR
G'), ('August', 'DATE'), ('Samsung', 'ORG'), ('US', 'GPE'), ('Apple', 'ORG'),
('1.05bn', 'MONEY'), ('0.66bn', 'MONEY'), ('iPad', 'ORG'), ('Galaxy', 'FAC'), ('S
amsung', 'ORG'), ('UK', 'GPE'), ('Samsung', 'ORG'), ('Apple', 'ORG'), ('South Kor
ean', 'NORP'), ('iPad', 'ORG')]
```

[total points: 4] Compare spacy and NLTK

Compare the processing of spaCy and NLTK, answering the following questions:

- The number of sentences
- The number of tokens
- the number of entities

Discuss the quality of the entities for each. Which is better why? Which mistakes are made by both?

In [ ]:
```
print(f"Number of sentences: NLTK={len(nltk.sent_tokenize(text))}, spaCy={len(li

print(f"Number of tokens: NLTK={len(tokens)}, spaCy={len(doc)}")

print(f"Number of entities: NLTK={len(nltk_entities)}, spaCy={len(spacy_entities
```

```
Number of sentences: NLTK=6, spaCy=6
Number of tokens: NLTK=209, spaCy=213
Number of entities: NLTK=25, spaCy=31
```

Discussion:

The first difference is the missing link in the the NLTK entities output. The second difference is that sometimes the nltk entities do not see the whole entity but only parts such as 'bean' instead of 'jelly bean'. This is not the case with spaCy. The third difference, is in the entity 'South Korean'. SpaCy refers this to 'NORP' which is a nationality, religion etc. and nltk has refered it to a location. When we check the context of the text we think it implies the location of the firm more than the nationality. Furthermore, nltk has an issue with naming entities such as 'Apple' or 'Samsung' a PERSON or GPE, while it should have been ORG. SpaCy on the other hand, has no issues with 'Apple' and 'Samsung', however it is labelling iPad and Galaxy as ORG as well.

So, we think Spacy is slightly better and also easier to work with. However, we do think that you have more control over your steps of the pipeline while working with nltk.

End of the assignment 1