Lab1-assignment-toolkits

February 17, 2025

1 Lab1-Assignment

This notebook describes the assignment for Lab 1 of the text mining course.

Points: each exercise is prefixed with the number of points you can obtain for the exercise.

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 (**Lab1-apple-samsung-example.txt**) with both NLTK and spaCy and discuss the similarities and differences.

1.1 Credits

The notebooks in this block have been originally created by Marten Postma. Adaptations were made by Filip Ilievski.

1.2 Tip: how to read a file from disk

Let's open the file **Lab1-apple-samsung-example.txt** from disk.

[1]:
```
from pathlib import Path
```

[2]:
```
cur_dir = Path().resolve() # this should provide you with the folder in which␣
 ↪this notebook is placed
path_to_file = Path.joinpath(cur_dir, 'Lab1-apple-samsung-example.txt')
print(path_to_file)
print('does path exist? ->', Path.exists(path_to_file))
```

```
/Users/iliamarkov/work/VU-TM-2022-changes/ba-text-mining/Lab1-apple-samsung-
example.txt
does path exist? -> False
```

If the output from the code cell above states that **does path exist? -> False**, please check that the file **Lab1-apple-samsung-example.txt** is in the same directory as this notebook.

[3]:
```
with open(path_to_file) as infile:
    text = infile.read()

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

```
number of characters 1139
```

1.3 [total points: 4] Exercise 1: NLTK

In this exercise, we use NLTK to apply **Part-of-speech (POS) tagging**, **Named Entity Recognition (NER)**, and **Constituency parsing**. The following code snippet already performs sentence splitting and tokenization.

[4]:
```
import nltk
from nltk.tokenize import sent_tokenize
from nltk import word_tokenize
```

[5]:
```
sentences_nltk = sent_tokenize(text)
```

[6]:
```
tokens_per_sentence = []
for sentence_nltk in sentences_nltk:
    sent_tokens = word_tokenize(sentence_nltk)
    tokens_per_sentence.append(sent_tokens)
```

We will use lists to keep track of the output of the NLP tasks. We can hence inspect the output for each task using the index of the sentence.

[7]:
```
sent_id = 1
print('SENTENCE', sentences_nltk[sent_id])
print('TOKENS', tokens_per_sentence[sent_id])
```

```
SENTENCE The six phones and tablets affected are the Galaxy S III, running the
new Jelly Bean system, the Galaxy Tab 8.9 Wifi tablet, the Galaxy Tab 2 10.1,
Galaxy Rugby Pro and Galaxy S III mini.
TOKENS ['The', 'six', 'phones', 'and', 'tablets', 'affected', 'are', 'the',
'Galaxy', 'S', 'III', ',', 'running', 'the', 'new', 'Jelly', 'Bean', 'system',
',', 'the', 'Galaxy', 'Tab', '8.9', 'Wifi', 'tablet', ',', 'the', 'Galaxy',
'Tab', '2', '10.1', ',', 'Galaxy', 'Rugby', 'Pro', 'and', 'Galaxy', 'S', 'III',
'mini', '.']
```

1.3.1 [point: 1] Exercise 1a: Part-of-speech (POS) tagging

Use `nltk.pos_tag` to perform part-of-speech tagging on each sentence.

Use `print` to **show** the output in the notebook (and hence also in the exported PDF!).

[8]:
```
pos_tags_per_sentence = []
for tokens in tokens_per_sentence:
    print()
```

[9]:
```
print(pos_tags_per_sentence)
```

```
[]
```

1.3.2 [point: 1] Exercise 1b: Named Entity Recognition (NER)

Use `nltk.chunk.ne_chunk` to perform Named Entity Recognition (NER) on each sentence.

Use `print` to **show** the output in the notebook (and hence also in the exported PDF!).

[ ]:
```
ner_tags_per_sentence = []
```

[ ]:
```
print(ner_tags_per_sentence)
```

1.3.3 [points: 2] Exercise 1c: Constituency parsing

Use the `nltk.RegexpParser` to perform constituency parsing on each sentence.

Use `print` to **show** the output in the notebook (and hence also in the exported PDF!).

[ ]:
```
constituent_parser = nltk.RegexpParser('''
NP: {<DT>? <JJ>* <NN>*} # NP
P: {<IN>}           # Preposition
V: {<V.*>}          # Verb
PP: {<P> <NP>}      # PP -> P NP
VP: {<V> <NP|PP>*}  # VP -> V (NP|PP)*''')
```

[ ]:
```
constituency_output_per_sentence = []
```

[ ]:
```
print(constituency_output_per_sentence)
```

Augment the RegexpParser so that it also detects Named Entity Phrases (NEP), e.g., that it detects *Galaxy S III* and *Ice Cream Sandwich*

[ ]:
```
constituent_parser_v2 = nltk.RegexpParser('''
NP: {<DT>? <JJ>* <NN>*} # NP
P: {<IN>}           # Preposition
V: {<V.*>}          # Verb
PP: {<P> <NP>}      # PP -> P NP
VP: {<V> <NP|PP>*}  # VP -> V (NP|PP)*
NEP: {}             # ???''')
```

[ ]:
```
constituency_v2_output_per_sentence = []
```

[ ]:
```
print(constituency_v2_output_per_sentence)
```

1.4 [total points: 1] Exercise 2: spaCy

Use Spacy to process the same text as you analyzed with NLTK.

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

[ ]:
```
doc = nlp(text) # insert code here
```

small tip: You can use **sents = list(doc.sents)** to be able to use the index to access a sentence like **sents[2]** for the third sentence.

1.5 [total points: 7] Exercise 3: Comparison NLTK and spaCy

We will now compare the output of NLTK and spaCy, i.e., in what do they differ?

1.5.1 [points: 3] Exercise 3a: Part of speech tagging

Compare the output from NLTK and spaCy regarding part of speech tagging.

- To compare, you probably would like to compare sentence per sentence. Describe if the sentence splitting is different for NLTK than for spaCy. If not, where do they differ?
- After checking the sentence splitting, select a sentence for which you expect interesting results and perhaps differences. Motivate your choice.
- Compare the output in `token.tag` from spaCy to the part of speech tagging from NLTK for each token in your selected sentence. Are there any differences? This is not a trick question; it is possible that there are no differences.

[10]:
```
import pandas as pd

pd.set_option('display.max_colwidth', None)

df = pd.read_excel("Group43_lab1/text_mining_1.xlsx")
display(df)
```

```
        NLTK (POS TAGGING)        SpaCy (TOKEN.TAG)  \
0    ('Documents', 'NNS'),  Documents: NNPS (PROPN)
1        ('filed', 'VBN'),        filed: VBD (VERB)
2            ('to', 'TO'),            to: IN (ADP)
3           ('the', 'DT'),           the: DT (DET)
4          ('San', 'NNP'),         San: NNP (PROPN)
5         ('Jose', 'NNP'),        Jose: NNP (PROPN)
6       ('federal', 'JJ'),      federal: JJ (ADJ)\n
7         ('court', 'NN'),         court: NN (NOUN)
8            ('in', 'IN'),            in: IN (ADP)
9   ('California', 'NNP'),  California: NNP (PROPN)
10           ('on', 'IN'),            on: IN (ADP)
11    ('November', 'NNP'),    November: NNP (PROPN)
12           ('23', 'CD'),            23: CD (NUM)
13         ('list', 'NN'),          list: NN (NOUN)
14          ('six', 'CD'),           six: CD (NUM)
15     ('Samsung', 'NNP'),     Samsung: NNP (PROPN)
16    ('products', 'NNS'),     products: NNS (NOUN)
17     ('running', 'VBG'),    running: VBG (VERB)\n
```

```
18          ('the', 'DT'),            the: DT (DET)
19           ('``', '``'),             ": `` (PUNCT)
20        ('Jelly', 'RB'),        Jelly: NNP (PROPN)
21        ('Bean', 'NNP'),         Bean: NNP (PROPN)
22           ("''", "''"),             ": '' (PUNCT)
23          ('and', 'CC'),            and: CC (CCONJ)
24           ('``', '``'),             ": `` (PUNCT)
25         ('Ice', 'NNP'),        Ice: NNP (PROPN)\n
26       ('Cream', 'NNP'),        Cream: NNP (PROPN)
27    ('Sandwich', 'NNP'),     Sandwich: NNP (PROPN)
28           ("''", "''"),             ": '' (PUNCT)
29   ('operating', 'VBG'),      operating: NN (NOUN)
30     ('systems', 'NNS'),      systems: NNS (NOUN)
31             (',', ','),              ,: , (PUNCT)
32       ('which', 'WDT'),       which: WDT (PRON)
33       ('Apple', 'NNP'),       Apple: NNP (PROPN)
34      ('claims', 'VBZ'),      claims: VBZ (VERB)
35      ('infringe', 'VB'),    infringe: VBP (VERB)
36        ('its', 'PRP$'),        its: PRP$ (PRON)
37     ('patents', 'NNS'),     patents: NNS (NOUN)
38           ('.', '.')],              .: . (PUNCT)


                                                   Notes  \
0
                                                     NaN
1
                                                     NaN
2   NLTK just tags to as TO, but SpaCy uses IN to tag preposition or
subordinating conjunction (e.g., "to", "in", "on")*
3
                                                     NaN
4
                                                     NaN
5
                                                     NaN
6
                                                     NaN
7
                                                     NaN
8                                            NLTK has a seperate IN tag, where
SpaCy uses the IN tag for multiple words*
9
                                                     NaN
10
                                                       *
```

```
11    NaN
12    NaN
13    NaN
14    NaN
15    NaN
16    NaN
17    NaN
18    NaN
19    NaN
20    NLTK tags jelly as an adverb, SpaCy tags it as proper noun singular
21    NaN
22    NaN
23    NaN
24    NaN
25    NaN
26    NaN
27    NaN
28    NaN
29    NLTK tags operating as "verb, present participle or gerund", amd SpaCy tags it as  "noun, singular or mass"
30    NaN
31    NaN
32    NaN
33    NaN
```

```
34                                                                             ␣
↪                                    NaN
35                                                                             ␣
↪                                    NaN
36                                                                             ␣
↪                                    NaN
37                                                                             ␣
↪                                    NaN
38                                                                             ␣
↪                                    NaN

    Unnamed: 3                                        Unnamed: 4
0          NaN                                               NaN
1          NaN  * every instance where i noticed the use of the IN tag
2          NaN                                               NaN
3          NaN                                             SPACY
4          NaN                Named Entities, Phrases, and Concepts:
5          NaN                                     San Jose: GPE
6          NaN                                   California: GPE
7          NaN                                 November 23: DATE
8          NaN                                    six: CARDINAL
9          NaN                                     Samsung: ORG
10         NaN                          Jelly Bean: WORK_OF_ART
11         NaN                                       Apple: ORG
12         NaN                                               NaN
13         NaN                                               NaN
14         NaN                                               NaN
15         NaN                                               NaN
16         NaN                                               NaN
17         NaN                     (ORGANIZATION San/NNP Jose/NNP)
18         NaN                              (GPE California/NNP)
19         NaN                        (ORGANIZATION Samsung/NNP)
20         NaN                                    (GPE Bean/NNP)
21         NaN                               (PERSON Apple/NNP)
22         NaN                                               NaN
23         NaN                                               NaN
24         NaN                                               NaN
25         NaN                                               NaN
26         NaN                                               NaN
27         NaN                                               NaN
28         NaN                                               NaN
29         NaN                                               NaN
30         NaN                                               NaN
31         NaN                                               NaN
32         NaN                                               NaN
33         NaN                                               NaN
34         NaN                                               NaN
```

35	NaN	NaN
36	NaN	NaN
37	NaN	NaN
38	NaN	NaN

1.5.2 [points: 2] Exercise 3b: Named Entity Recognition (NER)

- Describe differences between the output from NLTK and spaCy for Named Entity Recognition. Which one do you think performs better?

We looked at the first sentence to compare the outputs from both NLTK and SpaCY for Named Entity Recognition and we believe that SpaCy works better. For example, SpaCy recognized San Jose as a GPE (geopolitical entity) while NLTK recognized it as an organization. NLTK also did not recognize November 23 as a date, while SpaCy did. The two did classify Samsung correctly as an organization. However, both also misclassified Jelly Bean, SpaCy giving us that its a WORK_OF_ART and NLTK recognizing only Bean as a GPE. Lastly, SpaCy classified Apple as an organization while NLTK classified it as a person. Though both are clearly not perfect, we felt that SpaCY made less mistakes overall.

1.5.3 [points: 2] Exercise 3c: Constituency/dependency parsing

Choose one sentence from the text and run constituency parsing using NLTK and dependency parsing using spaCy. * describe briefly the difference between constituency parsing and dependency parsing * describe differences between the output from NLTK and spaCy.

Constituency parsing breaks a sentence into hierachical tree of phrases based on grammar rules. It represent the syntactic structure of the sentence by identifying noun phrases, verb phrases and other components. On the other hand, dependency parsing represents a sentence in terms of dependencies between words, showing grammatical relationships. Each word is connected to a headword forming a directed graph. NLTK uses the constituency based approach while spaCy uses a dependency based approach. In the output there are still some differences between the two due to the two different approaches. In 1 NLTK treats it as a plural noun while spaCy as a proper noun plural, in 2 NLTK tags it as past participle while spaCy as a simple past verb, in 3 NLTK tags it as TO while spaCy as ADP which means adposition, in 21 NLTK tags it as an adverb while spaCy as a proper noun and then as last in 30 NLTK tags it as a verb while spaCy as a noun.

2 End of this notebook