Homework1

September 28, 2020

1 Homework 1

1.1 Problem 1

[179]:
```
import timeit
import re
import spacy

from urllib import request
# Tokenization
from nltk.tokenize import word_tokenize
from spacy.tokenizer import Tokenizer
from spacy.lang.en import English
# Stemming / Lemmatization
from nltk.stem import WordNetLemmatizer
```

[238]:
```
# Sourcing this text document which consists of the book Crime and Punishment
# This is part of the nltk book instructions found here: https://www.nltk.org/
 ↪book/ch03.html
url = "http://www.gutenberg.org/files/2554/2554-0.txt"
response = request.urlopen(url)
raw = response.read().decode('utf8')
len(raw)
```

[238]:
```
1176967
```

1.1.1 Tokenization

[159]:
```
# with regex
re_tokens = re.findall('\w+', raw)

%timeit re.findall('\w+', raw)
```

```
54.4 ms ± 415 µs per loop (mean ± std. dev. of 7 runs, 10 loops each)
```

[175]:
```
print('Number of words from regex: ', len(re_tokens))
print('Size of vocabulary: ', len(set(re_tokens)))
```

```
Number of words from regex:  212001
Size of vocabulary:  10729
```

[161]:
```
# with nltk
nltk_tokens = word_tokenize(raw)

%timeit word_tokenize(raw)
```

```
1.27 s ± 17.4 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)
```

[174]:
```
print('Number of words from nltk: ', len(nltk_tokens))
print('Size of vocabulary: ', len(set(nltk_tokens)))
```

```
Number of words from regex:  257727
Size of vocabulary:  11539
```

[235]:
```
# A sample of tokens captured by re, but not by nltk
print(list(set(re_tokens)-set(nltk_tokens))[:50])
```

```
['passers', '64', 'menage_', 'india', 'Tailor', 'tête', 'merry', 'tilted',
'approval', 'F', 'abasement', 'sparrow', 'turvy', 'cannot', 'hearted', '6',
'self', 'reliance', '6221541', '7', 'Strong', 'sill', 'knacks', 'Literally',
'topsy', 'serfdom', 'crest', 'Holstein', 'E', 'Cannot', 'maidish', 'cheeked',
'sized', 'skinned', 'UTF', 'ton', 'comer', 'tray', 'Folk', 'flint', 'chop',
'riff', '_Pani_', 'passer', 'ant', 'bye', 'bred', 'raff', 'intentioned',
'freshly']
```

[236]:
```
# A sample of tokens captured by nltk, but not by re
print(list(set(nltk_tokens)-set(re_tokens))[:50])
```

```
['reason.', 'solid-looking', 'late.', 'off.', 'common-sense', 'minute.', 'Tut-
tut-tut', 'deal.', 'well-to-do', 'unfortunate.', 'night.', 'Cough-cough',
'alone.', 'dressed.', 'half-way', '\ufeffThe', 'ill-treated', 'well.', 'began.',
'half-educated', 'egg-shells', 'detail.', 'five-and-thirty', 'funeral.',
'attention.', 'Svidrigaïlov.', 'king.', 'poor-looking', 'cart-horses', 'bell-
ringing', 'truth-like', 'free-thinking', 'healthy-looking', 'old-maidish',
'listen.', 'then.', 'minutes.', 'pot-house', 'promise.', 'initial.', 'grown-up',
'yes.', 'hours.', 'begin.', 'pavement.', 'He-he', 'colours.', 'quietly.',
'eating-houses', '[']
```

[151]:
```
# with SpaCy
# Initializing English language
nlp = English()

# blank Tokenizer with just the English vocab
tokenizer = Tokenizer(nlp.vocab)
```

[164]:
```
spacy_tokens = tokenizer(raw)

%timeit tokenizer(raw)
```

```
403 ms ± 46.2 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)
```

[189]:
```
print('Number of words from spacy: ', len(spacy_tokens))
```

```
Number of words from spacy:  224902
```

Observation Regex are much faster, but catch way fewer words, which is probably a problem.

1.1.2 Stemming / Lemmatization

[171]:
```
# NLTK
wordnet_lemmatizer = WordNetLemmatizer()

lemmatized_tokens = [wordnet_lemmatizer.lemmatize(token) for token in␣
 ↪nltk_tokens]
len(set(lemmatized_tokens))
```

[171]:
```
10662
```

[169]:
```
%timeit [wordnet_lemmatizer.lemmatize(token) for token in nltk_tokens]
```

```
663 ms ± 5.78 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)
```

1.1.3 Part of Speech Tagging

[243]:
```
nlp = spacy.load("en_core_web_sm") # small english model

'''
When I tried to directly use the English model on the raw text
I get an error that the text is larger than 1,000,000, which is the limit.
I am warned that I need ~1GB of temporary memory per 100,000 characters of␣
 ↪text, if using the entire pipeline
The model has 3 steps in its pipeline, which execute after the tokenizer:
tokenizer --> tagger --> parser --> ner
So, I will disable the parser and ner, leaving just the (POS) tagger
'''

nlp.disable_pipes('parser', 'ner')
nlp.max_length = 1_500_000 # increasing maximum number of characters
doc = nlp(raw)
```

[244]:
```
%timeit nlp(raw)
```

```
5.78 s ± 143 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)
```

Note: I originally tried just the tokenizer (without the tagger) and the average was ~109ms. In other words, including the tagger added ~5.5 seconds to the runtime. A significant slowdown!

[245]:
```
len(doc)
```

[245]: 274765

[246]:
```
# Let's look at some of the interesting things that we got for a random word
print('Text: ', doc[1204].text)
print('Lemma: ', doc[1204].lemma_)
print('POS: ', doc[1204].pos_)
print('Tag: ', doc[1204].tag_)
print('Shape: ', doc[1204].shape_)
print('Dependency: ', doc[1204].dep_, ' # This is None, because we disabled the␣
 ↪parser!')
```

```
Text:  dinners
Lemma:  dinner
POS:  NOUN
Tag:  NNS
Shape:  xxxx
Dependency:    # This is None, because we disabled the parser!
```

[247]:
```
# It's pretty cool how you can ask SpaCy to explain an acronym!
spacy.explain('NNS')
```

[247]: 'noun, plural'

1.2 Problem 2

1.2.1 Problem 2.1 - Regular Expression for all emails in text

[248]:
```
text = '''This is some sample text with some @email addresses.
Note that in the previous sentence there is the special symbol "@", which is␣
 ↪typical of emails.
However, there is no word in front of it. It would also be problematic if we␣
 ↪have @ without anything after it either.
My regex should capture my own email: kristiyan.t.dimitrov@gmail.com. I need to␣
 ↪be careful to capture all the full-stop-separated words;
as well as the full-stop at the end.
Here is another email@example.com, just for testing purposes.
Also, what about kristiyan_dimitrov@mail.com'''
```

[250]:
```
pattern = r'[a-zA-Z0-9_\.\#\!\?]+@\w+\.\w+'
re.findall(pattern, text)
```

[250]:
```
['kristiyan.t.dimitrov@gmail.com',
 'email@example.com',
 'kristiyan_dimitrov@mail.com']
```

1.2.2 Problem 2.2 - Dates

[117]:
```
dates = '''This is some text with dates such as 04/12/2019 or another date␣
↪being April 20th 2019 or December 23th 2020, or even what we do in Bulgaria␣
↪which is 23.12.1991'''

pattern1 = r'\d{1,2}/\d{1,2}/\d{4}'
re.findall(pattern1, dates)
```

[117]:
```
['04/12/2019']
```

[121]:
```
pattern2 = r'\w+ \d{2}\w{2} \d{4}'
re.findall(pattern2, dates)
```

[121]:
```
['April 20th 2019', 'December 23th 2020']
```

[122]:
```
pattern3 = r'\d{2}\.\d{2}\.\d{4}'
re.findall(pattern3, dates)
```

[122]:
```
['23.12.1991']
```