TokenizationExamples

January 25, 2024

1 SpaCy tokenization and other examples

[7]:
```
import spacy

nlp = spacy.load("en_core_web_sm")
doc = nlp("MS is looking at buying U.K. startup for $11.1 million.")
for token in doc:
  print(token.text, token.lemma_, token.pos_, token.tag_, token.dep_,
        token.shape_, token.is_alpha, token.is_stop)
```

```
MS MS PROPN NNP nsubj XX True False
is be AUX VBZ aux xx True True
looking look VERB VBG ROOT xxxx True False
at at ADP IN prep xx True True
buying buy VERB VBG pcomp xxxx True False
U.K. U.K. PROPN NNP dobj X.X. False False
startup startup NOUN NN dep xxxx True False
for for ADP IN prep xxx True True
$ $ SYM $ quantmod $ False False
11.1 11.1 NUM CD compound dd.d False False
million million NUM CD pobj xxxx True False
. . PUNCT . punct . False False
```

1.0.1 Display syntactic dependencies

[8]:
```
import spacy
from spacy import displacy

nlp = spacy.load("en_core_web_sm")
doc = nlp("Microsoft plans to buy Activision for $69 billion.")
displacy.render(doc, style = "dep")
```

```
<IPython.core.display.HTML object>
```

[9]:
```
import spacy
from spacy import displacy

nlp = spacy.load("en_core_web_sm")
```

```
doc = nlp("Microsoft plans to buy Activision for $69 billion.")
displacy.render(doc, style = "ent")
```

```
<IPython.core.display.HTML object>
```

1.0.2 Use directly the tokenizer component

[17]:
```
from spacy.lang.en import English

nlp = English()
tokenizer = nlp.tokenizer
tokens = tokenizer("U.S. economy is healing, but there's a long way to go. "
                   "The spread of Covid-19 led to surge in orders for factory␣
 ↪robots."
                  "Fine-tuning models is time-consuming.")

for token in tokens:
  print(token.text, end = ' ')
print()
```

```
U.S. economy is healing , but there 's a long way to go . The spread of Covid-19
led to surge in orders for factory robots . Fine - tuning models is time -
consuming .
```

[11]:
```
from spacy.lang.en import English

nlp = English()
tokenizer = nlp.tokenizer
tokens = tokenizer("I think what she said is soooo craaaazy!")
for token in tokens:
  print(token.text, end = ' ')
print()
```

```
I think what she said is soooo craaaazy !
```

1.0.3 Use only the tokenizer component by disabling other pipeline modules

[19]:
```
import spacy
nlp = spacy.load("en_core_web_sm", exclude = ['tagger, ner, parser'])

doc = nlp("U.S. economy is healing, but there's a long way to go. "
          "The spread of Covid-19 led to surge in orders for factory robots.")

for token in doc:
  print(token.text, end = ' ')
print()
print()
```

```
for sent in doc.sents:
    for token in sent:
        print(token.text, end = ' ')
    print()
```

```
U.S. economy is healing , but there 's a long way to go . The spread of Covid-19
led to surge in orders for factory robots .

U.S. economy is healing , but there 's a long way to go .
The spread of Covid-19 led to surge in orders for factory robots .
```

1.0.4 Use a special sentencizer that does not require syntactic parsing, for efficiency

[25]:
```
from spacy.lang.en import English

nlp = English()
nlp.add_pipe("sentencizer")

doc = nlp("U.S. economy is healing, but there's a long way to go. "
          "The spread of Covid-19 led to surge in orders for factory robots.")

# Print tokens, one sentence per line.
for sent in doc.sents:
  for token in sent:
    print (token, end = ' ')
  print()
```

```
U.S. economy is healing , but there 's a long way to go .
The spread of Covid-19 led to surge in orders for factory robots .
```

1.0.5 By default, it seems spaCy creates tokens for newline characters

[23]:
```
nlp = spacy.load("en_core_web_sm")

stanza = "I am a contract-drafting em,\n" \
  "The loyalest of lawyers!\n" \
  "I draw up terms for deals 'twixt firms\n" \
  "To service my employers!"
print(stanza, '\n')

doc = nlp(stanza)

print(f"Stanza has {len(list(doc.sents))} sentences.\n")

# Print tokens, one sentence per line.
for sent in doc.sents:
    for token in sent:
        if token.text == '\n':
            print('<NL>', end = ' ')
        else:
            print(token, end = ' ')
    print()
```

```
I am a contract-drafting em,
The loyalest of lawyers!
I draw up terms for deals 'twixt firms
To service my employers!

Stanza has 2 sentences.

I am a contract - drafting em , <NL> The loyalest of lawyers ! <NL>
I draw up terms for deals ' twixt firms <NL> To service my employers !
```

1.0.6 Replace newlines with white spaces

[24]:
```
stanza = "I am a contract-drafting em, " \
  "The loyalest of lawyers! " \
  "I draw up terms for deals 'twixt firms " \
  "To service my employers!"
print(stanza)
print()

doc = nlp(stanza)

print(f"Stanza has {len(list(doc.sents))} sentences.\n")

# Print tokens, one sentence per line.
for sent in doc.sents:
  for token in sent:
    print (token, end = ' ')
  print()

#for token in doc:
#  print(token.text, token.lemma_, token.pos_, token.tag_, token.dep_,
#         token.shape_, token.is_alpha, token.is_stop)
```

```
I am a contract-drafting em, The loyalest of lawyers! I draw up terms for deals
'twixt firms To service my employers!

Stanza has 2 sentences.

I am a contract - drafting em , The loyalest of lawyers !
I draw up terms for deals ' twixt firms To service my employers !
```

2 GPT Tokenization using BPE

[26]:
```
# Uncomment this line if tiktoken is not yet installed on your machine.
#!pip install tiktoken

import tiktoken

# To get the tokeniser corresponding to a specific model in the OpenAI API:
enc = tiktoken.encoding_for_model("gpt-4")

tokens = [enc.decode_single_token_bytes(token) for token in enc.encode("His \t␣
↪\n \n\t  ambivalence was perplexing.")]
#tokens = enc.decode_single_token_bytes(enc.encode("His ambivalence was␣
↪perplexing."))
print(tokens)
```

```
[b'His', b' \t', b' \n \n', b'\t ', b' amb', b'ivalence', b' was', b' perplex',
b'ing', b'.']
```

[28]:
```
tokens = [enc.decode_single_token_bytes(token) for token in enc.encode("I think␣
↪what she said is soooo craaaazy!")]
print(tokens)
```

```
[b'I', b' think', b' what', b' she', b' said', b' is', b' so', b'ooo', b' cra',
b'aa', b'azy', b'!']
```

[29]:
```
tokens[1].strip().decode('ASCII')
```

[29]:
```
'think'
```

[30]:
```
tokens = [enc.decode_single_token_bytes(token) for token in enc.encode("The␣
↪perplexing cat sat on the mat.")]
print(tokens)
```

```
[b'The', b' perplex', b'ing', b' cat', b' sat', b' on', b' the', b' mat', b'.']
```

[ ]: