Open in Colab

Homework 1: Tokenization

In this homework, you'll compare the tokenizations outputs from different classes of tokenizers. This homework is also an opportunity for you to check in on your Python proficiency; for all of the operations below (downloading a file, reading it in, counting objects), you should either be comfortable implementing them already or know how to find out how to do so yourself (if you find yourself struggling with them, we encourage you to take this class at a later date, with a bit more Python experience under your belt).

We've added some space for you to write the code for each section, but feel free to create more code cells if you'd like.

Part 1

Tokenize the following document with each of these models. Feel free to use the documentation linked (and AI Assistance) to do so for this low-level operation (but again remember that you have to be able to explain what your code is doing). For each of the tokenizers above, we want to see a list of tokens for this document (not numeric token IDs, but legible words) -- e.g., ["London", ".", ...]

- NLTK `word_tokenize` (https://www.nltk.org/book/ch03.html)
- Spacy `tokenize` (https://spacy.io/usage/spacy-101#annotations-token)
- Tiktoken BPE tokenization (https://github.com/openai/tiktoken) -- cl100k_base (GPT-3.5, GPT-4).

```
# download the packages first here
!pip install nltk
import nltk

nltk.download('punkt_tab')
import spacy
nlp = spacy.blank("en")
!pip install tiktoken
import tiktoken
```

```
Requirement already satisfied: nltk in /usr/local/lib/python3.12/dist-packages (3.9.1)
Requirement already satisfied: click in /usr/local/lib/python3.12/dist-packages (from nltk) (8.2.1)
Requirement already satisfied: joblib in /usr/local/lib/python3.12/dist-packages (from nltk) (1.5.2)
Requirement already satisfied: regex>=2021.8.3 in /usr/local/lib/python3.12/dist-packages (from nltk) (2024.11.6)
Requirement already satisfied: tqdm in /usr/local/lib/python3.12/dist-packages (from nltk) (4.67.1)
[nltk_data] Downloading package punkt_tab to /root/nltk_data...
[nltk_data]   Package punkt_tab is already up-to-date!
Requirement already satisfied: tiktoken in /usr/local/lib/python3.12/dist-packages (0.11.0)
Requirement already satisfied: regex>=2022.1.18 in /usr/local/lib/python3.12/dist-packages (from tiktoken) (2024.11.6)
Requirement already satisfied: requests>=2.26.0 in /usr/local/lib/python3.12/dist-packages (from tiktoken) (2.32.4)
Requirement already satisfied: charset_normalizer<4,>=2 in /usr/local/lib/python3.12/dist-packages (from requests>=2.26.
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.12/dist-packages (from requests>=2.26.0->tiktoken)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.12/dist-packages (from requests>=2.26.0->tik
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.12/dist-packages (from requests>=2.26.0->tik
```

```
document = "London. Michaelmas term lately over, and the Lord Chancellor sitting in Lincoln’s Inn Hall. Implacable November

# Your code here:

# NLTK Tokenizer
from nltk.tokenize import sent_tokenize, word_tokenize

def tokenize_nltk(text):
    tokens = word_tokenize(text)
    return tokens

# Spacy Tokenizer

def tokenize_spacy(text):
    doc = nlp(text)
    tokens = [token.text for token in doc]
    return tokens


# Tiktoken BPE tokenization

def tokenize_tiktoken(text):
  enc = tiktoken.get_encoding("cl100k_base")
  token_ids_bpe = enc.encode(text)
  print(token_ids_bpe)
```

```
  # Since these are token ids, we need to decode them to various token in an array
  # Decode each token ID into its string form
  words_bpe = [enc.decode_single_token_bytes(tok).decode("utf-8", errors="replace") for tok in token_ids_bpe]
  return words_bpe

words_nltk = tokenize_nltk(document)
print("These are the NLTK Tokens",words_nltk)
words_spacy = tokenize_spacy(document)
print("These are the Spacy Tokens",words_spacy)
words_bpe = tokenize_tiktoken(document)
print("These are the BPE Tokens",words_bpe)
```

```
These are the NLTK Tokens ['London', '.', 'Michaelmas', 'term', 'lately', 'over', ',', 'and', 'the', 'Lord', 'Chancellor
These are the Spacy Tokens ['London', '.', 'Michaelmas', 'term', 'lately', 'over', ',', 'and', 'the', 'Lord', 'Chancello
[40672, 13, 8096, 7044, 4751, 31445, 927, 11, 323, 279, 10425, 48063, 11961, 304, 25379, 753, 17382, 11166, 13, 89192, 5
These are the BPE Tokens ['London', '.', ' Michael', 'mas', ' term', ' lately', ' over', ',', ' and', ' the', ' Lord', '
```

Part 2

Examine the different tokenizations for the passage above -- i.e., actually read through them and see how they differ. In a paragraph or two, characterize the salient differences in tokenization between a.) NLTK and Spacy and b.) NLTK and BPE. Reference real examples in the text. At the end of this homework, you want to be able to discuss the practical differences between tokenization methods.

```
# Get diff of words_nltk and words_spacy I want two arrays, the one in words_nltk and not in words spacy and vice versa
diff_nltk_spacy = set(words_nltk) - set(words_spacy)
print("Words present in NLTK and NOT in Spacy", diff_nltk_spacy)

diff_spacy_nltk = set(words_spacy) - set(words_nltk)
print("Words present in SPacy and NOT in NLTK", diff_spacy_nltk)



# Get diff of words_nltk and words_bpe
diff_nltk_bpe = set(words_nltk) - set(words_bpe)
diff_bpe_nltk = set(words_bpe) - set(words_nltk)
print("Words present in NLTK and NOT in BPE", diff_nltk_bpe)
print("Words present in BPE and NOT in NLTK", diff_bpe_nltk)
```

```
Words present in NLTK and NOT in Spacy {'street-corners', '’', 's', 'full-grown', 'foot-hold', 'chimney-pots', 'snowflak
Words present in SPacy and NOT in NLTK {'chimney', '-', 'pots', 'grown', 'corners', '’s', 'snowflakes', 'hold', 'full',
Words present in NLTK and NOT in BPE {'waddling', 'losing', 'one', 'splashed', 'drizzle', 'sticking', 'at', 'lately', 'H
Words present in BPE and NOT in NLTK {' lowering', ' lately', ' day', ' Lord', ' but', ' long', 'ers', ' ten', ' spl', '
```

Response:

A. From these two arrays:
Words present in NLTK and NOT in Spacy {'street-corners', '’', 's', 'full-grown', 'foot-hold', 'chimney-pots', 'snowflakes—gone'} Words present in SPacy and NOT in NLTK {'chimney', '-', 'pots', 'grown', 'corners', '’s', 'snowflakes', 'hold', 'full', 'street', 'gone', '—'}

We can see that for NLTK, it does treat hyphenated words as a single token while Spacy does treat it as a single token. Example are words like '**snowflakes—gone**' which in NLTK, it is present while in SPacy, it is broken into snowflakes and gone. Same case as '**street-corners**' ''**foot-hold**' , '**full-grownd**''**chimney-pots**'words.

B. From these arrays: Words present in NLTK and NOT in BPE {'waddling', 'losing', 'one', 'splashed', 'drizzle', 'sticking', 'at', 'lately', ……..}
Words present in BPE and NOT in NLTK {' lowering', ' lately', ' day', ' Lord', ' but', ' long', 'ers', ' ten', ' spl', '—', ' ……..}

We can see that there is so much difference between NLTK and BPE. One of the key diffrenece observed is the treatment of space as a token in BPE and not in NLTK. Eg, the word 'Lord' is present in NLTK but in BPE it is ' Lord' with a space infront. Same case with other words like 'lately' and ' lately'.

Part 3

Download the full text of *Pride and Prejudice* (https://raw.githubusercontent.com/dbamman/anlp25/main/data/1342_pride_and_prejudice.txt) and tokenize it using each of the methods above. How many word types (in the formal sense we discussed in class) does each tokenization method have for that complete file?

```
# Your code here:

# download from https://raw.githubusercontent.com/dbamman/anlp25/main/data/1342_pride_and_prejudice.txt
```

```
def download_file(url):
  import requests

  try:
      response = requests.get(url)
      response.raise_for_status() # Raise an exception for bad status codes (4xx or 5xx)

      # Read the content as text (for text files)
      file_content = response.text
      return file_content
  except requests.exceptions.RequestException as e:
      print(f"Error retrieving file: {e}")

pride_prejudice_file_content =download_file("https://raw.githubusercontent.com/dbamman/anlp25/main/data/1342_pride_and_preju
words_nltk_file = tokenize_nltk(pride_prejudice_file_content)
print("These are the NLTK Tokens",words_nltk_file)
words_spacy_file = tokenize_spacy(pride_prejudice_file_content)
print("These are the Spacy Tokens",words_spacy_file)
words_bpe_file = tokenize_tiktoken(pride_prejudice_file_content)
print("These are the BPE Tokens", words_bpe_file)
print("    ")
print("    ")
print("    ")


##  We know types are the unique words in each set

unique_words_nltk = set(words_nltk_file)
unique_words_spacy = set(words_spacy_file)
unique_words_bpe = set(words_bpe_file)

# Print the lengths of the above sets
print("Number of unique words or word types in NLTK:", len(unique_words_nltk))
print("Number of unique words or word types in Spacy:", len(unique_words_spacy))
print("Number of unique words or word types in BPE:", len(unique_words_bpe))
```

```
These are the NLTK Tokens ['Chapter', '1', 'It', 'is', 'a', 'truth', 'universally', 'acknowledged', ',', 'that', 'a', 's
These are the Spacy Tokens ['Chapter', '1', '\n\n\n', 'It', 'is', 'a', 'truth', 'universally', 'acknowledged', ',', 'tha
[26072, 220, 16, 1432, 2181, 374, 264, 8206, 61528, 26579, 11, 430, 264, 3254, 893, 304, 19243, 198, 1073, 264, 1695, 33
These are the BPE Tokens ['Chapter', ' ', '1', '\n\n\n', 'It', ' is', ' a', ' truth', ' universally', ' acknowledged', '
```

```
Number of unique words or word types in NLTK: 7475
Number of unique words or word types in Spacy: 6780
Number of unique words or word types in BPE: 8364
```

Part 4

Which text has the greater type-token ratio, *Pride and Prejudice* (https://raw.githubusercontent.com/dbamman/anlp25/main/data/1342_pride_and_prejudice.txt) or *Emma* (https://raw.githubusercontent.com/dbamman/anlp25/main/data/158_emma.txt)? Calculate the TTR for both texts using the NLTK tokenizer, but only use the first 1,000 tokens from each text when calculating its TTR.

```
# Your code here:

# Get the first 1000 tokens from the txt file

emma_file_content =download_file("https://raw.githubusercontent.com/dbamman/anlp25/main/data/158_emma.txt")
words_nltk_file_emma = tokenize_nltk(emma_file_content)

# Get the first 1000 in both arrays
words_nltk_file_emma = words_nltk_file_emma[:1000]
words_nltk_file_pride_prejudice = words_nltk_file[:1000]

print("    ")
print("    ")


pride_and_prejudice_type = set(words_nltk_file_pride_prejudice)
emma_type = set(words_nltk_file_emma)
print("Number of unique words or word types in pride and prejudice file:", len(pride_and_prejudice_type))
print("Number of unique words or word types in Emma file:", len(emma_type))
```

```
    Number of unique words or word types in pride and prejudice file: 360
    Number of unique words or word types in Emma file: 410
```

```
pp_ttr = len(pride_and_prejudice_type)/1000
emma_ttr = len(emma_type)/1000
answer = "Pride and Prejudice file" if pp_ttr > emma_ttr else "Emma file"

print("The TTR for 'Pride and Prejudice' is", pp_ttr)
print("The TTR for 'Emma' is", emma_ttr)
print(f"{answer} has the higher TTR.")
```

```
The TTR for 'Pride and Prejudice' is 0.36
The TTR for 'Emma' is 0.41
Emma file has the higher TTR.
```