TokenizationExamples

January 20, 2026

1 BPE Tokenization using `tiktoken`

BPE tokenization is done to using statistics over a large corpus of text in order to determine how strings should be segmented into *subword* tokens. Here we use the `tiktoken` tokenizer from OpenAI.

[1]:
```
# Uncomment this line if tiktoken is not yet installed on your machine.
#!pip install tiktoken

import tiktoken

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

# The .encode() method converts a text string into a list of token integers.
ltokens = enc.encode("soooo much rrrracing in Kannapolis this Summer!")
print(ltokens)
```

```
[708, 39721, 1790, 436, 637, 81, 4628, 304, 78311, 24751, 420, 19367, 0]
```

[2]:
```
# The .decode() method converts a list of token integers to a string.
enc.decode(ltokens)
```

[2]:
```
'soooo much rrrracing in Kannapolis this Summer!'
```

[3]:
```
# The .decode_single_token_bytes() method safely converts a single integer␣
 ↪token to the bytes it represents.
tokens = [enc.decode_single_token_bytes(token) for token in ltokens]
print(tokens)
```

```
[b'so', b'ooo', b' much', b' r', b'rr', b'r', b'acing', b' in', b' Kann',
b'apolis', b' this', b' Summer', b'!']
```

[4]:
```
# We usually combine .encode() with .decode_single_token_bytes() into one list␣
 ↪comprehension
#    to get the list of tokens as byte strings.
tokens = [enc.decode_single_token_bytes(token) for token in enc.encode("soooo␣
 ↪much rrrracing in Kannapolis this Summer!")]
```

```
# Note the 'b' in front of each string, which means that the string you see is␣
  ↪a sequence of bytes.
print(tokens)
```

```
[b'so', b'ooo', b' much', b' r', b'rr', b'r', b'acing', b' in', b' Kann',
b'apolis', b' this', b' Summer', b'!']
```

[5]:
```
# To translate to the standard representation (utf-8), you can use token.
  ↪decode('utf-8').
utf8_tokens = [token.decode('utf-8') for token in tokens]
print(utf8_tokens)
```

```
['so', 'ooo', ' much', ' r', 'rr', 'r', 'acing', ' in', ' Kann', 'apolis', '
this', ' Summer', '!']
```

[ ]:
```

```

[6]:
```
tokens = [enc.decode_single_token_bytes(token) for token in enc.encode("I think␣
  ↪what she said is soooo craaaazy!")]
print(tokens)

tokens[1].strip().decode('utf-8')
```

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

[6]:
```
'think'
```

[7]:
```
# Another example showing subword tokens.
tokens = [enc.decode_single_token_bytes(token) for token in enc.encode("The␣
  ↪perplexing cat sat on the mat.")]
print(tokens)

utf8_tokens = [token.decode('utf-8') for token in tokens]
print(utf8_tokens)
```

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

1.1 Tokenization of multiple lines of text

Let's try `tiktoken` on the first stanza of The Contract Drafting Em.

[13]:
```
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!\n" \
  "I like drafting poems.\n"
print(stanza, '\n')
```

```
tokens = [enc.decode_single_token_bytes(token) for token in enc.encode(stanza)]
utf8_tokens = [token.decode('utf-8') for token in tokens]
print(utf8_tokens)
```

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


['I', ' am', ' a', ' contract', '-d', 'raft', 'ing', ' em', ',\n', 'The', ' lo',
'y', 'ale', 'st', ' of', ' lawyers', '!\n', 'I', ' draw', ' up', ' terms', '
for', ' deals', " 'twixt", ' firms', '\n', 'To', ' service', ' my',
' employers', '!\n', 'I', ' like', ' drafting', ' poems', '.\n']
```

[12]:
```
# Let's see how tiktoken deals with different types of white space.
tokens = [enc.decode_single_token_bytes(token) for token in enc.encode("His \t␣
 ↪\n \n\t     \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'\t ', b' amb', b'ivalence', b' was',
b'  ', b' perplex', b'ing', b'.']
```

[ ]:
```

```