4.10 Code Exercise 4

Max Ryoo (hr2ee)

Set Up

Configs

```
In [1]: OHCO = ['book_id', 'chap_num', 'para_num', 'sent_num', 'token_num']
        epub_dir = 'epubs'
```

Imports

```
In [2]: #pip install nltk
```

```
In [3]: import pandas as pd
        import numpy as np
        from glob import glob
        import re
        import nltk
```

```
In [4]: %matplotlib inline
```

```
In [5]: nltk.download('punkt')
        nltk.download('averaged_perceptron_tagger')
        nltk.download('stopwords')
        nltk.download('tagsets')
```

```
[nltk_data] Downloading package punkt to /Users/maxryoo/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     /Users/maxryoo/nltk_data...
[nltk_data]   Package averaged_perceptron_tagger is already up-to-
[nltk_data]       date!
[nltk_data] Downloading package stopwords to
[nltk_data]     /Users/maxryoo/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package tagsets to /Users/maxryoo/nltk_data...
[nltk_data]   Package tagsets is already up-to-date!
```

```
Out[5]: True
```

Inspecting

We will be looking at the three books

- Middlemarch http://www.gutenberg.org/files/145/145-0.txt
- The Mill on the Floss http://www.gutenberg.org/files/6688/6688-0.txt
- Adam Bede http://www.gutenberg.org/files/507/507-0.txt

```
In [6]: roman = '[IVXLCM]+'
```

```
caps = "[A-Z';, -]+"
chap_pats = {
    145: {
        'start_line': 23,
        'end_line': 33311,
        'volume': re.compile('^\s*BOOK\s+{}\s*$'.format(roman)),
        'chapter': re.compile('^\s*CHAPTER\s+{}\.s*$'.format(roman))
    },
    6688: {
        'start_line': 24,
        'end_line': 21270,
        # 'volume': re.compile('^\s*BOOK\s+{}\s*$'.format(roman)),
        'chapter': re.compile('^\s*Chapter\s+{}\.s*$'.format(roman))
    },
    507: {
        'start_line': 24,
        'end_line': 20702,
        # 'volume': re.compile('^\s*BOOK\s+{}\s*$'.format(roman)),
        'chapter': re.compile('^\s*Chapter\s+{}\s*$'.format(roman))
    },
}
```

In [7]:
```
def acquire_epubs(epub_list, chap_pats, OHCO=OHCO):

    my_lib = []
    my_doc = []

    for epub_file in epub_list:

        # Get PG ID from filename
        # book_id = int(epub_file.split('-')[-1].split('.')[0].replace('pg',''
        book_id = int(epub_file.split('/')[1].split('-')[0])
        print("BOOK ID", book_id)

        # Import file as lines
        lines = open(epub_file, 'r', encoding='utf-8-sig').readlines()
        df = pd.DataFrame(lines, columns=['line_str'])
        df.index.name = 'line_num'
        df.line_str = df.line_str.str.strip()
        df['book_id'] = book_id

        # FIX CHARACTERS TO IMPROVE TOKENIZATION
        df.line_str = df.line_str.str.replace('—', ' — ')
        df.line_str = df.line_str.str.replace('-', ' - ')

        # Get book title and put into LIB table -- note problems, though
        book_title = re.sub(r"The Project Gutenberg eBook( of|,) ", "", df.loc[
        book_title = re.sub(r"Project Gutenberg's ", "", book_title, flags=re.I

        # Remove cruft
        a = chap_pats[book_id]['start_line'] - 1
        b = chap_pats[book_id]['end_line'] + 1
        df = df.iloc[a:b]

        # Chunk by chapter
        chap_lines = df.line_str.str.match(chap_pats[book_id]['chapter'])
        chap_nums = [i+1 for i in range(df.loc[chap_lines].shape[0])]
        df.loc[chap_lines, 'chap_num'] = chap_nums
        df.chap_num = df.chap_num.ffill()
        # Clean up
        df = df[~df.chap_num.isna()] # Remove chapter heading lines
        df = df.loc[~chap_lines] # Remove everything before Chapter 1
        df['chap_num'] = df['chap_num'].astype('int')
        # Group -- Note that we exclude the book level in the OHCO at this poi
        df = df.groupby(OHCO[1:2]).line_str.apply(lambda x: '\n'.join(x)).to_fr

        # Split into paragrpahs
        df = df['line_str'].str.split(r'\n\n+', expand=True).stack().to_frame()
        df.index.names = OHCO[1:3] # MAY NOT BE NECESSARY UNTIL THE END
        df['para_str'] = df['para_str'].str.replace(r'\n', ' ').str.strip()
        df = df[~df['para_str'].str.match(r'^\s*$')] # Remove empty paragraphs

        # Set index
        df['book_id'] = book_id
        df = df.reset_index().set_index(OHCO[:3])

        # Register
        my_lib.append((book_id, book_title, epub_file))
        my_doc.append(df)

    docs = pd.concat(my_doc)
    library = pd.DataFrame(my_lib, columns=['book_id', 'book_title', 'book_file
    print("Done.")
    return library, docs
```

In [8]:
```
epubs = [epub for epub in sorted(glob('data/*.txt'))]
epubs
```

Out[8]:
```
['data/145-0.txt', 'data/507-0.txt', 'data/6688-0.txt']
```

In [9]:
```
LIB, DOC = acquire_epubs(epubs, chap_pats)
```

```
/var/folders/pn/dgy7ckd90nl7mlj6g6rc_1kw0000gn/T/ipykernel_2458/4092773172.py:
49: FutureWarning: The default value of regex will change from True to False i
n a future version.
  df['para_str'] = df['para_str'].str.replace(r'\n', ' ').str.strip()
BOOK ID 145
BOOK ID 507
BOOK ID 6688
Done.
```

In [10]:
```
LIB
```

Out[10]:

	book_title	book_file
book_id		
145	Middlemarch, by George Eliot	data/145-0.txt
507	Adam Bede, by George Eliot	data/507-0.txt
6688	The Mill on the Floss, by George Eliot	data/6688-0.txt

In [11]:
```
DOC.sample(10)
```

Out[11]:

			para_str
book_id	chap_num	para_num	
145	32	0	BOOK IV. THREE LOVE PROBLEMS.
507	14	2	“Eh, I’m loath to see the last on her,” she sa...
	45	34	“Hetty,” she said gently, “do you know who it ...
	8	2	“You are only a visitor in this neighbourhood,...
6688	52	41	“Oh, what shall I do?” cried Maggie, in an ago...
145	144	64	“No.”
6688	5	14	Maggie’s answer was to throw her arms round To...
145	108	62	“I mean what you said about the necessity of k...
	100	34	He had longed not only to be set free from his...
	143	67	There was no time to say any more before Mr. F...

Tokenize and Annotate

In [12]:
```python
def tokenize(doc_df, OHCO=OHCO, remove_pos_tuple=False, ws=False):

    # Paragraphs to Sentences
    df = doc_df.para_str\
        .apply(lambda x: pd.Series(nltk.sent_tokenize(x)))\
        .stack()\
        .to_frame()\
        .rename(columns={0:'sent_str'})

    # Sentences to Tokens
    # Local function to pick tokenizer
    def word_tokenize(x):
        if ws:
            s = pd.Series(nltk.pos_tag(nltk.WhitespaceTokenizer().tokenize(x)))
        else:
            s = pd.Series(nltk.pos_tag(nltk.word_tokenize(x))) # Discards stuf
        return s

    df = df.sent_str\
        .apply(word_tokenize)\
        .stack()\
        .to_frame()\
        .rename(columns={0:'pos_tuple'})

    # Grab info from tuple
    df['pos'] = df.pos_tuple.apply(lambda x: x[1])
    df['token_str'] = df.pos_tuple.apply(lambda x: x[0])
    if remove_pos_tuple:
        df = df.drop('pos_tuple', 1)

    # Add index
    df.index.names = OHCO

    return df
```

In [13]:
```
%%time
TOKEN = tokenize(DOC, ws=False)
```

```
CPU times: user 39.7 s, sys: 696 ms, total: 40.4 s
Wall time: 40.6 s
```

In [14]:
```
TOKEN.head()
```

Out[14]:

					pos_tuple	pos	token_str
book_id	**chap_num**	**para_num**	**sent_num**	**token_num**			
145	**12**	**0**	**0**	**0**	(BOOK, NNP)	NNP	BOOK
				1	(II, NNP)	NNP	II
				2	(., .)	.	.
			1	**0**	(OLD, NNP)	NNP	OLD
				1	(AND, CC)	CC	AND

In [15]:
```
TOKEN[TOKEN.pos.str.match('^CC')]
```

Out[15]:

					pos_tuple	pos	token_str
book_id	**chap_num**	**para_num**	**sent_num**	**token_num**			
145	**12**	**0**	**1**	**1**	(AND, CC)	CC	AND
	52	**0**	**1**	**2**	(AND, CC)	CC	AND
	86	**2**	**0**	**11**	(and, CC)	CC	and
				69	(and, CC)	CC	and
			1	**10**	(and, CC)	CC	and
...	**...**	**...**	**...**	**...**	...	...	...
6688	**58**	**65**	**0**	**26**	(and, CC)	CC	and
				43	(and, CC)	CC	and
		66	**0**	**14**	(but, CC)	CC	but
		68	**0**	**7**	(and, CC)	CC	and
				11	(and, CC)	CC	and

29728 rows × 3 columns

Reduce

Extract a vocabulary from the TOKEN table

In [16]:
```
TOKEN['term_str'] = TOKEN['token_str'].str.lower().str.replace('[\W_]', '')
```

```
/var/folders/pn/dgy7ckd90nl7mlj6g6rc_1kw0000gn/T/ipykernel_2458/1858674674.py:
1: FutureWarning: The default value of regex will change from True to False in
a future version.
  TOKEN['term_str'] = TOKEN['token_str'].str.lower().str.replace('[\W_]', '')
```

In [17]:
```
VOCAB = TOKEN.term_str.value_counts()\
    .to_frame()\
    .rename(columns={'index':'term_str', 'term_str':'n'})\
    .sort_index()\
    .reset_index()\
    .rename(columns={'index':'term_str'})
VOCAB.index.name = 'term_id'
```

In [18]:
```
VOCAB['num'] = VOCAB.term_str.str.match("\d+").astype('int')
```

In [19]:
```
VOCAB.head()
```

Out[19]:

term_id	term_str	n	num
0		145933	0
1	1	1	1
2	1790	1	1
3	1799	2	1
4	1801	1	1

Annotate (VOCAB)

Add Stopwords

In [20]:
```
sw = pd.DataFrame(nltk.corpus.stopwords.words('english'), columns=['term_str'])
sw = sw.reset_index().set_index('term_str')
sw.columns = ['dummy']
sw.dummy = 1
```

In [21]:
```
sw.sample(10)
```

Out[21]:

term_str	dummy
so	1
on	1
has	1
under	1
while	1
here	1
at	1
weren't	1
up	1
herself	1

In [22]:
```
VOCAB['stop'] = VOCAB.term_str.map(sw.dummy)
VOCAB['stop'] = VOCAB['stop'].fillna(0).astype('int')
```

In [23]:
```
VOCAB[VOCAB.stop == 1].sample(10)
```

Out[23]:

term_id	term_str	n	num	stop
2483	but	5707	0	1
21060	who	2060	0	1
59	above	145	0	1
12863	or	1550	0	1
5548	do	1693	0	1
4512	d	664	0	1
20996	where	725	0	1
21010	which	3530	0	1
21408	y	64	0	1
10171	isn	133	0	1

Add Stems

In [24]:
```
from nltk.stem.porter import PorterStemmer
stemmer1 = PorterStemmer()
VOCAB['stem_porter'] = VOCAB.term_str.apply(stemmer1.stem)
```

In [25]:
```
VOCAB.sample(10)
```

Out[25]:

term_id	term_str	n	num	stop	stem_porter
10015	interruptions	2	0	0	interrupt
778	ants	3	0	0	ant
16384	scold	8	0	0	scold
12206	mysticism	2	0	0	mystic
5368	dismalness	2	0	0	dismal
6077	elephants	2	0	0	eleph
10639	laughable	1	0	0	laughabl
6088	elinor	9	0	0	elinor
4502	cuttle	2	0	0	cuttl
8416	grouse	1	0	0	grous

pos_max feature

Finally, add a feature named "pos_max" to the VOCAB table that contains the most frequently associated part-of-speech tag, as found in the TOKEN table, with each term

In [26]:
```python
TOKEN.sample(10)
```

Out[26]:

					pos_tuple	pos	token_str	term_str
book_id	**chap_num**	**para_num**	**sent_num**	**token_num**				
6688	**5**	**46**	**0**	**18**	(buy, VB)	VB	buy	buy
145	**159**	**6**	**3**	**15**	(medical, JJ)	JJ	medical	medical
	144	**34**	**5**	**9**	(Vincy, NNP)	NNP	Vincy	vincy
507	**6**	**8**	**1**	**15**	(a, DT)	DT	a	a
6688	**6**	**55**	**0**	**10**	(mind, NN)	NN	mind	mind
	7	**35**	**0**	**9**	(,, ,)	,	,	
145	**172**	**63**	**3**	**8**	(he, PRP)	PRP	he	he
507	**45**	**19**	**1**	**3**	(askance, NN)	NN	askance	askance
6688	**25**	**12**	**0**	**26**	(offices, NNS)	NNS	offices	offices
507	**27**	**18**	**2**	**3**	(in, IN)	IN	in	in

In [27]:
```python
part_in_token = TOKEN.groupby(['pos','term_str']).size().reset_index()
part_in_token
```

Out[27]:

	pos	term_str	0
0	''		190
1	''	harriet	1
2	''	him	1
3	''	lad	1
4	''	madam	1
...	...	...	...
32326	WRB	where	719
32327	WRB	wherever	3
32328	WRB	whichever	1
32329	WRB	why	569
32330	WRB	wi	3

32331 rows × 3 columns

In [28]:
```
part_in_token.groupby(['pos','term_str'])[0].max().reset_index()
```

Out[28]:

	pos	term_str	0
0	''		190
1	''	harriet	1
2	''	him	1
3	''	lad	1
4	''	madam	1
...	...	...	...
32326	WRB	where	719
32327	WRB	wherever	3
32328	WRB	whichever	1
32329	WRB	why	569
32330	WRB	wi	3

32331 rows × 3 columns

In [29]:
```
pos_dict = part_in_token.groupby(['pos','term_str'])[0]\
        .max()\
        .reset_index()\
        .set_index('term_str')\
        .to_dict()['pos']
```

In [30]:
```
VOCAB.sample(10)
```

Out[30]:

	term_str	n	num	stop	stem_porter
term_id					
10731	leaved	3	0	0	leav
21288	worldliness	7	0	0	worldli
801	apocryphal	2	0	0	apocryph
6517	events	37	0	0	event
12426	nicest	3	0	0	nicest
12596	numbers	3	0	0	number
12543	nothingness	1	0	0	nothing
11897	mizraim	2	0	0	mizraim
11815	mirrors	4	0	0	mirror
2374	brusquely	2	0	0	brusqu

In [31]:
```
VOCAB['term_str'].map(pos_dict)
```

```
Out[31]: term_id
         0          WRB
         1           CD
         2           CD
         3           CD
         4           CD
                   ...
         21503       NN
         21504      NNP
         21505      NNP
         21506       JJ
         21507      NNP
         Name: term_str, Length: 21508, dtype: object
```

```
In [32]: VOCAB['pos_max'] = VOCAB['term_str'].map(pos_dict)
```

```
In [33]: VOCAB
```

Out[33]:

term_id	term_str	n	num	stop	stem_porter	pos_max
0		145933	0	0		WRB
1	1	1	1	0	1	CD
2	1790	1	1	0	1790	CD
3	1799	2	1	0	1799	CD
4	1801	1	1	0	1801	CD
...	...	...	...	...	...	...
21503	œuvre	1	0	0	œuvr	NN
21504	μέγεθος	1	0	0	μέγεθος	NNP
21505	τι	1	0	0	τι	NNP
21506	ἀπέρωτος	1	0	0	ἀπέρωτος	JJ
21507	ἒρως	1	0	0	ἒρως	NNP

21508 rows × 6 columns

```
In [ ]:
```