M02 Homework

Michael Vaden, mtv2eva

In this exercise, you will convert a different text from raw text into a data frame of tokens and preserving its OHCO. Then you will extract some statistical features from the resulting corpus.

Follow these instructions:

Download the attached Gutenberg version of Jane Austen's Sense and Sensibility (pg161.txt Download pg161.txt).

Create a notebook to convert the raw text into a data frame of tokens, just as we did with Persuasion.

You may use the notebook from the lab as your guide.

Specifically, make sure your complete these tasks:

Remove Gutenberg's front and back matter using the lines that indicate the start and end of the project.

Chunk by chapter, using the pattern of locating the headers in the data frame, assigning them numbers, forward-filling those numbers, and then grouping by number (and cleaning up).

Split resulting data frame into paragraphs using the regex provided.

Split resulting data frame into sentences using the regex provided.

Split resulting data frame into tokens using the regex provided.

Be sure to include the OHCO of Chapters, Paragraphs, and Sentences in your data frame's index.

Once you have done this, combine both Persuasion and Sense and Sensibility into a single data frame with an appropriately modified OHCO list. In other words, make sure your index includes a new index level for the book. Use the attached CSV (austen-persuasion.csv Download austen-persuasion.csv) to get the Persuasion data and then import it into your notebook as a data frame.

From the combined data frame, extract a vocabulary, i.e. a data frame with term string as index, along with term frequency and term length as features.

After you have done all this, answer the following questions by extracting features from the corpus.

1. How many raw tokens are in the combined data frame?
2. How many distinct terms are there in the combined data frame (i.e. how big is the vocabulary)?
3. How many more terms does the vocabulary of Sense and Sensibility have than that of Persuasion?
4. What is the average number of tokens, rounded to an integer, per chapter in the corpus?
5. What is the average number of tokens, rounded to an integer, per paragraph in the corpus?

In [ ]:
```
import pandas as pd
```

In [ ]:
```
import configparser
config = configparser.ConfigParser()
config.read("../env.ini")
data_home = config['DEFAULT']['data_home']
output_dir = config['DEFAULT']['output_dir']
```

In [ ]:
```
text_file = f"{data_home}/pg161.txt"
```

Create a notebook to convert the raw text into a data frame of tokens:

In [ ]:
```
OHCO = ['chap_num', 'para_num', 'sent_num', 'token_num']

LINES = pd.DataFrame(open(text_file, 'r', encoding='utf-8-sig').readlines(), columns=['line_str'])
LINES.index.name = 'line_num'
```

```
LINES.line_str = LINES.line_str.str.replace(r'\n+', ' ', regex=True).str.strip()

title = LINES.loc[0].line_str.replace('The Project Gutenberg EBook of ', '')
print(title)
```

```
Sense and Sensibility, by Jane Austen
```

Clip Cruft

In [ ]:
```
clip_pats = [
    r"\*\*\*\s*START OF (?:THE|THIS) PROJECT",
    r"\*\*\*\s*END OF (?:THE|THIS) PROJECT"
]

pat_a = LINES.line_str.str.match(clip_pats[0])
pat_b = LINES.line_str.str.match(clip_pats[1])

line_a = LINES.loc[pat_a].index[0] + 1
line_b = LINES.loc[pat_b].index[0] - 1
print(line_a, line_b)

LINES = LINES.loc[line_a : line_b]
```

```
20 12666
```

chapter headers

In [ ]:
```
chap_pat = r"^\s*(?:chapter|letter)\s+\d+"

chap_lines = LINES.line_str.str.match(chap_pat, case=False) # Returns a truth vector
```

In [ ]:
```
LINES.loc[chap_lines, 'chap_num'] = [i+1 for i in range(LINES.loc[chap_lines].shape[0])]
#LINES.loc[chap_lines]
```

forward fill

In [ ]:
```
LINES.chap_num = LINES.chap_num.ffill()

LINES = LINES.dropna(subset=['chap_num']) # Remove everything before Chapter 1
# LINES = LINES.loc[~LINES.chap_num.isna()] # Remove everything before Chapter 1 (alternate method)
LINES = LINES.loc[~chap_lines] # Remove chapter heading lines; their work is done
LINES.chap_num = LINES.chap_num.astype('int') # Convert chap_num from float to int
```

group chapters

In [ ]:
```
OHCO[:1]
['chap_num']
# Make big string for each chapter
CHAPS = LINES.groupby(OHCO[:1])\
    .line_str.apply(lambda x: '\n'.join(x))\
    .to_frame('chap_str')

CHAPS['chap_str'] = CHAPS.chap_str.str.strip()
```

get paragraphs

In [ ]:
```
para_pat = r'\n\n+'
# CHAPS['chap_str'].str.split(para_pat, expand=True).head()
PARAS = CHAPS['chap_str'].str.split(para_pat, expand=True).stack()\
    .to_frame('para_str').sort_index()
PARAS.index.names = OHCO[:2]
#PARAS.head()
```

In [ ]:
```
PARAS['para_str'] = PARAS['para_str'].str.replace(r'\n', ' ', regex=True)
PARAS['para_str'] = PARAS['para_str'].str.strip()
PARAS = PARAS[~PARAS['para_str'].str.match(r'^\s*$')] # Remove empty paragraphs
#PARAS.head()
```

get sentences

In [ ]:
```
# sent_pat = r'[.?!;:"]+'
sent_pat = r'[.?!;:]+'
SENTS = PARAS['para_str'].str.split(sent_pat, expand=True).stack()\
    .to_frame('sent_str')
SENTS.index.names = OHCO[:3]
SENTS = SENTS[~SENTS['sent_str'].str.match(r'^\s*$')] # Remove empty paragraphs
```

```
SENTS.sent_str = SENTS.sent_str.str.strip() # CRUCIAL TO REMOVE BLANK TOKENS
SENTS.head()
```

Out[ ]:

chap_num	para_num	sent_num	sent_str
1	0	0	The family of Dashwood had long been settled i...
		1	Their estate was large, and their residence wa...
		2	The late owner of this estate was a single man...
		3	But her death, which happened ten years before...
		4	for to supply her loss, he invited and receive...

get tokens

In [ ]:
```
token_pat = r"[\s',-]+"
TOKENS = SENTS['sent_str'].str.split(token_pat, expand=True).stack()\
    .to_frame('token_str')
TOKENS.index.names = OHCO[:4]

TOKENS['term_str'] = TOKENS.token_str.replace(r'[\W_]+', '', regex=True).str.lower()
TOKENS
```

Out[ ]:

chap_num	para_num	sent_num	token_num	token_str	term_str
1	0	0	0	The	the
			1	family	family
			2	of	of
			3	Dashwood	dashwood
			4	had	had
...	...	...	...	...	...
50	22	0	8	and	and
			9	Sensibility	sensibility
			10	by	by
			11	Jane	jane
			12	Austen	austen

122882 rows × 2 columns

download and combine Persuasion

In [ ]:
```
persuasion_csv = f"{data_home}/austen-persuasion.csv"
persuasion_df = pd.read_csv(persuasion_csv)
```

In [ ]:
```
persuasion_gb = persuasion_df.groupby(['chap_num', 'para_num', 'sent_num', 'token_num']).sum()
persuasion_gb
```

Out[ ]:

chap_num	para_num	sent_num	token_num	token_str	term_str
1	0	0	0	Sir	sir
			1	Walter	walter
			2	Elliot	elliot
			3	of	of
			4	Kellynch	kellynch
...	...	...	...	...	...
24	13	0	6	of	of
			7	Persuasion	persuasion
			8	by	by
			9	Jane	jane
			10	Austen	austen

85014 rows × 2 columns

In [ ]:
```
persuasion_gb['book'] = 'Persuasion'
TOKENS['book'] = 'Sense and Sensibility'

combined = pd.concat([persuasion_gb, TOKENS])

combined_gb = combined.reset_index().groupby(['book', 'chap_num', 'para_num', 'sent_num', 'token_num'])[['toke
combined_gb
```

Out[ ]:

book	chap_num	para_num	sent_num	token_num	token_str	term_str
Persuasion	1	0	0	0	Sir	sir
				1	Walter	walter
				2	Elliot	elliot
				3	of	of
				4	Kellynch	kellynch
...	...	...	...	...	...	...
Sense and Sensibility	50	22	0	8	and	and
				9	Sensibility	sensibility
				10	by	by
				11	Jane	jane
				12	Austen	austen

207896 rows × 2 columns

After you have done all this, answer the following questions by extracting features from the corpus.

1. How many raw tokens are in the combined data frame?

In [ ]:
```
combined_gb.shape[0]
```

Out[ ]:
```
207896
```

There are **207896 raw tokens** in the combined dataframe

2. How many distinct terms are there in the combined data frame (i.e. how big is the vocabulary)?

In [ ]:
```
combined['term_str'].value_counts().shape[0]
```

Out[ ]:
```
8240
```

There are **8240 distinct terms** in the combined dataframe (ignoring case)

3. How many more terms does the vocabulary of Sense and Sensibility have than that of Persuasion?

In [ ]:
```
combined_gb.query("book == 'Sense and Sensibility'")['term_str'].value_counts().shape[0] - combined_gb.query("
```

Out[ ]:
```
520
```

There are **520 more terms** in the vocab of Sense and Sensibility than Persuasion

4. What is the average number of tokens, rounded to an integer, per chapter in the corpus?

In [ ]:
```
combined_gb.reset_index().groupby(['book', 'chap_num'])['token_num'].count().mean().round(0)
```

Out[ ]:
```
2809.0
```

The average number of tokens per chapter in the corpus is **2809**

5. What is the average number of tokens, rounded to an integer, per paragraph in the corpus?

In [ ]:
```
combined_gb.reset_index().groupby(['book', 'chap_num', 'para_num'])['token_num'].count().mean().round(0)
```

Out[ ]:
```
74.0
```

The average number of tokens per paragraph in the corpus is **74**