2019-01-24 Lab 2

Contents

Exercise: Determine the OHCO elements of Moby Dick

- [ ] In Spyder, go back to the Project you worked on last week and open up the source text of *Moby Dick*.
- [ ] Observe the structure of the text.
- [ ] Follow the instructor's questions to class to determine the text's discursive units (content objects).
- [ ] Record the results of the discussion.

Exercise: Create a Data frame of Moby Dick with OCHO

- [ ] Observe instructor's overview of demonstration code.
- [ ] Reproduce code on your own, using the embedded code below (or the files provided.)

Homework

- [ ] Download this Gutenberg version of *Austin's Persuasion*. Be sure to grab the UTF-8 version.
- [ ] Using a Python script (or Jupyter Notebook if you prefer), convert it into a data frame of tokens as we did with *Moby Dick*. Note that you do not need to do anything else in the demonstration file.
- [ ] Be sure to include the hierarchy of Chapters, Paragraphs, and Sentences in your data frame's index.
- [ ] Upload your code to the assignment page in Collab.

Files

- moby2.ipynb
- moby2.py
- 2701-0.txt

Source Code

```
#!/usr/bin/env python
# coding: utf-8

# %% Libraries

import pandas as pd
import seaborn as sns; sns.set()

# %% Configs

# Regular expressions for OHCO
BODY_START = 341
BODY_END = 21964
CHAP_PAT = r'^\s*(?:CHAPTER|ETYMOLOGY|Epilogue).*$'
PARA_PAT = r'\n\n+'
SENT_PAT = r'([.;?!"“”]+)'
TOKEN_PAT = r'(\W+)'

# The file
src_file = '2701-0.txt'

# %% Get the text as a list of lines

lines = open(src_file, 'r', encoding='utf-8').readlines()

# %% OHCO 1: Get Body

# Remove front and backmatter
# Clip at points discovered by visual inspection
lines = lines[BODY_START - 1 : BODY_END + 1]

# %% Convert list to data frame

df = pd.DataFrame({'line_str':lines})
df.index.name = 'line_id'

# %% OCHO 2: Chunk by Chapter

df.loc[df.line_str.str.match(CHAP_PAT), 'chap_id'] = df.apply(lambda x:
x.name, 1)
df.chap_id = df.chap_id.ffill().astype('int')

# %% Convert temporary IDs to sequential numbers

# We get the unique chapter names and convert them to a list
chap_ids = df.chap_id.unique().tolist()
df['chap_num'] = df.chap_id.apply(lambda x: chap_ids.index(x))

# %% Group and gather lines into chapter chunks

chaps = df.groupby('chap_num').apply(lambda x: ''.join(x.line_str))\
    .to_frame()\
    .rename(columns={0:'chap_str'})

# %% OHCO 3: Chunk by Paragraph

# We follow the **SPLIT-EXPAND-STACK** pattern.
paras = chaps.chap_str.str.split(PARA_PAT,
expand=True).stack().to_frame().rename(columns={0:'para_str'})
```

```
paras.index.names = ['chap_num', 'para_num']

# %% Clean up
paras.para_str = paras.para_str.str.strip()
paras.para_str = paras.para_str.str.replace(r'\n', ' ')
paras.para_str = paras.para_str.str.replace(r'\s+', ' ')
paras = paras[~paras.para_str.str.match(r'^\s*$')]

# %% OHCO 4: Chunk by Sentence

# We follow the **SPLIT-GROUP-JOIN** pattern again.
sents = paras.para_str.str.split(SENT_PAT,
expand=True).stack().to_frame().rename(columns={0:'sent_str'})
sents.index.names = ['chap_num', 'para_num', 'sent_num']

# %% Tokenize

# Again, use the **SPLIT-GROUP-JOIN** pattern.
tokens = sents.sent_str.str.split(TOKEN_PAT,
expand=True).stack().to_frame().rename(columns={0:'token_str'})
tokens.index.names = ['chap_num', 'para_num', 'sent_num', 'token_num']

# %% Now get paragraphs and chapters back

paras2 = tokens.groupby(['chap_num','para_num']).token_str.apply(lambda x:
''.join(x)).to_frame().rename(columns={'token_str':'para_str'})
print(paras2.head())

chaps2 = paras2.groupby(['chap_num']).para_str.apply(lambda x: '
'.join(x)).to_frame().rename(columns={'para_str':'chap_str'})
print(chaps2.head())

# %% Define a general function

def gather_chunks(df, div_names, doc_str = 'token_str', sep=''):
    chunks = df.groupby(div_names)[doc_str].apply(lambda x:
x.str.cat(sep=sep))
    chunks.columns = ['doc_content']
    return chunks

# %% Test on sentences

sents2 = gather_chunks(tokens, ['chap_num', 'para_num',
'sent_num']).to_frame()
print(sents2.head())

# %% Tag puncuation

tokens['punc'] = tokens.token_str.str.match(r'^\W*$').astype('int')

# %% View non-puncuation token counts

token_counts = tokens[tokens.punc ==
0].token_str.str.lower().value_counts()
print(token_counts.head(25))

# %% Visualize
```

```
token_counts.head(25).sort_values().plot(kind='barh', figsize=(10,20))
```