M04_HW_KEY

February 5, 2023

1 Metadata

```
Course:   DS 5001
Module:   04 HW KEY
Author:   R.C. Alvarado
```

2 Instructions

In this week's code exercise, you will use NLTK to help tokenize and annotate a small corpus of George Eliot's novels to create an `F3` level digital analytical edition from them.

Using this week's Lab notebook as a guide (`M04_01_Pipeline.ipynb`), which uses the `TextParser` class in the `/lib` directory of the notebook repository, import and combine the novels contained in the directory `/data/gutenberg/eliot-set`.

You should produce the following related dataframes:

- A library `LIB` with the following metadata (and data) about each book:
  - The `book_id`, matching the first level of the index in the `CORPUS`.
  - The raw book title will be sufficient, i.e. with title and author combined.
  - The path of the source file.
  - The regex used to parse chapter milestones.
  - The length of the book (number of tokens).
  - The number of chapters in the book.
- A an aggregate of all the novels' tokens `CORPUS` with an appropriate `OHCO` index, with following features:
  - The token string.
  - The term string.
  - THe part-of-speech tag inferred by NLTK.
- A vocabulary `VOCAB` of terms extracted from `CORPUS`, with the following annotation features derived from either NLTK or by using operations presented in the notebook:
  - Stopwords.
  - Porter stems.
  - Maximum POS; i.e. the most frequently associated POS tag for the term using `.idxmax()`. Note that ties are handled by the method.
  - POS ambiguity expressed a number of POS tags associated with a term's tokens.

Once you have these, use the dataframes to answer the questions below.

Hints: * You will need to edit the `ohco_pats` config to match the downloaded texts. * You may

also need to edit the code that reads files from disk and parses their names. * In defining the milestone regexes, be sure to include all chapter-level sections.

3 Questions

3.1 Q1

What regular expression did you use to chunk *Middlemarch* into chapters?

Answer: `^(?:PRELUDE|CHAPTER|FINALE)` or something similar.

3.2 Q2

What is the title of the book has the most tokens?

Answer: *Middlemarch*.

3.3 Q3

How many chapter level chunks are there in this novel?

Answer: 88

3.4 Q4

Among the three stemming algorithms – Porter, Snowball, and Lancaster – which is the most aggressive, in terms of the number of words associated with each stem?

Answer: Lancaster (1.8 stems/term)

3.5 Q5

Using the most aggressive stemmer from the previous question, what is the stem with the most associated terms?

Answer: 'cont'

4 Code

4.1 Setup

[3]:
```
data_home = "../labs-repo/data"
local_lib = "../labs-repo/lib"
source_files = f'{data_home}/gutenberg/eliot-set'
data_prefix = 'eliot'
```

[4]:
```
OHCO = ['book_id', 'chap_num', 'para_num', 'sent_num', 'token_num']
```

[5]:
```
import pandas as pd
import numpy as np
from glob import glob
```

```
import re
import nltk
```

[6]:
```
import sys
sys.path.append(local_lib)
```

[7]:
```
from textparser import TextParser
```

4.2 Inspect

Since Project Gutenberg texts vary widely in their markup, we define our chunking patterns by hand.

[47]:
```
roman = '[IVXLCM]+'
caps = "[A-Z';, -]+"
clip_pats = [
    r"\*\*\*\s*START OF",
    r"\*\*\*\s*END OF"
]
# All are 'chap'and 'm'
ohco_pat_list = [
    (6688,  rf"^Chapter\s+{roman}\.\s*$"),
    (507,   rf"^(?:Chapter\s+{roman}|Epilogue)\s*$"),
    (145,   rf"^(?:PRELUDE|BOOK|CHAPTER|FINALE)")
]
```

4.3 Register

We get each file and add to a library `LIB`.

[48]:
```
source_file_list = sorted(glob(f"{source_files}/*.*"))
```

[49]:
```
source_file_list
```

[49]:
```
['../labs-repo/data/gutenberg/eliot-set/ELIOT_GEORGE_ADAM_BEDE-pg507.txt',
 '../labs-repo/data/gutenberg/eliot-set/ELIOT_GEORGE_MIDDLEMARCH-pg145.txt',
 '../labs-repo/data/gutenberg/eliot-set/ELIOT_GEORGE_THE_MILL_ON_THE_FLOSS-
pg6688.txt']
```

[50]:
```
book_data = []
for source_file_path in source_file_list:
    book_id = int(source_file_path.split('-')[-1].split('.')[0].
 ↪replace('pg',''))
    book_title = source_file_path.split('/')[-1].split('-')[0].replace('_', ' ')
    book_data.append((book_id, source_file_path, book_title))
```

[51]:
```
```

```
LIB = pd.DataFrame(book_data,␣
 ↪columns=['book_id','source_file_path','raw_title'])\
    .set_index('book_id').sort_index()
```

[52]:
```
LIB
```

[52]:
```
                                          source_file_path  \
book_id
145      ../labs-repo/data/gutenberg/eliot-set/ELIOT_GE…
507      ../labs-repo/data/gutenberg/eliot-set/ELIOT_GE…
6688     ../labs-repo/data/gutenberg/eliot-set/ELIOT_GE…

                                  raw_title
book_id
145                ELIOT GEORGE MIDDLEMARCH
507                  ELIOT GEORGE ADAM BEDE
6688     ELIOT GEORGE THE MILL ON THE FLOSS
```

4.4 Tokenize

We tokenize each book and add each TOKENS table to a list to be concatenated into a single CORPUS.

[53]:
```
books = []
for pat in ohco_pat_list:

    book_id, chap_regex = pat
    print("Tokenizing", book_id, LIB.loc[book_id].raw_title)
    ohco_pats = [('chap', chap_regex, 'm')]
    src_file_path = LIB.loc[book_id].source_file_path

    text = TextParser(src_file_path, ohco_pats=ohco_pats, clip_pats=clip_pats,␣
 ↪use_nltk=True)
    text.verbose = False
    text.strip_hyphens = True
    text.strip_whitespace = True
    text.import_source().parse_tokens();
    text.TOKENS['book_id'] = book_id
    text.TOKENS = text.TOKENS.reset_index().set_index(['book_id'] + text.OHCO)

    books.append(text.TOKENS)
```

```
Tokenizing 6688 ELIOT GEORGE THE MILL ON THE FLOSS
Tokenizing 507 ELIOT GEORGE ADAM BEDE
Tokenizing 145 ELIOT GEORGE MIDDLEMARCH
```

4.5 Create Corpus

[54]:
```
CORPUS = pd.concat(books).sort_index()
```

[55]:
```
CORPUS.loc[145]
```

[55]:
```
                                          pos_tuple  pos  token_str  \
chap_id para_num sent_num token_num
1       0        0        0               (Who, WP)   WP        Who
                          1              (that, WDT)  WDT       that
                          2             (cares, VBZ)  VBZ      cares
                          3               (much, RB)   RB       much
                          4                 (to, TO)   TO         to
…                                                …  …        …
88      0        85       56                (in, IN)   IN         in
                          57         (unvisited, JJ)   JJ  unvisited
                          58            (tombs., NN)   NN     tombs.
                 86       0                 (THE, DT)   DT        THE
                          1                 (END, NN)   NN        END

                                        term_str
chap_id para_num sent_num token_num
1       0        0        0                 who
                          1                that
                          2               cares
                          3                much
                          4                  to
…                                           …
88      0        85       56                 in
                          57           unvisited
                          58               tombs
                 86       0                 the
                          1                 end

[317844 rows x 4 columns]
```

4.6 Extract some features for LIB

[56]:
```
LIB['book_len'] = CORPUS.groupby('book_id').term_str.count()
```

[57]:
```
LIB['n_chaps'] = CORPUS.reset_index()[['book_id','chap_id']]\
    .drop_duplicates()\
    .groupby('book_id').chap_id.count()
```

[58]:
```
LIB['chap_regex'] = LIB.index.map(pd.Series({x[0]:x[1] for x in ohco_pat_list}))
```

[59]:
```
LIB.sort_values('book_len')
```

[59]:
```
                                          source_file_path  \
book_id
6688     ../labs-repo/data/gutenberg/eliot-set/ELIOT_GE…
507      ../labs-repo/data/gutenberg/eliot-set/ELIOT_GE…
145      ../labs-repo/data/gutenberg/eliot-set/ELIOT_GE…

                                  raw_title  book_len  n_chaps  \
book_id
6688     ELIOT GEORGE THE MILL ON THE FLOSS    207459       58
507                  ELIOT GEORGE ADAM BEDE    215402       57
145                ELIOT GEORGE MIDDLEMARCH    317799       88

                                    chap_regex
book_id
6688                 ^Chapter\s+[IVXLCM]+\.\s*$
507      ^(?:Chapter\s+[IVXLCM]+|Epilogue)\s*$
145           ^(?:PRELUDE|BOOK|CHAPTER|FINALE)
```

4.7 Exract VOCAB

Extract a vocabulary from the CORPUS as a whole

[60]:
```
# CORPUS[CORPUS.term_str == '']
```

[61]:
```
CORPUS[CORPUS.term_str == ''].token_str.value_counts()
```

[61]:
```
&      10
…       3
),      2
);      2
(&)     1
):      1
;"      1
Name: token_str, dtype: int64
```

[62]:
```
CORPUS = CORPUS[CORPUS.term_str != '']
```

[63]:
```
VOCAB = CORPUS.term_str.value_counts().to_frame('n').sort_index()
VOCAB.index.name = 'term_str'
VOCAB['n_chars'] = VOCAB.index.str.len()
VOCAB['p'] = VOCAB.n / VOCAB.n.sum()
VOCAB['i'] = -np.log2(VOCAB.p)
```

4.8 Annotate VOCAB

[64]:
```
VOCAB['max_pos'] = CORPUS[['term_str','pos']].value_counts().
↪unstack(fill_value=0).idxmax(1)
```

[65]:
```
TPM = CORPUS[['term_str','pos']].value_counts().unstack()
```

[66]:
```
VOCAB['n_pos'] = TPM.count(1)
```

[67]:
```
VOCAB['cat_pos'] = CORPUS[['term_str','pos']].value_counts().to_frame('n').
↪reset_index()\
   .groupby('term_str').pos.apply(lambda x: set(x))
```

[68]:
```
VOCAB
```

[68]:
```
           n  n_chars         p          i max_pos  n_pos cat_pos
term_str
1          1        1  0.000001  19.498413      CD      1    {CD}
1790       1        4  0.000001  19.498413      CD      1    {CD}
1799       2        4  0.000003  18.498413      CD      1    {CD}
1801more   1        8  0.000001  19.498413      CD      1    {CD}
1807       1        4  0.000001  19.498413      CD      1    {CD}
…       ..       …        …         …     …    …      …
œdipus     2        6  0.000003  18.498413      NN      1    {NN}
       1        7  0.000001  19.498413     NNP      1   {NNP}
         1        2  0.000001  19.498413     NNP      1   {NNP}
       1        8  0.000001  19.498413      JJ      1    {JJ}
         1        4  0.000001  19.498413     NNP      1   {NNP}

[26351 rows x 7 columns]
```

4.9 Add Stopwords

We use NLTK's built in stopword list for English. Note that we can add and subtract from this list, or just create our own list and keep it in our data model.

[28]:
```
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
```

[29]:
```
VOCAB['stop'] = VOCAB.index.map(sw.dummy)
VOCAB['stop'] = VOCAB['stop'].fillna(0).astype('int')
```

[30]:
```
VOCAB
```

[30]:
```
           n  n_chars         p          i max_pos  n_pos cat_pos  stop
term_str
```

```
1          1         1  0.000001  19.498411      CD      1    {CD}     0
1790       1         4  0.000001  19.498411      CD      1    {CD}     0
1799       2         4  0.000003  18.498411      CD      1    {CD}     0
1801more   1         8  0.000001  19.498411      CD      1    {CD}     0
1807       1         4  0.000001  19.498411      CD      1    {CD}     0
…       ..       …        …         …     …     …     …    …
œdipus     2         6  0.000003  18.498411      NN      1    {NN}     0
       1         7  0.000001  19.498411     NNP      1   {NNP}     0
          1         2  0.000001  19.498411     NNP      1   {NNP}     0
       1         8  0.000001  19.498411      JJ      1    {JJ}     0
         1         4  0.000001  19.498411     NNP      1   {NNP}     0

[26352 rows x 8 columns]
```

4.10 Add Stems

[31]:
```
from nltk.stem.porter import PorterStemmer
stemmer1 = PorterStemmer()
VOCAB['stem_porter'] = VOCAB.apply(lambda x: stemmer1.stem(x.name), 1)

from nltk.stem.snowball import SnowballStemmer
stemmer2 = SnowballStemmer("english")
VOCAB['stem_snowball'] = VOCAB.apply(lambda x: stemmer2.stem(x.name), 1)

from nltk.stem.lancaster import LancasterStemmer
stemmer3 = LancasterStemmer()
VOCAB['stem_lancaster'] = VOCAB.apply(lambda x: stemmer3.stem(x.name), 1)
```

[32]:
```
VOCAB.sample(10)
```

[32]:
```
                  n  n_chars         p          i max_pos  n_pos  \
term_str
extricate         1        9  0.000001  19.498411      VB      1
protestantism     3       13  0.000004  17.913448     NNP      2
recovered        32        9  0.000043  14.498411     VBN      4
energumena        1       10  0.000001  19.498411      JJ      1
are            1559        3  0.002105   8.892006     VBP     10
conferences       1       11  0.000001  19.498411     NNS      1
fear            166        4  0.000224  12.123372      NN      6
bonaventure       1       11  0.000001  19.498411     NNP      1
shroud            3        6  0.000004  17.913448      NN      2
culture           8        7  0.000011  16.498411      NN      2

                                                  cat_pos  stop  \
term_str
extricate                                            {VB}     0
protestantism                                    {NNP, NN}     0
```

```
recovered                                {JJ, VBD, VBN, NN}     0
energumena                                             {JJ}     0
are            {VB, IN, NNP, VBP, JJ, NN, VBD, NNS, VBZ, RB}     1
conferences                                           {NNS}     0
fear                           {VB, NN, VBP, JJ, VBN, RB}     0
bonaventure                                           {NNP}     0
shroud                                             {NN, VB}     0
culture                                            {NN, JJ}     0

              stem_porter stem_snowball stem_lancaster
term_str
extricate          extric        extric            ext
protestantism  protestant    protestant        protest
recovered           recov         recov          recov
energumena     energumena    energumena      energumen
are                   are           are            ar
conferences        confer        confer          conf
fear                 fear          fear          fear
bonaventure    bonaventur    bonaventur         bonav
shroud             shroud        shroud         shroud
culture            cultur        cultur          cult
```

[33]:
```
VOCAB[VOCAB.stem_porter != VOCAB.stem_snowball]
```

[33]:
```
               n  n_chars         p          i max_pos  n_pos  \
term_str
abjectly       1        8  0.000001  19.498411      RB      1
abruptly      16        8  0.000022  15.498411      RB      5
abstractedly   3       12  0.000004  17.913448      NN      2
abundantly     4       10  0.000005  17.498411      VB      3
accordingly   11       11  0.000015  16.038979      NN      6
…           ..      …       …         …     …    …
yeswellyou     1       10  0.000001  19.498411      NN      1
yous           3        4  0.000004  17.913448      NN      2
zealous       10        7  0.000014  16.176483      JJ      1
æschylus       2        8  0.000003  18.498411     NNP      2
œdipus         2        6  0.000003  18.498411      NN      1

                                cat_pos  stop   stem_porter stem_snowball  \
term_str
abjectly                            {RB}     0      abjectli        abject
abruptly          {NN, VBD, JJ, RB, RP}     0      abruptli        abrupt
abstractedly                   {NN, RB}     0  abstractedli      abstract
abundantly                {RB, VB, NNS}     0    abundantli         abund
accordingly   {IN, NNP, NN, VBP, JJ, RB}     0   accordingli        accord
…                                   …   …           …           …
yeswellyou                         {NN}     0      yeswelly    yeswellyou
```

```
yous                          {NN, RB}     0          you          yous
zealous                           {JJ}     0       zealou       zealous
æschylus                     {NNP, NN}     0       æschylu       æschylus
œdipus                            {NN}     0        œdipu        œdipus

             stem_lancaster
term_str
abjectly             abject
abruptly             abrupt
abstractedly       abstract
abundantly            abund
accordingly          accord
…                         …
yeswellyou       yeswellyou
yous                     yo
zealous                zeal
æschylus           æschylus
œdipus                 œdip

[655 rows x 11 columns]
```

5 Answers

5.1 Q1

[36]:
```
ohco_pats[0][1]
```

[36]:
```
'^(?:PRELUDE|BOOK|CHAPTER|FINALE)'
```

5.2 Q2

[40]:
```
LIB.loc[LIB.book_len.idxmax()].raw_title
```

[40]:
```
'ELIOT GEORGE MIDDLEMARCH'
```

5.3 Q3

How many chapter level chunks are there in this novel?

[42]:
```
LIB.loc[145].n_chaps
```

[42]:
```
88
```

5.4 Q4

Among the three stemming algorithms – Porter, Snowball, and Lancaster – which is the most aggressive, defined as the average number of terms associated with each stem?

[55]:
```
for stem_type in ['porter', 'snowball', 'lancaster']:
    x = VOCAB[f"stem_{stem_type}"].value_counts().mean()
    print(stem_type, round(x,2))
```

```
porter 1.5
snowball 1.53
lancaster 1.8
```

lancaster

5.5 Q5

Using the most aggressive stemmer from the previous question, what is the stem with the most associated terms?

[66]:
```
most_aggressive_stem = VOCAB.stem_lancaster.value_counts().head(1).index.
↪values[0]
```

[68]:
```
most_aggressive_stem
```

[68]:
```
'cont'
```

[67]:
```
VOCAB.query(f"stem_lancaster == '{most_aggressive_stem}'")
```

[67]:

term_str	n	n_chars	p	i	max_pos	n_pos
conceal	12	7	0.000016	15.913448	VB	1
concealed	4	9	0.000005	17.498411	VBD	3
concealing	3	10	0.000004	17.913448	VBG	2
concealment	17	11	0.000023	15.410948	NN	2
concealments	1	12	0.000001	19.498411	NNS	1
conceals	1	8	0.000001	19.498411	VBZ	1
concede	1	7	0.000001	19.498411	VB	1
conceded	1	8	0.000001	19.498411	JJ	1
conceding	1	9	0.000001	19.498411	VBG	1
concentrate	3	11	0.000004	17.913448	VB	1
concentrated	14	12	0.000019	15.691056	VBN	4
concentrating	3	13	0.000004	17.913448	VBG	1
concentration	8	13	0.000011	16.498411	NN	2
concentric	2	10	0.000003	18.498411	JJ	1
conciliate	5	10	0.000007	17.176483	VB	1
conciliated	1	11	0.000001	19.498411	VBN	1
conciliating	2	12	0.000003	18.498411	VBG	1
conciliation	2	12	0.000003	18.498411	JJ	2
conciliatory	2	12	0.000003	18.498411	JJ	1
concur	5	6	0.000007	17.176483	VB	2
content	22	7	0.000030	15.038979	JJ	3
contented	41	9	0.000055	14.140859	VBN	4
contentedly	2	11	0.000003	18.498411	RB	2

contentment	19	11	0.000026	15.250483	NN	3
contents	14	8	0.000019	15.691056	NNS	2
contrarieties	1	13	0.000001	19.498411	NNS	1
contrariness	1	12	0.000001	19.498411	NN	1
contrary	56	8	0.000076	13.691056	NN	2
contrivance	6	11	0.000008	16.913448	NN	1
contrivances	7	12	0.000009	16.691056	NNS	2
contrive	4	8	0.000005	17.498411	VB	2
contrived	6	9	0.000008	16.913448	VBN	1
contrives	1	9	0.000001	19.498411	NN	1
contriving	7	10	0.000009	16.691056	VBG	2

term_str	cat_pos	stop	stem_porter	stem_snowball	\
conceal	{VB}	0	conceal	conceal	
concealed	{JJ, VBD, VBN}	0	conceal	conceal	
concealing	{NN, VBG}	0	conceal	conceal	
concealment	{NN, JJ}	0	conceal	conceal	
concealments	{NNS}	0	conceal	conceal	
conceals	{VBZ}	0	conceal	conceal	
concede	{VB}	0	conced	conced	
conceded	{JJ}	0	conced	conced	
conceding	{VBG}	0	conced	conced	
concentrate	{VB}	0	concentr	concentr	
concentrated	{JJ, VBD, VBN, NN}	0	concentr	concentr	
concentrating	{VBG}	0	concentr	concentr	
concentration	{NN, JJ}	0	concentr	concentr	
concentric	{JJ}	0	concentr	concentr	
conciliate	{VB}	0	concili	concili	
conciliated	{VBN}	0	concili	concili	
conciliating	{VBG}	0	concili	concili	
conciliation	{NN, JJ}	0	concili	concili	
conciliatory	{JJ}	0	conciliatori	conciliatori	
concur	{NN, VB}	0	concur	concur	
content	{JJ, VB, NN}	0	content	content	
contented	{JJ, VBD, VBN, NN}	0	content	content	
contentedly	{NN, RB}	0	contentedli	content	
contentment	{NN, VBN, JJ}	0	content	content	
contents	{NN, NNS}	0	content	content	
contrarieties	{NNS}	0	contrarieti	contrarieti	
contrariness	{NN}	0	contrari	contrari	
contrary	{NN, JJ}	0	contrari	contrari	
contrivance	{NN}	0	contriv	contriv	
contrivances	{NN, NNS}	0	contriv	contriv	
contrive	{VBP, VB}	0	contriv	contriv	
contrived	{VBN}	0	contriv	contriv	
contrives	{NN}	0	contriv	contriv	

```
contriving              {NN, VBG}     0       contriv       contriv

              stem_lancaster
term_str
conceal                 cont
concealed               cont
concealing              cont
concealment             cont
concealments            cont
conceals                cont
concede                 cont
conceded                cont
conceding               cont
concentrate             cont
concentrated            cont
concentrating           cont
concentration           cont
concentric              cont
conciliate              cont
conciliated             cont
conciliating            cont
conciliation            cont
conciliatory            cont
concur                  cont
content                 cont
contented               cont
contentedly             cont
contentment             cont
contents                cont
contrarieties           cont
contrariness            cont
contrary                cont
contrivance             cont
contrivances            cont
contrive                cont
contrived               cont
contrives               cont
contriving              cont
```

[ ]: