Metadata

```
Course:   DS 5001
Module:   02 Homework KEY
Topic:    Text Models
Author:   R.C. Alvaraddo
Date:     30 January 2023 (revised)
```

Overview

Students parse a second text following the pattern of the first, and then combine them to create a corpus.

With the corpus, students observe basic descriptive statistical features.

Instructions

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:

1. Download the attached Gutenberg version of Jane Austen's *Sense and Sensibility* (`pg161.txt`).
2. 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.
3. Specifically, make sure your complete these tasks:
    (a) Remove Gutenberg's front and back matter using the lines that indicate the start and end of the project.
    (b) 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).
    (c) Split resulting data frame into paragraphs using the regex provided.
    (d) Split resulting data frame into sentences using the regex provided.
    (e) Split resulting data frame into tokens using the regex provided.
    (f) Be sure to include the OHCO of Chapters, Paragraphs, and Sentences in your data frame's index.
4. 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`) to get the *Persuasion* data and then import it into your notebook as a data frame.
5. 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.

6. After you have done all this, answer the following questions by extracting features from the corpus.
   (a) How many raw tokens are in the combined data frame?
   (b) How many distinct terms are there in the combined data frame (i.e. how big is the vocabulary)?
   (c) How many more terms does the vocabulary of Sense and Sensibility have than that of Persuasion?
   (d) What is the average number of tokens, rounded to an integer, per chapter in the corpus?
   (e) What is the average number of tokens, rounded to an integer, per paragraph in the corpus?

Summary

- Convert `pg161.txt` into `austen-sense.csv` with OHCO of chapters, paragraphs, sentences, and tokens.
- Combine tokenized dataframes of `austen-sense.csv` and `austen-persuasion.csv` into `austen-combo.csv`.
- Extract a vocabulary with term frequencies.
- Answer the questions.

Set Up

```
import pandas as pd
import seaborn as sns
```

```
sns.set()
```

```
data_home = '../../../repo/lessons/data'
```

```
text_file = f"{data_home}/gutenberg/pg161.txt"
csv_file1 = f"{data_home}/output/austen-sense.csv" # To be created
csv_file2 = f"{data_home}/output/austen-persuasion.csv" # Already created
csv_combo = f"{data_home}/output/austen-combo.csv" # To be created
OHCO = ['chap_num', 'para_num', 'sent_num', 'token_num']
```

Import Text

Import *Sense and Sensibility* into a dataframe.

```
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.strip()
```

```
LINES.sample(10)
```

```
                                                   line_str
line_num
3690                      they say it is a sweet pretty place."
9395      since it can advance him so little towards wha...
5744      arrival, without once stirring from her seat, ...
12762     1.E.1.  The following sentence, with active li...
8713      barbarous have I been to you!--you, who have b...
5656             of her real situation with respect to him.
5911      circumstances, it was better for both that the...
2804      Mrs. Dashwood was sorry for what she had said;...
3870      "Is Mr. Willoughby much known in your part of ...
2501      will always be welcome; for I will not press y...
```

Extract title of work from first line

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

```
title
```

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

```
LINES.head()
```

```
                                                   line_str
line_num
0         The Project Gutenberg EBook of Sense and Sensi...
1
2         This eBook is for the use of anyone anywhere a...
3         almost no restrictions whatsoever.  You may co...
4         re-use it under the terms of the Project Guten...
```

Remove Gutenberg's front and back matter

```
a = LINES.line_str.str.match(r"\*\*\*\s*START OF (THE|THIS) PROJECT")
b = LINES.line_str.str.match(r"\*\*\*\s*END OF (THE|THIS) PROJECT")
```

```
an = LINES.loc[a].index[0]
bn = LINES.loc[b].index[0]
```

```
LINES = LINES.loc[an + 1 : bn - 2]
```

```
LINES
```

```
                                                   line_str
line_num
20
21
22
```

```
23
24
...                                                    ...
12661
12662
12663
12664
12665     End of the Project Gutenberg EBook of Sense an...

[12646 rows x 1 columns]
```

Chunk by chapter

Find all chapter headers

```
chap_lines = LINES.line_str.str.match(r"^\s*(chapter|letter)\s+(\d+)", case=False)
```

```
LINES.loc[chap_lines]
```

```
            line_str
line_num
42         CHAPTER 1
196        CHAPTER 2
399        CHAPTER 3
561        CHAPTER 4
756        CHAPTER 5
858        CHAPTER 6
986        CHAPTER 7
1112       CHAPTER 8
1244       CHAPTER 9
1448      CHAPTER 10
1665      CHAPTER 11
1816      CHAPTER 12
1997      CHAPTER 13
2281      CHAPTER 14
2440      CHAPTER 15
2718      CHAPTER 16
2945      CHAPTER 17
3153      CHAPTER 18
3331      CHAPTER 19
3632      CHAPTER 20
3913      CHAPTER 21
4214      CHAPTER 22
4532      CHAPTER 23
4767      CHAPTER 24
5001      CHAPTER 25
```

```
5197       CHAPTER 26
5454       CHAPTER 27
5732       CHAPTER 28
5883       CHAPTER 29
6324       CHAPTER 30
6628       CHAPTER 31
7004       CHAPTER 32
7278       CHAPTER 33
7601       CHAPTER 34
7888       CHAPTER 35
8152       CHAPTER 36
8456       CHAPTER 37
8900       CHAPTER 38
9205       CHAPTER 39
9408       CHAPTER 40
9706       CHAPTER 41
9977       CHAPTER 42
10155      CHAPTER 43
10490      CHAPTER 44
11060      CHAPTER 45
11278      CHAPTER 46
11571      CHAPTER 47
11838      CHAPTER 48
11986      CHAPTER 49
12410      CHAPTER 50
```

Assign numbers to chapters

```python
chap_nums = [i+1 for i in range(LINES.loc[chap_lines].shape[0])]
```

```python
LINES.loc[chap_lines, 'chap_num'] = chap_nums
```

Forward-fill chapter numbers to following text lines

```python
LINES.chap_num = LINES.chap_num.ffill()
```

Clean up

```python
# LINES = LINES.loc[~LINES.chap_num.isna()] # Remove chapter heading lines
LINES = LINES.dropna(subset=['chap_num'])
LINES = LINES.loc[~chap_lines] # Remove everything before Chapter 1
LINES.chap_num = LINES.chap_num.astype('int') # Convert chap_num from float to int
```

```python
LINES.sample(10)
```

```
                                                   line_str  chap_num
line_num
```

```
3994       demands which this politeness made on it, was ...         21
10943      happy, and afterwards returned to town to be g...         44
8132                                                                 35
7473                                                                 33
1436                           destroyed all its ingenuity."          9
602        perceive how you could express yourself more w...          4
7157       Mrs. Jennings, who knew nothing of all this, w...         32
11596      In the evening, when they were all three toget...         47
10595                                                                44
12002                                                                49
```

Group lines by chapter num

```
CHAPS = LINES.groupby(OHCO[:1]).line_str.apply(lambda x: '\n'.join(x)).to_frame('chap_str')
```

```
CHAPS.head()
```

```
                                                    chap_str
chap_num
1          \n\nThe family of Dashwood had long been settl...
2          \n\nMrs. John Dashwood now installed herself m...
3          \n\nMrs. Dashwood remained at Norland several ...
4          \n\n"What a pity it is, Elinor," said Marianne...
5          \n\nNo sooner was her answer dispatched, than ...
```

Split into paragraphs

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

```
PARAS.head()
```

```
                                                             para_str
chap_num para_num
1        0
         1         The family of Dashwood had long been settled i...
         2         By a former marriage, Mr. Henry Dashwood had o...
         3         The old gentleman died: his will was read, and...
         4         Mr. Dashwood's disappointment was, at first, s...
```

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

```
PARAS.head()
```

```
                                                             para_str
chap_num para_num
1        1         The family of Dashwood had long been settled i...
```

```
2         By a former marriage, Mr. Henry Dashwood had o...
3         The old gentleman died: his will was read, and...
4         Mr. Dashwood's disappointment was, at first, s...
5         His son was sent for as soon as his danger was...
```

Split into sentences

NOTE: ADDED " to regex in `split()`

```
SENTS = PARAS['para_str'].str.split(r'[.?!;:"]+', expand=True).stack()\
    .to_frame().rename(columns={0:'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()
```

```
SENTS.head()
```

```
                                                                 sent_str
chap_num para_num sent_num
1        1        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...
```

Split into tokens

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

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

```
TOKENS
```

```
                                     token_str     term_str
chap_num para_num sent_num token_num
1        1        0        0               The          the
                           1            family       family
                           2                of           of
                           3          Dashwood     dashwood
                           4               had          had
...                                        ...          ...
50       23       0        8               and          and
                           9       Sensibility  sensibility
                           10               by           by
                           11             Jane         jane
```

```
                         12              Austen        austen

[122257 rows x 2 columns]
```

Save work to CSV

```
TOKENS.to_csv(csv_file1)
```

Combine the two into a Corpus

```
csv_file2 = f"{data_home}/output/austen-persuasion.csv"
```

```
df1 = pd.read_csv(csv_file1)
df2 = pd.read_csv(csv_file2)
```

```
len(df1), len(df2)
```

```
(122257, 85014)
```

```
df1['book_id'] = 1 # They may use the string for the titles here
df2['book_id'] = 2
```

```
LIB = {
    1: 'Sense & Sensibility',
    2:'Persuasion'
}
```

```
CORPUS = pd.concat([df1, df2])
```

```
OHCO2 = ['book_id'] + OHCO
```

```
CORPUS = CORPUS.set_index(OHCO2)
```

```
# CORPUS.sample(10)
```

```
len(CORPUS), CORPUS.shape[0], CORPUS.token_str.count()
```

```
(207271, 207271, 205599)
```

Extract a vocabulary V

```
CORPUS['term_str'] = CORPUS.token_str.str.replace(r"\W+", "", regex=True).str.lower()
V = CORPUS.term_str.value_counts().to_frame('n')
V.index.name = 'term_str'
V['n_chars'] = V.index.str.len()
```

```
len(V)
```

```
8239
```

```
V.n_chars.mean()
```

```
7.5543148440344705
```

```
CORPUS
```

```
                                                    token_str    term_str
book_id chap_num para_num sent_num token_num
1       1        1        0        0                      The         the
                                   1                   family      family
                                   2                       of          of
                                   3                 Dashwood    dashwood
                                   4                      had         had
...                                                       ...         ...
2       24       13       0        6                       of          of
                                   7               Persuasion  persuasion
                                   8                       by          by
                                   9                     Jane        jane
                                   10                  Austen      austen

[207271 rows x 2 columns]
```

```
V
```

```
                  n  n_chars
term_str
the            7435        3
to             6923        2
and            6290        3
of             6146        2
her            3747        3
...             ...      ...
unconquerable     1       13
outgrown          1        8
prosperously      1       12
nominal           1        7
finis             1        5

[8239 rows x 2 columns]
```

Save Combo

Do this for safe keeping.

Students are not asked to do this, so don't worry if it's not there.

```
CORPUS.to_csv(csv_combo)
```

Answer Questions

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

```
CORPUS.shape[0]
```

```
207271
```

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

```
V.shape[0]
```

```
8239
```

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

Method 1

```
vc_sense = CORPUS.loc[1].term_str.value_counts().shape[0]
vc_persu = CORPUS.loc[2].term_str.value_counts().shape[0]
```

```
vc_sense - vc_persu
```

```
520
```

Method 2

Students don't have to do this, but it's a good idea to put features where they belong.

In this case, we can think of the the term counts per book as features of V.

```
V['in_1'] = CORPUS.loc[1].term_str.value_counts()
V['in_2'] = CORPUS.loc[2].term_str.value_counts()
```

```
V.in_1.count() - V.in_2.count()
```

```
520
```

A second way to do this, which does not rely on the existences on `NA`s, is to convert the values to booleans.

```
V.in_1.fillna(0).astype('bool').sum() - V.in_2.fillna(0).astype('bool').sum()
```

```
520
```

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

```
CORPUS.groupby(OHCO2[:2]).term_str.count().mean().round().astype('int')
```

```
2778
```

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

```
CORPUS.groupby(OHCO2[:3]).term_str.count().mean().round().astype('int')
```

```
73
```