M01_03_first-foray

January 19, 2023

1 Metadata

```
Course:   DS 5001 Exploratory Text Analytics
Module:   01 Getting Started
Topic:    Lab: First Foray
Author:   R.C. Alvarado
Date:     14 October 2022 (revised)
```

Purpose: We take a raw text file from Project Gutenberg and convert it into a dataframe of tokens. We then explore some properties of the data. The main idea is to get acquainted with the primary process of convert text into analytical form.

2 Set Up

[1]:
```
import pandas as pd
```

3 Import File

[2]:
```
lines = open('pg105.txt', 'r').readlines()
```

[3]:
```
lines[:5]
```

[3]:
```
['Persuasion by Jane Austen (1818)\n', '\n', '\n', '\n', 'Chapter 1\n']
```

[4]:
```
lines[-5:]
```

[4]:
```
['the tax of quick alarm for belonging to that profession which is, if\n',
 'possible, more distinguished in its domestic virtues than in its\n',
 'national importance.\n',
 '\n',
 'Finis']
```

4 Convert to Dataframe

[5]:
```
text = pd.DataFrame(lines)
```

[6]:
```
text.sample(10)
```

[6]:
```
                                                      0
2101                                       you did."\n
5486  morning; sure to have plenty of chat; and then…
478   desirable tenants as any set of people one sho…
5744                                                \n
5470  "Did you say that you had something to tell me…
3867  dear Miss Louisa.  Vague wishes of getting Sar…
3437  luckily Mary did not much attend to their havi…
3308                                                \n
4170                                                \n
3014  last, had brought intelligence of Captain Harv…
```

[7]:
```
text.columns = ['line_str']
```

[8]:
```
text.head()
```

[8]:
```
                            line_str
0  Persuasion by Jane Austen (1818)\n
1                                  \n
2                                  \n
3                                  \n
4                        Chapter 1\n
```

[9]:
```
text.index.name = 'line_num'
```

[10]:
```
text.head()
```

[10]:
```
                                    line_str
line_num
0         Persuasion by Jane Austen (1818)\n
1                                         \n
2                                         \n
3                                         \n
4                               Chapter 1\n
```

5 Extract Simple Features

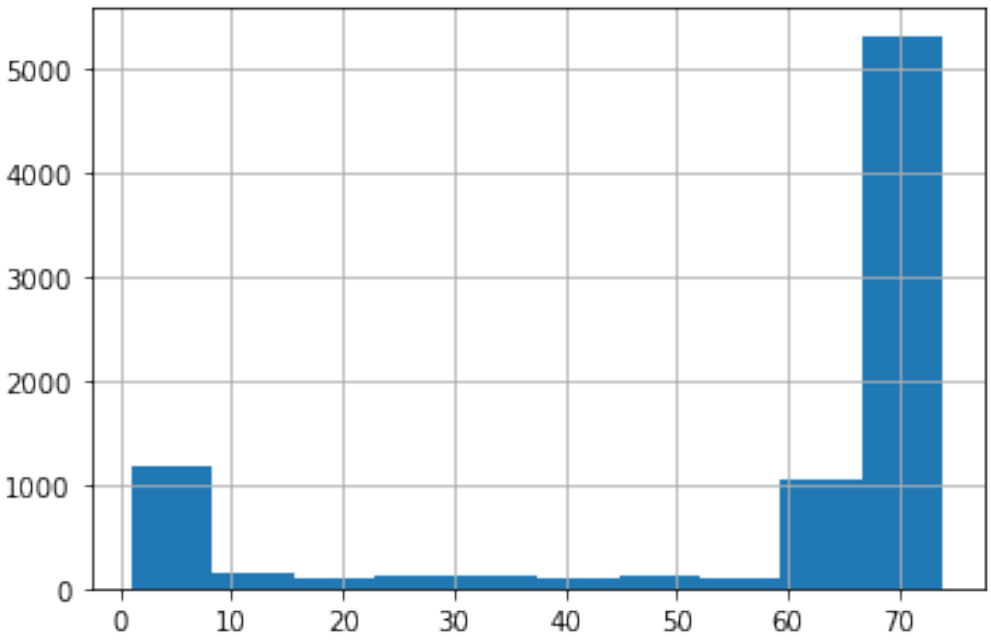
[11]:
```
text['len'] = text.line_str.str.len()
```

[12]:
```
text.len.describe()
```

[12]:
```
count    8317.000000
mean       56.131658
std        25.013216
min         1.000000
```

```
25%        62.000000
50%        69.000000
75%        71.000000
max        74.000000
Name: len, dtype: float64
```

[13]:
```
text.len.hist();
```



Why two humps? What might this bimodal distribution indicate?

Let's look at the first hump for characters.

[14]:
```
text[text['len'] < 5].sample(10)
```

[14]:

line_num	line_str	len
3517	\n	1
4753	\n	1
1048	\n	1
1572	\n	1
1840	\n	1
513	\n	1
2952	\n	1
7688	\n	1
1155	\n	1
7454	\n	1

6 Import Again

Now that we know what line breaks mean, we can use this information to import the file with a more accurate structure. Note also that we could have inferred this from visual inspection, too. But the principle that statistical features can provide evidence for structure remains – we will use this throughout the course.

6.1 Interpret line breaks \n

[15]:
```
chunk_pat = '\n\n'
```

[16]:
```
chunks = open('pg105.txt', 'r').read().split(chunk_pat)
```

[17]:
```
text = pd.DataFrame(chunks, columns=['chunk_str'])
text.index.name = 'chunk_id'
```

[18]:
```
text.head()
```

[18]:

chunk_id	chunk_str
0	Persuasion by Jane Austen (1818)
1	
2	Chapter 1
3	Sir Walter Elliot, of Kellynch Hall, in Somers…
4	"ELLIOT OF KELLYNCH HALL.

[19]:
```
text.shape
```

[19]:
```
(1056, 1)
```

6.2 Remove remaining breaks

[20]:
```
text.chunk_str = text.chunk_str.str.replace('\n+', ' ', regex=True).str.strip()
```

[21]:
```
text.head()
```

[21]:

chunk_id	chunk_str
0	Persuasion by Jane Austen (1818)
1	
2	Chapter 1
3	Sir Walter Elliot, of Kellynch Hall, in Somers…
4	"ELLIOT OF KELLYNCH HALL.

7 Convert Lines to Tokens

K: A dataframe of tokens.

Note the `expand` argument to the `.split()` method.

[47]:
```
K = text.chunk_str.str.split(expand=True).stack().to_frame('token_str')
K.index.names = ['chunk_num','token_num']
```

[48]:
```
K
```

[48]:
```
                      token_str
chunk_num token_num
0         0           Persuasion
          1                   by
          2                 Jane
          3               Austen
          4               (1818)
…                          …
1054      165                in
          166               its
          167          national
          168       importance.
1055      0                Finis

[83283 rows x 1 columns]
```

Broken down into steps

[24]:
```
# text.chunk_str.str.split()
```

[25]:
```
# text.chunk_str.str.split(expand=True)
```

[26]:
```
# text.chunk_str.str.split(expand=True).stack()
```

[27]:
```
# text.chunk_str.str.split(expand=True).stack().to_frame('token_str')
```

[50]:
```
K.iloc[100:120]
```

[50]:
```
                     token_str
chunk_num token_num
3         93               his
          94               own
          95           history
          96              with
          97                an
          98          interest
          99             which
          100            never
          101          failed.
          102             This
          103              was
```

```
104              the
105             page
106               at
107            which
108              the
109        favourite
110           volume
111           always
112         opened:
```

8 Do Some Cleaning

[51]:
```
K['term_str'] = K.token_str.str.replace(r'\W+', '', regex=True).str.lower()
```

[52]:
```
K.sample(10)
```

[52]:

chunk_num	token_num	token_str	term_str
947	55	very	very
1019	20	for	for
433	118	of	of
95	29	over-anxious	overanxious
763	28	many	many
918	43	ever	ever
625	170	to	to
648	13	bore	bore
689	31	come	come
644	380	the	the

9 Extract a Vocabulary

`V`: A table of terms. As opposed to tokens, which are term *instances*.

Terms are symbol **types**.

Tokens are symbol **instances**.

[53]:
```
V = K.term_str.value_counts().to_frame('n')
V.index.name = 'term_str'
```

[54]:
```
V.head(10)
```

[54]:

term_str	n
the	3326
to	2782
and	2781

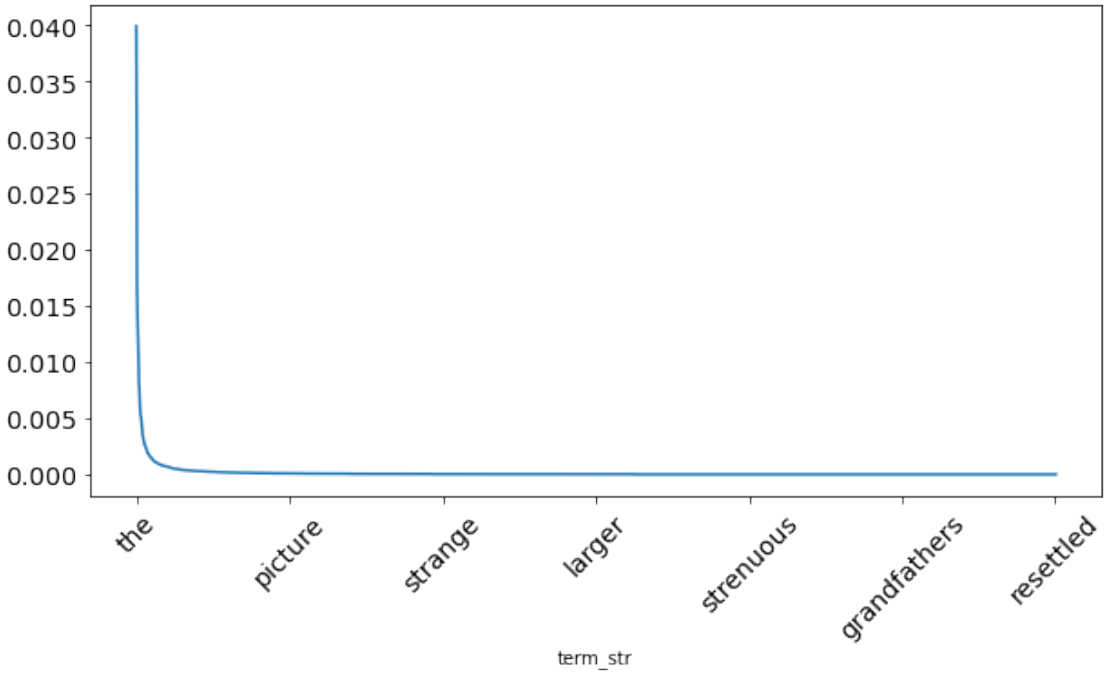
```
of      2565
a       1591
in      1382
was     1335
her     1202
had     1187
she     1142
```

Define relative frequency, an estimate of the probability of the word.

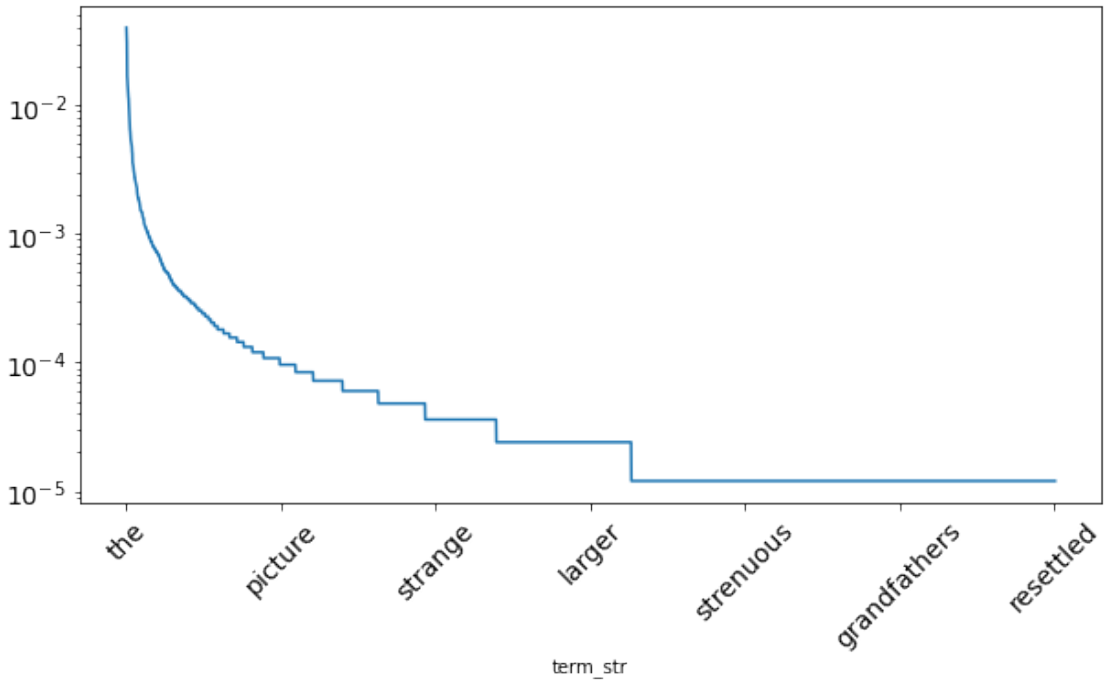
[55]:
```
V['p'] = V.n / V.n.sum()
```

10 Visualize Frequent Words

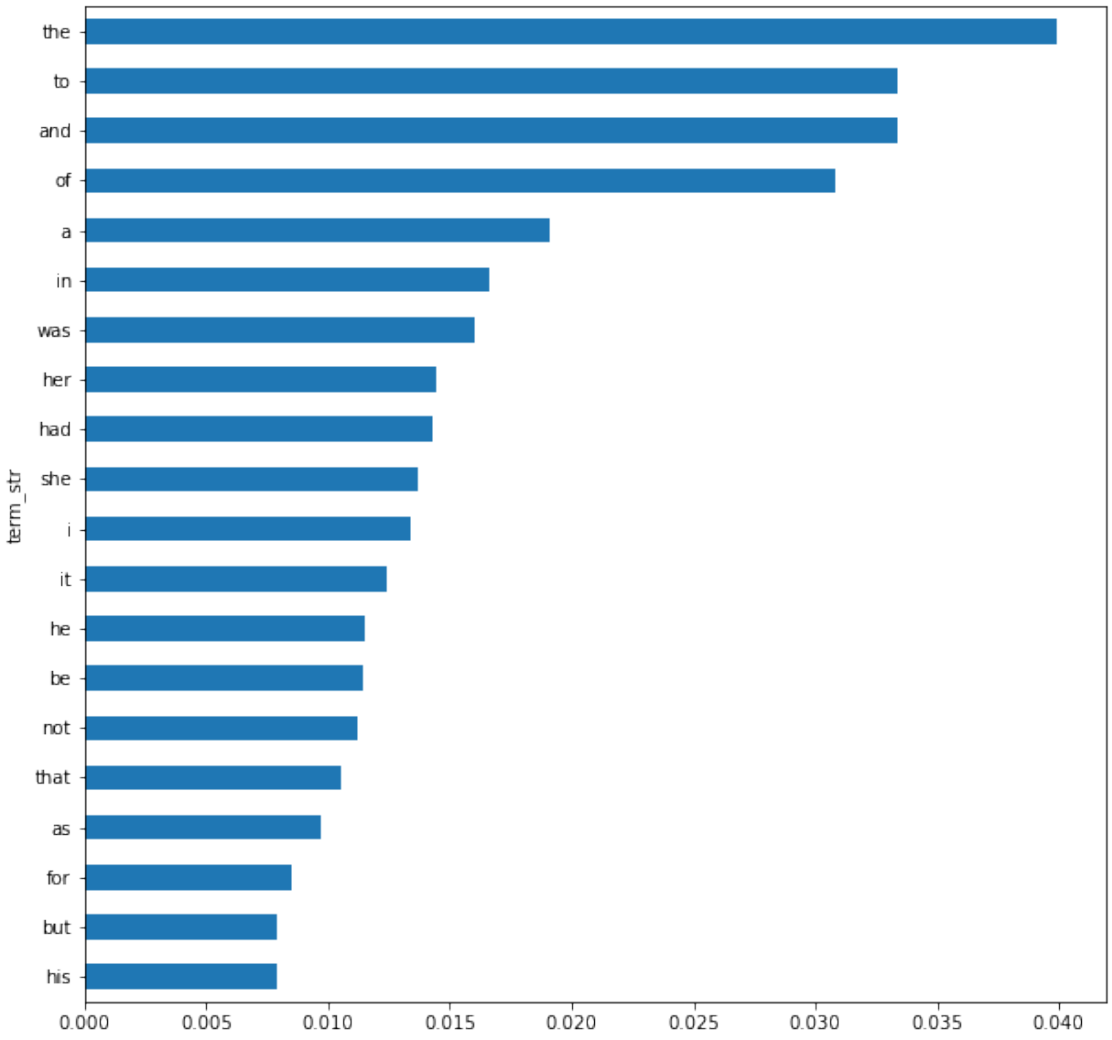
[56]:
```
V.p.plot(figsize=(10,5), fontsize=14, rot=45, legend=False);
```



[57]:
```
V.p.plot(figsize=(10,5), fontsize=14, rot=45, legend=False, logy=True);
```



[58]:
```
V.p.head(20).sort_values().plot.barh(figsize=(10,10));
```



11 The The

Why is “the” the most frequent word?

Consider that “the” is “The Most Powerful Word in the English Language.”

> … ‘the’ lies at the heart of English grammar, having a function rather than a meaning. Words are split into two categories: expressions with a semantic meaning and functional words like ‘the’, ‘to’, ‘for’, with a job to do. ‘The’ can function in multiple ways. This is typical, explains Gary Thoms, assistant professor in linguistics at New York University: “a super high-usage word will often develop a real flexibility”, with different subtle uses that make it hard to define. Helping us understand what is being referred to, ‘the’ makes sense of nouns as a subject or an object. So even someone with a rudimentary grasp of English can tell the difference between ‘I ate an apple’ and ‘I ate the apple’.

Note: function vs. meaning …

Function words are very specific to each language. So, someone who is a native Hindi or Russian speaker is going to have to think very differently when constructing a sentence in English. Murphy says that she has noticed, for instance, that sometimes her Chinese students hedge their bets and include 'the' where it is not required. Conversely, Smith describes Russian friends who are so unsure when to use 'the' that they sometimes leave a little pause: 'I went into… bank. I picked up… pen.' English speakers learning a language with no equivalent of 'the' also struggle and might overcompensate by using words like 'this' and 'that' instead.

12 Save Work

[59]:
```
K.to_csv("ff-TOKENS.csv")
V.to_csv("ff-VOCAB.csv")
```