Metadata

```
Course:   DS 5001
Module:   05 HW
Topic:    Create and Apply a TF-IDF Function
Author:   R.C. Alvarado
```

Instructions

Using the notebook from this module (`M05_01_BOW_TFIDF.ipynb`) and the `LIB` and `CORPUS` tables generated from the collection of texts (Austen and Melville) in Module 4, create a notebook to perform the following tasks:

Write a function to generate a bag-of-words `BOW` represenation of the `CORPUS` table (or some subset of it) that takes the following arguments:

- A tokens dataframe which can be a filtered version of the dataframe you import. This will be the `CORPUS` table or some subset of it.
- A choice of bag, i.e. `OHCO` level, such as book, chapter, or paragraph.

Write a function that returns the `TFIDF` values for a given `BOW`, with the following arguments:

- The `BOW` table.
- The type of `TF` measure to use. To compute `IDF`, use the formula $log_2(\frac{N}{DF})$ where N is the number of documents (aka 'bags') in your `BOW`.

Use these functions to get the appropriate `TFIDF` to answer the questions below.

Hints

- Update the course GitHub repository to make sure you are working with the latest files.
- Remember that the `CORPUS` table is a `TOKENS` table; it's just the combination of several such tables into one.
- You will need to generate your own `VOCAB` table from `CORPUS` and compute `max_pos`.
- When generating your own `VOCAB` table from `CORPUS`, be sure to name your index `term_str`.
- Remember that the mean `TFIDF` is an aggregate statistic computed from the `TFIDF` results, and which shares the same domain as the `VOCAB` table.
- `OHCO = ['book_id', 'chap_id', 'para_num', 'sent_num', 'token_num']`

Questions

Q1

Paste your functions here.

Answer: `PASTED FUNCTIONS`

Q2

What are the top 20 words in the corpus by TFIDF mean using the `max` count method and `book` as the bag?

Answer:

```
elinor 0.033840 NNP
pierre 0.030911 NNP
vernon 0.025980 NNP
marianne 0.021347 NNP
emma 0.021164 NNP
darcy 0.019302 NNP
reginald 0.018486 NNP
babbalanja 0.018252 NNP
catherine 0.018238 NNP
frederica 0.017986 NNP
crawford 0.017749 NNP
fanny 0.017167 NNP
elliot 0.017053 NNP
weston 0.016591 NNP
media 0.015986 NNP
israel 0.015428 NNP
knightley 0.015184 NNP
tilney 0.013815 NNP
elton 0.013648 NNP
bingley 0.013264 NNP
```

Q3

What are the top 20 words in the corpus by TFIDF mean, if you using the `sum` count method and `paragraph` (or `chapter`) as the bag? Note, beccause of the greater number of bags, this will take longer to compute.

NOTE: Students can use Chapter as a bag if they run into performance issues.

Answer:

Paragraphs:

```
i 0.025729 PRP
you 0.024533 PRP
the 0.021601 DT
of 0.017819 IN
a 0.016895 DT
to 0.016776 TO
and 0.016728 CC
```

```
is 0.016105 VBZ
he 0.016027 PRP
said 0.015729 VBD
her 0.015453 PRP$
it 0.015185 PRP
was 0.015107 VBD
his 0.014842 PRP$
in 0.014713 IN
my 0.014284 PRP$
not 0.014022 RB
that 0.013608 IN
she 0.013250 PRP
but 0.012186 CC
```

Chapters:

```
her 0.004327 PRP$
she 0.004150 PRP
cosmopolitan 0.003485 NN
pierre 0.003317 NNP
communion 0.003004 NN
i 0.002771 PRP
sailors 0.002668 NNS
you 0.002620 PRP
hypothetical 0.002437 NNP
mr 0.002084 NNP
and 0.002054 CC
confidential 0.002042 JJ
the 0.001972 DT
dream 0.001942 NN
boon 0.001857 NN
mrs 0.001747 NNP
elephants 0.001731 NN
whale 0.001715 NN
thou 0.001696 NN
acquaintance 0.001690 NN
```

Q4

Characterize the general difference between the words in Question 3 and those in Qestion 2 in terms of part-of-speech.

Answer: TFIDF by book just captures proper nouns.

Q5

Compute mean `TFIDF` for vocabularies conditioned on individual author, using *chapter* as the bag and `max` as the `TF` count method. Among the two authors, whose work has the most significant ajective?

Answer: Melville.

Solution

Setup

```
data_home = "../../../repo/lessons/data"
data_prefix = 'austen-melville'
```

```
OHCO = ['book_id', 'chap_id', 'para_num', 'sent_num', 'token_num']
```

```
SENTS = OHCO[:4]
PARAS = OHCO[:3]
CHAPS = OHCO[:2]
BOOKS = OHCO[:1]
```

Import

```
import pandas as pd
import numpy as np
import seaborn as sns
import plotly_express as px
```

```
sns.set()
```

Prepare the data

```
LIB = pd.read_csv(f"{data_home}/output/{data_prefix}-LIB.csv").set_index('book_id')
CORPUS = pd.read_csv(f"{data_home}/output/{data_prefix}-CORPUS.csv").set_index(OHCO)
```

```
VOCAB = CORPUS.term_str.value_counts().to_frame('n')
VOCAB.index.name = 'term_str'
VOCAB['p'] = VOCAB.n / VOCAB.n.sum()
VOCAB['i'] = np.log2(1/VOCAB.p)
VOCAB['max_pos'] = CORPUS.reset_index().value_counts(['term_str','pos']).sort_index().unstac
```

```
VOCAB
```

```
                 n              p          i max_pos
term_str
the         109921   5.338676e-02   4.227374      DT
of           65525   3.182438e-02   4.973724      IN
and          62954   3.057569e-02   5.031471      CC
to           56271   2.732987e-02   5.193378      TO
```

```
a            44174  2.145456e-02   5.542572      DT
...            ...           ...        ...     ...
lawfulness       1  4.856830e-07  20.973482      NN
equipages        1  4.856830e-07  20.973482     NNP
location         1  4.856830e-07  20.973482     NNP
rhodian          1  4.856830e-07  20.973482      JJ
scalpest         1  4.856830e-07  20.973482     JJS

[40281 rows x 4 columns]
```

Define Functions

```
def create_bow(CORPUS, bag, item_type='term_str'):
    BOW = CORPUS.groupby(bag+[item_type])[item_type].count().to_frame('n')
    return BOW

def get_tfidf(BOW, tf_method='max', df_method='standard', item_type='term_str'):

    DTCM = BOW.n.unstack(fill_value=0) # Create Doc-Term Count Matrix

    if tf_method == 'sum':
        TF = (DTCM.T / DTCM.T.sum()).T
    elif tf_method == 'max':
        TF = (DTCM.T / DTCM.T.max()).T
    elif tf_method == 'log':
        TF = (np.log2(1 + DTCM.T)).T
    elif tf_method == 'raw':
        TF = DTCM
    elif tf_method == 'bool':
        TF = DTCM.astype('bool').astype('int')
    else:
        raise ValueError(f"TF method {tf_method} not found.")

    DF = DTCM.astype('bool').sum()
    N_docs = len(DTCM)

    if df_method == 'standard':
        IDF = np.log2(N_docs/DF) # This what the students were asked to use
    elif df_method == 'textbook':
        IDF = np.log2(N_docs/(DF + 1))
    elif df_method == 'sklearn':
        IDF = np.log2(N_docs/DF) + 1
    elif df_method == 'sklearn_smooth':
        IDF = np.log2((N_docs + 1)/(DF + 1)) + 1
    else:
        raise ValueError(f"DF method {df_method} not found.")
    TFIDF = TF * IDF

    return TFIDF
```

Get Top Words by Bag

Q2

```
BOW_books = create_bow(CORPUS, bag=BOOKS)
```

```
BOW_books
```

```
                  n
book_id term_str
105     1         2
        15        1
        16        1
        1760      1
        1784      1
...              ..
34970   zero      1
        zest      1
        zone      3
        zones     2
        æniad     1

[177357 rows x 1 columns]
```

```
TFIDF_books = get_tfidf(BOW_books, tf_method='max', df_method='standard')
```

```
TFIDF_books
```

```
term_str         0         1        10       100      1000     10000  \
book_id
105       0.000000  0.000865  0.000000  0.000000  0.000000  0.000000
121       0.000000  0.000000  0.000000  0.000000  0.000000  0.000000
141       0.000000  0.000000  0.001086  0.000000  0.000000  0.000000
158       0.000000  0.000000  0.000000  0.000000  0.000000  0.001239
161       0.000000  0.000350  0.000000  0.000000  0.000000  0.000000
946       0.000000  0.000000  0.002512  0.000000  0.000000  0.000000
1212      0.006485  0.000000  0.000000  0.000000  0.000000  0.000000
1342      0.000000  0.000000  0.000000  0.000000  0.000000  0.000000
1900      0.000000  0.000000  0.000277  0.000000  0.000000  0.000000
2701      0.000000  0.000101  0.000000  0.000000  0.000000  0.000000
4045      0.000000  0.000000  0.000000  0.000000  0.000443  0.000443
8118      0.000000  0.000207  0.000323  0.000000  0.000000  0.000000
10712     0.000000  0.001542  0.000000  0.000316  0.000316  0.000000
```

```
13720     0.000000  0.000000  0.000000  0.000498  0.000000  0.000000
13721     0.000000  0.000000  0.000000  0.000000  0.000000  0.000000
15422     0.000000  0.000000  0.000000  0.000000  0.000000  0.000000
15859     0.000000  0.000571  0.000000  0.000000  0.000000  0.000000
21816     0.000000  0.001459  0.000000  0.000000  0.000000  0.000000
34970     0.000000  0.000000  0.000000  0.000000  0.000000  0.000000

term_str  10000000     10440     10800      10th  ...  zoroaster      zozo  \
book_id                                           ...
105       0.000000  0.000000  0.000000  0.000000  ...   0.000000  0.000000
121       0.000000  0.000000  0.000000  0.000000  ...   0.000000  0.000000
141       0.000000  0.000000  0.000000  0.000000  ...   0.000000  0.000000
158       0.000000  0.000000  0.000000  0.000000  ...   0.000000  0.000000
161       0.000000  0.000000  0.000000  0.000000  ...   0.000000  0.000000
946       0.000000  0.000000  0.000000  0.000000  ...   0.000000  0.000000
1212      0.000000  0.000000  0.000000  0.006485  ...   0.000000  0.000000
1342      0.000000  0.000000  0.000000  0.000000  ...   0.000000  0.000000
1900      0.000000  0.000000  0.000000  0.000000  ...   0.000000  0.000000
2701      0.000000  0.000296  0.000593  0.000000  ...   0.000227  0.000000
4045      0.000000  0.000000  0.000000  0.000000  ...   0.000000  0.000000
8118      0.000000  0.000000  0.000000  0.000000  ...   0.000000  0.000000
10712     0.000000  0.000000  0.000000  0.000000  ...   0.000000  0.000000
13720     0.000000  0.000000  0.000000  0.000000  ...   0.000000  0.000000
13721     0.000000  0.000000  0.000000  0.000000  ...   0.000653  0.000854
15422     0.000000  0.000000  0.000000  0.000000  ...   0.000000  0.000000
15859     0.000842  0.000000  0.000000  0.000000  ...   0.000000  0.000000
21816     0.000000  0.000000  0.000000  0.000000  ...   0.000000  0.000000
34970     0.000000  0.000000  0.000000  0.000000  ...   0.000000  0.000000

term_str      zuma       zur         à     æneas     æniad      æson  \
book_id
105       0.000000  0.000000  0.000000  0.000000  0.000000  0.000000
121       0.000000  0.000000  0.000000  0.000000  0.000000  0.000000
141       0.000000  0.000000  0.000000  0.000000  0.000000  0.000000
158       0.000000  0.000000  0.000000  0.000000  0.000000  0.000000
161       0.000000  0.000000  0.000000  0.000000  0.000000  0.000000
946       0.000000  0.000000  0.000000  0.000000  0.000000  0.000000
1212      0.000000  0.000000  0.000000  0.000000  0.000000  0.000000
1342      0.000000  0.000000  0.000000  0.000000  0.000000  0.000000
1900      0.000000  0.000000  0.000000  0.000000  0.000000  0.000000
2701      0.000000  0.000000  0.000000  0.000000  0.000000  0.000000
4045      0.000000  0.001158  0.000000  0.000000  0.000000  0.000000
8118      0.000000  0.000000  0.000000  0.000000  0.000000  0.000000
10712     0.000000  0.000000  0.000000  0.000000  0.000000  0.000000
13720     0.005209  0.000000  0.000000  0.000000  0.000000  0.000000
13721     0.000000  0.000000  0.000000  0.000000  0.000000  0.000000
```

```
15422     0.000000  0.000000  0.000691  0.000000  0.000000  0.000000
15859     0.000000  0.000000  0.000000  0.000000  0.000000  0.000000
21816     0.000000  0.000000  0.002632  0.000861  0.000000  0.001721
34970     0.000000  0.000000  0.000000  0.000000  0.000503  0.000000

term_str     æsops    ł20000
book_id
105       0.000000  0.000000
121       0.000000  0.000000
141       0.000000  0.000684
158       0.000000  0.000000
161       0.000000  0.000000
946       0.000000  0.000000
1212      0.000000  0.000000
1342      0.000000  0.000000
1900      0.000000  0.000000
2701      0.000000  0.000000
4045      0.000000  0.000000
8118      0.000000  0.000000
10712     0.000000  0.000000
13720     0.000000  0.000000
13721     0.000000  0.000000
15422     0.000000  0.000000
15859     0.000000  0.000000
21816     0.000861  0.000000
34970     0.000000  0.000000

[19 rows x 40281 columns]
```

```
TFIDF_books.mean().sort_values(ascending=False)\
    .head(20).to_frame('mean_tfidf').join(VOCAB.max_pos)
```

```
            mean_tfidf max_pos
term_str
elinor        0.033840     NNP
pierre        0.030911     NNP
vernon        0.025980     NNP
marianne      0.021347     NNP
emma          0.021164     NNP
darcy         0.019302     NNP
reginald      0.018486     NNP
babbalanja    0.018252     NNP
catherine     0.018238     NNP
frederica     0.017986     NNP
crawford      0.017749     NNP
fanny         0.017167     NNP
elliot        0.017053     NNP
```

```
weston        0.016591     NNP
media         0.015986     NNP
israel        0.015428     NNP
knightley     0.015184     NNP
tilney        0.013815     NNP
elton         0.013648     NNP
bingley       0.013264     NNP
```

Q3

Paragraphs

```
BOW_paras = create_bow(CORPUS, bag=PARAS)
```

```
BOW_paras
```

```
                                     n
book_id chap_id para_num term_str     
105     1       1        a           2
                         admiration  1
                         affairs     1
                         almost      1
                         always      1
...                                 ..
34970   114     24       of          1
                         or          1
                         pierre      1
                         project     1
                         the         1

[1470642 rows x 1 columns]
```

```
TFIDF_paras_max = get_tfidf(BOW_paras, tf_method='sum')
```

```
TFIDF_paras_max
```

```
term_str                    0         1   10  100  1000  10000  10000000  \
book_id chap_id para_num                                                    
105     1       1         0.0  0.000000  0.0  0.0   0.0    0.0       0.0   
                2         0.0  0.000000  0.0  0.0   0.0    0.0       0.0   
                3         0.0  0.399552  0.0  0.0   0.0    0.0       0.0   
                4         0.0  0.000000  0.0  0.0   0.0    0.0       0.0   
                5         0.0  0.000000  0.0  0.0   0.0    0.0       0.0   
...                       ...       ...  ...  ...   ...    ...       ...   
34970   114     18        0.0  0.000000  0.0  0.0   0.0    0.0       0.0   
                19        0.0  0.000000  0.0  0.0   0.0    0.0       0.0   
                20        0.0  0.000000  0.0  0.0   0.0    0.0       0.0   
                21        0.0  0.000000  0.0  0.0   0.0    0.0       0.0   
                24        0.0  0.000000  0.0  0.0   0.0    0.0       0.0   
```

```
term_str                    10440  10800  10th  ...  zoroaster  zozo  zuma  zur  \
book_id chap_id para_num                        ...
105     1       1             0.0    0.0   0.0  ...        0.0   0.0   0.0  0.0
                2             0.0    0.0   0.0  ...        0.0   0.0   0.0  0.0
                3             0.0    0.0   0.0  ...        0.0   0.0   0.0  0.0
                4             0.0    0.0   0.0  ...        0.0   0.0   0.0  0.0
                5             0.0    0.0   0.0  ...        0.0   0.0   0.0  0.0
...                           ...    ...   ...  ...        ...   ...   ...  ...
34970   114     18            0.0    0.0   0.0  ...        0.0   0.0   0.0  0.0
                19            0.0    0.0   0.0  ...        0.0   0.0   0.0  0.0
                20            0.0    0.0   0.0  ...        0.0   0.0   0.0  0.0
                21            0.0    0.0   0.0  ...        0.0   0.0   0.0  0.0
                24            0.0    0.0   0.0  ...        0.0   0.0   0.0  0.0

term_str                      à  æneas  æniad  æson  æsops  Ł20000
book_id chap_id para_num
105     1       1           0.0    0.0    0.0   0.0    0.0     0.0
                2           0.0    0.0    0.0   0.0    0.0     0.0
                3           0.0    0.0    0.0   0.0    0.0     0.0
                4           0.0    0.0    0.0   0.0    0.0     0.0
                5           0.0    0.0    0.0   0.0    0.0     0.0
...                         ...    ...    ...   ...    ...     ...
34970   114     18          0.0    0.0    0.0   0.0    0.0     0.0
                19          0.0    0.0    0.0   0.0    0.0     0.0
                20          0.0    0.0    0.0   0.0    0.0     0.0
                21          0.0    0.0    0.0   0.0    0.0     0.0
                24          0.0    0.0    0.0   0.0    0.0     0.0

[30459 rows x 40281 columns]
```

```
TFIDF_paras_max.mean().sort_values(ascending=False)\
    .head(20).to_frame('mean_tfidf').join(VOCAB.max_pos)
```

```
          mean_tfidf max_pos
term_str
i           0.025729     PRP
you         0.024533     PRP
the         0.021601      DT
of          0.017819      IN
a           0.016895      DT
to          0.016776      TO
and         0.016728      CC
is          0.016105     VBZ
he          0.016027     PRP
said        0.015729     VBD
her         0.015453    PRP$
```

```
it          0.015185     PRP
was         0.015107     VBD
his         0.014842    PRP$
in          0.014713      IN
my          0.014284    PRP$
not         0.014022      RB
that        0.013608      IN
she         0.013250     PRP
but         0.012186      CC
```

Chapters

```python
BOW_chaps = create_bow(CORPUS, bag=CHAPS)
```

```python
BOW_chaps
```

```
                          n
book_id chap_id term_str   
105     1       1         2
                15        1
                16        1
                1760      1
                1784      1
...                      ..
34970   114     ye        1
                yes       2
                yet       1
                young     2
                your      1

[726847 rows x 1 columns]
```

```python
TFIDF_chaps_max = get_tfidf(BOW_chaps, tf_method='sum')
```

```python
TFIDF_chaps_max
```

```
term_str           0         1   10  100  1000  10000  10000000  10440  10800  \
book_id chap_id                                                                 
105     1        0.0  0.005048  0.0  0.0   0.0    0.0       0.0    0.0    0.0   
        2        0.0  0.000000  0.0  0.0   0.0    0.0       0.0    0.0    0.0   
        3        0.0  0.000000  0.0  0.0   0.0    0.0       0.0    0.0    0.0   
        4        0.0  0.000000  0.0  0.0   0.0    0.0       0.0    0.0    0.0   
        5        0.0  0.000000  0.0  0.0   0.0    0.0       0.0    0.0    0.0   
...              ...       ...  ...  ...   ...    ...       ...    ...    ...   
34970   110      0.0  0.000000  0.0  0.0   0.0    0.0       0.0    0.0    0.0   
        111      0.0  0.000000  0.0  0.0   0.0    0.0       0.0    0.0    0.0   
        112      0.0  0.000000  0.0  0.0   0.0    0.0       0.0    0.0    0.0   
        113      0.0  0.000000  0.0  0.0   0.0    0.0       0.0    0.0    0.0   
        114      0.0  0.000000  0.0  0.0   0.0    0.0       0.0    0.0    0.0

term_str         10th  ...  zoroaster  zozo  zuma  zur    à  æneas  æniad  \
book_id chap_id        ...
105     1          0.0  ...        0.0   0.0   0.0  0.0  0.0    0.0    0.0
        2          0.0  ...        0.0   0.0   0.0  0.0  0.0    0.0    0.0
        3          0.0  ...        0.0   0.0   0.0  0.0  0.0    0.0    0.0
        4          0.0  ...        0.0   0.0   0.0  0.0  0.0    0.0    0.0
        5          0.0  ...        0.0   0.0   0.0  0.0  0.0    0.0    0.0
...                ...  ...        ...   ...   ...  ...  ...    ...    ...
34970   110        0.0  ...        0.0   0.0   0.0  0.0  0.0    0.0    0.0
        111        0.0  ...        0.0   0.0   0.0  0.0  0.0    0.0    0.0
        112        0.0  ...        0.0   0.0   0.0  0.0  0.0    0.0    0.0
        113        0.0  ...        0.0   0.0   0.0  0.0  0.0    0.0    0.0
        114        0.0  ...        0.0   0.0   0.0  0.0  0.0    0.0    0.0

term_str         æson  æsops  ł20000
book_id chap_id
105     1          0.0    0.0     0.0
        2          0.0    0.0     0.0
        3          0.0    0.0     0.0
        4          0.0    0.0     0.0
        5          0.0    0.0     0.0
...                ...    ...     ...
34970   110        0.0    0.0     0.0
        111        0.0    0.0     0.0
        112        0.0    0.0     0.0
        113        0.0    0.0     0.0
        114        0.0    0.0     0.0

[1185 rows x 40281 columns]
```

```
TFIDF_chaps_max.mean().sort_values(ascending=False)\
    .head(20).to_frame('mean_tfidf').join(VOCAB.max_pos)
```

```
              mean_tfidf max_pos
term_str
her             0.004327    PRP$
she             0.004150     PRP
cosmopolitan    0.003485      NN
pierre          0.003317     NNP
communion       0.003004      NN
i               0.002771     PRP
sailors         0.002668     NNS
you             0.002620     PRP
hypothetical    0.002437     NNP
mr              0.002084     NNP
```

```
and             0.002054      CC
confidential    0.002042      JJ
the             0.001972      DT
dream           0.001942      NN
boon            0.001857      NN
mrs             0.001747     NNP
elephants       0.001731      NN
whale           0.001715      NN
thou            0.001696      NN
acquaintance    0.001690      NN
```

Q5

```
AUS_IDX = LIB[LIB.author.str.contains('AUS')].index
MEL_IDX = LIB[LIB.author.str.contains('MEL')].index
```

```
aus_chap_bow = create_bow(CORPUS.loc[AUS_IDX], bag=CHAPS)
mel_chap_bow = create_bow(CORPUS.loc[MEL_IDX], bag=CHAPS)
```

```
aus_chap_bow
```

```
                          n
book_id chap_id term_str
105     1       1         2
                15        1
                16        1
                1760      1
                1784      1
...                      ..
1342    61      you       7
                young     1
                younger   1
                yours     1
                youth     1

[233724 rows x 1 columns]
```

```
mel_chap_bow
```

```
                            n
book_id chap_id term_str
1900    1       1595        1
                a          54
                abandoned   1
                aboard      1
                abortive    1
...                        ..
34970   114     ye          1
```

```
                yes         2
                yet         1
                young       2
                your        1

[493123 rows x 1 columns]
```

```
TFIDF_aus = get_tfidf(aus_chap_bow, tf_method='max')
TFIDF_mel = get_tfidf(mel_chap_bow, tf_method='max')
```

Method 1

```
A = TFIDF_aus.mean().sort_values(ascending=False).to_frame('mean_tfidf').join(VOCAB.max_pos
```

```
A[A.max_pos == 'JJ'].head(20).mean_tfidf
```

```
term_str
sure              0.013167
dear              0.012992
poor              0.012213
upper             0.011347
old               0.011327
agreeable         0.011301
young             0.010834
happy             0.010687
handsome          0.010642
general           0.010385
present           0.010303
few               0.010130
afraid            0.009894
impossible        0.009860
sorry             0.009823
amiable           0.009712
glad              0.009678
same              0.009620
last              0.009538
many              0.009340
Name: mean_tfidf, dtype: float64
```

```
M = TFIDF_mel.mean().sort_values(ascending=False).to_frame('mean_tfidf').join(VOCAB.max_pos
```

```
M[M.max_pos == 'JJ'].head(20).mean_tfidf
```

```
term_str
thy         0.028653
old         0.021042
ugh         0.015733
little      0.014585
good        0.014173
```

```
white     0.013809
many      0.013759
such      0.013335
much      0.013215
poor      0.012750
own       0.012663
great     0.012603
other     0.012348
sweet     0.012195
dear      0.012165
young     0.011964
hard      0.011697
last      0.011420
new       0.011265
dead      0.011121
Name: mean_tfidf, dtype: float64
```

Method 2

```
A[A.max_pos == 'JJ'].mean_tfidf.idxmax(), A[A.max_pos == 'JJ'].mean_tfidf.max()
```

```
('sure', 0.013166788165010412)
```

```
M[M.max_pos == 'JJ'].mean_tfidf.idxmax(), M[M.max_pos == 'JJ'].mean_tfidf.max()
```

```
('thy', 0.02865278371527089)
```