homework-1-pardes

February 2, 2021

[1]:
```
'''
IST 664
Homework 1
Sammy Pardes
1/28/21

data source: https://www.kaggle.com/tunguz/200000-jeopardy-questions

additionals sources:
https://pbpython.com/currency-cleanup.html
https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.
 ↪rename.html
https://pbpython.com/currency-cleanup.html
https://www.geeksforgeeks.org/
 ↪selecting-rows-in-pandas-dataframe-based-on-conditions/
https://stackoverflow.com/questions/8478602/
 ↪convert-a-list-of-string-sentences-to-words
https://stackoverflow.com/questions/45516207/
 ↪removing-stop-words-and-string-punctuation
https://stackoverflow.com/questions/38597503/
 ↪in-nltk-get-the-number-of-occurrences-of-a-trigram
'''
```

[1]:
```
'\nIST 664\nHomework 1\nSammy Pardes\n1/28/21\n\ndata source:
https://www.kaggle.com/tunguz/200000-jeopardy-questions \n\nadditionals
sources:\nhttps://pbpython.com/currency-
cleanup.html\nhttps://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.
DataFrame.rename.html\nhttps://pbpython.com/currency-
cleanup.html\nhttps://www.geeksforgeeks.org/selecting-rows-in-pandas-dataframe-
based-on-conditions/\nhttps://stackoverflow.com/questions/8478602/convert-a-
list-of-string-sentences-to-
words\nhttps://stackoverflow.com/questions/45516207/removing-stop-words-and-
string-punctuation\nhttps://stackoverflow.com/questions/38597503/in-nltk-get-
the-number-of-occurrences-of-a-trigram\n'
```

[2]:
```
#import statements
import pandas as pd
```

```
import nltk
from nltk import FreqDist
nltk.download("stopwords")
import string
from nltk.collocations import *
import re
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\slpar\AppData\Roaming\nltk_data…
[nltk_data]   Package stopwords is already up-to-date!
```

[3]:
```
#load data and preview first few rows
jeopardy = pd.read_csv("JEOPARDY_CSV.csv")

jeopardy.head()
```

[3]:

	Show Number	Air Date	Round	Category	Value	Question	Answer
0	4680	2004-12-31	Jeopardy!	HISTORY	$200	For the last 8 years of his life, Galileo was …	Copernicus
1	4680	2004-12-31	Jeopardy!	ESPN's TOP 10 ALL-TIME ATHLETES	$200	No. 2: 1912 Olympian; football star at Carlisl…	Jim Thorpe
2	4680	2004-12-31	Jeopardy!	EVERYBODY TALKS ABOUT IT…	$200	The city of Yuma in this state has a record av…	Arizona
3	4680	2004-12-31	Jeopardy!	THE COMPANY LINE	$200	In 1963, live on "The Art Linkletter Show", th…	McDonald's
4	4680	2004-12-31	Jeopardy!	EPITAPHS & TRIBUTES	$200	Signer of the Dec. of Indep., framer of the Co…	John Adams

[4]:
```
#rename columns to remove extra spaces

#create list of new column names
columns = ["show_number", "air_date", "round", "category", "value", "question",␣
 ↪"answer"]

#overwrite column names with values in the columns list
jeopardy.columns = columns

jeopardy.head()
```

[4]:

	show_number	air_date	round	category	value	question	answer
0	4680	2004-12-31	Jeopardy!	HISTORY	$200	For the last 8 years of his life, Galileo was …	Copernicus
1	4680	2004-12-31	Jeopardy!	ESPN's TOP 10 ALL-TIME ATHLETES	$200	No. 2: 1912 Olympian; football star at Carlisl…	Jim Thorpe
2	4680	2004-12-31	Jeopardy!	EVERYBODY TALKS ABOUT IT…	$200	The city of Yuma in this state has a record av…	Arizona
3	4680	2004-12-31	Jeopardy!	THE COMPANY LINE	$200	In 1963, live on "The Art Linkletter Show", th…	McDonald's
4	4680	2004-12-31	Jeopardy!	EPITAPHS & TRIBUTES	$200	Signer of the Dec. of Indep., framer of the Co…	John Adams

```
                                              question      answer
0  For the last 8 years of his life, Galileo was …  Copernicus
1  No. 2: 1912 Olympian; football star at Carlisl…  Jim Thorpe
2  The city of Yuma in this state has a record av…     Arizona
3  In 1963, live on "The Art Linkletter Show", th…  McDonald's
4  Signer of the Dec. of Indep., framer of the Co…  John Adams
```

[5]:
```
#determine unique question values in data set

#initialize empty list
values = []

#add unqiue values only to the values list
for value in jeopardy["value"]:
    if value not in values:
        values.append(value)

print(values)
```

```
['$200', '$400', '$600', '$800', '$2,000', '$1000', '$1200', '$1600', '$2000',
'$3,200', 'None', '$5,000', '$100', '$300', '$500', '$1,000', '$1,500',
'$1,200', '$4,800', '$1,800', '$1,100', '$2,200', '$3,400', '$3,000', '$4,000',
'$1,600', '$6,800', '$1,900', '$3,100', '$700', '$1,400', '$2,800', '$8,000',
'$6,000', '$2,400', '$12,000', '$3,800', '$2,500', '$6,200', '$10,000',
'$7,000', '$1,492', '$7,400', '$1,300', '$7,200', '$2,600', '$3,300', '$5,400',
'$4,500', '$2,100', '$900', '$3,600', '$2,127', '$367', '$4,400', '$3,500',
'$2,900', '$3,900', '$4,100', '$4,600', '$10,800', '$2,300', '$5,600', '$1,111',
'$8,200', '$5,800', '$750', '$7,500', '$1,700', '$9,000', '$6,100', '$1,020',
'$4,700', '$2,021', '$5,200', '$3,389', '$4,200', '$5', '$2,001', '$1,263',
'$4,637', '$3,201', '$6,600', '$3,700', '$2,990', '$5,500', '$14,000', '$2,700',
'$6,400', '$350', '$8,600', '$6,300', '$250', '$3,989', '$8,917', '$9,500',
'$1,246', '$6,435', '$8,800', '$2,222', '$2,746', '$10,400', '$7,600', '$6,700',
'$5,100', '$13,200', '$4,300', '$1,407', '$12,400', '$5,401', '$7,800',
'$1,183', '$1,203', '$13,000', '$11,600', '$14,200', '$1,809', '$8,400',
'$8,700', '$11,000', '$5,201', '$1,801', '$3,499', '$5,700', '$601', '$4,008',
'$50', '$2,344', '$2,811', '$18,000', '$1,777', '$3,599', '$9,800', '$796',
'$3,150', '$20', '$1,810', '$22', '$9,200', '$1,512', '$8,500', '$585',
'$1,534', '$13,800', '$5,001', '$4,238', '$16,400', '$1,347', '$2547',
'$11,200']
```

[6]:
```
#clean up value column

#remove questions where the Value is "None" and convert values to numbers
jeopardy = jeopardy[jeopardy.value != "None"]

#remove "$" and "," from values
jeopardy["value"] = jeopardy["value"].str.replace("$", "")
jeopardy["value"] = jeopardy["value"].str.replace(",", "")
```

```
#convert from string type to float type
jeopardy["value"] = jeopardy["value"].astype("int")

jeopardy["value"].head()
```

[6]:
```
0    200
1    200
2    200
3    200
4    200
Name: value, dtype: int32
```

[7]:
```
#create two data sets: lowest and highest value questions
numquestions = 20000

jeopardylow = jeopardy.nsmallest(numquestions, "value")

jeopardyhigh = jeopardy.nlargest(numquestions, "value")

print(jeopardylow.head())
print(jeopardyhigh.head())
```

```
       show_number    air_date              round         category  value  \
20789          496  1986-11-03         Jeopardy!     LEAD SINGERS      5
39193          838  1988-04-06  Double Jeopardy!          "WATER"      5
47067         1194  1989-11-09         Jeopardy!  NAME'S THE SAME      5
53179         4813  2005-07-06  Double Jeopardy!             ANTS      5
75426         4643  2004-11-10  Double Jeopardy!           TRAVEL      5

                                                question  \
20789  [Audio] Called "Buffoons of '60s British Rock …
39193  It's reported the Rolling Stones took their na…
47067  John Dos Passos work, or the group heard <a hr…
53179  These insects also known as plant lice are cap…
75426  The Peer Gynt ski area has been called this co…

                        answer
20789  Freddie And The Dreamers
39193              Muddy Waters
47067       Manhattan Transfer
53179                   aphids
75426                   Norway
        show_number    air_date              round                 category  \
150825         6246  2011-11-14  Double Jeopardy!                LANGUAGES
195755         6217  2011-10-04  Double Jeopardy!                   PLUS 8
88937          6221  2011-10-10  Double Jeopardy!              "A" IN MATH
32508          4140  2002-09-06  Double Jeopardy!       SAINTS ON THE MAP
```

```
188391         5839  2010-01-21  Double Jeopardy!  REMEMBERING TED KENNEDY

        value                                          question        answer
150825  18000  Although Dutch is the official language, Srana…      Suriname
195755  16400       Number of days in a leap year times 2 plus 8           740
88937   14200  It's the length from the base of a cone to the…  the altitude
32508   14000  Jesse James was terminated in this city, once …    St. Joseph
188391  13800  Kennedy called this the cause of his life & wa…   health care
```

[8]:
```
#get questions only from high/low lists and convert to lowercase
jeopardylowqs = jeopardylow["question"].str.lower()
jeopardyhighqs = jeopardyhigh["question"].str.lower()

print(jeopardylowqs.head())
print(jeopardyhighqs.head())
```

```
20789    [audio] called "buffoons of '60s british rock …
39193    it's reported the rolling stones took their na…
47067    john dos passos work, or the group heard <a hr…
53179    these insects also known as plant lice are cap…
75426    the peer gynt ski area has been called this co…
Name: question, dtype: object
150825    although dutch is the official language, srana…
195755         number of days in a leap year times 2 plus 8
88937     it's the length from the base of a cone to the…
32508     jesse james was terminated in this city, once …
188391    kennedy called this the cause of his life & wa…
Name: question, dtype: object
```

[9]:
```
#tokenize, remove stopwords, remove punctuation, remove HTML

#use regex to clean out HTML
#initialize empty list
html = []

#define HTML tag pattern with regex
pattern = '<.*[\s]?/?.*>?|target="_blank">.*\.?'

#if the word in any question appears in the regex patter, append to html list
for question in jeopardylowqs:
    for word in question.split(' '):
        if nltk.regexp_tokenize(word, pattern):
            html.append(word)

for question in jeopardyhighqs:
    for word in question.split(' '):
        if nltk.regexp_tokenize(word, pattern):
            html.append(word)
```

```
print(len(html), '\n')
print(html[:100])

#define stopwords
mystop = ['clue', 'crew', 'like', '"it\'ll', '"the', 'this,', 'also', 'may',␣
 ↪'1', '2', '3']

#get stopwords from nltk and add additional words
stopwords = nltk.corpus.stopwords.words('english')
stopwords = stopwords + mystop
```

```
6701

['<a', 'href="http://www.j-archive.com/media/1989-11-09_j_29.mp3">here</a>:',
'<i>"ooh', 'city…"</i>', '<a', 'following</a>', '<i>"i\'ll',
'faithfully…"</i>', '<a', 'target="_blank">here</a>', '<a',
'target="_blank">here</a>', '<a', 'target="_blank">here</a>', '<a',
'target="_blank"><big>&divide;</big></a>', '<a',
'href="http://www.j-archive.com/media/1987-11-12_j_13.mp3">following</a>',
'(<a', 'target="_blank">hi,', 'hasselhoff.</a>)', '(<a', 'laboratory.</a>)',
'<a', 'music]</a>', '<a', 'target="_blank">here</a>', '<a',
'target="_blank">here</a>:', '<i>vogue</i>', '<a', 'target="_blank">here</a>',
'(<a', 'evening!</a>)', '(<a', 'background.</a>)', '<a',
'target="_blank">here</a>:', '<i>"suddenly', 'me"</i>', '<a',
'target="_blank">following</a>', '<i><a', 'plane!"</a></i>', '(<a',
'target="_blank">hi,', 'russ.</a>)', '(<a', 'clue.</a>)', '(<a', 'reads.</a>)',
'(<a', 'language.</a>)', '<a', 'target="_blank">-le-', '--r-et</a>', '<a',
'target="_blank">here</a>', '<a', 'target="_blank">here</a>', '<a',
'target="_blank">here</a>:', '(<i>thunderball</i>)', '<a', 'target="_blank">56',
'(4)</a>', '<a',
'href="http://www.j-archive.com/media/2001-01-09_j_30.mp3">here</a>', '<a',
'href="http://www.j-archive.com/media/1989-09-12_j_06.mp3">[audio]</a>',
'<i>meet', 'wrong.</i>', '<a', 'target="_blank">here</a>', '<a',
'target="_blank">here</a>:', '<a', 'target="_blank">[state', 'outline]</a>',
'<b>whse</b>', '<b>whsle</b>', '(<a', 'target="_blank">hi.', '<i>nypd',
'blue</i>.</a>)', '<i>buffalo', 'news</i>', '<i><a', 'dwell…"</a></i>', '<a',
'href="http://www.j-archive.com/media/2001-09-21_j_17.wmv">here</a>', '<i>at',
'government…"</i>', '<a', 'target="_blank">here</a>', '(<a',
'target="_blank">sofia', 'lab.</a>)', '<i>"hello,', 'friend"</i>']
```

[10]:
```
#split questions on whitespace to get tokenized words
#put in list if not a stopword and not punctuation

#initialize empty token lists
lowtokens = []
hightokens = []
```

```
#for each question, split on white space, add to token list if not stopword,␣
 ↪punctuation, or html
for question in jeopardylowqs:
    for word in question.split(' '):
        if word not in stopwords and word not in string.punctuation and word␣
 ↪not in html:
            lowtokens.append(word)

for question in jeopardyhighqs:
    for word in question.split(' '):
        if word not in stopwords and word not in string.punctuation and word␣
 ↪not in html:
            hightokens.append(word)

print(lowtokens[:100])
print(hightokens[:100])
```

```
['[audio]', 'called', '"buffoons', "'60s", 'british', 'rock', 'invasion",',
'led', 'ex-milkman', 'named', 'garrity:', 'reported', 'rolling', 'stones',
'took', 'name', 'following', 'blues', 'song', 'singer:', 'john', 'dos',
'passos', 'work,', 'group', 'heard', 'wah', 'ooh', 'wah', 'cool', 'cool',
'kitty', 'asks', 'boy', 'new', 'york', 'insects', 'known', 'plant', 'lice',
'captured', '"milked"', 'many', 'ants', 'honeydew', 'liquid', 'produce', 'peer',
'gynt', 'ski', 'area', 'called', "country's", 'best', 'place', 'cross-country',
'skiing', 'latin', '"to', 'correct",', 'adjective', 'someone', "can't",
'corrected', 'reformed', '(jimmy', 'carnegie', 'mellon', 'university',
'pittsburgh)', 'greek', '"self-acting",', 'another', 'word', 'robot', 'mimics',
'human', 'actions', 'master', 'craftsman,', 'invented', 'axe', 'built',
'labyrinth', '1964', 'elvis', 'bought', 'yacht', 'owned', 'ex-president',
'$55,000,', 'donated', 'march', 'dimes', 'john', 'howard', 'griffin',
'chemically', 'darkened', 'skin']
['although', 'dutch', 'official', 'language,', 'sranan', 'tongo', 'spoken',
'people', 'south', 'american', 'country', 'number', 'days', 'leap', 'year',
'times', 'plus', '8', 'length', 'base', 'cone', 'apex', 'jesse', 'james',
'terminated', 'city,', 'home', 'terminus', 'pony', 'express', 'kennedy',
'called', 'cause', 'life', 'hoping', 'see', 'reform', 'bill', 'passed', 'died',
'english', 'means', '"truthful";', 'german', '"frenchman"', 'tiller', 'engine,',
'start"', 'receiving', '"rock', 'solid"', 'education', 'stanford,', 'lou',
'hoover', 'first', 'woman', 'earn', 'degree', 'aconcagua', 'one',
'alliteratively', 'known', '"7"', 'pd:', 'great', 'place', 'live', 'music',
'small', 'masses', 'lymphoid', 'tissue', 'nasopharynx', 'film', 'title',
'"eternal', 'sunshine', 'spotless', 'mind"', 'comes', 'poem', 'ill-fated',
'medieval', 'lovers', 'annual', 'fishing', 'derby', 'fish', 'features', 'one',
'tagged', 'specimen', 'worth', '$100,000', 'africa', 'asia', 'joined',
'isthmus', 'separates']
```

[16]:
```
#list the top 50 low value words by frequency (normalized by the length of the␣
↪document)

#get number of words in each list
lowlen = len(lowtokens)
highlen = len(hightokens)

#get frequency distribution
lowdist = nltk.FreqDist(lowtokens)
lowdist

#get normalized frequencies
for word, freq in lowdist.most_common(50):
    print(word, freq/lowlen)
```

```
one 0.007513674129871113
first 0.006195153648642681
name 0.004624284749954262
city 0.0034256297670193234
called 0.0029777113260278467
u.s. 0.0029209329320993497
new 0.002681201935512362
state 0.0025297928850363696
made 0.002466705780671373
named 0.0024603970702348735
country 0.0023973099658698765
type 0.0023783838345603774
film 0.0021449615484098895
seen 0.00213865283797339
used 0.0020440221814258946
man 0.001924156683132401
known 0.0018989218413864022
became 0.0018547608683309046
played 0.001709660528291412
capital 0.001709660528291412
title 0.001690734396981913
years 0.0015897950299979182
president 0.001570868898688419
part 0.001495164373450423
john 0.00135637274384743
term 0.0013437553229744307
famous 0.0013248291916649318
word 0.0012869769290459337
said 0.0012680507977364348
people 0.001261742087299935
home 0.0012554333768634355
book 0.0012491246664269357
largest 0.001242815955990436
```

```
world 0.0012365072455539363
born 0.001204963693371438
last 0.0011923462724984387
hit 0.001179728851625439 3
show 0.001179728851625439 3
american 0.0011797288516254393
make 0.0011734201411889396
war 0.0011481852994429408
many 0.0011355678785699415
get 0.0011229504576969421
tv 0.0011040243263874432
day 0.001085098195077944
national 0.0010850981950779 44
time 0.0010787894846414445
song 0.0010409372220224465
found 0.0010409372220224465
island 0.0010220110907129473
```

[17]:
```
#list the top 50 high value words by frequency (normalized by the length of the␣
 ↪document)

#get frequency distribution
highdist = nltk.FreqDist(hightokens)

#get normalized frequencies
for word, freq in highdist.most_common(50):
    print(word, freq/highlen)
```

```
name 0.006952940329821803
one 0.005535780517374047
first 0.005054167612362505
called 0.0033048609688723063
named 0.00328271784680281
seen 0.0029948572598993595
title 0.0027457471366175274
type 0.002695925111961161
city 0.0026461030873047944
new 0.002568602160061558
known 0.0024523507691967027
used 0.0024246718666098327
word 0.002341635158849222
u.s. 0.00214788284074113
film 0.0020980608160847637
made 0.001976273644702535
state 0.0019652020836677868
country 0.0019541305226330385
man 0.0019264516200461683
french 0.0018434149122855577
```

```
novel 0.0017880571071118173
last 0.0017382350824554508
wrote 0.0016551983746948401
became 0.0016330552526253438
term 0.0016109121305558478
part 0.0015832332279689775
means 0.0015500185448647332
american 0.0015168038617604889
capital 0.0014448387150346263
greek 0.001417159812447756
president 0.0013894809098608859
latin 0.0013784093488261378
years 0.0013728735683087637
island 0.0013009084215829011
british 0.001273229518996031
played 0.0012621579579612826
war 0.0012566221774439087
play 0.0012510863969265346
work 0.0012510863969265346
king 0.0012455506164091607
whose 0.0012400148358917866
famous 0.0012400148358917866
south 0.0012289432748570385
reports 0.0012234074943396644
century 0.0012123359333049164
said 0.0012012643722701683
get 0.0012012643722701683
john 0.0011846570307180461
meaning 0.0011846570307180461
book 0.0011403707865790538
```

[18]:
```
#list the top 50 low-value bigrams by frequencies

#create shorthand for full measures function
bgmeasures = nltk.collocations.BigramAssocMeasures()

qfinderlow = BigramCollocationFinder.from_words(lowtokens)
qscoredlow = qfinderlow.score_ngrams(bgmeasures.raw_freq)

for bigram in qscoredlow[:50]:
    print(bigram)
```

```
(('new', 'york'), 0.0006750320167054652)
(('became', 'first'), 0.0005930187810309695)
(('one', 'these,'), 0.00029020068007898505)
(('last', 'name'), 0.0002838919696424854)
(('capital', 'city'), 0.000233422286150488)
(('seen', 'here,'), 0.0002081874444044893)
```

```
(('prime', 'minister'), 0.00019557002353148993)
(('whose', 'name'), 0.00019557002353148993)
(('first', 'lady'), 0.00018926131309499026)
(('19th', 'century'), 0.00018295260265849058)
(('civil', 'war'), 0.00018295260265849058)
(('first', 'name'), 0.00018295260265849058)
(('title', 'character'), 0.0001766438922219909)
(('york', 'city'), 0.00017033518178549123)
(('tv', 'show'), 0.00016402647134899155)
(('white', 'house'), 0.00015140905047599220)
(('world', 'war'), 0.00014510034003949252)
(("world's", 'largest'), 0.00014510034003949252)
(('became', '1st'), 0.00013879162960299285)
(('first', 'woman'), 0.00013879162960299285)
(('hall', 'fame'), 0.00013879162960299285)
(('national', 'park'), 0.00013879162960299285)
(('shares', 'name'), 0.00013879162960299285)
(('name', 'means'), 0.00013248291916649317)
(('san', 'francisco'), 0.00013248291916649317)
(('seen', 'here:'), 0.00013248291916649317)
(('u.s.', 'president'), 0.00013248291916649317)
(('years', 'later'), 0.00013248291916649317)
(('could', 'tell'), 0.00012617420872999350)
(('first', 'u.s.'), 0.00012617420872999350)
(('ice', 'cream'), 0.00012617420872999350)
(('new', 'jersey'), 0.00012617420872999350)
(('years', 'ago'), 0.00012617420872999350)
(('better', 'known'), 0.00011986549829349382)
(('gave', 'us'), 0.00011986549829349382)
(('high', 'school'), 0.00011986549829349382)
(('united', 'states'), 0.00011986549829349382)
(('said,', '"i'), 0.00011355678785699416)
(('body', 'part'), 0.00010724807742049448)
(('first', 'time'), 0.00010724807742049448)
(('largest', 'city'), 0.00010724807742049448)
(('made', 'first'), 0.00010724807742049448)
(('no.', 'hit'), 0.00010724807742049448)
(('south', 'american'), 0.00010724807742049448)
(('state', 'capital'), 0.00010724807742049448)
(('washington,', 'd.c.'), 0.00010724807742049448)
(('best', 'known'), 0.00010093936698399480)
(('north', 'american'), 0.00010093936698399480)
(('theme', 'song'), 0.00010093936698399480)
(('used', 'make'), 0.00010093936698399480)
```

[19]:
```
#list the top 50 high-value bigrams by frequencies
qfinderhigh = BigramCollocationFinder.from_words(hightokens)
```

```
qscoredhigh = qfinderhigh.score_ngrams(bgmeasures.raw_freq)

for bigram in qscoredhigh[:50]:
    print(bigram)
```

```
(('new', 'york'), 0.0005535780517374047)
(('last', 'name'), 0.0004594697829420459)
(('whose', 'name'), 0.00044286244138992377)
(('became', 'first'), 0.0003985761972509314)
(('19th', 'century'), 0.0003930404167335573)
(('title', 'character'), 0.00028786058690345045)
(('first', 'name'), 0.0002767892586870236)
(('prime', 'minister'), 0.0002767892586870236)
(('shares', 'name'), 0.00024357434276445808)
(('south', 'american'), 0.00023803856224708403)
(('world', 'war'), 0.00023803856224708403)
(('name', 'means'), 0.00022143122069496189)
(('word', 'meaning'), 0.00022143122069496189)
(('nobel', 'prize'), 0.00019375231810809165)
(('daily', 'double):'), 0.00018268075707334356)
(('comes', 'latin'), 0.0001771449765559695)
(('one', 'these,'), 0.0001771449765559695)
(('supreme', 'court'), 0.0001771449765559695)
(('seen', 'here,'), 0.00017160919603859546)
(('african', 'country'), 0.00016607341552122141)
(('best', 'known'), 0.00016053763500384737)
(('gave', 'us'), 0.00015500185448647332)
(('national', 'park'), 0.00015500185448647332)
(('20th', 'century'), 0.00014393029345172523)
(('capital', 'city'), 0.00013839451293435118)
(('new', 'jersey'), 0.00013839451293435118)
(('united', 'states'), 0.00013839451293435118)
(('latin', '"to'), 0.00013285873241697713)
(('name', 'greek'), 0.00013285873241697713)
(('civil', 'war'), 0.00012732295189960308)
(('first', 'woman'), 0.00012732295189960308)
(('hall', 'fame'), 0.00012732295189960308)
(('white', 'house'), 0.00012732295189960308)
(('(video', 'daily'), 0.00012178717138222904)
(('body', 'water'), 0.00012178717138222904)
(('name', 'comes'), 0.00012178717138222904)
(('takes', 'place'), 0.00012178717138222904)
(('word', 'means'), 0.00012178717138222904)
(('work', 'seen'), 0.00012178717138222904)
(('bears', 'name'), 0.00011625139086485499)
(('comes', 'greek'), 0.00011625139086485499)
(('north', 'carolina'), 0.00011625139086485499)
(('secretary', 'state'), 0.00011625139086485499)
```

```
(('shows', 'map'), 0.00011625139086485499)
(('used', 'make'), 0.00011625139086485499)
(('17th', 'century'), 0.00011071561034748094)
(('often', 'used'), 0.00011071561034748094)
(('state', 'university'), 0.00011071561034748094)
(('washington,', 'd.c.'), 0.00011071561034748094)
(('high', 'school'), 0.0001051798298301069)
```

[20]:
```
#list the top 50 low-value bigrams by their Mutual Information scores (using␣
↪min frequency 5)

#score by PMI metric, filtering to be sure the bigrams appear at least 5 times
qfinderlow.apply_freq_filter(5)
lowqpmi = qfinderlow.score_ngrams(bgmeasures.pmi)

for bigram in lowqpmi[:50]:
    print(bigram)
```

```
(('"annie', 'hall"'), 14.95229534038602)
(('"robinson', 'crusoe"'), 14.95229534038602)
(('"moby', 'dick"'), 14.689260934552223)
(('"wheel', 'fortune"'), 14.466868513215775)
(('barbra', 'streisand'), 14.466868513215775)
(('mick', 'jagger'), 14.466868513215775)
(('orson', 'welles'), 14.274223435273381)
(('potent', 'potable'), 14.203834107381983)
(('da', 'vinci'), 14.104298433831065)
(('puerto', 'rico'), 14.011189029439585)
(('conan', 'doyle'), 13.981441686045535)
(('"joy', 'cooking"'), 13.782370338943704)
(('steven', 'spielberg'), 13.596151530160743)
(('bruce', 'willis'), 13.42622652871843)
(('ronald', 'reagan'), 13.32936498946584)
(('debbie', 'reynolds'), 13.274223435273381)
(('babe', 'ruth'), 13.256301527276118)
(('heavyweight', 'boxing'), 13.077826222469875)
(('warner', 'bros.'), 13.077826222469875)
(('los', 'angeles'), 13.026295921829792)
(('eddie', 'murphy'), 12.841264027997273)
(('fairy', 'godmother'), 12.689260934552223)
(('"star', 'wars"'), 12.677288292886146)
(('"star', 'trek"'), 12.592399395299633)
(('"happy', 'days"'), 12.573783717132287)
(('"tonight', 'show"'), 12.487627073382573)
(('martin', 'luther'), 12.367332839664861)
(('las', 'vegas'), 12.367332839664858)
(('tonight', 'show"'), 12.265234652046125)
(('woody', 'allen'), 12.211213637747578)
```

```
(('nursery', 'rhyme,'), 12.174687761722463)
(('nursery', 'rhyme'), 12.077826222469875)
(('patron', 'saint'), 12.05022176107527)
(('jimmy', 'stewart'), 12.03390910593967)
(('jimmy', 'carter'), 12.007436894578479)
(('active', 'volcano'), 11.988821216411132)
(('ivy', 'league'), 11.95229534038602)
(('stephen', 'crane'), 11.923726188189248)
(('grand', 'slam'), 11.881906012494618)
(('washington,', 'd.c.'), 11.85389163632502)
(('super', 'bowl'), 11.739301617051815)
(('degrees', 'fahrenheit'), 11.641955219773866)
(('monetary', 'unit'), 11.618871606660825)
(('al', 'gore'), 11.573783717132288)
(('johnny', 'cash'), 11.573783717132288)
(('declaration', 'independence'), 11.554492378523527)
(('yellow', 'brick'), 11.551757410802288)
(('golden', 'gate'), 11.537257841107174)
(('peanut', 'butter'), 11.528269057879918)
(('prime', 'minister'), 11.51417422799413)
```

[21]:
```
#list the top 50 high-value bigrams by their Mutual Information scores (using␣
↪min frequency 5)

#score by PMI metric, filtering to be sure the bigrams appear at least 5 times
qfinderhigh.apply_freq_filter(5)
highqpmi = qfinderhigh.score_ngrams(bgmeasures.pmi)

for bigram in highqpmi[:50]:
    print(bigram)
```

```
(('agatha', 'christie'), 14.655426903131012)
(('e.m.', 'forster'), 14.655426903131012)
(('los', 'angeles'), 14.655426903131012)
(('"gone', 'wind"'), 14.39239249729722)
(('midsummer', "night's"), 14.292856823746305)
(('nicolas', 'cage'), 14.240389403852166)
(('clint', 'eastwood'), 14.140853730301252)
(('h.g.', 'wells'), 14.140853730301252)
(('ralph', 'waldo'), 13.87781932446746)
(('edgar', 'allan'), 13.72581623102241)
(('sherlock', 'holmes'), 13.655426903131012)
(('t.s.', 'eliot'), 13.655426903131012)
(('hong', 'kong'), 13.614784918633667)
(("night's", 'dream"'), 13.292856823746302)
(('headquarters', 'tokyo,'), 13.276915279877283)
(('spinal', 'cord'), 13.26638461238511)
(('julius', 'caesar'), 13.240389403852168)
```

```
(('teddy', 'roosevelt'), 13.225742627887765)
(('allan', 'poe'), 13.199747419354823)
(('h.w.', 'bush'), 13.112284578104484)
(('eugene', "o'neill"), 13.044929310302718)
(('en', 'route'), 12.887872989131383)
(('julia', 'roberts'), 12.740315800717525)
(('las', 'vegas'), 12.555891229580098)
(('homeland', 'security'), 12.462781825188618)
(('double):', '"(hi,'), 12.418387705830163)
(('super', 'bowl'), 12.352357835494965)
(('sony', 'headquarters'), 12.276915279877283)
(("earth's", 'crust'), 12.233963134692736)
(('woody', 'allen'), 12.233963134692734)
(('lewis', 'carroll'), 12.140853730301256)
(('martial', 'arts'), 12.112284578104484)
(('coat', 'arms'), 12.062851218299977)
(('ivy', 'league'), 12.036517070486518)
(('fits', 'category'), 11.887872989131383)
(('(video', 'daily'), 11.818925635413892)
(('(audio', 'daily'), 11.81892563541389)
(('daily', 'double):'), 11.81892563541389)
(('marine', 'corps'), 11.814809526941659)
(('coen', 'brothers'), 11.810705128608923)
(('f.', 'scott'), 11.778283650916544)
(('anatomical', 'animation'), 11.769294867689291)
(('periodic', 'table'), 11.687262990253014)
(('gold', 'medalist'), 11.681422111663956)
(('string', 'quartet'), 11.629891811023874)
(('jimmy', 'carter'), 11.555891229580098)
(('san', 'francisco,'), 11.555891229580096)
(('plane', 'crash'), 11.51524924508275)
(('fit', 'category'), 11.482234187854788)
(('running', 'mate'), 11.451554569765362)
```

[22]:
```
#list the top 50 low-value trigrams by frequencies

#create shorthand for full measures function
trimeasures = nltk.collocations.TrigramAssocMeasures()

trifinderlow = TrigramCollocationFinder.from_words(lowtokens)
triscoredlow = trifinderlow.score_ngrams(trimeasures.raw_freq)

for trigram in triscoredlow[:50]:
    print(trigram)
```

```
(('new', 'york', 'city'), 0.00017033518178549123)
(('became', 'first', 'woman'), 8.832194611099545e-05)
(('world', 'war', 'ii'), 6.939581480149642e-05)
```

(('whose', 'name', 'means'), 6.308710436499675e-05)
(('british', 'prime', 'minister'), 4.4160973055497726e-05)
(('nobel', 'peace', 'prize'), 4.4160973055497726e-05)
(('feet', 'sea', 'level'), 3.1543552182498374e-05)
(('gave', 'us', 'name'), 3.1543552182498374e-05)
(('john', 'f.', 'kennedy'), 3.1543552182498374e-05)
(('john', 'paul', 'ii'), 3.1543552182498374e-05)
(('north', 'american', 'country'), 3.1543552182498374e-05)
(('real', 'first', 'name'), 3.1543552182498374e-05)
(('talk', 'show', 'host'), 3.1543552182498374e-05)
(('top', '40', 'hit'), 3.1543552182498374e-05)
(('world', 'war', 'i,'), 3.1543552182498374e-05)
(('"monday', 'night', 'football"'), 2.52348417459987e-05)
(('"saturday', 'night', 'fever"'), 2.52348417459987e-05)
(('american', 'red', 'cross'), 2.52348417459987e-05)
(('arthur', 'conan', 'doyle'), 2.52348417459987e-05)
(('became', 'first', 'black'), 2.52348417459987e-05)
(('became', 'first', 'u.s.'), 2.52348417459987e-05)
(('celebrated', '50th', 'anniversary'), 2.52348417459987e-05)
(('e', 'street', 'band'), 2.52348417459987e-05)
(('francis', 'ford', 'coppola'), 2.52348417459987e-05)
(('future', 'first', 'lady'), 2.52348417459987e-05)
(('ice', 'cream', 'flavor'), 2.52348417459987e-05)
(('ivy', 'league', 'school'), 2.52348417459987e-05)
(('july', '4,', '1826'), 2.52348417459987e-05)
(('late', '19th', 'century'), 2.52348417459987e-05)
(('league', 'baseball', 'team'), 2.52348417459987e-05)
(('made', 'first', 'appearance'), 2.52348417459987e-05)
(('major', 'league', 'baseball'), 2.52348417459987e-05)
(('mayor', 'new', 'york'), 2.52348417459987e-05)
(('new', 'south', 'wales'), 2.52348417459987e-05)
(('new', "year's", 'eve'), 2.52348417459987e-05)
(('new', 'york', 'city,'), 2.52348417459987e-05)
(('president', 'united', 'states'), 2.52348417459987e-05)
(('sir', 'arthur', 'conan'), 2.52348417459987e-05)
(('st.', "patrick's", 'day'), 2.52348417459987e-05)
(('world', 'war', 'ii,'), 2.52348417459987e-05)
(('"a', 'midsummer', "night's"), 1.8926131309499025e-05)
(('"as', 'good', 'gets"'), 1.8926131309499025e-05)
(('"beautiful', 'blue"', 'river'), 1.8926131309499025e-05)
(('"blond', 'ambition"', 'tour'), 1.8926131309499025e-05)
(('"goodbye', 'yellow', 'brick'), 1.8926131309499025e-05)
(('"i', 'love', 'lucy"'), 1.8926131309499025e-05)
(('"joy', 'cooking"', 'says'), 1.8926131309499025e-05)
(('"little', 'women"', 'author'), 1.8926131309499025e-05)
(('"my', 'heart', 'go'), 1.8926131309499025e-05)
(('"saturday', 'night', 'live"'), 1.8926131309499025e-05)

[23]:
```
#list the top 50 high-value trigrams by frequencies

trifinderhigh = TrigramCollocationFinder.from_words(hightokens)
triscoredhigh = trifinderhigh.score_ngrams(trimeasures.raw_freq)

for trigram in triscoredhigh[:50]:
    print(trigram)
```

```
(('whose', 'name', 'means'), 0.00013285873241697713)
(('(video', 'daily', 'double):'), 0.00011625139086485499)
(('world', 'war', 'ii'), 9.41082687953588e-05)
(('new', 'york', 'city'), 8.303670776061071e-05)
(('(audio', 'daily', 'double):'), 6.642936620848857e-05)
(('nobel', 'peace', 'prize'), 4.9822024656366424e-05)
(('whose', 'name', 'comes'), 4.9822024656366424e-05)
(('"a', 'midsummer', "night's"), 4.428624413899238e-05)
(('early', '20th', 'century'), 4.428624413899238e-05)
(('shows', 'anatomical', 'animation'), 4.428624413899238e-05)
(('name', 'comes', 'latin'), 3.875046362161833e-05)
(('south', 'american', 'country'), 3.875046362161833e-05)
(('world', 'war', 'ii,'), 3.875046362161833e-05)
(('became', 'first', 'woman'), 3.321468310424428e-05)
(('chief', 'justice', 'u.s.'), 3.321468310424428e-05)
(('civil', 'rights', 'leader'), 3.321468310424428e-05)
(('comes', 'greek', 'words'), 3.321468310424428e-05)
(('comes', 'latin', 'word'), 3.321468310424428e-05)
(('comes', 'word', 'meaning'), 3.321468310424428e-05)
(('daily', 'double):', '"(hi,'), 3.321468310424428e-05)
(('double):', '"(hi,', "i'm"), 3.321468310424428e-05)
(('gave', 'us', 'word'), 3.321468310424428e-05)
(('midsummer', "night's", 'dream"'), 3.321468310424428e-05)
(('new', 'york', 'state'), 3.321468310424428e-05)
(('new', 'york', 'times'), 3.321468310424428e-05)
(('real', 'first', 'name'), 3.321468310424428e-05)
(('supreme', 'court', 'justice'), 3.321468310424428e-05)
(('air', 'force', 'base'), 2.7678902586870236e-05)
(('became', 'first', 'black'), 2.7678902586870236e-05)
(('best', 'picture', 'oscar'), 2.7678902586870236e-05)
(('central', 'american', 'country'), 2.7678902586870236e-05)
(('edgar', 'allan', 'poe'), 2.7678902586870236e-05)
(('george', 'h.w.', 'bush'), 2.7678902586870236e-05)
(('grand', 'central', 'terminal'), 2.7678902586870236e-05)
(('major', 'league', 'baseball'), 2.7678902586870236e-05)
(('million', 'years', 'ago'), 2.7678902586870236e-05)
(('named', '19th', 'century'), 2.7678902586870236e-05)
(('nobel', 'prize', 'literature'), 2.7678902586870236e-05)
(('north', 'carolina', 'state'), 2.7678902586870236e-05)
(('sony', 'headquarters', 'tokyo,'), 2.7678902586870236e-05)
```

```
(('whose', 'work', 'seen'), 2.7678902586870236e-05)
(('word', 'meaning', '"to'), 2.7678902586870236e-05)
(('19th', 'century', 'french'), 2.214312206949619e-05)
(('5th', 'century', 'b.c.'), 2.214312206949619e-05)
(('add', 'letter', 'country'), 2.214312206949619e-05)
(('arthur', 'miller', 'play'), 2.214312206949619e-05)
(('aung', 'san', 'suu'), 2.214312206949619e-05)
(('became', 'first', 'man'), 2.214312206949619e-05)
(('comes', 'words', 'meaning'), 2.214312206949619e-05)
(('daphne', 'du', 'maurier'), 2.214312206949619e-05)
```

[ ]: