Taller 9

Métodos Computacionales para Políticas Públicas - URosario

Entrega: viernes 24-abr-2020 11:59 PM

Juan Sebastián Gómez

juansebastian.gomezm@urosario.edu.co (mailto:juansebastian.gomezm@urosario.edu.co)

Instrucciones:

- Guarde una copia de este *Jupyter Notebook* en su computador, idealmente en una carpeta destinada al material del curso.
- Modifique el nombre del archivo del *notebook*, agregando al final un guión inferior y su nombre y apellido, separados estos últimos por otro guión inferior. Por ejemplo, mi *notebook* se llamaría: mcpp_taller9_santiago_matallana
- Marque el *notebook* con su nombre y e-mail en el bloque verde arriba. Reemplace el texto "[Su nombre acá]" con su nombre y apellido. Similar para su e-mail.
- Desarrolle la totalidad del taller sobre este *notebook*, insertando las celdas que sea necesario debajo de cada pregunta. Haga buen uso de las celdas para código y de las celdas tipo *markdown* según el caso.
- Recuerde salvar periódicamente sus avances.
- Cuando termine el taller:
  1. Descárguelo en PDF. Si tiene algún problema con la conversión, descárguelo en HTML.
  2. Suba todos los archivos a su repositorio en GitHub, en una carpeta destinada exclusivamente para este taller, antes de la fecha y hora límites.

NLTK Book (http://www.nltk.org/book/ (http://www.nltk.org/book/)), ejercicios:

- Capítulo 1: 22, 26, 28
- Capítulo 2: 2, 4, 11

Capítulo 1

Punto 22

- Find all the four-letter words in the Chat Corpus (text5). With the help of a frequency distribution (FreqDist), show these words in decreasing order of frequency.

In [1]:
```
import nltk
import matplotlib.pyplot as plt
from nltk.book import *
```

```
*** Introductory Examples for the NLTK Book ***
Loading text1, ..., text9 and sent1, ..., sent9
Type the name of the text or sentence to view it.
Type: 'texts()' or 'sents()' to list the materials.
text1: Moby Dick by Herman Melville 1851
text2: Sense and Sensibility by Jane Austen 1811
text3: The Book of Genesis
text4: Inaugural Address Corpus
text5: Chat Corpus
text6: Monty Python and the Holy Grail
text7: Wall Street Journal
text8: Personals Corpus
text9: The Man Who Was Thursday by G . K . Chesterton 1908
```

In [2]:
```
frec_4 = FreqDist(x for x in text5 if len(x) == 4)
print(frec_4.most_common(100))
```

```
[('JOIN', 1021), ('PART', 1016), ('that', 274), ('what', 183), ('here', 181),
('....', 170), ('have', 164), ('like', 156), ('with', 152), ('chat', 142), ('yo
ur', 137), ('good', 130), ('just', 125), ('lmao', 107), ('know', 103), ('room',
98), ('from', 92), ('this', 86), ('well', 81), ('back', 78), ('hiya', 78), ('th
ey', 77), ('dont', 75), ('yeah', 75), ('want', 71), ('love', 60), ('guys', 58),
('some', 58), ('been', 57), ('talk', 56), ('nice', 52), ('time', 50), ('when',
48), ('haha', 44), ('make', 44), ('girl', 43), ('need', 43), ('U122', 42), ('MO
DE', 41), ('will', 40), ('much', 40), ('then', 40), ('over', 39), ('work', 38),
('were', 38), ('take', 37), ('U121', 36), ('U115', 36), ('song', 36), ('even',
35), ('does', 35), ('seen', 35), ('U156', 35), ('U105', 35), ('more', 34), ('da
mn', 34), ('only', 33), ('come', 33), ('hell', 29), ('long', 28), ('them', 28),
('name', 27), ('tell', 27), ('away', 26), ('sure', 26), ('look', 26), ('baby',
26), ('call', 26), ('play', 25), ('U110', 25), ('U114', 25), ('NICK', 24), ('do
wn', 24), ('cool', 24), ('sexy', 23), ('many', 23), ('hate', 23), ('said', 23),
('last', 22), ('ever', 22), ('hear', 21), ('life', 21), ('live', 20), ('feel',
19), ('very', 19), ('mean', 19), ('give', 19), ('same', 19), ('must', 19), ('st
op', 19), ('LMAO', 19), ('!!!!', 18), ('hugs', 18), ('What', 18), ('find', 18),
('cant', 18), ('left', 17), ('????', 17), ('shit', 17), ('nite', 17)]
```

Punto 26

- What does the following Python code do? sum(len(w) for w in text1) Can you use it to work out the average word length of a text?

In [3]:
```
# El código va por las palabras del texto 1 y cuenta el número de caracteres en e

caracteres = sum(len(w) for w in text1)
print('Los caracteres en el texto son: ', caracteres)

palabras = len(text1)
average = caracteres/palabras
print('El promedio de tamaño de las palabras es: ', average)
```

```
Los caracteres en el texto son:  999044
El promedio de tamaño de las palabras es:  3.830411128023649
```

Punto 28

- Define a function percent(word, text) that calculates how often a given word occurs in a text, and expresses the result as a percentage.

In [8]:
```
def percent(word, text):
    conteo = text.count(word)
    num_palabras = len(text)
    percentage = (conteo/num_palabras)*100
    return percentage
percent('love', text5)
```

Out[8]:
```
0.13330371028660298
```

Capítulo 2

Punto 2

- Use the corpus module to explore austen-persuasion.txt. How many word tokens does this book have? How many word types?

In [5]:
```
from nltk.corpus import gutenberg

austen_persuasion = gutenberg.words('austen-persuasion.txt')
print('Número de tokens: ', len(austen_persuasion))
print('Tipos de Palabras: ', len(set(austen_persuasion)))
```

```
Número de tokens:  98171
Tipos de Palabras:  6132
```

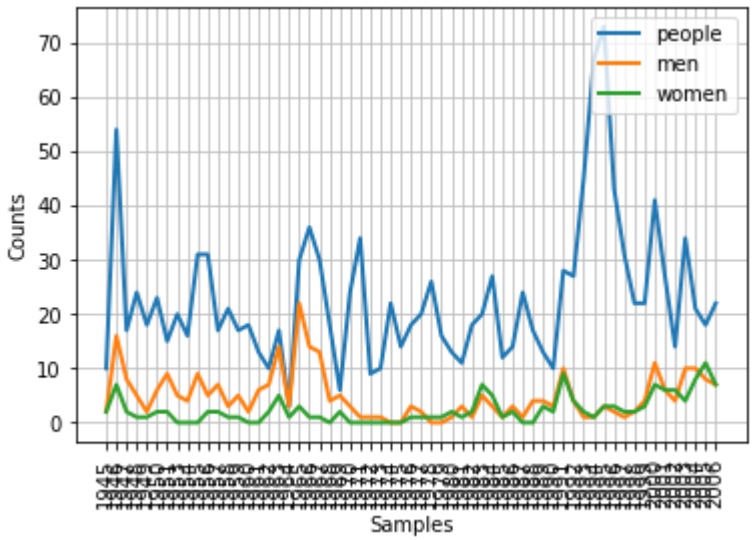
Punto 4

- Read in the texts of the State of the Union addresses, using the state_union corpus reader. Count occurrences of men, women, and people in each document. What has happened to the usage of these words over time?

In [6]:
```
from nltk.corpus import state_union

palabras = ['men','women','people']
frecuencia = nltk.ConditionalFreqDist(
    (x, fileid[:4])
    for fileid in state_union.fileids()
    for a in state_union.words(fileid)
    for x in palabras
    if a.lower().startswith(x))

frecuencia.plot()
```



Out[6]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x22716391a88>
```

Punto 11

- Investigate the table of modal distributions and look for other patterns. Try to explain them in terms of your own impressionistic understanding of the different genres. Can you find other closed classes of words that exhibit significant differences across different genres?

In [7]:
```
from nltk.corpus import brown
brown.categories()
modal_verb = ['can','could', 'may','might','will','would','must','shall','should
frecuencia = nltk.ConditionalFreqDist(
    (tema, modal_verb.lower())
    for tema in brown.categories()
    for modal_verb in brown.words(categories=tema))
genres = ['news', 'religion', 'hobbies', 'lore', 'belles_lettres', 'government',

frecuencia.tabulate(conditions=genres, samples=modal_verb)
```

	can	could	may	might	will	would	must	shall	should
news	94	87	93	38	389	246	53	5	61
religion	84	59	79	12	72	69	54	23	45
hobbies	276	59	143	22	269	83	84	5	73
lore	170	142	170	50	178	188	96	12	78
belles_lettres	249	216	221	113	246	397	171	35	105
government	119	38	179	13	244	120	102	99	113
fiction	39	168	10	44	56	291	55	3	38
mystery	45	145	15	57	25	189	31	4	29
science_fiction	16	49	4	12	17	80	8	3	3
romance	79	195	11	51	49	247	46	4	33
adventure	48	154	7	59	51	194	27	11	17
humor	17	33	8	8	13	56	9	2	7

Podemos ver que la categoría de belles lettres, al ser una categoría que incluye los trabajos literarios que no encajan en los otros géneros literarios, hace sentido que al ser una categoría tan amplia, sea la que tiene el mayor uso de 5 de los 9 verbos modales.

Ahora, vemos que 'shall' es un verbo poco usado en general, tal vez por la formalidad del mismo. De hecho, vemos que el genre de gobierno es el que más lo utiliza, debido a la formalidad propia de la categoría. Adicionalmente, este genre también es el que más utiliza 'should', lo que puede ser explicado por el uso de ese verbo para hacer o pedir sugerencias, opiniones u acciones.

Hce sentido que el verbo 'would' sea de los más usados por los genres de romance, aventura y ficcion, ya que esta palabra expresa una intención de poner condiciones.

In [8]:
```
palabras = ['life','death','love','hate','darkness','threat']
frecuencia.tabulate(conditions=genres, samples=palabras)
```

	life	death	love	hate	darkness	threat
news	20	10	3	1	0	8
religion	55	42	13	3	5	0
hobbies	36	3	8	0	0	0
lore	76	29	19	2	2	4
belles_lettres	200	61	72	4	4	11
government	14	2	1	0	0	3
fiction	45	17	16	9	9	1
mystery	21	18	7	2	3	0
science_fiction	5	2	4	0	1	0
romance	54	12	36	9	5	1
adventure	29	20	9	8	12	1
humor	12	1	5	0	0	1

lore, es el genre que más emplea la palabra 'life', (no tomando en cuenta a belles_lettres por ser una categoría muy amplia). Esto salta a la vista porque asumiríamos que el genre religion hubiese sido el que más veces usara 'life', pero cobra un poco de sentido al saber que son los conocimientos y tradiciones de un grupo lo que abarce el genre lore, por lo que hace sentido que se hable de la vida, al ser de las cosas tradicionalmente primordiales.

Es curioso que en todos los géneros se hable más de la vida que de la muerte, así como pasa con el amor y el odio, mostrándo una predominancia de las palabras con connotaciones positivas, por lo que también vemos que 'darkness' y 'threat' sean palabras muy poco usadas.