Taller 9

Métodos Computacionales para Políticas Públicas - URosario

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

[Santiago Ortiz Ortiz]

[santiago.ortizo@urosario.edu.co (mailto:santiago.ortizo@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

In [ ]:
```
pip install nltk
```

In [1]:
```
import nltk
```

In [4]:
```
nltk.download('inaugural')
```

```
[nltk_data] Downloading package inaugural to
[nltk_data]     C:\Users\tatoo\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping corpora\inaugural.zip.
```

Out[4]: True

In [6]:
```
nltk.download('nps_chat')
```

```
[nltk_data] Downloading package nps_chat to
[nltk_data]     C:\Users\tatoo\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping corpora\nps_chat.zip.
```

Out[6]: True

In [8]:
```
nltk.download('webtext')
```

```
[nltk_data] Downloading package webtext to
[nltk_data]     C:\Users\tatoo\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping corpora\webtext.zip.
```

Out[8]: True

In [10]:
```
nltk.download('treebank')
```

```
[nltk_data] Downloading package treebank to
[nltk_data]     C:\Users\tatoo\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping corpora\treebank.zip.
```

Out[10]: True

In [11]:
```
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
```

Capítulo 1:

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 [14]:
```
print(text5)
```

```
<Text: Chat Corpus>
```

In [15]:
```
chat_corpus = list(text5)
four_letters_word = [w for w in chat_corpus if len(w) == 4]
```

In [16]:
```
four_letters_word
```

```
 'chat',
 'with',
 'PART',
 'JOIN',
 'JOIN',
 'fast',
 'U116',
 'bowl',
 'bong',
 'JOIN',
 'well',
 'glad',
 'hard',
 'from',
 'here',
 'back',
 'PART',
 'PART',
 'JOIN',
 'U121',
```

In [20]:
```
frecuencia = FreqDist(four_letters_word)
frecuencia
```

In [24]:
```
frecuencia.most_common(40) # Los cuarenta más comunes
```

Out[24]:
```
[('JOIN', 1021),
 ('PART', 1016),
 ('that', 274),
 ('what', 183),
 ('here', 181),
 ('....', 170),
 ('have', 164),
 ('like', 156),
 ('with', 152),
 ('chat', 142),
 ('your', 137),
 ('good', 130),
 ('just', 125),
 ('lmao', 107),
 ('know', 103),
 ('room', 98),
 ('from', 92),
 ('this', 86),
 ('well', 81),
 ('back', 78),
 ('hiya', 78),
 ('they', 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),
 ('MODE', 41),
 ('will', 40)]
```

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?

- Lo que el comando hace es arrojar el total de caracteres que hay en el text1 (Moby Dick by Herman Melville 1851)

In [25]:
```
sum(len(w) for w in text1)
```

Out[25]:
```
999044
```

- Si se puede hacer el promedio. Si se tiene el número total de palabras que tiene el texto.

In [26]:
```
len(text1)
```

Out[26]:
```
260819
```

In [27]:
```
print("En promedio cada palabra tiene", sum(len(w) for w in text1)/len(text1), "
```

```
En promedio cada palabra tiene 3.830411128023649 caracteres
```

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 [29]:
```
# 1. ¿Cómo saber la frecuencia de las palabras?
frecuencia = FreqDist(text1)
frecuencia
```

Out[29]:
```
FreqDist({',': 18713, 'the': 13721, '.': 6862, 'of': 6536, 'and': 6024, 'a': 45
69, 'to': 4542, ';': 4072, 'in': 3916, 'that': 2982, ...})
```

In [30]:
```
# 2. Queremos una palabra específica:
frecuencia["in"]
```

Out[30]:
```
3916
```

In [33]:
```
def percent(word, text):
    freq_by_word = FreqDist(text)
    especifica = freq_by_word[word]
    total_word = len(text)
    porcentaje = (freq_by_word[word]/len(text))*100
    print(f"La palabra {word} aparece en el {porcentaje}% del {text}")
```

In [ ]:
```
percent('whale', text1)
```

In [39]:
```
percent('stock', text7)
```

```
La palabra stock aparece en el 0.1350868131431523% del <Text: Wall Street Journ
al>
```

Capítulo 2

2.

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

In [40]:
```
# De esta forma traemos el .txt
austen = nltk.corpus.gutenberg.words('austen-persuasion.txt')
```

In [43]:
```
# Si queremos ver el número de caracteres que tiene txt
print("El total de tokes que tiene el texto es de", len(austen))
print("El total de palabras que tiene el texto es de",len(set(austen)))
```

```
El total de tokes que tiene el texto es de 98171
El total de palabras que tiene el texto es de 6132
```

4.

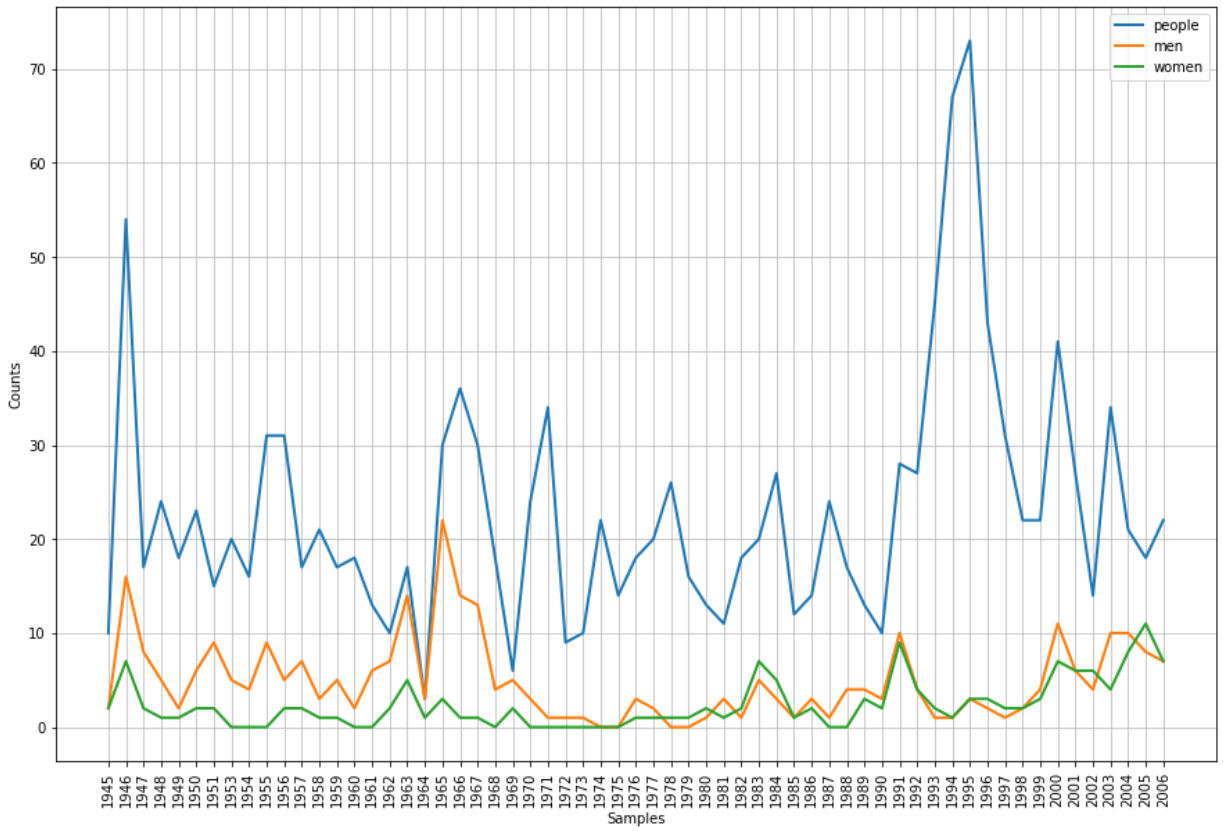
Read in the texts of the State of the Union addresses, using the state_union corpus reader. Count occurrences of men, women, and people in each document. What has happened to the usage of these words over time?

In [51]:
```
nltk.download('state_union')
import matplotlib.pyplot as plt
```

```
[nltk_data] Downloading package state_union to
[nltk_data]     C:\Users\tatoo\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping corpora\state_union.zip.
```

In [52]:
```
# Palabras que queremos buscar
lista=['men', 'women', 'people']
## Realizamos la busqueda por año
DistF= nltk.ConditionalFreqDist(
    (i, fileid[:4])
    # Recorremos el texto y buscamos que si cada palabra coincide con las que est
    for fileid in state_union.fileids()
    for j in state_union.words(fileid)
    for i in lista
    # En caso tal de que si coincida
    if j.lower().startswith(i))
```

In [53]:
```
plt.figure(figsize=(15,10))
DistF.plot()
```



Out[53]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x27711700fc8>
```

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 [54]:
```
from nltk.corpus import brown
## Tomado del libro los modals serian
modals = ['can', 'could', 'may', 'might', 'must', 'will']
## Tomado del libro
genres = ['news', 'religion', 'hobbies', 'science_fiction', 'romance', 'humor']
tabla = nltk.ConditionalFreqDist(
            (genre, word.lower())
            for genre in brown.categories()
            for word in brown.words(categories=genre))
tabla.tabulate(conditions=genres, samples=modals)
```

	can	could	may	might	must	will
news	94	87	93	38	53	389
religion	84	59	79	12	54	72
hobbies	276	59	143	22	84	269
science_fiction	16	49	4	12	8	17
romance	79	195	11	51	46	49
humor	17	33	8	8	9	13

```
- News: Es el que usa mas la palabra will, es comun que una noticia estos
indican que sucederá despues del acontecimiento.
- Religion: Can, may y will. Tal vez hablan de como Dios puede hacer las cosas
y como las hará.
- Hobbies: Can, may y will. indica que las personas pueden o realizaran algunas
actividades
- Romance: Es el que usa mas la palabra "could" indicando la posible existencia
de diferentes cosas que no son seguras que sucedan en el futuro.
- Humor: No utiliza tantos modales. Tal vez es un género más conciso.
```

In [55]:
```
# Mis palabras
julian_pal = ['but','then','also','however','thus','so']
## Tomado del libro
genres = ['news', 'religion', 'hobbies', 'science_fiction', 'romance', 'humor']
tabla = nltk.ConditionalFreqDist(
            (genre, word.lower())
            for genre in brown.categories()
            for word in brown.words(categories=genre))
tabla.tabulate(conditions=genres, samples=julian_pal)
```

	but	then	also	however	thus	so
news	283	73	129	49	10	81
religion	175	59	57	19	24	99
hobbies	221	88	110	41	21	121
science_fiction	89	18	2	7	2	26
romance	387	143	16	2	0	192
humor	101	37	13	10	5	56

- News y romance usan mucho "but" seguramente porque siempre se muestran diferentes puntos de vista.
- So y Then son usadas en hobbies y romance indicando que dado que paso algo, esto conllevo a que les guste algo o que dos personas se separaran.
- En humor tambien se exponen diferentes puntos de vista pero con menor frecuencia