GoT

November 18, 2015

1 Introduction to Python

1.1 Biostats Computing Workshop

1.1.1 Motivational Example

```
In [1]:  ## SO MANY LIBRARIES
         from bs4 import BeautifulSoup as bs ## Importing an Object!
         from urllib2 import urlopen ## Importing some functions!
         from urllib import urlretrieve
         import unicodedata
         import os
         import sys
         from __future__ import division ## Oddity with division
```

```
In [2]: URL = "http://www.readbooksvampire.com/George_R.R._Martin/A_Game_of_Thrones.html" ## CAPS = GLO
```

```
In [12]: soup = bs(urlopen(URL)) ### Creating a 'bs' object here
```

```
In [12]: i = 0
         # chapters = []
         for link in soup.find_all('a'): ## This is what a 'for loop' looks like in python
             if "George_R.R._Martin/A_Game_of_Thrones/" in link.get('href'): # some if statements!
                 print(link.get('href'))
                 # chapters.append(link.get('href'))
                 i+=1
             if i > 6:
                 break
```

```
/George_R.R._Martin/A_Game_of_Thrones/01.html
/George_R.R._Martin/A_Game_of_Thrones/02.html
/George_R.R._Martin/A_Game_of_Thrones/03.html
/George_R.R._Martin/A_Game_of_Thrones/04.html
/George_R.R._Martin/A_Game_of_Thrones/05.html
/George_R.R._Martin/A_Game_of_Thrones/06.html
/George_R.R._Martin/A_Game_of_Thrones/07.html
```

1.2 OOOOOO LIST COMPREHENSION

```
In [13]: chapters = [ link.get('href') for link in soup.find_all('a')
                      if "George_R.R._Martin/A_Game_of_Thrones/" in link.get('href')]
         chapters[:5]
```

```
Out[13]: ['/George_R.R._Martin/A_Game_of_Thrones/01.html',
          '/George_R.R._Martin/A_Game_of_Thrones/02.html',
```

```
          '/George_R.R._Martin/A_Game_of_Thrones/03.html',
          '/George_R.R._Martin/A_Game_of_Thrones/04.html',
          '/George_R.R._Martin/A_Game_of_Thrones/05.html']
```

```
In [14]: ## lets check this out for chapter 1
         chapter1_url = URL[:-5] + chapters[0][len(chapters[0])-8:] ## look at this fancy string slicin
```

```
URL[:-5] ## gets everything but the last five characters in the string
http://www.readbooksvampire.com/George_R.R._Martin/A_Game_of_Thrones     .html

chapters[0][len(chapters[0])-8:] ## lets break this down one piece at a time

chapters[0] ## this is the object holding the text string for the first chapter
>>/George_R.R._Martin/A_Game_of_Thrones/01.html'

## I'm now accessing a string object, but now I only want the last 8 characters!
len(chapters[0]) ## gives me the entire length of the string
len(chapters[0])-8 ## Length of string minus 8
chapters[0][len(chapters[0])-8:]
#Gives me the entire string of the first chapter url, starting 8 characters back
>>/01.html

URL[:-5] + chapters[0][len(chapters[0])-8:] ## I stick the two strings together with the '+' sign!
```

```
In [15]: chapter1_url
```

```
Out[15]: 'http://www.readbooksvampire.com/George_R.R._Martin/A_Game_of_Thrones/01.html'
```

```
In [16]: soup_1 = bs(urlopen(chapter1_url)) ## Creating a Soup Object!
```

```
In [80]: # soup_1.get_text()
```

1.3 lets get the whole book

What would this look like in a for loop?

```
In [15]: chapter_num = 1
         for chapter in chapters: ## notice the arbitrary indexing -- this is a property called __iter_
             chapter_num_url = URL[:-5] + chapters[0][len(chapter)-8:]
             bs(urlopen(chapter_num_url)).get_text()
```

1.3.1 Lets be cool and use a list comprehension

```
In [29]: def get_GoT(chapter):
             """
             Returns all the GoT text within the parameter chapter
             P.S. Docstrings are great for making sure other people
             who read your code know what you're doing with your
             custom functions and objects
             """
             chapter_num_url = URL[:-5] + chapter[len(chapter)-8:]
             return bs(urlopen(chapter_num_url)).get_text()
```

```
In [30]: GoT = [get_GoT(chapter) for chapter in chapters] ## takes a couple minutes
```

1.3.2 Not always wise to do this, servers and their admins will be upset if you make a lot of requests in a short amount of time

```
In [31]: len(GoT),len(chapters)
```

```
Out[31]: (72, 72)
```

```
In [32]: GoT[0][:100]
```

```
Out[32]: u'\n\nA Game of Thrones(Song of Ice and Fire Book 1) by George R.R. Martin | Chapter One | Reac
```

```
In [33]: GoT[71][:100] ## zero indexed arrays/lists
```

```
Out[33]: u'\n\nA Game of Thrones(Song of Ice and Fire Book 1) by George R.R. Martin | Chapter Seventy-tw
```

```
In [34]: type(GoT[0]),type(unicodedata.normalize('NFKD', GoT[0]).encode('ascii','ignore'))
```

```
Out[34]: (unicode, str)
```

```
In [35]: GoT = [unicodedata.normalize('NFKD',chapter).encode('ascii','ignore') for chapter in GoT] ## A
```

```
In [102]: !mkdir GoT
```

```
In [16]: os.chdir("GoT")
```

```
In [122]: for chapter_num in range(1,73):
              with open("Chapter "+str(chapter_num) + ".txt","w") as chap: ## file access!
                  chap.write(GoT[chapter_num-1])
```

```
In [36]: book_string = ""
         for chapter in GoT:
             book_string += chapter
```

```
In [37]: len(book_string)
```

```
Out[37]: 1616629
```

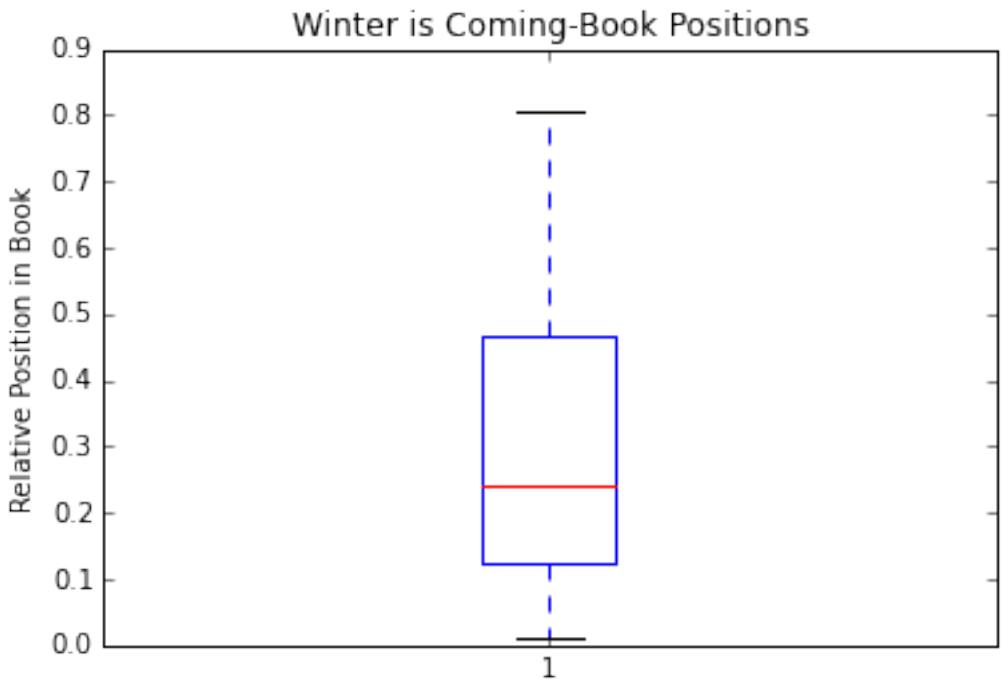
```
In [38]: with open("GoT_bookone.txt","w") as book:
             book.write(book_string)
```

```
In [10]: !ls
```

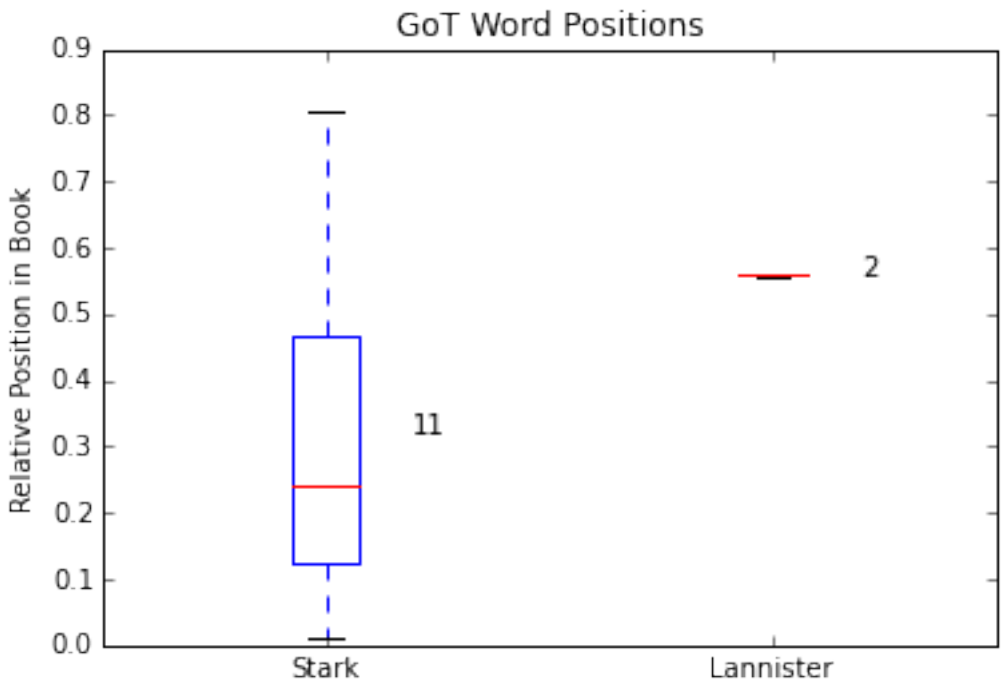
```
Chapter 1.txt   Chapter 23.txt  Chapter 37.txt  Chapter 50.txt  Chapter 64.txt
Chapter 10.txt  Chapter 24.txt  Chapter 38.txt  Chapter 51.txt  Chapter 65.txt
Chapter 11.txt  Chapter 25.txt  Chapter 39.txt  Chapter 52.txt  Chapter 66.txt
Chapter 12.txt  Chapter 26.txt  Chapter 4.txt   Chapter 53.txt  Chapter 67.txt
Chapter 13.txt  Chapter 27.txt  Chapter 40.txt  Chapter 54.txt  Chapter 68.txt
Chapter 14.txt  Chapter 28.txt  Chapter 41.txt  Chapter 55.txt  Chapter 69.txt
Chapter 15.txt  Chapter 29.txt  Chapter 42.txt  Chapter 56.txt  Chapter 7.txt
Chapter 16.txt  Chapter 3.txt   Chapter 43.txt  Chapter 57.txt  Chapter 70.txt
Chapter 17.txt  Chapter 30.txt  Chapter 44.txt  Chapter 58.txt  Chapter 71.txt
Chapter 18.txt  Chapter 31.txt  Chapter 45.txt  Chapter 59.txt  Chapter 72.txt
Chapter 19.txt  Chapter 32.txt  Chapter 46.txt  Chapter 6.txt   Chapter 8.txt
Chapter 2.txt   Chapter 33.txt  Chapter 47.txt  Chapter 60.txt  Chapter 9.txt
Chapter 20.txt  Chapter 34.txt  Chapter 48.txt  Chapter 61.txt  GoT_bookone.txt
Chapter 21.txt  Chapter 35.txt  Chapter 49.txt  Chapter 62.txt
Chapter 22.txt  Chapter 36.txt  Chapter 5.txt   Chapter 63.txt
```

```
In [22]: import re ## regular expressions library
```

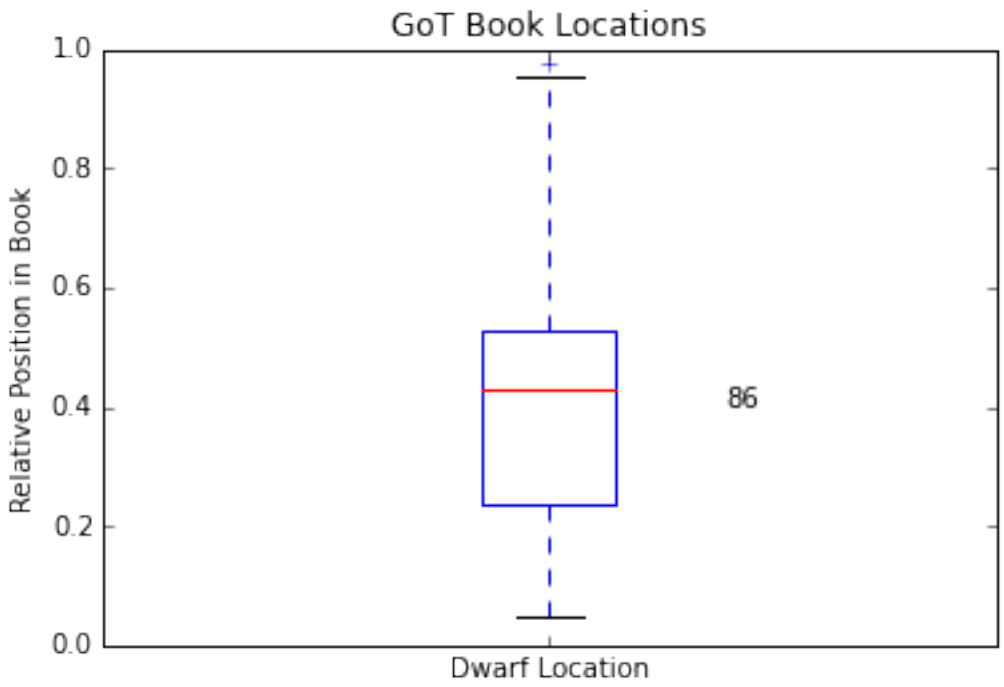
```
In [40]: wic = re.findall("[Ww][Ii][Nn][Tt][Ee][Rr] [Ii][Ss] [Cc][Oo][Mm][Ii][Nn][Gg]",book_string)
         wic,len(wic)

Out[40]: (['winter is coming',
           'Winter is coming',
           'Winter is coming',
           'winter is coming',
           'Winter is coming',
           'Winter is coming',
           'Winter is coming',
           'Winter is coming',
           'Winter is coming',
           'winter is coming',
           'Winter is coming'],
          11)

In [41]: Starks= "[Ww][Ii][Nn][Tt][Ee][Rr] [Ii][Ss] [Cc][Oo][Mm][Ii][Nn][Gg]"
         Lannisters = "[Aa] [Ll]annister [Aa]lways [Pp]ays [Hh]is debts"
         Lannister_actual = "[Hh]ear [Mm]e [Rr]oar!"

In [42]: search_string = re.compile(Starks)
         find = search_string.search(book_string)

In [43]: find.span()

Out[43]: (21266, 21282)

In [44]: for position in search_string.finditer(book_string):
             print position.start()

21266
21561
103321
300227
313673
392475
420761
730895
780740
1299233
1299288

In [45]: Stark_Positions = [position.start()/len(book_string) for position in search_string.finditer(boo

In [48]: from matplotlib import pyplot as plt
         import numpy as np
         %matplotlib inline

In [49]: plt.boxplot(Stark_Positions)
         plt.title("Winter is Coming-Book Positions")
         plt.ylabel("Relative Position in Book")
         plt.show()
```




```
In [88]: search_string = re.compile(Lannisters)
         Lannister_positions = [position.start()/len(book_string) for position in search_string.finditer
         plt.boxplot([Stark_Positions,Lannister_positions])
         plt.title("GoT Word Positions")
         plt.xticks([1,2],["Stark","Lannister"])
         plt.text(1.2,np.mean(Stark_Positions),str(len(Stark_Positions)))
         plt.text(2.2,np.mean(Lannister_positions),str(len(Lannister_positions)))
         plt.ylabel("Relative Position in Book")
         plt.show()
```




```
In [98]: search_string = re.compile("[Dd]warf")
         dwarf_pos = [position.start()/len(book_string) for position in search_string.finditer(book_str
         plt.boxplot([dwarf_pos])
         plt.title("GoT Book Locations")
         plt.xticks([1],["Dwarf Location"])
         plt.text(1.2,np.mean(dwarf_pos),str(len(dwarf_pos)))
         plt.ylabel("Relative Position in Book")
         plt.show()
```



1.4 bonus nltk discourse if time

```
In [8]: import nltk ## another library...seriously?

In [10]: tokens = nltk.word_tokenize(book_string)

In [60]: from collections import Counter,OrderedDict ## This is getting ridiculous

In [30]: word_freq = Counter(tokens) ## example of dictionary
         word_freq

Out[30]: Counter({'raining': 4,
                  'mustachio': 3,
                  'Greyjoy': 68,
                  'yellow': 30,
                  'four': 65,
                  'woods': 40,
                  'clotted': 2,
                  'spiders': 8,
                  'ornate': 12,
                  'woody': 2,
                  'Until': 12,
                  'marching': 15,
                  'dragonbone': 9,
                  'unanswered': 1,
                  'sunlit': 1,
                  'Visitors': 1,
                  'Vengeance': 1,
                  'caned': 1,
```

```
'Western': 6,
'crossbar': 3,
'lord': 410,
'meadows': 1,
'sinking': 2,
'kennels': 5,
'wracked': 2,
'dreamers': 1,
'oceans': 1,
'bile': 1,
'foul': 6,
'Old': 99,
'stabbed': 7,
'bringing': 17,
'banqueting': 1,
'disturb': 3,
'prize': 2,
'Less': 4,
'wooden': 66,
'Jeren': 5,
'Emmon': 1,
'solid': 11,
'persisted': 1,
'Does': 18,
'crotch': 1,
'Towerso-called': 1,
'provided': 8,
'festered': 3,
'commented': 5,
'guardsmen': 39,
'Cradling': 1,
'Baelor': 14,
'nigh': 2,
'tired': 36,
'spell-forged': 2,
'hanging': 6,
'bacon': 8,
'appropriated': 1,
'elegant': 3,
'second': 69,
'crisply': 2,
'valiant': 7,
'sailed': 4,
'scraped': 4,
'loathing': 7,
'snuggled': 1,
'haughtily': 1,
'milled': 1,
'captain': 29,
'dangled': 5,
'Redwynes': 3,
'swaggering': 1,
'thunder': 10,
'cooking': 1,
```

```
'fingers': 184,
'Wrong': 1,
'hull': 3,
'pawed': 1,
'pressed': 33,
'fisherfolk': 3,
'hero': 6,
'avert': 1,
'chins': 2,
'herb': 1,
'Mohor': 5,
'leaning': 13,
'Straighten': 1,
'here': 396,
'herd': 6,
'reported': 2,
'hers': 34,
'Targaryens': 10,
'splinted': 1,
'chink': 1,
'strewn': 1,
'pretensions': 1,
'elaborate': 3,
'climbed': 43,
'Feathers': 1,
'encouragement': 2,
'reports': 5,
'Elmar': 1,
'powdering': 1,
'Farther': 3,
'Choosing': 1,
'waycastle': 3,
'NOW': 1,
'delicacy': 6,
'Isles': 9,
'rebel': 1,
'golden': 84,
'Compared': 1,
'vaults': 3,
'explained': 16,
'Three': 33,
'replace': 2,
'brought': 119,
'stern': 9,
'scarier': 1,
'presents': 2,
'spoke': 61,
'Danwell': 5,
'glimmering': 2,
'blacksmiths': 1,
'backing': 3,
'music': 17,
'strike': 17,
'heralded': 2,
```

```
'care': 61,
'until': 188,
'sideface': 1,
'whorls': 1,
'holy': 2,
'relax': 1,
'brings': 6,
'gashes': 2,
'whirling': 6,
'Rule': 1,
'hurt': 71,
'glass': 20,
'tying': 1,
'midst': 4,
'hold': 73,
'chaos': 6,
'circumstances': 2,
'Maddened': 1,
'intake': 1,
'locked': 4,
'dreadful': 5,
'Ibben': 4,
'blade': 106,
'wineskin': 4,
'Hollow': 1,
'cajoled': 1,
'warhorns': 2,
'Mern': 2,
'lordperfume': 1,
'Mere': 1,
'misfortunes': 1,
'melons': 3,
'owes': 3,
'slow-moving': 2,
'unjust': 1,
'household': 20,
'dragons': 43,
'caution': 4,
'want': 258,
'absolute': 1,
'groaned': 6,
'complaining': 1,
'travel': 4,
'drying': 1,
'damage': 1,
'Killing': 1,
'how': 213,
'hot': 61,
'uninvited': 2,
'dared': 27,
'blue-black': 1,
'preferable': 1,
'cadence': 1,
'retreating': 2,
```

```
'A': 726,
'Tobho': 7,
'beauty': 14,
'envied': 1,
'twenty-third': 2,
'shores': 3,
'wrong': 64,
'twined': 1,
'short-tempered': 1,
'Fogo': 3,
'Addam': 14,
'Centuries': 1,
'winy': 1,
'calves': 1,
'welts': 1,
'scourge': 1,
'Dragonlord': 2,
'revolt': 1,
'sickening': 5,
'wink': 1,
'snowballs': 2,
'Another': 29,
'keeps': 10,
'Tumblestone': 6,
'wing': 9,
'wind': 83,
'wine': 133,
'sprawling': 6,
'snugly': 1,
'Karstark': 32,
'welcomed': 10,
'dreamed': 15,
'govern': 2,
'feign': 1,
'profaned': 1,
'butcher': 29,
'Forty-one': 2,
'rewarded': 3,
'wrought': 6,
'His': 465,
'Hit': 2,
'fit': 25,
'screaming': 32,
'fix': 1,
'survivors': 7,
'fig': 5,
'nobler': 2,
'hidden': 27,
'easier': 14,
'undercut': 1,
'grievous': 9,
'bristle': 1,
'thrones': 11,
'wooded': 1,
```

```
’slicing’: 2,
’wasteland’: 3,
’drinking’: 18,
’half-dozen’: 3,
’sixteen’: 12,
’slats’: 2,
’silver’: 131,
’saddened’: 2,
’Previous232425262728293031323334353637 38Next’: 1,
’Shield’: 3,
’clumsy’: 11,
’blushes’: 1,
’towered’: 4,
’black-and-scarlet’: 2,
’crops’: 2,
’Hobber’: 2,
’arrow’: 27,
’loyalties’: 3,
"m’lady": 7,
’preceded’: 1,
’whim’: 2,
’Perchance’: 1,
’snakes’: 5,
’woes’: 1,
’garment’: 3,
’spider’: 4,
’bowls’: 2,
’solution’: 2,
’snaked’: 1,
’turnip’: 4,
’fortnight’: 24,
’waddled’: 5,
’whit’: 1,
’blinded’: 1,
’whip’: 22,
’borne’: 2,
’drove’: 21,
’centaurs’: 1,
’smirk’: 2,
’whoosh’: 1,
’bearclaw’: 1,
’branch’: 7,
’Curiosity’: 1,
’unarmored’: 2,
’pardons’: 15,
’Awkwardly’: 1,
’outburst’: 1,
’impetuous’: 1,
’Word’: 1,
’assured’: 9,
’underfoot’: 6,
’threatened’: 6,
’fugitive’: 2,
’pumpkins’: 2,
```

```
'grapes': 2,
'tinkled': 1,
'crowned': 18,
'estimate': 1,
'enormous': 1,
'BearIsland': 2,
'exposing': 1,
'piecemeal': 1,
'shelves': 2,
'moment': 163,
'scruff': 1,
'waned': 1,
'shipped': 1,
'disturbed': 1,
'tempting': 2,
'wobbling': 1,
'potions': 3,
'breed': 3,
'disfigured': 1,
'Please': 38,
'channels': 1,
'wash': 13,
'instruct': 5,
'red-brown': 3,
'elms': 1,
'her.': 1,
'bitten': 3,
'Man': 5,
'service': 24,
'whorehouse': 4,
'tsking': 1,
'engagement': 1,
'returns': 5,
'needed': 38,
'molder': 1,
'master': 29,
'legs': 108,
'bitter': 17,
'ranging': 3,
'rummaged': 2,
'listen': 52,
'collapse': 3,
'cushions': 12,
'frowned': 41,
'wisdom': 12,
'predictable': 1,
'Previous575859606162636465666768697071 72Next': 1,
'crawl': 4,
'lunged': 5,
'positively': 1,
'Guardsmen': 1,
'trek': 1,
'peril': 1,
'showed': 21,
```

```
'coward': 10,
'Alongside': 1,
'tree': 46,
'likely': 21,
'idly': 3,
'nations': 1,
'shower': 3,
'exclaimed': 7,
'endure': 5,
'feeling': 19,
'groaning': 2,
'fevered': 1,
'exploring': 1,
'shyly': 5,
'spent': 17,
'shrubs': 1,
'alive': 46,
'dozen': 54,
'Ass': 1,
'Then': 105,
'forgive': 20,
'wed.': 2,
'regretting': 1,
'Staggering': 1,
'soaked': 11,
'nagged': 1,
'eagerly': 15,
'metallic': 2,
'absorbed': 2,
'amusing': 5,
'They': 442,
'Ask': 5,
'castellan': 2,
'Bank': 1,
'disdainfully': 1,
'horselords': 7,
'Wise': 2,
'committing': 1,
'lithely': 1,
'shall': 141,
'thrilled': 2,
'object': 2,
'looming': 15,
"'m": 167,
'affirmation': 1,
'mouth': 135,
'letter': 63,
'entry': 4,
'relieving': 1,
'WheelTower': 2,
'laughing': 30,
'dummy': 1,
'singer': 35,
'Staggered': 1,
```

```
'grove': 6,
'camp': 32,
'singed': 1,
'braying': 1,
'ugliest': 3,
'greenwoods': 1,
'mating': 2,
'scream': 23,
'came': 316,
'saying': 53,
'reclined': 2,
'meetings': 2,
'droppings': 1,
'padded': 18,
'Lhazareen': 4,
'Raymun': 14,
'Daryn': 4,
'tempted': 1,
'cheaply': 1,
'abreast': 5,
'hounded': 1,
'lessons': 6,
'capes': 1,
'touches': 1,
'busy': 5,
'clicked': 6,
'lazing': 1,
'shuffling': 1,
'buxom': 1,
'Banefort': 1,
'bush': 3,
'touched': 61,
'rich': 29,
'Mollen': 17,
'greatsword': 16,
'arakh': 25,
'pocked': 4,
'plate': 41,
'Presters': 1,
'cartwheels': 1,
'Marillion': 26,
'denounce': 2,
'terrors': 1,
'foremost': 1,
'pocket': 6,
'cushion': 4,
'altogether': 4,
'rugged': 1,
'relish': 3,
'sers': 2,
'lurches': 1,
'half-submerged': 1,
'spilling': 6,
'nicely': 8,
```

```
'blanched': 1,
'avenged': 1,
'lurched': 14,
'flanked': 4,
'release': 3,
'green-and-bronze': 1,
'boarded': 1,
'Clearly': 2,
'respond': 3,
'blew': 16,
'fair': 31,
'consummation': 1,
'.if': 1,
'deemed': 3,
'unexpectedly': 1,
'bled': 2,
'flaxen': 1,
'result': 2,
'silver-and-purple': 1,
'fail': 4,
'resigned': 1,
'skulls': 9,
'Dogs': 3,
'gift-givers': 1,
'Dwarf': 2,
'lots': 2,
'fanged': 1,
'rings': 16,
'solicitude': 1,
'Porther': 6,
'score': 3,
'scorn': 2,
'preserve': 2,
'claws': 8,
'splinter': 1,
'khaleesi': 19,
'nature': 5,
'rolled': 25,
'smelled': 24,
'lapping': 3,
'Bronn': 119,
'twinkling': 1,
'horseflesh': 8,
'medallion': 6,
'defiance': 2,
'debt': 5,
'pity': 20,
'accident': 2,
'trickster': 1,
'Logs': 2,
'weirwood': 17,
'disdain': 5,
'country': 5,
'pits': 3,
```

```
'Blessed': 4,
'askew': 1,
'Direwolves': 2,
'planned': 7,
'hated': 19,
'argue': 4,
'asked': 272,
'fearlessly': 1,
'mortified': 3,
'Howland': 4,
'fissures': 1,
'unprotected': 2,
'gleaming': 6,
'pregnancy': 1,
'Sorry': 3,
'grazing': 1,
'armored': 24,
'shouting': 37,
'holdings': 2,
'shriek': 5,
'remained': 32,
'.': 25270,
'breathlessly': 1,
'much': 192,
'fry': 2,
'Flowers': 22,
'tallest': 4,
'Wendish': 4,
'life': 163,
'crying': 18,
'spit': 19,
'Mothers': 3,
'onyx': 6,
'athwart': 1,
'dragonspawn': 4,
'joust': 7,
'lift': 18,
'direwolf': 96,
'child': 150,
'worked': 18,
'spin': 1,
'cunt': 1,
'wildcat': 1,
'Chips': 1,
'taper': 1,
'rests': 1,
'Previous51525354555657585960616263646566Next': 1,
'hates': 4,
'flooding': 1,
'skirts': 7,
'remembering': 19,
'played': 11,
'Swyft': 4,
'hunkered': 1,
```

```
'piercing': 1,
'taunts': 2,
'eighteen': 10,
'Sword': 6,
'Sworn': 10,
'trusted': 9,
'golds': 1,
'credit': 3,
'things': 98,
'Oh': 88,
'oozed': 2,
'rebellion': 3,
'split': 13,
'babies': 3,
'big': 68,
'supped': 1,
'boiled': 16,
'marches': 2,
'Maybe': 20,
'pitchforks': 1,
'firepit': 6,
'refit': 1,
'dexterous': 1,
'marched': 11,
'astonishingly': 1,
'Apples': 1,
'supper': 14,
'metalwork': 1,
'haggard': 3,
'tune': 4,
'Martyn': 3,
'askance': 1,
'Cruel': 3,
'echoed': 21,
'climber': 1,
'stillness': 1,
'coldness': 4,
'snowmelt': 1,
'echoes': 5,
'veils': 1,
'eyebrows': 1,
'spurred': 11,
'solace': 1,
'Thank': 21,
'sleeps': 1,
'masons': 2,
'sleepy': 6,
'forsaken': 1,
'half-starved': 1,
'uncombed': 1,
'slayers': 1,
'rushing': 11,
'succeeding': 1,
'previous': 3,
```

```
'wandered': 6,
'haj': 1,
'enters': 1,
'ham': 5,
'beef-and-bacon': 1,
'Offer': 1,
'ease': 6,
'had': 2811,
'Wardens': 3,
'cast-outs': 1,
'hay': 3,
'Surely': 12,
'innocent': 13,
'east': 42,
'hat': 3,
'Apart': 1,
'abashed': 5,
'casually': 6,
'desertions': 1,
'possible': 6,
'lengthen': 1,
'possibly': 4,
'bathwaters': 1,
'birth': 10,
'clustered': 2,
'shadow': 68,
'summons': 5,
'bushy': 1,
'desire': 3,
'Lords': 17,
'strengths': 3,
'obvious': 5,
'undertunic': 3,
'remind': 12,
'steps': 76,
'Previous151617181920212223242526272829 30Next': 1,
'grotesquely': 2,
'guests': 6,
'battlefield': 4,
'right': 170,
'old': 300,
'creek': 1,
'crowd': 21,
'night-eyes': 1,
'crown': 34,
'begin': 25,
'crows': 20,
'baseborn': 5,
'Whatever': 22,
'Between': 10,
'left.': 1,
'Smell': 1,
'guttering': 1,
'creep': 1,
```

```
'enemies': 27,
'gasps': 2,
'perchance': 5,
'ruffled': 4,
'for': 1569,
'bottom': 15,
'fox': 2,
'killing': 13,
'plucked': 9,
'foe': 13,
'fog': 3,
'summoned': 14,
'serpents': 1,
'Royce': 20,
'pinched': 2,
'ever-so-subtle': 1,
'Alia': 1,
'shifting': 3,
'losing': 9,
'bowing': 9,
'Think': 5,
'stableboys': 4,
'First': 48,
'Suckling': 1,
'restlessness': 1,
'benches': 11,
'visitors': 8,
'grievously': 1,
'o': 4,
'despair': 5,
'lacked': 3,
'plainfaced': 1,
'slightly': 3,
'Targaryen': 69,
'hulking': 2,
'raised': 65,
'sob': 7,
'Missed': 1,
'scythes': 4,
'nought': 1,
'disentangled': 3,
'brutes': 1,
'son': 284,
'Few': 3,
'Forty-three': 2,
'wrap': 3,
'despised': 2,
'fabric': 4,
'waits': 1,
'altitude': 1,
'constantly': 1,
'Hand': 154,
'avail': 1,
'Blackfish': 16,
```

```
'grasping': 5,
'innkeeps': 1,
'halls': 15,
'overhang': 1,
'grooms': 3,
'perfumes': 4,
'offer': 23,
'fascination': 6,
'forming': 1,
'tigers': 2,
'hot-tempered': 2,
'canopied': 1,
'shrilly': 2,
'Eel': 1,
'perfumed': 4,
'caromed': 1,
'sprinting': 3,
'duel': 6,
'suspect': 5,
'rippling': 8,
'inside': 81,
'devices': 1,
'monkey-tail': 1,
'vanguard': 6,
'nicest': 1,
'smashed': 15,
'Damn': 22,
'panels': 1,
'yelped': 2,
'disgrace': 7,
'Sit': 7,
'wetness': 4,
'Six': 13,
"'": 7830,
'exist': 1,
'curious': 18,
'splotch': 1,
'Fifty': 5,
'sympathy': 1,
'dealer': 1,
'crutches': 1,
'aurochs': 9,
'protested': 7,
'dotted': 3,
'floor': 59,
'glacier': 1,
'Vhaghar': 3,
'crowns': 1,
'uttered': 1,
'flood': 1,
'warms': 1,
'role': 2,
'ambitious': 2,
'snick': 1,
```

```
'smell': 41,
'roll': 7,
'Cracks': 1,
"'s": 2672,
'hurled': 2,
'palms': 3,
'sortie': 2,
'hawked': 1,
'intimate': 1,
'Summon': 2,
'maimed': 3,
'comely': 1,
'intent': 9,
'smelling': 1,
'rolling': 16,
'cleaver': 2,
'entrust': 2,
'lunge': 1,
'fastened': 14,
'Inform': 1,
'happenstance': 2,
'sniffed': 15,
'openmouthed': 1,
'time': 388,
'push': 5,
'banners': 52,
'gown': 21,
'spat.': 1,
'chain': 35,
'whoever': 3,
'resources': 1,
'avalanche': 3,
'Sullen': 1,
'doted': 1,
'chair': 32,
'hole': 10,
'Goldengrove': 1,
'uneven': 4,
'rambling': 1,
'timbered': 6,
'rousing': 1,
'vex': 2,
'balled': 2,
'grappled': 1,
'Lighting': 1,
'carpenters': 3,
'blustering': 1,
'veterans': 1,
'dragonfly': 1,
'jerk': 2,
'trading': 8,
'Yield': 3,
'choice': 48,
'Heddle': 7,
```

```
'gloomy': 5,
'mourn': 2,
'fatigued': 1,
'stays': 1,
'pranced': 2,
'fretting': 3,
'exact': 1,
'sun-browned': 1,
'minute': 2,
'pillowed': 1,
'cooks': 9,
'reining': 2,
'tear': 13,
'Fifteen': 6,
'teat': 1,
'leave': 132,
'settle': 6,
'skewer': 3,
'unaware': 2,
'prevent': 1,
'accomplishment': 1,
'findings': 1,
'insignificant': 1,
'plunged': 10,
'soft-spoken': 1,
'sign': 13,
'chopping': 2,
'shirts': 6,
'handscarf': 1,
'Rhaggat': 3,
'coppers': 9,
'axes': 8,
'entrusts': 1,
'deep-cut': 1,
'melt': 6,
'current': 5,
'hamstringing': 1,
'lazily': 2,
'view': 12,
'Brorm': 2,
'falling': 32,
'ground': 107,
'badge': 8,
'Me': 6,
'drafted': 1,
'Dates': 1,
'sweeter': 4,
'blunted': 2,
'funeral': 5,
'plucking': 1,
'understanding': 2,
'contemplate': 4,
'thwacks': 1,
'snack': 3,
```

```
'yards': 3,
'address': 2,
'alone': 104,
'along': 84,
'My': 288,
'hurtling': 2,
'summerwine': 5,
'One-armed': 2,
'brilliant': 1,
'studied': 24,
'wherever': 7,
'commonly': 1,
'Born': 1,
'accomplished': 1,
'sprouted': 1,
'MY': 1,
'reclaim': 1,
'respite': 2,
'Shattered': 2,
'tasks': 2,
'love': 100,
'bloods': 1,
'Lie': 1,
'cacophony': 2,
'prefer': 13,
'bloody': 42,
'marvelous': 2,
'Enough': 12,
'betraying': 2,
'crime': 9,
'sky': 70,
'crammed': 1,
'fresher': 1,
'working': 7,
'perished': 1,
'whittled': 1,
'angry': 43,
'tightly': 16,
'by': 647,
'sisterly': 1,
'filmy': 1,
'opposed': 1,
'wondering': 31,
'shouldered': 4,
'sunburst': 6,
'pavilions': 7,
'loving': 4,
'scratched': 9,
'‘‘': 8164,
'guardsman': 15,
'consoled': 1,
'afford': 4,
'Dontos': 2,
'tines': 2,
```

```
'Rivers': 3,
'scratches': 3,
'riders': 49,
'behalf': 1,
'honey': 20,
'wool': 25,
'deserved': 3,
'pretend': 9,
'privy': 1,
'inflamed': 1,
'believes': 2,
'stature': 1,
'believed': 22,
'Our': 29,
'indolence': 1,
'pumpkin': 1,
'Out': 8,
'pillageand': 1,
'Holding': 1,
'mirrors': 1,
'Anointed': 1,
'weasels': 1,
'wrenched': 24,
'hides': 4,
'allowed': 22,
'Matthar': 5,
'offense': 4,
'Aegon': 26,
'stole': 6,
'listens': 1,
'winter': 33,
'savor': 2,
'divided': 2,
'Who': 50,
'poking': 4,
'iron-grey': 1,
'thanking': 1,
'Symeon': 1,
'Oswell': 4,
'Why': 130,
'annoyed': 4,
'spot': 10,
'Tomard': 16,
'explored': 1,
'such': 75,
'suck': 5,
'filthy': 10,
'reined': 14,
'natural': 6,
'Thirty-nine': 2,
'conscious': 5,
'ordinarily': 1,
'darkened': 8,
'Catching': 1,
```

```
                'so': 766,
                'tangled': 15,
                'swollen': 14,
                'wolves': 32,
                'Harwin': 14,
                'pulled': 103,
                'Mole': 8,
                'scents': 4,
                'drunken': 14,
                'oak-and-bronze': 1,
                ...})
```

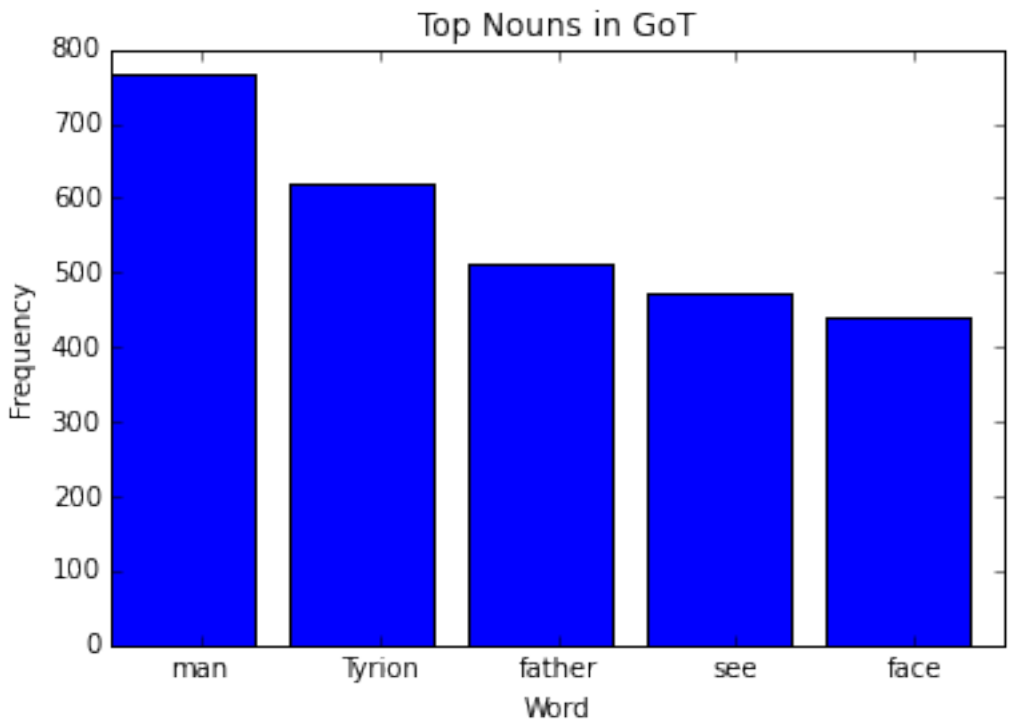
```
In [28]: word_freq.most_common()[:10]
```

```
Out[28]: [('.', 25270),
          (',', 20777),
          ('the', 14940),
          ('and', 8551),
          ('‘‘', 8164),
          ("’’", 7830),
          ('to', 6442),
          ('of', 6195),
          ('a', 5766),
          ('his', 4529)]
```

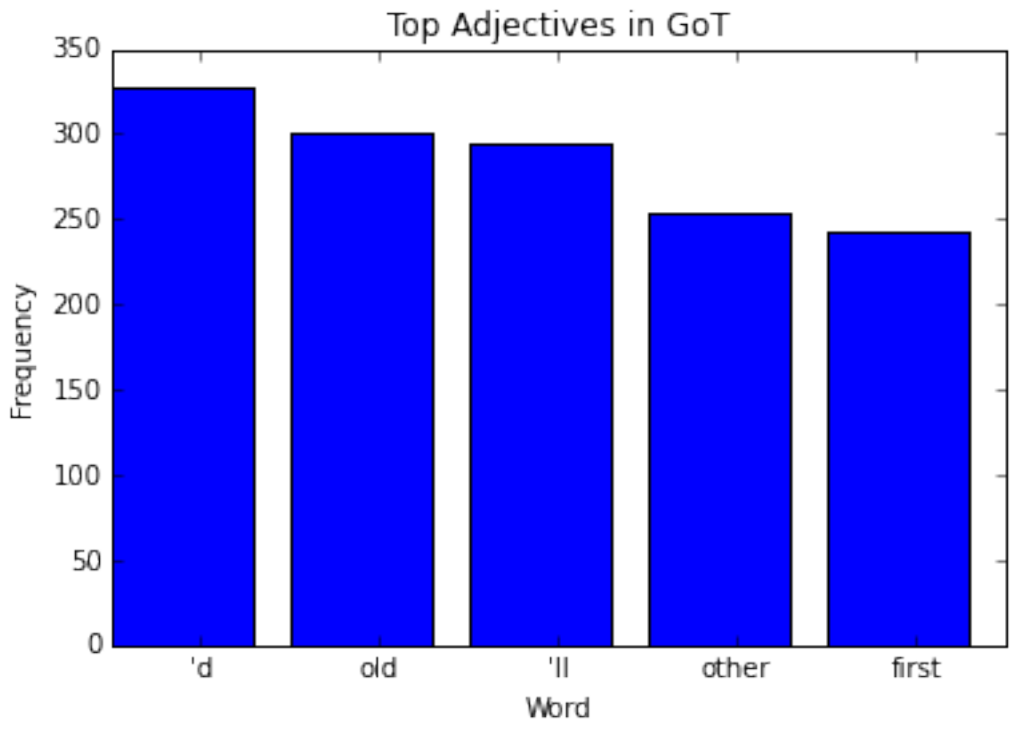
```
In [78]: top_nouns = {}
         top_adjectives = {}
         for word,freq in word_freq.most_common()[:1000]:
             if nltk.pos_tag(nltk.word_tokenize(word))[0][1] == "NN": ## getting nouns
                 top_nouns[word] = freq
             elif nltk.pos_tag(nltk.word_tokenize(word))[0][1] == "JJ": ## getting adjectives
                 top_adjectives[word] = freq
```

```
In [107]: ordered_top_nouns = OrderedDict(sorted(top_nouns.items(), key=lambda x: x[1], reverse=True))
          ordered_top_adjectives = OrderedDict(sorted(top_adjectives.items(),key=lambda x: x[1], revers
```

```
In [111]: plt.bar(np.arange(5),ordered_top_nouns.values()[:5])
          plt.xticks(np.arange(5)+.5,ordered_top_nouns.keys()[:5])
          plt.title("Top Nouns in GoT")
          plt.xlabel("Word")
          plt.ylabel("Frequency")
          plt.show()
```




```
In [114]: plt.bar(np.arange(5),ordered_top_adjectives.values()[:5])
          plt.xticks(np.arange(5)+.5,ordered_top_adjectives.keys()[:5])
          plt.title("Top Adjectives in GoT")
          plt.xlabel("Word")
          plt.ylabel("Frequency")
          plt.show()
```


Top Adjectives in GoT
Frequency
350
300
250
200
150
100
50
0
'd
old
'll
other
first
Word