Real-World Python

A Hacker's Guide to Solving Problems with Code

by Lee Vaughan

errata updated to print 2

Page	Error	Correction	Print corrected

15	`self.sailor_actual[0] = np.random.choice(self.sa1.shape[1], 1)`
	`self.sailor_actual[1] = np.random.choice(self.sa1.shape[0], 1)`
		`self.sailor_actual[0] = np.random.choice(self.sa1.shape[1])`
		`self.sailor_actual[1] = np.random.choice(self.sa1.shape[0])`
			Print 2

15	Selecting `[0]` with `random.choice()` means that rows are used, **and the final argument, 1, selects a single element**.	Selecting `[0]` with `random.choice()` means that rows are used.	Print 2

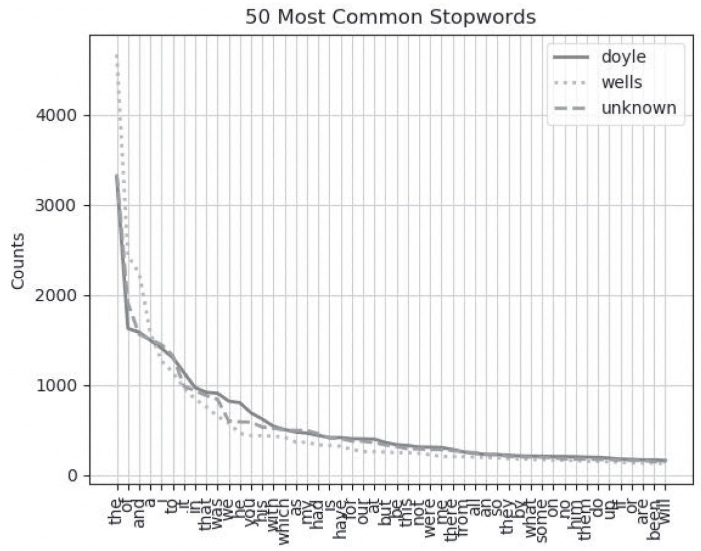
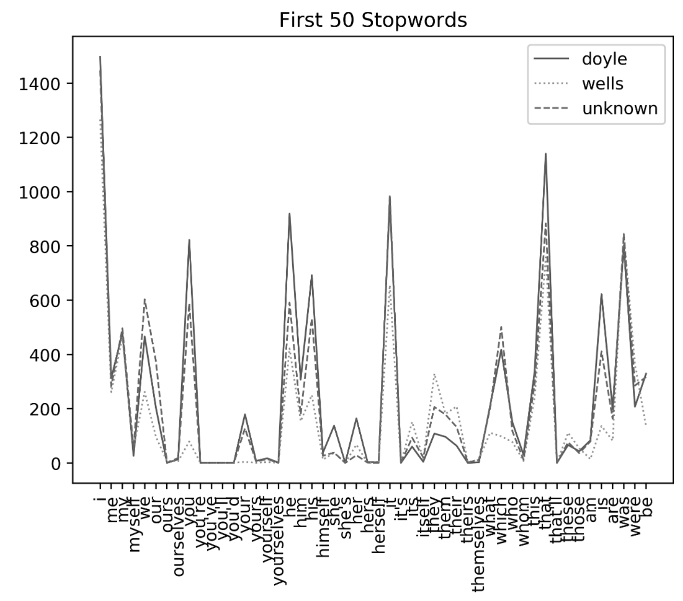
32	`import nltk`
	`from nltk.corpus import stopwords`
	`import matplotlib.pyplot as plt`
		`from collections import Counter`
		`import nltk`
		`from nltk.corpus import stopwords`
		`import matplotlib.pyplot as plt`
			Print 3

33	Start by importing **NLTK and the Stopwords Corpus. Then import `matplotlib`** .	Start by importing **the `collections` module's `Counter` class (for counting list items), `matplotlib`, NLTK, and the Stopwords Corpus**.	Print 3

39	`def stopwords_test(words_by_author, len_shortest_corpus):`
	`    """Plot stopwords freq by author, truncated to shortest corpus length."""`
	`    stopwords_by_author_freq_dist = dict()`
	`    plt.figure(2)`
	`    stop_words = set(stopwords.words('english')) # Use set for speed.`
	`    #print('Number of stopwords = {}\n'.format(len(stop_words)))`
	`    #print('Stopwords = {}\n'.format(stop_words))`
	
	`    for i, author in enumerate(words_by_author):`
	`        stopwords_by_author = [word for word in words_by_author[author]`
	`                               [:len_shortest_corpus] if word in stop_words]`
	`        stopwords_by_author_freq_dist[author] = nltk.FreqDist(stopwords_by_`
	`        author)`
	`        stopwords_by_author_freq_dist[author].plot(50,`
	`                                                   label=author,`
	`                                                   linestyle=LINES[i],`
	`                                                   title=`
	`                                                   '50 Most Common Stopwords')`
		`def stopwords_test(words_by_author, len_shortest_corpus):`
		`    """Plot stopwords freq by author, truncated to shortest corpus length."""`
		`    fdist = dict()`
		`    plt.figure(2)`
		`    stop_words = stopwords.words('english')`
		
		`    for i, author in enumerate(words_by_author):`
		`        stopwords_by_author = [word for word in words_by_author[author]`
		`                               [:len_shortest_corpus] if word in stop_words]`
		`        fdist[author] = {word: stopwords_by_author.count(word) for word in`
		`                         stop_words[:50]}  # Use first 50 of 179 stopwords.`
		`        k, v = list(fdist[author].keys()), list(fdist[author].values())`
		`        plt.plot(k, v, label=author, linestyle=LINES[i], lw=1)`
		
		`##    plt.xticks([])  # Turn off labels if plotting >50 stopwords.`
		`    plt.title('First 50 Stopwords')`
		`    plt.legend()`
			Print 3

Page	Error	Correction	Print corrected

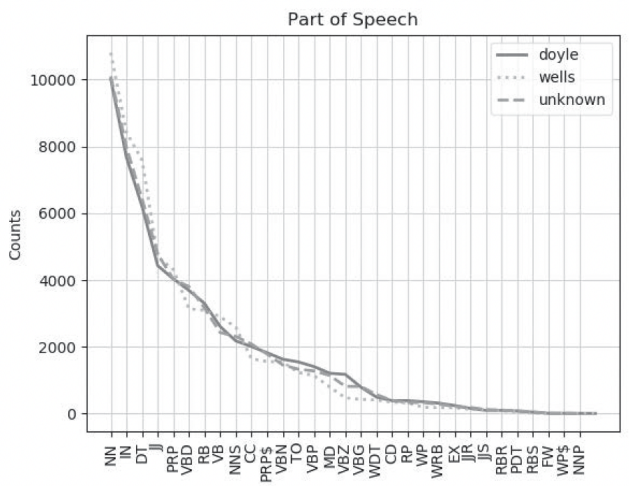
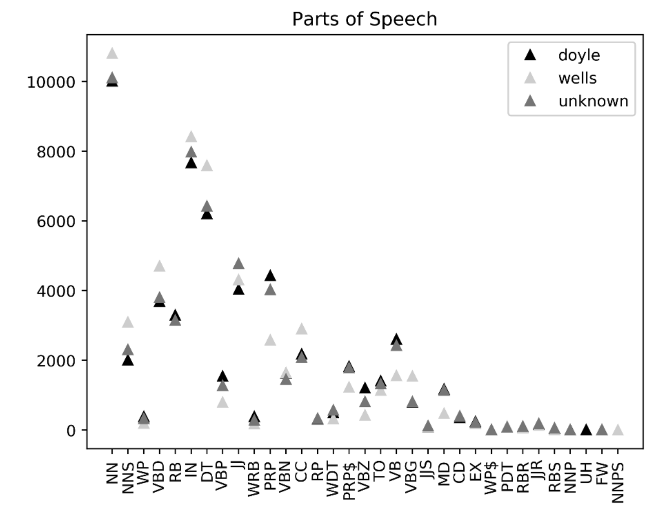
	`    plt.legend()`
	`##    plt.show() # Uncomment to see plot while coding function.`
	
	Listing 2-5: Defining the `stopwords_test()` *function*
	
	Define a function that takes the words dictionary and the length of the shortest corpus variables as arguments. Then initialize a dictionary to hold the frequency distribution of stop words for each author.
		`    plt.xticks(rotation=90)`
		`##    plt.show() # Uncomment to see plot while coding function.`
		
		Listing 2-5: Defining the `stopwords_test()` *function*
		
		Define a function that takes the words dictionary **(`fdist`)** and the length of the shortest corpus variables as arguments. Then initialize a dictionary to hold the frequency distribution of stop words for each author.	

40	Assign a local variable, `stop_words`, to the NLTK stop words corpus for English. **Sets are quicker to search than lists, so make the corpus a set for faster lookups later. The next two lines, currently commented out, print the number of stop words (179) and the stop words themselves.** Now, start looping through the authors in the `words_by_author` dictionary. Use list comprehension to pull out all the stop words in each author's corpus **and use these as the value in a new dictionary** named `stopwords_by_author`. **In the next line, you'll pass this dictionary to NLTK's `FreqDist()` method and use the output to populate the `stopwords_by_author_freq_dist` dictionary. This dictionary will contain the data needed to make the frequency distribution plots for each author.** **Repeat the code you used to plot the word lengths in Listing 2-4, but set the number of samples to `50` and give it a different title. This will plot the top 50 stop words in use (Figure 2-4).**  *Figure 2-4: Frequency plot of* ***top*** *50 stop words by author*	Assign a local variable, `stop_words`, to the NLTK stop words corpus for English. Now, start looping through the authors in the `words_by_author` dictionary. Use list comprehension to pull out all the stop words in each author's corpus **to make a new list** named `stopwords_by_author`. **In the next line, fill the `fdist` dictionary with the stop words and their count, per author, truncated to the first 50 stop words. (As there are 179 stop words, it's best to plot them in chunks.)** **The next step is to extract the `fdist` keys and values into lists and then pass these variables to the `plt.plot()` function. This produces Figure 2-4.**  *Figure 2-4: Frequency plot of* ***the first*** *50 stop words by author*	Print 3
41	Deletion	~~The taggers are typically trained on large datasets like the Penn Treebank or Brown Corpus, making them highly accurate though not perfect. You can also find training data and taggers for languages other than English. You don't need to worry about all~~	Print 3

Page	Error	Correction	Print corrected
		~~these various terms and their abbreviations. As with the previous tests, you'll just need to compare lines in a chart.~~	

41	`def parts_of_speech_test(words_by_author, len_shortest_corpus):`
	`    """Plot author use of parts-of-speech such as nouns, verbs, adverbs."""`
	`    by_author_pos_freq_dist = dict()`
	`    plt.figure(3)`
	`    for i, author in enumerate(words_by_author):`
	`        pos_by_author = [pos[1] for pos in nltk.pos_tag(words_by_author[author]`
	`                        [:len_shortest_corpus])]`
	`        by_author_pos_freq_dist[author] = nltk.FreqDist(pos_by_author)`
	`        by_author_pos_freq_dist[author].plot(35,`
	`                                             label=author,`
	`                                             linestyle=LINES[i],`
	`                                             title='Part of Speech')`
	`    plt.legend()`
	`    plt.show()`
	
	Listing 2-6: Defining the `parts_of_speech_test()` *function*
		`def parts_of_speech_test(words_by_author, len_shortest_corpus):`
		`    """Plot author use of parts-of-speech such as nouns, verbs, adverbs."""`
		`    fdist = dict()`
		`    colors = ['k', 'lightgrey', 'grey']`
		`    plt.figure(3)`
		`    for i, author in enumerate(words_by_author):`
		`        pos_by_author = [pos[1] for pos in nltk.pos_tag(words_by_author[author]`
		`                        [:len_shortest_corpus])]`
		`        fdist[author] = Counter(pos_by_author)`
		`        k, v = list(fdist[author].keys()), list(fdist[author].values())`
		`        plt.plot(k, v, linestyle='', marker='^', c=colors[i], label=author)`
		
		`    plt.title('Parts of Speech')`
		`    plt.legend()`
		`    plt.xticks(rotation=90)`
		`    plt.show()`
		
		Listing 2-6: Defining the `parts_of_speech_test()` *function*
			Print 3

42	Next, make a frequency distribution **of the POS list and with each loop plot the curve, using the top 35 samples. Note that there are only 36 POS tags and several, such as *list item markers*, rarely appear in novels**.	Next, make a frequency distribution **by calling the `Counter` class we imported previously. Then, extract the keys and values as lists and pass them to the plotting function. Use a triangle marker and turn off the connecting curves with `linestyle=''`**.	Print 3

Page	Error	Correction	Print corrected
42	 *Figure 2-5: Frequency plot of* ***top 35*** *parts of speech by author* Once again, the match between the Doyle and unknown **curves** is clearly better than the match of unknown to Wells. This suggests that Doyle is the author of the unknown corpus.	 *Figure 2-5: Frequency plot of parts of speech by author* Once again, the match between the Doyle and unknown **data** is clearly better than the match of unknown to Wells. This suggests that Doyle is the author of the unknown corpus.	Print 3
54	`speech = ''.join(p_elems)`	`speech = ''.join(p_elems) # Use a space to join the paragraph elements.`	Print 2
60	Because there are only **7** sentences in the whole speech with 10 or fewer words . . .	Because there are only **about a dozen** sentences in the whole speech with 10 or fewer words . . .	Print 2
61	Insertion	After March 2021 install version 3.8.3 (*https://pypi.org/project/gensim/3.8.3/*).	Print 2
62	`speech = ''.join(p_elems)`	`speech = ''.join(p_elems) # Use a space to join the paragraph elements.`	Print 2

62	`print("\nSummary of Make Your Bed speech:")`
	`summary = summarize(speech, word_count=225)`
	`sentences = sent_tokenize(summary)`
	`sents = set(sentences)`
	`print(' '.join(sents))`
		`print("\nSummary of Make Your Bed speech:")`
		`print(summarize(speech, word_count=225))`
			Print 2

62	Then, **call the `genism summarize()` function to summarize the speech in 225 words. This word count will produce about 15 sentences, assuming the average sentence has** 15 words.	Then, **summarize the speech and print the result in one step by calling the `genism summarize()` function within the `print()` function. Set the word_count argument to 225. In theory, this will produce a summary of 15 sentences, assuming the average sentence contains** 15 words.	Print 2

Page	Error	Correction	Print corrected
62	Deletion	~~Ideally, you could summarize the speech and print the summary in one step.~~ `print(summarize(speech`~~`, word_count=225))`~~	Print 2

63–64	Unfortunately, `gensim` sometimes duplicates sentences in summaries, and that occurs here:
	
	`Summary of Make Your Bed speech:`
	`Basic SEAL training is six months of long torturous runs in the soft sand, midnight swims in the cold water off San Diego, obstacle courses, unending calisthenics, days without sleep and always being cold, wet and miserable.`
	`Basic SEAL training is six months of long torturous runs in the soft sand, midnight swims in the cold water off San Diego, obstacle courses, unending calisthenics, days without sleep and always being cold, wet and miserable.`
	--snip--
	
	To avoid duplicating text, you first need to break out the sentences in the summary variable using the `NLTK sent_tokenize()` function. Then make a set from these sentences, which will remove duplicates. Finish by printing the results.
	Because sets are unordered, the arrangement of the sentences may change if you run the program multiple times.
	
	`Summary of Make Your Bed speech:`
	`If you can't do the little things right, you will never do the big things right.And, if by chance you have a miserable day, you will come home to a bed that is made – that you made – and a made bed gives you encouragement that tomorrow will be better.If you want to change the world, start off by making your bed.During SEAL training the students are broken down into boat crews. It's just the way life is sometimes.If you want to change the world get over being a sugar cookie and keep moving forward.Every day during training you were challenged with multiple physical events – long runs, long swims, obstacle courses, hours of calisthenics – something designed to test your mettle. Basic SEAL training is six months of long torturous runs in the soft sand, midnight swims in the cold water off San Diego, obstacle courses, unending calisthenics, days without sleep and always being cold, wet and miserable.`
	`>>>`
	`======= RESTART: C:\Python372\sequel\wordcloud\bed_summary.py =======`
	
	`Summary of Make Your Bed speech:`
	`It's just the way life is sometimes.If you want to change the world get over being a sugar cookie and keep moving forward.Every day during training you were challenged with multiple physical events – long runs, long swims, obstacle courses, hours of calisthenics – something designed to test your mettle. If you can't do the little things right, you will never do the big things right.And, if by chance you have a miserable day, you will come home to`
		After running the program, you should get the following output:
		
		`Summary of Make Your Bed speech:`
		`"What starts here changes the world." Tonight there are almost 8,000 students graduating from UT. Basic SEAL training is six months of long torturous runs in the soft sand, midnight swims in the cold water off San Diego, obstacle courses, unending calisthenics, days without sleep and always being cold, wet and miserable. So, here are the 10 lessons I learned from basic SEAL training that hopefully will be of value to you as you move forward in life. Every morning in basic SEAL training, my instructors, who at the time were all Vietnam veterans, would show up in my barracks room and the first thing they would inspect was your bed. If you want to change the world, start off by making your bed. During SEAL training the students are broken down into boat crews. Over a few weeks of difficult training my SEAL class, which started with 150 men, was down to just 35. Every day during training you were challenged with multiple physical events – long runs, long swims, obstacle courses, hours of calisthenics – something designed to test your mettle. So, if you want to change the world, start singing when you're up to your neck in mud. If you want to change the world don't ever, ever ring the bell. Moments away from starting to change the world – for the better.`
		
		If you increase the word count parameter to 450, the "make your bed" aspects of the speech are stressed even more (I've shortened the output for brevity).
		
		`Summary of Make Your Bed speech:`
		`The University's slogan is, "What starts here changes the world." I have to admit – I kinda like it. "What starts here changes the world." Tonight there are almost 8,000 students graduating from UT. And while these lessons were learned during my time in the military, I can assure you that it matters not whether you ever served a day in uniform. Basic SEAL training is six months of long torturous runs in the soft sand, midnight swims in the cold water off San Diego, obstacles courses, unending calisthenics, days without sleep and always being cold, wet and miserable. So, here are the 10 lessons I learned from basic SEAL training that hopefully will be of value to you as you move forward in life. Every morning in basic SEAL training, my instructors, who at the time were all Vietnam veterans, would show up in my barracks room and the first thing they would inspect was your bed. Making your bed will also reinforce the fact that little things in life matter. If you want to change the world, start off by making your bed. During SEAL training the students are broken down into boat crews. If you want to change the world, find someone to help you paddle. Over a few weeks of difficult training my SEAL class,`
			Print 2

Page	Error	Correction	Print corrected
	`a bed that is made – that you made – and a made bed gives you encouragement that tomorrow will be better.If you want to change the world, start off by making your bed.During SEAL training the students are broken down into boat crews. Basic SEAL training is six months of long torturous runs in the soft sand, midnight swims in the cold water off San Diego, obstacle courses, unending calisthenics, days without sleep and always being cold, wet and miserable.`  If you take the time to read the full speech, you'll probably conclude that `gensim` produced a fair summary. **Although these two results are different, both** extracted the key points of the speech, including the reference to making your bed.	`which started with 150 men, was down to just 35.` *--snip--* If you take the time to read the full speech, you'll probably conclude that `gensim` produced a fair summary. **It** extracted **many of** the key points of the speech, including the reference to making your bed.	
169	`interpolation = cv.INTER_AREA)`	`interpolation=cv.INTER_AREA)`	Print 2
179	We'll look at this case in "Experimenting with Transit Photometry" on page 182.	We'll look at this case in "Experimenting with Transit Photometry" on page 186.	Print 2

309	`"""Use NLP (nltk) to make dispersion plot."""`
	`import nltk`
	`import file_loader`
	
	`corpus = file_loader.text_to_string('hound.txt')`
	`tokens = nltk.word_tokenize(corpus)`
	`tokens = nltk.Text(tokens) # NLTK wrapper for automatic text analysis.`
		`"""Use NLP (nltk) to make dispersion plot."""`
		`import nltk`
		
		`def text_to_string(filename):`
		`    strings = []`
		`    with open(filename) as f:`
		`        strings.append(f.read())`
		`    return '\n'.join(strings)`
		
		`corpus = text_to_string('hound.txt')`
		`tokens = nltk.word_tokenize(corpus)`
		`tokens = nltk.Text(tokens) # NLTK wrapper for automatic text analysis.`
			Print 3