*Natural Language Procesing with Python Chapter 1*

Computing With Language: Text and Words

|  |  |
| --- | --- |
| Problem | Solution |
| Floating point and integer division | 1 / 3 >> 0 # floor. 1.0 / 3.0 >> 0.3333 |
| Loading the different example texts | from nltk.book import \* |
| Text object type | type(text1) >> <class 'nltk.text.Text'> |
| Concordance | text1.concordance("monstrous") #shows the context of each instance of monstrous |
| Similar Contexts | text1.similar("monstrous") #given the context of monstrous what other words often appear in this context |
| Common Contexts | text2.common\_contexts(['monstrous', 'very']) #contexts where both words appear. |
| Dispersion Plot | text4.dispersion\_plot(["citizens", "democracy", "freedom"]) #plot wheres in the text the word occurs |
| Generate Random Text (using N grams) | text3.generate() |
| Find the length of a text | len(text3) >> 44764 tokens |
| Number of unique tokens | len(set(text3)) |
| How often each word occurs | len(text3) / len(set(text3)) >> 16 #floating point float(len(text3)) / float(len(set(text3))) >> 16.05 |
| Count how often a word appears | text3.count('smote') #easy to calculates its percentage occurance text3.count('smote') / len(text3) |
| Lexical Diversity | len(text) / len(set(text)) #number of words / number of unique words |

Closer Look at Python: Texts as Lists of Words

|  |  |
| --- | --- |
| Problem | Solution |
| Concatenate lists | ['one', 'two'] + ['three', 'four'] >> ['one', 'two', 'three', 'four'] |
| ***Using count in lists*** | lstText = ['one', 'one', 'three', 'one']; lstText.count('one') >>> 3 |
| Slicing (normal syntax) | lstNumber = ['one', 'two', 'three', 'four']; lstNumber[1:2] >> two three |
| Slicing (to end / from beginning) | Assumes previous list: lstNumber[1:] from position 2 to the end |
| Slicing (to end / from beginning) | Assumes previous list: lstNumber[:2] from position 0 to position 3 |
| Slicing (negative) | Counts from the end. Last two entries: lstNumber[-2:] |
| Using Index in lists | lstText = ['one', ' one', 'three', 'one'] lstText.index('three') >> 2 |
| Joining Elements of a list | lstText = ['one', 'one', 'three', 'one']  ' '.join(lstText) >> 'one one three one' |
| Sorting a list | sorted(lstText) >> ['one', 'one', 'one', 'three'] |
| Dictionary list of types | dict.items() |

Computing With Language: Simple Statistics

|  |  |
| --- | --- |
| Problem | Solution |
| Basic Frequency Distribution | fdist = FreqDist(text1)  type(fdist) >> <class 'nltk.probability.FreqDist'> |
| Get the nth element of fdist | fdist.keys()[4] >>> 'and'  fdist[fdist.keys()[4]] >>> 6024  fdist[‘and’] >>> 6024 |
| Great cumerlative graph of words | Page 18 |
| Hapaxes (words that occur once) | fdist.hapaxes() |
| Get list of long words | setWords = set(text1)  longWords = [w for w in setWords if len(w) > 15] |
| Long and frequently used | longFrequent = [w for w in setWords if len(w) > 7 and fdist[w] > 7] |
| Collocation | Sequence of words that occur together unusally often. |
| Bigram | lstTest = ['I', 'have', 'a', 'dream']  bigrams(lstTest) >>> [('I', 'have'), ('have', 'a'), ('a', 'dream')] |
| Find Collocations | Text8.collocations() >>> returns popular bigrams. |
| Create a freqency distribution of word lengths | # get lengths of all words [len(w) for w in text1]  fdist = FreqDist([len(w) for w in text1]) |
| Frequency of three letter words | fdist[3] >> 50223 |
| Relative Freqency | fdist.freq(3) >>> this is number of times wordLength(3) occurs divided by the total number of words |
| Number of Words | fdist.N() |
| Highest Count | fdist.max() |

Back to Python: Making Decisions and Taking Control

|  |  |
| --- | --- |
| Problem | Solution |
| Python String Functions | s.startswith(), s.endswith(), s.islower(), s.isupper(), s.isalpha(), s.isalnum(), s.isdigit(), s.istitle(). |
| Find strings with ‘gnt’ in them | [term for term in set(text4) if ‘gnt’ in term] |
| Prevent double counting of words | [word.lower for word in in text1] |