



Q

**Discussion Forums** 

## Week 4

## **SUBFORUMS** ΑII Assignment: Assignment 4 Submission

## **←** Week 4



## 🌄 Week 4 Notebook Provided Here 🖈

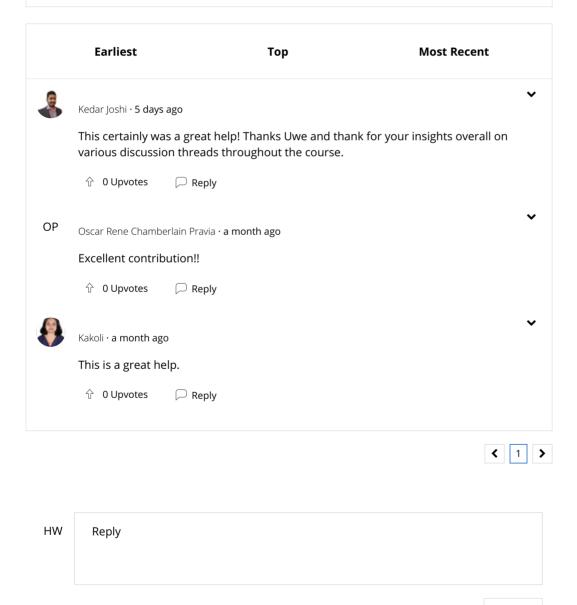
Uwe F Mayer Mentor Week  $4 \cdot 2$  months ago

There isn't a provided one. I assembled a workbook when I took the class from the material presented. Here's the code, each block is a cell.

```
import re
    import pandas as pd
                                         coursera
    import numpy as np
    import nltk
    from nltk.corpus import wordnet as wn
 6
    # Use path length in wordnet to find word similarity
    # find sense of words via synonym set
 8
    # n=noun, 01=synonym set for first meaning of the word
    deer = wn.synset('deer.n.01')
10
11
12
    elk = wn.synset('elk.n.01')
13
    deer.path_similarity(elk)
14
15
16
    horse = wn.synset('horse.n.01')
    deer.path_similarity(horse)
17
18
19
    # Use an information criteria to find word similarity
20
    from nltk.corpus import wordnet_ic
    brown_ic = wordnet_ic.ic('ic-brown.dat')
21
22
    deer.lin_similarity(elk, brown_ic)
23
24
    deer.lin_similarity(horse, brown_ic)
25
26
    # Use NLTK Collocation and Association Measures
27
    from nltk.collocations import *
28
    # load some text for examples
29
    from nltk.book import *
   # text1 is the book "Moby Dick"
30
31
   # extract just the words without numbers and sentence marks and
        make them lower case
32
    text = [w.lower() for w in list(text1) if w.isalpha()]
33
34
    bigram_measures = nltk.collocations.BigramAssocMeasures()
35
    finder = BigramCollocationFinder.from_words(text)
    finder.nbest(bigram_measures.pmi,10)
36
37
38
    # find all the bigrams with occurrence of at least 10, this
        modifies our "finder" object
39
    finder.apply_freq_filter(10)
40
    finder.nbest(bigram_measures.pmi,10)
41
    # Working with Latent Dirichlet Allocation (LDA) in Python
42
43
    # Several packages available, such as gensim and lda. Text needs
        to be
44
    # preprocessed: tokenizing, normalizing such as lower-casing,
        stopword
45
    # removal, stemming, and then transforming into a (sparse)
        matrix for
46
    # word (bigram, etc) occurences.
    # generate a set of preprocessed documents
47
48
    from nltk.stem.porter import PorterStemmer
49
    from nltk.corpus import stopwords
50
    from nltk.book import *
51
52
    len(stopwords.words('english'))
53
54
    stopwords.words('english')
55
56
   # extract just the stemmed words without numbers and sentence
        marks and make them lower case
57
    p_stemmer = PorterStemmer()
    sw = stopwords.words('english')
58
   doc1 = [p_stemmer.stem(w.lower()) for w in list(text1) if w
        .isalpha() and not w.lower() in sw]
60
    doc2 = [p_stemmer.stem(w.lower()) for w in list(text2) if w
        .isalpha() and not w.lower() in sw]
61
    doc3 = [p_stemmer.stem(w.lower()) for w in list(text3) if w
        .isalpha() and not w.lower() in sw]
62
    doc4 = [p_stemmer.stem(w.lower()) for w in list(text4) if w
        .isalpha() and not w.lower() in sw]
63
    doc5 = [p_stemmer.stem(w.lower()) for w in list(text5) if w
        .isalpha() and not w.lower() in sw]
64
    doc_set = [doc1, doc2, doc3, doc4, doc5]
65
66
    # under Windows this generates a warning
67
    import gensim
    from gensim import corpora, models
68
```

Q





Reply

Q