Ανάκτηση Πληροφορίας - Εργαστήριο

Άσκηση 2

Όνομα: Άγγελος Τζώρτζης
Α.Μ.: 18390094

Κώδικας για τις προτάσεις μας και τόν κατακερματισμός τους:

```
import nltk.tokenize
import numpy as np

sentence1 = "Thomas Jeffeson began building Monticello at the age of 26."
sentence2 = "This sentence is an example for the lab exercise. lab."

# Οι τρόποι tokenization των προτάσεων μας.
# Με την συνάρτηση split().
sentence1.split()
# Με την συνάρτηση nltk.word_tokenize().
nltk.word_tokenize(sentence1)

sentence2.split()
nltk.word_tokenize(sentence2)
```

Αποτέλεσμα της εκτέλεσης του κώδικα:
Για την πρώτη πρόταση:

```
In [4]: sentence1.split()
Out[4]:
['Thomas',
 'Jeffeson',
 'began',
 'building',
 'Monticello',
 'at',
 'the',
 'age',
 'of',
 '26.']

In [5]: nltk.word_tokenize(sentence1)
Out[5]:
['Thomas',
 'Jeffeson',
 'began',
 'building',
 'Monticello',
 'at',
 'the',
 'age',
 'of',
 '26',
 '.']
```

Για την δεύτερη πρόταση:

```
In [6]: sentence2.split()
Out[6]:
['This',
 'sentence',
 'is',
 'an',
 'example',
 'for',
 'the',
 'lab',
 'exercise.',
 'lab.']

In [7]: nltk.word_tokenize(sentence2)
Out[7]:
['This',
 'sentence',
 'is',
 'an',
 'example',
 'for',
 'the',
 'lab',
 'exercise',
 '.',
 'lab',
 '.']
```

Ερώτηση 1:

Κώδικας για τον πίνακα συμπτώσεων της sentence1 με την split:

```
# Ερώτηση 1.

# Πίνακας συμπτώσεων για την sentence1.
# Για κατακερματισμό με την split().
tokens_sent1_split = str.split(sentence1)
# Φτιάχνουμε το λεξιλόγιο της πρότασης με κάθε μοναδική λέξη και την ταξινομούμε.
vocab_sent1_split = sorted(set(tokens_sent1_split))

# Συνένωση όλων των στοιχείων του λεξιλογίου σε ένα string και εμφάνιση στην κονσόλα.
', '.join(vocab_sent1_split)

# Μήκος της πρότασης.
num_tokens1_split = len(tokens_sent1_split)
# Μήκος του λεξιλογίου.
vocab_size1_split = len(vocab_sent1_split)

# Φτιάχουμε έναν πίνακα με στήλες όσα τα στοιχεία του λεξιλογίου μας
# και γραμμές όσες οι λέξεις της πρότασης.
onehot_vector1_split = np.zeros((num_tokens1_split, vocab_size1_split), int)

# Για κάθε λέξη στην πρόταση σημειώνουμε στην αντίστοιχη στήλη του λεξιλογίου με 1.
for i, word in enumerate(tokens_sent1_split):
    onehot_vector1_split[i, vocab_sent1_split.index(word)] = 1

# Εμφάνιση του πίνακα στην κονσόλα.
onehot_vector1_split
```

Αποτελέσματα του κώδικα:

```
In [11]: ', '.join(vocab_sent1_split)
Out[11]: '26., Jeffeson, Monticello, Thomas, age, at, began, building, of, the'

In [12]: onehot_vector1_split
Out[12]:
array([[0, 0, 0, 1, 0, 0, 0, 0, 0, 0],
       [0, 1, 0, 0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 1, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 1, 0, 0],
       [0, 0, 1, 0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 1, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 1],
       [0, 0, 0, 0, 1, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 1, 0],
       [1, 0, 0, 0, 0, 0, 0, 0, 0, 0]])
```

Κώδικας για τον πίνακα συμπτώσεων της sentence1 με την nltk.word_tokenize:

```
# Για κατακερματισμό με την nltk.word_tokenize.
tokens_sent1_nltk = nltk.word_tokenize(sentence1)
vocab_sent1_nltk = sorted(set(tokens_sent1_nltk))

', '.join(vocab_sent1_nltk)

num_tokens1_nltk = len(tokens_sent1_nltk)
vocab_size1_nltk = len(vocab_sent1_nltk)

onehot_vector1_nltk = np.zeros((num_tokens1_nltk, vocab_size1_nltk), int)

for i, word in enumerate(tokens_sent1_nltk):
    onehot_vector1_nltk[i, vocab_sent1_nltk.index(word)] = 1

onehot_vector1_nltk
```

Αποτέλεσμα του κώδικα:

```
In [22]: ', '.join(vocab_sent1_nltk)
Out[22]: '., 26, Jeffeson, Monticello, Thomas, age, at, began, building, of, the'

In [23]: onehot_vector1_nltk
Out[23]:
array([[0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0],
       [0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0],
       [0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1],
       [0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0],
       [0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0],
       [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]])
```

Κώδικας για τον πίνακα συμπτώσεων της sentence2 με την split:

```
# Πίνακας συμπτώσεων για την sentence2.
# Για κατακερματισμό με την split().
tokens_sent2_split = str.split(sentence2)
vocab_sent2_split = sorted(set(tokens_sent2_split))

', '.join(vocab_sent2_split)

num_tokens2_split = len(tokens_sent2_split)
vocab_size2_split = len(vocab_sent2_split)

onehot_vector2_split = np.zeros((num_tokens2_split, vocab_size2_split), int)

for i, word in enumerate(tokens_sent2_split):
    onehot_vector2_split[i, vocab_sent2_split.index(word)] = 1

onehot_vector2_split
```

Αποτέλεσμα του κώδικα:

```
In [17]: ', '.join(vocab_sent2_split)
Out[17]: 'This, an, example, exercise., for, is, lab, lab., sentence, the'

In [18]: onehot_vector2_split
Out[18]:
array([[1, 0, 0, 0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 1, 0],
       [0, 0, 0, 0, 0, 1, 0, 0, 0, 0],
       [0, 1, 0, 0, 0, 0, 0, 0, 0, 0],
       [0, 0, 1, 0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 1, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 1],
       [0, 0, 0, 0, 0, 0, 1, 0, 0, 0],
       [0, 0, 0, 1, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 1, 0, 0]])
```

Κώδικας για τον πίνακα συμπτώσεων της sentence2 με την nltk.word_tokenize:

```
# Για κατακερματισμό με την nltk.word_tokenize.
tokens_sent2_nltk = nltk.word_tokenize(sentence2)
vocab_sent2_nltk = sorted(set(tokens_sent2_nltk))

', '.join(vocab_sent2_nltk)

num_tokens2_nltk = len(tokens_sent2_nltk)
vocab_size2_nltk = len(vocab_sent2_nltk)

onehot_vector2_nltk = np.zeros((num_tokens2_nltk, vocab_size2_nltk), int)

for i, word in enumerate(tokens_sent2_nltk):
    onehot_vector2_nltk[i, vocab_sent2_nltk.index(word)] = 1

onehot_vector2_nltk
```

Αποτέλεσμα του κώδικα:

```
In [20]: ', '.join(vocab_sent2_nltk)
Out[20]: '., This, an, example, exercise, for, is, lab, sentence, the'

In [21]: onehot_vector2_nltk
Out[21]:
array([[0, 1, 0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 1],
       [0, 0, 0, 0, 0, 0, 1, 0, 0],
       [0, 0, 1, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 1, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 1, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 1],
       [0, 0, 0, 0, 0, 0, 0, 1, 0],
       [0, 0, 0, 0, 1, 0, 0, 0, 0],
       [1, 0, 0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 1, 0],
       [1, 0, 0, 0, 0, 0, 0, 0, 0]])
```

Η κάθε στήλη αντιπροσωπεύει μία λέξη από το λεξιλόγιο που έχουμε ορίσει και σημειώνεται με 1 όταν εμφανίζεται η λέξη στην πρόταση μας.

Συμπεράσματα:

Αρχικά μπορούμε να παρατηρήσουμε μια βασική διαφορά στον κατακερματισμό με split και με nltk.word_tokenize, και αυτό είναι ότι με split δέν γίνεται διαχωρισμός μεταξύ των λέξεων με τελεία στο τέλος ενώ στο nltk.word_tokenize γίνεται. Αυτό έχει ώς αποτέλεσμα να βγάζουμε διαφορετικό πίνακα συμπτώσεων για τις προτάσεις μας ανάλογα με το τρόπο που αποφασίζουμε να κάνουμε κατακερματισμό. Για την sentence 1 βλέπουμε πώς άμα συγκρίνουμε τους δύο πίνακες συμπτώσεων πώς για την split υπάρχει μικρότερο λεξιλόγιο και μικρότερο μήκος πρότασης. Αυτό προκύπτει καθώς το nltk.word_tokenize μετράει το ‘26’ και ‘.’ ώς ξεχωριστές λέξεις ενώ το split της πιάνει ώς μία λέξη. Στον πίνακα βλέπουμε πώς αφού δεν υπάρχουν επαναλαμβανόμενες λέξεις στην πρόταση υπάρχει μόνο ένα 1 σε κάθε στήλη. Για την sentence 2 παρατηρείται παρόμοιο φαινόμενο για το μήκος της πρότασης καθώς πάλι δεν γίνεται διαχωρισμός των λέξεων απο την τελεία. Το λεξιλόγιο όμως άν και ίδιου μήκους ανεξαρτήτως με ποιόν τρόπο κάναμε κατακερματισμό, έχει διαφορετικό λεξιλόγιο. Κυρίως πρέπει να δοθεί σημασία στο ότι έχουμε μετρήσει το “lab” και “lab.” ώς διαφορετικές λέξεις. Αυτό το φαινόμενο μπορεί να μας δημιουργήσει προβλήματα εάν θέλουμε να μετρήσουμε τις λέξεις ενός κειμένου επειδή θα μετρηθούν ώς διαφορετικές λέξεις ενώ φαινομενικά είναι ίδιες. Επίσης στον πίνακα παρατηρούμε ότι οι στήλες που αντιστοιχούν στις λέξεις “.” και “lab” έχουν δύο 1 στην στήλη καθώς εμφανίζονται δύο φορές στην πρόταση ένα φαινόμενο που δεν υπήρχε στην sentence1. Τέλος μπορούμε να καταλάβουμε ότι με χρήση του nltk.word_tokenize γίνεται πιο ακριβής κατακερματισμός και έτσι προκύπτει ένα πιο ορθός πίνακας συμπτώσεων.

Κώδικας για την εμφάνιση των αποτελεσμάτων με pandas:

```
pd.DataFrame(onehot_vector1_split, columns = vocab_sent1_split)
pd.DataFrame(onehot_vector1_nltk, columns = vocab_sent1_nltk)
pd.DataFrame(onehot_vector2_split, columns = vocab_sent2_split)
pd.DataFrame(onehot_vector2_nltk, columns = vocab_sent2_nltk)
```

Αποτελέσματα για την sentence1:

```
In [26]: pd.DataFrame(onehot_vector1_split, columns = vocab_sent1_split)
Out[26]:
   26.  Jeffeson  Monticello  Thomas  age  at  began  building  of  the
0    0         0           0       1    0   0      0         0   0    0
1    0         1           0       0    0   0      0         0   0    0
2    0         0           0       0    0   0      1         0   0    0
3    0         0           0       0    0   0      0         1   0    0
4    0         0           1       0    0   0      0         0   0    0
5    0         0           0       0    0   1      0         0   0    0
6    0         0           0       0    0   0      0         0   0    1
7    0         0           0       0    1   0      0         0   0    0
8    0         0           0       0    0   0      0         0   1    0
9    1         0           0       0    0   0      0         0   0    0

In [27]: pd.DataFrame(onehot_vector1_nltk, columns = vocab_sent1_nltk)
Out[27]:
    .  26  Jeffeson  Monticello  Thomas  age  at  began  building  of  the
0   0   0         0           0       1    0   0      0         0   0    0
1   0   0         1           0       0    0   0      0         0   0    0
2   0   0         0           0       0    0   0      1         0   0    0
3   0   0         0           0       0    0   0      0         1   0    0
4   0   0         0           1       0    0   0      0         0   0    0
5   0   0         0           0       0    0   1      0         0   0    0
6   0   0         0           0       0    0   0      0         0   0    1
7   0   0         0           0       0    1   0      0         0   0    0
8   0   0         0           0       0    0   0      0         0   1    0
9   0   1         0           0       0    0   0      0         0   0    0
10  1   0         0           0       0    0   0      0         0   0    0
```

Αποτελέσματα για την sentence2:

```
In [28]: pd.DataFrame(onehot_vector2_split, columns = vocab_sent2_split)
Out[28]:
   This  an  example  exercise.  for  is  lab  lab.  sentence  the
0     1   0        0          0    0   0    0     0         0    0
1     0   0        0          0    0   0    0     0         1    0
2     0   0        0          0    0   1    0     0         0    0
3     0   1        0          0    0   0    0     0         0    0
4     0   0        1          0    0   0    0     0         0    0
5     0   0        0          0    1   0    0     0         0    0
6     0   0        0          0    0   0    0     0         0    1
7     0   0        0          0    0   0    1     0         0    0
8     0   0        0          1    0   0    0     0         0    0
9     0   0        0          0    0   0    0     1         0    0

In [29]: pd.DataFrame(onehot_vector2_nltk, columns = vocab_sent2_nltk)
Out[29]:
    .  This  an  example  exercise  for  is  lab  sentence  the
0   0     1   0        0         0    0   0    0         0    0
1   0     0   0        0         0    0   0    0         1    0
2   0     0   0        0         0    0   1    0         0    0
3   0     0   1        0         0    0   0    0         0    0
4   0     0   0        1         0    0   0    0         0    0
5   0     0   0        0         0    1   0    0         0    0
6   0     0   0        0         0    0   0    0         0    1
7   0     0   0        0         0    0   0    1         0    0
8   0     0   0        0         1    0   0    0         0    0
9   1     0   0        0         0    0   0    0         0    0
10  0     0   0        0         0    0   0    1         0    0
11  1     0   0        0         0    0   0    0         0    0
```

Ερώτηση 2:

Η κάθε γραμμή αναπαριστά την θέση μιας λέξης στην πρόταση (π.χ. Βλέπουμε ότι στην πρώτη γραμμή είναι σημειωμένη με 1 η στήλη της λέξης “This”, άρα η πρώτη λέξη της πρότασης είναι το “This”. Ένα βασικό πλεονέκτημα της χρήσης του pandas είναι πως πλέον αφού υπάρχουν ετικέτες στις γραμμές και στις στήλες του πίνακα μπορούμε εύκολα να δούμε το λεξιλόγιο της πρότασης και σε ποιές θέσεις της βρίσκεται η κάθε λέξη. Έτσι μπορούμε μόνο με την χρήση του πίνακα μας και συνδυάζοντας τις πληροφορίες που μας δίνει, να αναπαραστήσουμε πλήρως την πρόταση μας με 100% ακρίβεια. Ακόμα και στις περιπτώσεις όπως με αυτή που κάναμε τον κατακερματισμό με split που η λέξεις με τελεία στο τέλος δεν χωρίστηκαν σωστά, η αναπαράσταση με χρήση του πίνακα παράγει σωστό αποτέλεσμα. Να σημειωθεί πως αν και ο πίνακας δεν μας δείχνει πότε υπάρχουν κενά και αλλαγή γραμμής στο κείμενο μας δεν μας επηρεάζει καθώς η πληροφορία τους είναι αμελητέα. Επίσης επειδή ο πίνακας αποτελείται αυστηρά απο “0” και “1” είναι σε μορφή που ο υπολογιστής μας μπορεί να καταλάβει (δυαδικό σύστημα). Γενικά η μέθοδος αυτή μπορεί να χρησιμοποιηθεί για την αναπαράσταση οποιουδήποτε εγγράφου με ακρίβεια.

α)

Κώδικας για την ομοιότητα των 2 προτάσεων:

```
# Οι προτάσεις μας που έχουν όλες τουλάχιστον μία κοινή λέξη με την sentence2.
# sentence2 = "This sentence is an example for the lab exercise. lab."

sentence3 = "The lab experiment went completely wrong."
sentence4 = "This exercise will help your cardio."
sentence5 = "Just another sentence."

# α)
# Φτιάχνουμε dictionary με την κάθε λέξη tokenized αφαιρώντας τις τελείες.
corpus = {}
corpus['sent2'] = dict((tok.strip('.'), 1) for tok in sentence2.split())
corpus['sent3'] = dict((tok.strip('.'), 1) for tok in sentence3.split())
corpus['sent4'] = dict((tok.strip('.'), 1) for tok in sentence4.split())
corpus['sent5'] = dict((tok.strip('.'), 1) for tok in sentence5.split())

# Παρουσιάζουμε ποιές λέξεις εμφανίζονται σε κάθε πρόταση.
df = pd.DataFrame.from_records(corpus).fillna(0).astype(int).T
df

# Βρίσκουμε πόσες κοινές λέξεις υπάρχουν μεταξύ των προτάσεων.
df = df.T

# Συγκρίνουμε όλες τις προτάσεις μεταξύ τους.
df.sent2.dot(df.sent3)
df.sent2.dot(df.sent4)
df.sent2.dot(df.sent5)
df.sent3.dot(df.sent4)
df.sent3.dot(df.sent5)
df.sent4.dot(df.sent5)
```

Πίνακας για τις κοινές λέξεις των προτάσεων:

```
Out[1]:
       This  sentence  is  an  example  ...  help  your  cardio  Just  another
sent2     1         1   1   1        1  ...     0     0       0     0        0
sent3     0         0   0   0        0  ...     0     0       0     0        0
sent4     1         0   0   0        0  ...     1     1       1     0        0
sent5     0         1   0   0        0  ...     0     0       0     1        1
```

Βλέπουμε πόσες κοινές λέξεις υπάρχουν μεταξύ της κάθε πρότασης:

```
In [4]: df.sent2.dot(df.sent3)
Out[4]: 1

In [5]: df.sent2.dot(df.sent4)
Out[5]: 2

In [6]: df.sent2.dot(df.sent5)
Out[6]: 1

In [7]: df.sent3.dot(df.sent4)
Out[7]: 0

In [8]: df.sent3.dot(df.sent5)
Out[8]: 0

In [9]: df.sent4.dot(df.sent5)
Out[9]: 0
```

β)

Κώδικας για την ομοιότητα των text4 και text7 (στις πρώτες 50 λέξεις):

```
# β)
# Φτιάχνουμε πίνακα για τις πρώτες 50 λέξεις των text4 και text7 απο τις οποίες έχουν αφαιρεθεί
# τα σημεία στίξης και τα stopwords.
texts_4_7_clean = {}
texts_4_7_clean['text4'] = dict((tok.strip('.'), 1)for tok in clean_text(text4[0:50]))
texts_4_7_clean['text7'] = dict((tok.strip('.'), 1)for tok in clean_text(text7[0:50]))

dft_clean = pd.DataFrame.from_records(texts_4_7_clean).fillna(0).astype(int).T
dft_clean

# Εδώ βρίσκουμε τις κοινές λέξεις των "καθαρων" εγγράφων.
dft_clean = dft_clean.T
[(k, v) for (k, v) in (dft_clean.text4 & dft_clean.text7).items() if v]


# Φτιάχνουμε πίνακα για τις πρώτες 50 λέξεις των text4 και text7 χωρίς να έχει
# γίνει κάποιος "καθαρισμός".
texts_4_7 = {}
texts_4_7['text4'] = dict((tok.strip('.'), 1)for tok in text4[0:50])
texts_4_7['text7'] = dict((tok.strip('.'), 1)for tok in text7[0:50])


dft = pd.DataFrame.from_records(texts_4_7).fillna(0).astype(int).T
dft

dft = dft.T
[(k, v) for (k, v) in (dft.text4 & dft.text7).items() if v]
```

Σημείωση: Χρησιμοποιείται η συνάρτηση cleaned_text απο την προηγούμενη εργαστηριακή άσκηση που αφαιρεί τα σημεία στίξης και τα προθήματα, αφού αυτές οι λέξεις δέν δίνουν κάποιοι νόημα στο κείμενο.

Εδώ φαίνονται οι κοινές λέξεις για τα έγγραφα ενώ έχουμε εφαρμόσει την cleaned_text:

```
Out[12]:
       Fellow  Citizens  Senate  House  ...  Fields  PLC  named  *-1
text4       1         1       1      1  ...       0    0      0    0
text7       0         0       0      0  ...       1    1      1    1

[2 rows x 50 columns]
```

```
In [13]: dft_clean = dft_clean.T
    ...: [(k, v) for (k, v) in (dft_clean.text4 & dft_clean.text7).items() if v]
Out[13]: []
```

Εδώ φαίνονται οι κοινές λέξεις για τα έγγραφα χωρίς την χρήση της cleaned_text:

```
Out[15]:
       Fellow  -  Citizens  of  the  ...  Gold  Fields  PLC  named  *-1
text4       1  1         1   1    1  ...     0       0    0      0    0
text7       0  0         0   1    1  ...     1       1    1      1    1

[2 rows x 73 columns]
```

```
In [16]: dft = dft.T
    ...: [(k, v) for (k, v) in (dft.text4 & dft.text7).items() if v]
Out[16]: [('of', 1), ('the', 1), ('and', 1), ('was', 1), (',', 1)]
```

Ερώτηση 3:

Μια μεγάλη ποσοστιαία ομοιότητα των δύο εγγράφων σημαίνει πως πιθανόν αυτά τα δύο έγγραφα έχουν το ίδιο θέμα ή τουλάχιστον δύο θέματα τα οποία σχετίζονται μεταξύ τους. Αντιθέτως όσο μικρότερη η ομοιότητα τόσο πιό πιθανό τα έγγραφα να έχουν τελείως διαφορετική θεματολογία. Όπως φαίνεται και παραπάνω για τα δύο έγγραφα που χρησιμοποιήσαμε δέν υπήρχαν κοινές λέξεις πέρα από τα σημεία στίξης και τα προθήματα οπότε καταλαβαίνουμε πως δεν έχουν κάποιο κοινό θέμα.

Ερώτηση 4:

Βρίσκουμε τις τις 3 πιο συχνά εμφανιζόμενες λέξεις του κάθε εγγράφου:

```
fdist4 = list(FreqDist(text4[0:50]))
fdist7 = list(FreqDist(text7[0:50]))

fdist4[0:3]
fdist7[0:3]
```

```
fdist4[0:3]
['the', 'of', 'and']

fdist7[0:3]
[',', 'Vinken', 'years']
```

Αν και συνήθως δεν προτιμούμε νά φτιάχνουμε posting list με stopwords και σημεία στίξης για το παράδειγμα μας δεν θα τα αφαιρέσουμε ώστε να φανεί καλύτερα η λειτουργία των posting lists.

Κώδικας για το posting list:

```
terms1 = fdist4[0:3] + fdist7[0:3]

terms2 = [" ".join(text4[0:50]), " ".join(text7[0:50])]
vocab = []
pl = {

}

for i, doc in enumerate(terms2):
    words = doc.split(" ")
    for word in words:
        if word not in vocab:
            vocab.append(word)
        #wordId = vocab.index(word)
        if word not in pl:
            pl[word] = [i]
        else:
            pl[word].append(i)
```

Θα ψάξουμε τον πίνακα για τις λέξεις που εμφανίζονται πιο συχνά:

```
'of': [0, 0, 0, 0, 0, 1, 1],
'the': [0, 0, 0, 0, 0, 0, 1, 1],
```

```
'and': [0, 0, 1],
```

```
',': [0, 1, 1, 1, 1, 1],
```

```
'years': [1, 1],
```

```
'Vinken': [1, 1],
```

Άν και τα posting lists δεν εμφανίζονται όπως θα ήθελα μπορούμε να βγάλουμε ένα γενικό νόημα. Ο αριθμός που εμφανίζεται το index κάθε κειμένου μας βοηθάει να καταλάβουμε πόσες εμφανίζεται η λέξη σε κάθε κείμενο. Π.χ.: για το “of” βλέπουμε πέντε “0” και 2 “1”. Αυτό σημαίνει ότι η λέξη εμφανίζεται 5 φορές στο κείμενο με index 0 και 2 φορές σε αυτό με index 1.

Ερώτηση 5:

Κώδικας για το συνημίτονο των 2 προτάσεων:

```python
# Ερώτηση 5.
# Ορίζουμε το tokenizer.
tokenizer = TreebankWordTokenizer()
# Κάνουμε κατακερματισμό στα κείμενα μας.
tokens = tokenizer.tokenize(sentence2.lower() + " ".join((text4[0:50])) + " ".join((text7[0:50])))

# Μετράμε πόσες φορές εμφανίζεται κάθε λέξη
bag_of_words = Counter(tokens)

bag_of_words_sent4 = Counter(clean_text(sent4))
bag_of_words_sent7 = Counter(clean_text(sent7))

# Συνάρτηση για το συνημίτονο.
def cosine_sim(vec1, vec2):

    vec1 = [val for val in vec1.values()]
    vec2 = [val for val in vec2.values()]

    dot_prod = 0
    for i, v in enumerate(vec1):
        dot_prod += v * vec2[i]

    mag_1 = math.sqrt(sum([x**2 for x in vec1]))
    mag_2 = math.sqrt(sum([x**2 for x in vec2]))

    return dot_prod/(mag_1 * mag_2)
```

Παραπάνω βλέπουμε ότι φτιάξαμε και το bag_of_words μας.

Ερώτηση 6:

Κώδικας για το συνημίτονο των 2 βιβλίων:

```
# Ερώτηση 6.
bag_of_words_text4 = Counter(clean_text(text4))
bag_of_words_text7 = Counter(clean_text(text7))

print(cosine_sim(bag_of_words_text4, bag_of_words_text7))

# Ερώτηση 7.
docs = [" ".join(sent4), " ".join(sent7)]

corpus = docs
vectorizer = TfidfVectorizer(min_df = 1)
model = vectorizer.fit_transform(corpus)
print(model.todense().round(2))
```

Αποτελέσματα:

```
In [34]: print(cosine_sim(bag_of_words_sent4, bag_of_words_sent7))
0.674199862463242

In [35]: print(cosine_sim(bag_of_words_text4, bag_of_words_text7))
0.12491249742804174
```

Συμπεράσματα:

Βλέπουμε από τους αριθμούς που προκύπτουν ότι υπάρχει μεγαλύτερη ομοιότητα μεταξύ των προτάσεων από ότι μεταξύ των κειμένων. Έτσι μπορούμε να καταλάβουμε πώς συνήθως ένα τμήμα του κειμένου δεν είναι αρκετό για να καταλάβουμε όλο το νόημα του καθώς αν κάποιος έβλεπε το πρώτο συνημίτονο θα νόμιζε πως τα κείμενα μας έχουν αρκετά παρόμοιο νόημα, αλλα όπως βλέπουμε όταν τα συγκρίνουμε ολόκληρα βλέπουμε πως δεν σχετίζονται μεταξύ τους.

Ερώτηση 7:

Κώδικας για την ομοιότητα TF-IDF:

```
# Ερώτηση 7.
docs = [" ".join(sent4), " ".join(sent7)]

corpus = docs
vectorizer = TfidfVectorizer(min_df = 1)
model = vectorizer.fit_transform(corpus)
print(model.todense().round(2))
```

Ο πίνακας μας:

```
[[0.   0.   0.24 0.   0.   0.24 0.   0.24 0.24 0.   0.   0.   0.73 0.
  0.   0.24 0.24 0.34 0.   0.   0.  ]
 [0.27 0.27 0.   0.27 0.27 0.   0.27 0.   0.   0.27 0.27 0.27 0.   0.27
  0.27 0.   0.   0.19 0.27 0.27 0.27]]
```

Συμπεράσματα:

Γενικά η μέθοδος TFIDF

Ο πίνακας αυτός μας δείχνει το TFIDF της κάθε λέξης όπου κάθε γραμμή αναπαριστά τις λέξεις που χρησιμοποιούνται σε μία πρόταση από όλο το λεξιλόγιο. Το TFIDF θεωρείται πιο σωστός τρόπος για να μετράμε το “βάρος” μιας λέξης από το να μετράμε πόσες φορές εμφανίζεται σε ένα έγγραφο.