Ανάκτηση Πληροφορίας - Εργαστήριο

Άσκηση 1

Τεχνικές προ-επεξεργασίας κειμένου

Όνομα: Άγγελος Τζώρτζης
A.M.: 18390094

Βήμα 1 Απλά στατιστικά

Α) Κώδικας για την συνάρτηση του υπολογισμό του λεξικού πλούτου σε ένα κείμενο:

```
# Συνάρτηση που υπολογίζει πόσο πλούσιο είναι το λεξιλόγιο.
def lexical_diversity(text):
    # Ορίζουμε το μήκος λέξεων του βιβλίου μας.
    length = len(text)
    # Φτιάχνουμε μία λίστα με όλες τις λεκτικές μονάδες που βρίσκονται στο βιβλίο.
    tokens = sorted(set(text))
    # Ορίζουμε το μήκος της λίστας.
    token_amount = len(tokens)
    # Υπολογίζουμε πόσο πλούσιο είναι το λεξιλόγιο του βιβλίου μας.
    richness = token_amount/length
    # Επιστρέφει το πόσο πλούσιο είναι το λεξιλόγιο.
    return richness

# Συνάρτηση που υπολογίζει σε τι ποσοστό εμφανίζεται μία λέξη μέσα σε ένα κείμενο.
def percentage(word_num, total_words):
    # Με βάση τον μαθηματικό τύπο: (Φορές που εμφανίζεται η λέξη/Μήκος κειμένου)*100%
    percentage = (word_num/total_words)*100
    # Επιστρέφει το ποσοστό της λέξης μέσα στο κείμενο
    return percentage
```

Ερώτηση 1α:

i) Κώδικας για την συνάρτηση percentage():

```
def percentage(word_num, total_words):
    # Με βάση τον μαθηματικό τύπο: (Φορές που εμφανίζεται η λέξη/Μήκος κειμένου)*100%
    percentage = (word_num/total_words)*100
    # Επιστρέφει το ποσοστό της λέξης μέσα στο κείμενο
    return percentage
```

Κώδικας που χρησιμοποιήθηκε για την υλοποίηση του ερωτήματος:

```
# Χρησιμοποιούμε την συνάρτηση "lexical_diversity" που φτιάξαμε παραπάνω για τον υπολογισμό
# του λεκτικού πλούτου ενός κειμένου.
lr6 = lexical_diversity(text6)
print("The lexical richness of the book 'Monty Python and the Holy Grail' is: " + str(lr6) + ".")

# Χρήση της συνάρτησης ".count()" για να βρούμε πόσες φορές εμφανίζεται μια λέξη σε ένα κείμενο.
LAUNCELOT_count = text6.count("LAUNCELOT")
print("The word 'LAUNCELOT' appears " + str(LAUNCELOT_count) + " times.")

# Υπολογίζουμε το ποσοστό που εμφανίζεται η λέξη "LAUNCELOT" στο βιβλίο
# "Monty Python and the Holy Grail" με χρήση της συνάρτησης "percentage".
LAUNCELOT_percentage = percentage(LAUNCELOT_count, len(text5))
print("The word 'LAUNCELOT' appears at rate of " + str(LAUNCELOT_percentage) + "%")
```

Αποτελέσματα της εκτέλεσης του κώδικα:

```
The lexical richness of the book 'Monty Python and the Holy Grail' is: 0.1276595744680851.
The word 'LAUNCELOT' appears 76 times.
The word 'LAUNCELOT' appears at rate of 0.16885136636303044%
```

ii) Κώδικας που χρησιμοποιήθηκε για την υλοποίηση του ερωτήματος:

```
lr5 = lexical_diversity(text5)
print("The lexical richness of the book 'Chat Corpus' is: " + str(lr5) + ".")

omg_count = text5.count("omg")
OMG_count = text5.count("OMG")
lol_count = text5.count("lol")
print("The word 'omg' appears " + str(omg_count) + " times.")
print("The word 'OMG' appears " + str(OMG_count) + " times.")
print("The word 'lol' appears " + str(lol_count) + " times.")

omg_percentage = percentage(omg_count, len(text5))
OMG_percentage = percentage(OMG_count, len(text5))
lol_percentage = percentage(lol_count, len(text5))
print("The word 'omg' appears at a rate of: " + str(omg_percentage) + "%")
print("The word 'OMG' appears at a rate of: " + str(OMG_percentage) + "%")
print("The word 'lol' appears at a rate of: " + str(lol_percentage) + "%")
```

Αποτελέσματα της εκτέλεσης του κώδικα:

```
The lexical richness of the book 'Chat Corpus' is: 0.13477005109975562.
The word 'omg' appears 29 times.
The word 'OMG' appears 6 times.
The word 'lol' appears 704 times.
The word 'omg' appears at a rate of: 0.06443012663852478%
The word 'OMG' appears at a rate of: 0.013330371028660299%
The word 'lol' appears at a rate of: 1.5640968673628082%
```

Ερώτηση 1β:

Κώδικας που χρησιμοποιήθηκε για την υλοποίηση του ερωτήματος:

```
# 3 Λέξεις απο το "Monty Python and the Holy Grail".
bones_count = text6.count("bones")
hiyaah_count = text6.count("hiyaah")
fwump_count = text6.count("fwump")
print("The word 'bones' appears " + str(bones_count) + " times.")
print("The word 'hiyaah' appears " + str(hiyaah_count) + " times.")
print("The word 'fwump' appears " + str(fwump_count) + " times.")

bones_percentage = percentage(bones_count, len(text6))
hiyaah_percentage = percentage(hiyaah_count, len(text6))
fwump_percentage = percentage(fwump_count, len(text6))
print("The word 'bones' appears at a rate of: " + str(bones_percentage) + "%")
print("The word 'hiyaah' appears at a rate of: " + str(hiyaah_percentage) + "%")
print("The word 'fwump' appears at a rate of: " + str(fwump_percentage) + "%")
print()

# 3 Λέξεις απο το "Chat Corpus".
clap_count = text5.count("clap")
pinch_count = text5.count("pinch")
hogs_count = text5.count("hogs")
print("The word 'clap' appears " + str(clap_count) + " times.")
print("The word 'pinch' appears " + str(pinch_count) + " times.")
print("The word 'hogs' appears " + str(hogs_count) + " times.")

clap_percentage = percentage(clap_count, len(text5))
pinch_percentage = percentage(pinch_count, len(text5))
hogs_percentage = percentage(hogs_count, len(text5))
print("The word 'clap' appears at a rate of: " + str(clap_percentage) + "%")
print("The word 'pinch' appears at a rate of: " + str(pinch_percentage) + "%")
print("The word 'hogs' appears at a rate of: " + str(hogs_percentage) + "%")
```

Αποτελέσματα της εκτέλεσης του κώδικα:

```
The word 'bones' appears 1 times.
The word 'hiyaah' appears 1 times.
The word 'fwump' appears 1 times.
The word 'bones' appears at a rate of: 0.005893793835091649%
The word 'hiyaah' appears at a rate of: 0.005893793835091649%
The word 'fwump' appears at a rate of: 0.005893793835091649%

The word 'clap' appears 3 times.
The word 'pinch' appears 1 times.
The word 'hogs' appears 1 times.
The word 'clap' appears at a rate of: 0.006665185514330149%
The word 'pinch' appears at a rate of: 0.00222172850477671 6%
The word 'hogs' appears at a rate of: 0.00222172850477671 6%
```

<u>Συμπεράσματα:</u>

Για τον λεξιλογικό πλούτο των δύο βιβλίων βλέπουμε ότι το “Monty Python and the Holy Grail” έχει λεξιλογικό πλούτο με βαθμό ίσο με 0.1276595744680851 και το βιβλίο “Chat Corpus” έχει λεξιλογικό πλούτο με βαθμό ίσο με 0.13477005109975562. Βλέποντας αυτούς τους 2 αριθμούς συμπεραίνουμε ότι το “Chat Corpus” έχει μεγαλύτερη λεξιλογική ποικιλομορφία. Όσο για τις λέξεις που εμφανίζονται στα κείμενα, βλέπουμε πόσες φορές εμφανίζεται η κάθε μια αλλα πιο σημαντικά βλέπουμε σε τι ποσοστό εμφανίζονται το οποίο μας βοηθάει να καταλάβουμε το περιεχόμενο και νόημα του βιβλίου. (π.χ. Η λέξη “hogs” εμφανίζεται μία φορά στο με ποσοστό 0.00222172850477671 6%, οπότε μάλλον το βιβλίο δεν έχει θέμα τα γουρούνια.)

Β)
Κώδικας για την αναπαράσταση της πρώτης πρότασης του βιβλίου 1 ώς λίστας με tokens:

```
# Τυπώνουμε την πρώτη πρόταση απο το βιβλίο 1.
print(sent1)
```

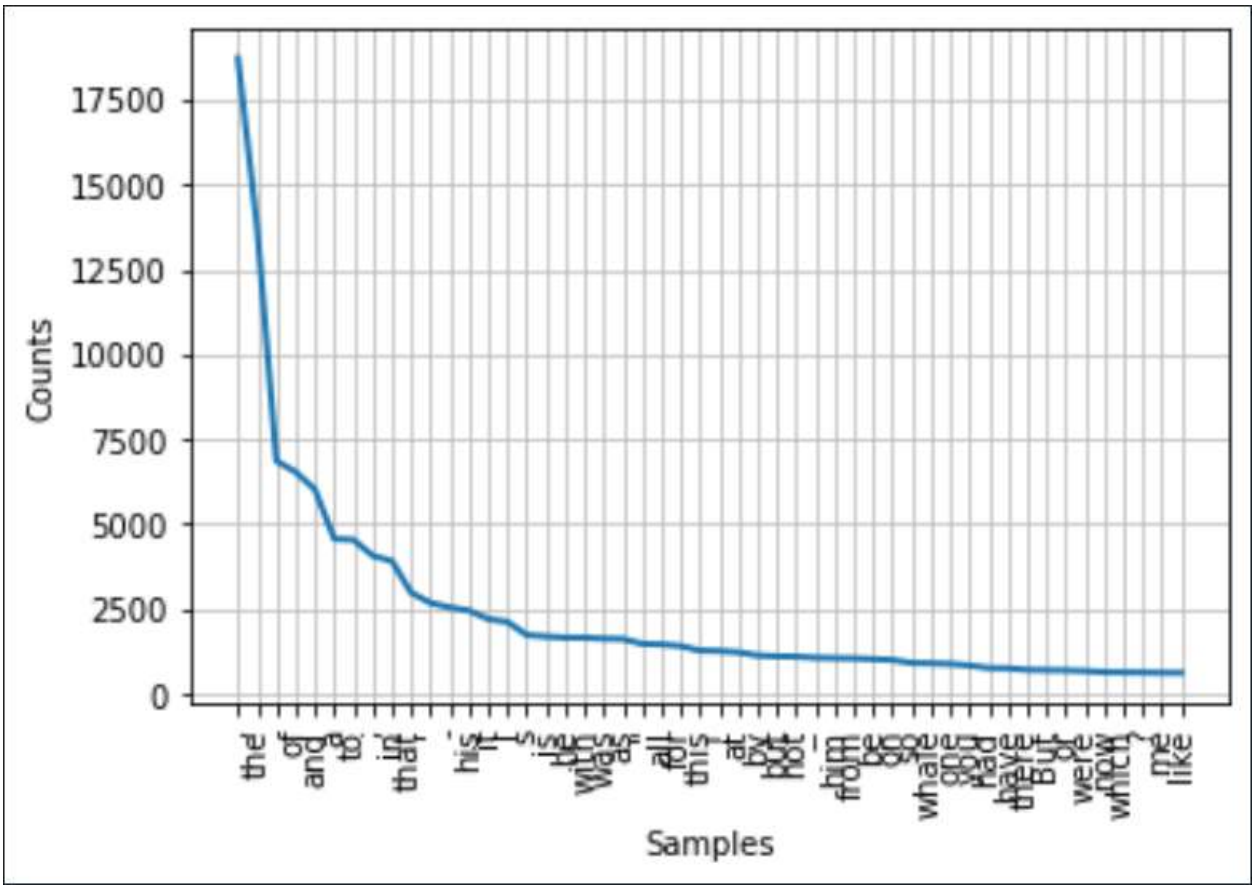
Αποτέλεσμα της εκτέλεσης του κώδικα:

```
['Call', 'me', 'Ishmael', '.']
```

Γ)
Κώδικας για τον υπολογισμό της κατανομής της συχνότητας κάθε στοιχείου του λεξιλογίου σε ένα κείμενο:

```
#Βάλε στην μεταβλητή fdist1 την κατανομή συχνότητας στο text1.
fdist1 = FreqDist(text1)
#Δείξε μου την μεταβλητή fdist1.
fdist1
#Εμφάνισε τα 50 στοιχεία που εμφανίζονται με τη μεγαλύτερη συχνότητα.
fdist1.most_common(50)
fdist1.plot(50)
```

Ερώτηση 2:
Γράφημα:



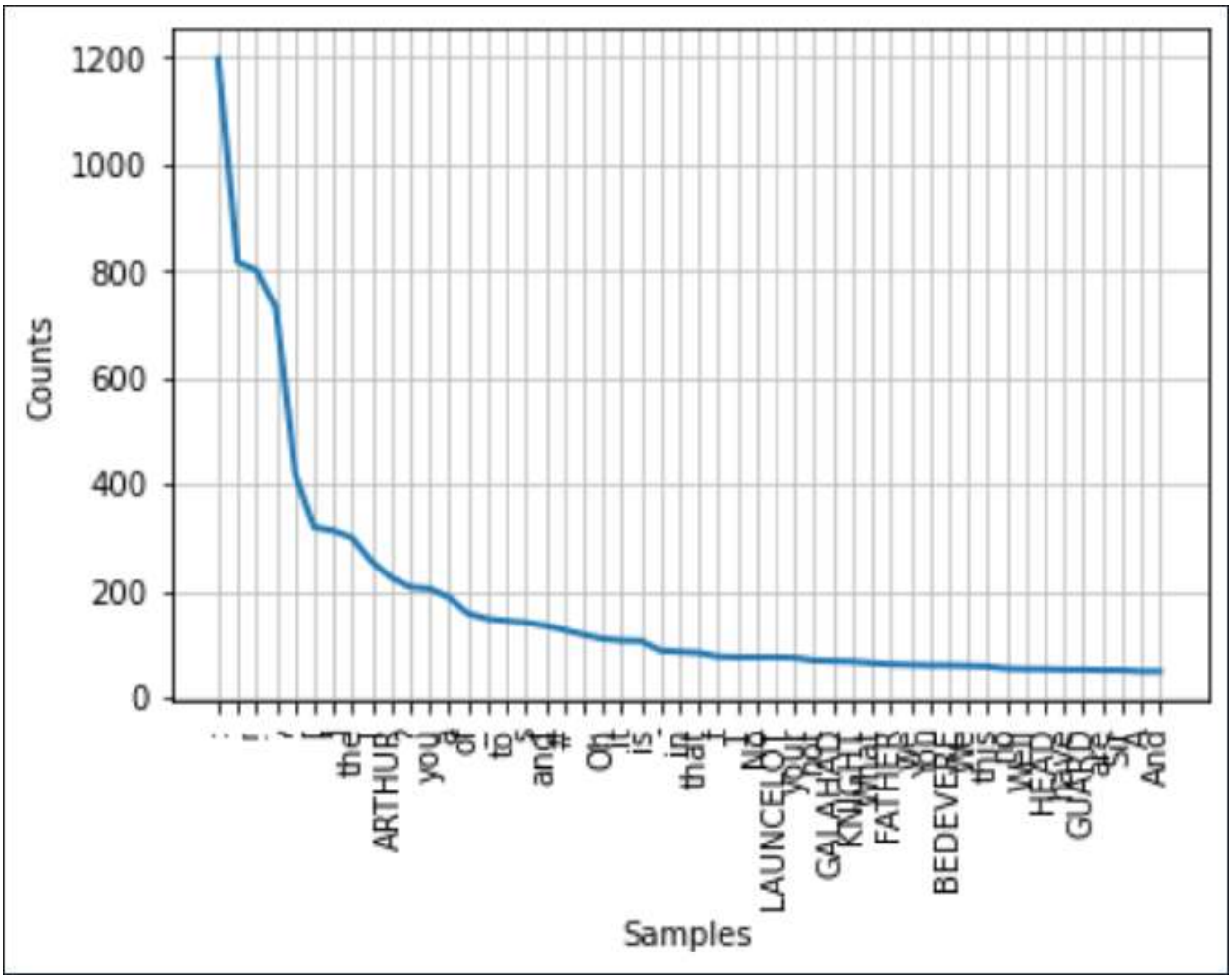
Συμπεράσματα:
Απο το γράφημα που φαίνεται παραπάνω βγάζουμε το συμπέρασμα ότι το συγκεκριμένο κείμενο έχει ως θεματολογία τις φάλαινες ή μία φάλαινα συγκεκριμένα καθώς η λέξη που εμφανίζεται σε μεγαλύτερη συχνότητα(εκτός από προθέσεις, σημεία στίξης κλπ) είναι η λέξη “whale”.

Ερώτηση 3:

Κώδικας που χρησιμοποιήθηκε για την υλοποίηση του ερωτήματος:

```
# Φτιάχνοντας ένα γράφημα για το "Monty Python and the Holy Grail", παρόμοιο με αυτό για το text1.
fdist6 = FreqDist(text6)
print(fdist6)
fdist6.most_common(50)
fdist6.plot(50)
```

Γράφημα:



Συμπεράσματα:

Με βάση το γράφημα μπορούμε να υποθέσουμε πώς το κείμενο σχετίζεται με ιππότες καθώς η λέξη που εμφανίζεται με μεγαλύτερη συχνότητα στο κείμενο (εκτός από ονόματα προθέσεις, σημεία στίξης κλπ) είναι η λέξη “KNIGHT”. Επίσης μπορούμε να παρατηρήσουμε πως τα ονόματα που εμφανίζονται σε μεγάλη συχνότητα είναι αυτά που συνήθως έχουν ιππότες σε διάφορες ιστορίες.

Βήμα 2 Κανονικοποίηση κειμένου (normalization)

Κώδικας για την κανονικοποίηση της πρώτης πρότασης του βιβλίου 1:

```
# Εμφάνισε την πρώτη πρόταση απο το βιβλίο 1.
print(sent1)
# Βάλε την sent1 στην μεταβλητή token1
tokens1 = sent1
# Για κάθε x που υπάρχει στο token1 κάνε "μικρά" τα γράμματα.
normalized_sent1 = [x.lower() for x in tokens1]
print(normalized_sent1)
```

Αποτέλεσμα της εκτέλεσης του κώδικα:

```
['Call', 'me', 'Ishmael', '.']
['call', 'me', 'ishmael', '.']
```

Ερώτηση 4:

Ο παραπάνω κώδικας στην πρώτη πρόταση του βιβλίου μετέτρεψε τα κεφαλαία γράμματα που υπήρχαν στην πρόταση σε πεζά και τα υπόλοιπα τα αφήνει όπως έχει. Μία επίπτωση αυτής της κανονικοποίησης είναι ότι λέξεις οι οποίες είναι επίθετα ή ονόματα αλλα έχουν και νοήμα πέρα από αυτό(π.χ. Bill Gates, Bill = Τιμολόγιο και Gates = Πύλες) θα μετρηθούν ώς η ίδια λέξη και ας έχουν τελείως διαφορετικό νόημα, και δεν δίνει μια ακριβή εικόνα για το κείμενο. Αντιθέτως η πρώτη λέξη μιας πρότασης η οποία έχει το πρώτο γράμμα ώς κεφαλαίο, μία λέξη η οποία γράφεται με όλα τα γράμματα κεφαλαία για να δοθεί έμφαση και μια λέξη η οποία γράφεται με όλα τα γράμματα πεζά, ακόμα και αν αποτελούνται από τα ίδια γράμματα(π.χ. Call, CALL, call) θα μετρηθούν ώς η ίδια λέξη και θα έχουμε μια πιο ακριβή εικόνα από ότι θα είχαμε χωρίς την κανονικοποίηση

Κώδικας για παράδειγμα stemming:

```
# Βάζουμε στο tokens1 τις 200 πρώτες λεκτικές μονάδες του "Sense and Sensibility".
tokens1 = text2[0:200]
# Φτιάχνουμε το αντικείμενο για το stemming.
porter = nltk.PorterStemmer()
# Εμφάνιση των στοιχείων.
# print([porter.stem(t) for t in tokens1])
[porter.stem(t) for t in tokens1]
```

Αποτελέσματα κώδικα:

```
['[', 'sens', 'and', 'sensibl', 'by', 'jane', 'austen', '1811', ']', 'chapter', '1', 'the', 'famili', 'of',
'dashwood', 'had', 'long', 'been', 'settl', 'in', 'sussex', '.', 'their', 'estat', 'wa', 'larg', ',',
'and', 'their', 'resid', 'wa', 'at', 'norland', 'park', ',', 'in', 'the', 'centr', 'of', 'their',
'properti', ',', 'where', ',', 'for', 'mani', 'gener', ',', 'they', 'had', 'live', 'in', 'so', 'respect',
'a', 'manner', 'as', 'to', 'engag', 'the', 'gener', 'good', 'opinion', 'of', 'their', 'surround',
'acquaint', '.', 'the', 'late', 'owner', 'of', 'thi', 'estat', 'wa', 'a', 'singl', 'man', ',', 'who',
'live', 'to', 'a', 'veri', 'advanc', 'age', ',', 'and', 'who', 'for', 'mani', 'year', 'of', 'hi', 'life',
',', 'had', 'a', 'constant', 'companion', 'and', 'housekeep', 'in', 'hi', 'sister', '.', 'but', 'her',
'death', ',', 'which', 'happen', 'ten', 'year', 'befor', 'hi', 'own', ',', 'produc', 'a', 'great', 'alter',
'in', 'hi', 'home', ';', 'for', 'to', 'suppli', 'her', 'loss', ',', 'he', 'invit', 'and', 'receiv', 'into',
'hi', 'hous', 'the', 'famili', 'of', 'hi', 'nephew', 'mr', '.', 'henri', 'dashwood', ',', 'the', 'legal',
'inheritor', 'of', 'the', 'norland', 'estat', ',', 'and', 'the', 'person', 'to', 'whom', 'he', 'intend',
'to', 'bequeath', 'it', '.', 'in', 'the', 'societi', 'of', 'hi', 'nephew', 'and', 'niec', ',', 'and',
'their', 'children', ',', 'the', 'old', 'gentleman', "'", 's', 'day', 'were', 'comfort', 'spent', '.',
'hi', 'attach', 'to', 'them', 'all', 'increas', '.', 'the', 'constant']
```

Κώδικας για παράδειγμα lemmatization:

```
# Φορτώνουμε την βιβλιοθήκη wordnet.
nltk.download('wordnet')
# Φτιάχνουμε το αντικείμενο για το lemmatization.
wnl = nltk.WordNetLemmatizer()
# Εμφάνιση των στοιχείων.
# print([wnl.lemmatize(t) for t in tokens1])
[wnl.lemmatize(t) for t in tokens1]
```

Αποτελέσματα κώδικα:

```
['[', 'sens', 'and', 'sensibl', 'by', 'jane', 'austen', '1811', ']', 'chapter', '1', 'the', 'famili', 'of',
'dashwood', 'had', 'long', 'been', 'settl', 'in', 'sussex', '.', 'their', 'estat', 'wa', 'larg', ',',
'and', 'their', 'resid', 'wa', 'at', 'norland', 'park', ',', 'in', 'the', 'centr', 'of', 'their',
'properti', ',', 'where', ',', 'for', 'mani', 'gener', ',', 'they', 'had', 'live', 'in', 'so', 'respect',
'a', 'manner', 'as', 'to', 'engag', 'the', 'gener', 'good', 'opinion', 'of', 'their', 'surround',
'acquaint', '.', 'the', 'late', 'owner', 'of', 'thi', 'estat', 'wa', 'a', 'singl', 'man', ',', 'who',
'live', 'to', 'a', 'veri', 'advanc', 'age', ',', 'and', 'who', 'for', 'mani', 'year', 'of', 'hi', 'life',
',', 'had', 'a', 'constant', 'companion', 'and', 'housekeep', 'in', 'hi', 'sister', '.', 'but', 'her',
'death', ',', 'which', 'happen', 'ten', 'year', 'befor', 'hi', 'own', ',', 'produc', 'a', 'great', 'alter',
'in', 'hi', 'home', ';', 'for', 'to', 'suppli', 'her', 'loss', ',', 'he', 'invit', 'and', 'receiv', 'into',
'hi', 'hous', 'the', 'famili', 'of', 'hi', 'nephew', 'mr', '.', 'henri', 'dashwood', ',', 'the', 'legal',
'inheritor', 'of', 'the', 'norland', 'estat', ',', 'and', 'the', 'person', 'to', 'whom', 'he', 'intend',
'to', 'bequeath', 'it', '.', 'in', 'the', 'societi', 'of', 'hi', 'nephew', 'and', 'niec', ',', 'and',
'their', 'children', ',', 'the', 'old', 'gentleman', "'", 's', 'day', 'were', 'comfort', 'spent', '.',
'hi', 'attach', 'to', 'them', 'all', 'increas', '.', 'the', 'constant']
```

Υποσημείωση: Τα αποτελέσματα του κώδικα εμφανίζονται οριζόντια ώστε να χωρέσουν στο pdf. Στην εκτέλεση του κώδικα θα εμφανίζονται κάθετα για πιο εύκολη ανάγνωση των αποτελεσμάτων από τον χρήστη.

Ερώτηση 5:

Κώδικας για την υλοποίηση του ερωτήματος:

```
# Δικά μάς κείμενα.
greek_text = "Τα παιδιά έπαιζαν στο πάρκο με τα σκυλιά τους."
english_text1 = "The children were playing in the park with their dogs."
english_text2 = "The dogs were chasing the cats all day."
# Πειράματα με το ελληνικό κείμενο.
# Χρησιμοποιούμε την συνάρτηση split για να χωριστεί σε tokens.
# print([porter.stem(gt) for gt in greek_text.split()])
# print([porter.stem(et) for et in english_text1.split()])
# print([porter.stem(et) for et in english_text2.split())
[porter.stem(gt) for gt in greek_text.split()]
[porter.stem(et1) for et1 in english_text1.split()]
[porter.stem(et2) for et2 in english_text2.split()]
# print([wnl.lemmatize(gt) for gt in greek_text.split()])
# print([wnl.lemmatize(et) for et in english_text1.split()])
# print([wnl.lemmatize(et) for et in english_text2.split()])
[wnl.lemmatize(gt) for gt in greek_text.split()]
[wnl.lemmatize(et1) for et1 in english_text1.split()]
[wnl.lemmatize(et2) for et2 in english_text2.split()]
```

Υποσημείωση: Χρησιμοποιείται το split για τον διαχωρισμό του κειμένου μας σε tokens γιατί αλλιώς θα χωριζόταν ανα γράμμα και θα εμφανίζαμε τελείως λάθος αποτέλεσμα.

Αποτελέσματα της εκτέλεσης του κώδικα:

```
In [93]: [porter.stem(gt) for gt in greek_text.split()]
Out[93]: ['τα', 'παιδιά', 'έπαιζαν', 'στο', 'πάρκο', 'με', 'τα', 'σκυλιά', 'τους.']

In [94]: [porter.stem(et1) for et1 in english_text1.split()]
Out[94]:
['the',
 'children',
 'were',
 'play',
 'in',
 'the',
 'park',
 'with',
 'their',
 'dogs.']

In [95]: [porter.stem(et2) for et2 in english_text2.split()]
Out[95]: ['the', 'dog', 'were', 'chase', 'the', 'cat', 'all', 'day.']

In [96]: [wnl.lemmatize(gt) for gt in greek_text.split()]
Out[96]: ['Τα', 'παιδιά', 'έπαιζαν', 'στο', 'πάρκο', 'με', 'τα', 'σκυλιά', 'τους.']

In [97]: [wnl.lemmatize(et1) for et1 in english_text1.split()]
Out[97]:
['The',
 'child',
 'were',
 'playing',
 'in',
 'the',
 'park',
 'with',
 'their',
 'dogs.']

In [98]: [wnl.lemmatize(et2) for et2 in english_text2.split()]
Out[98]: ['The', 'dog', 'were', 'chasing', 'the', 'cat', 'all', 'day.']
```

Υποσημείωση: Δεν γνωρίζω γιατί κάποιες φορές τυπώνονται κάθετα και κάποιες φορές οριζόντια τα αποτελέσματα του κώδικα μας.

Συμπεράσματα:

Αρχικά βλέπουμε πως το stemming δουλεύει μερικά για τα ελληνικά μετατρέποντας τα κεφαλαία σε πεζά. Το lemmatization δεν δουλεύει καθόλου για τα ελληνικά γράμματα. Και οι δύο μέθοδοι δουλεύουν για τα Αγγλικά. Η διαφορά των 2 μεθόδων φαίνεται σε δύο σημεία. Στην πρόταση english_text1 η αλλαγή από playing σε play στο stemming και η αλλαγή από children σε child στο lemmatization. Οι αλλαγές αυτές μας δείχνουν την λειτουργία της κάθε συνάρτησης καθώς η stem() μας αποκόπτει τις λέξεις ώστε να μείνει μόνο η “βάση” τούς και η χρήση της lemmatize() είναι να ομαδοποιηθούν οι λέξεις με το ίδιο νόημα. Η απλή κανονικοποίηση που είδαμε νωρίτερα λειτουργεί πολύ πιο φτωχά καθώς απλά μετατρέπει τα κεφαλαία γράμματα σε πεζά που βοηθάει ελάχιστα στο να ομαδοποιήσουμε λέξεις καθώς η ίδια λέξη που είναι σε διαφορετικό χρόνο ή σε άλλη κλήση θα διαχωριστεί. Το stemming ενω θα ομαδοποιήσει λέξεις που προέρχονται από την ίδια λέξη απλα αλλάζει ο χρόνος η κλήση δεν θα ομαδοποιήσει λέξεις συνώνυμες η που έχουν το ίδιο νόημα. Το lemmatizing θα ομαδοποιήσει τις λέξεις με βάση το νόημα αλλά δεν αλλάζει τις λέξεις με βάση τον χρόνο και την κλήση. Επίσης το lemmatize δεν μετατρέπει τα κεφαλαία γράμματα σε πεζά. Ο λόγος που βάλαμε το english_text2 είναι για να δείξουμε παρακάτω πως οι λέξεις που είναι κολλημένες σε σημείο στίξης δεν

διαβάζονται σωστά απο το stemming και το lemmatization(π.χ. το dogs δεν θα αλλάξει στο english_text1 αλλα θα αλλάξει στο english_text2)

Βήμα 3 Tokenization

Κώδικας για tokenization:

```
sentence = "Monticello wasn't designated as UNESCO World Heritage Site until 1987."
print(sentence.split())
print(str.split(sentence))
print(nltk.word_tokenize(sentence))
```

Αποτελεσματα:

```
['Monticello', "wasn't", 'designated', 'as', 'UNESCO', 'World', 'Heritage', 'Site', 'until', '1987.']
['Monticello', "wasn't", 'designated', 'as', 'UNESCO', 'World', 'Heritage', 'Site', 'until', '1987.']
['Monticello', 'was', "n't", 'designated', 'as', 'UNESCO', 'World', 'Heritage', 'Site', 'until', '1987', '.']
```

Ερώτηση 6:

Κώδικας που χρησιμοποιήθηκε για την υλοποίηση του ερωτήματος:

```
# Μετατροπή των tokens απο λίστα σε string.
text2_string = " ".join(text2[0:200])
greek_text = "Τα παιδιά δέν έπαιζαν στο πάρκο σήμερα."
english_text = "The children weren't playing in the park today."
# Tokenization με split().
str.split(text2_string)
str.split(greek_text)
str.split(english_text)
# Tokenization με nltk.word_tokenize.
nltk.word_tokenize(text2_string)
nltk.word_tokenize(greek_text)
nltk.word_tokenize(english_text)
```

Αποτελεσματα του tokenization των δικών μας προτάσεων:

```
In [153]: str.split(greek_text)
Out[153]: ['Τα', 'παιδιά', 'δέν', 'έπαιζαν', 'στο', 'πάρκο', 'σήμερα.']

In [154]: str.split(english_text)
Out[154]: ['The', 'children', "weren't", 'playing', 'in', 'the', 'park', 'today.']

In [155]: nltk.word_tokenize(greek_text)
Out[155]: ['Τα', 'παιδιά', 'δέν', 'έπαιζαν', 'στο', 'πάρκο', 'σήμερα', '.']

In [156]: nltk.word_tokenize(english_text)
Out[156]:
['The',
 'children',
 'were',
 "n't",
 'playing',
 'in',
 'the',
 'park',
 'today',
 '.']
```

Συμπεράσματα:

Δεν εμφανίζει τα αποτελέσματα για τις πρώτες 200 λέξεις του “Sense and Sensibility καθώς δεν θα χωρούσαν στην οθόνη. Με χρήση της split τα σημεία στίξης που είναι κολλητά σε μία λέξη δεν χωρίζονται όταν γίνεται tokenization. Αντιθέτως η συνάρτηση word_tokenize() τα διαχωρίζει αλλα δημιουργείται θέμα με την απόστροφο καθώς δεν κάνει σωστά την λέξη που την έχει(όπως φαίνεται παραπάνω με την λέξη weren’t. Στην συγκεκριμένη περίπτωση φαίνεται να λειτουργεί σωστά και με ελληνικό κείμενο.

Βήμα 4 Αφαίρεση σημείων στίξης και προθημάτων (stop words)

Κώδικας για την εμφάνιση των σημείων στίξης:

```
# Εμφάνιση των σημείων στίξης.
print(string.punctuation)
```

Σημεία στίξης:

```
In [162]: print(string.punctuation)
!"#$%&'()*+,-./:;<=>?@[\]^_`{|}~
```

Ερώτηση 7:

Κώδικας για την υλοποίηση του ερωτήματος για τις αγγλικές λέξεις:

```
# Ερώτηση 7.
# Για να δούμε τα προθήματα της Αγγλικής γλώσσας.
# nltk.download('stopwords')
stopwords = nltk.corpus.stopwords.words('english')
print(stopwords)
# Εμφάνιση αριθμών προθημάτων.
print(len(stopwords))
```

Αποτελέσματα του κώδικα:

```
['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "you've", "you'll", "you'd",
'your', 'yours', 'yourself', 'yourselves', 'he', 'him', 'his', 'himself', 'she', "she's", 'her', 'hers',
'herself', 'it', "it's", 'its', 'itself', 'they', 'them', 'their', 'theirs', 'themselves', 'what', 'which',
'who', 'whom', 'this', 'that', "that'll", 'these', 'those', 'am', 'is', 'are', 'was', 'were', 'be', 'been',
'being', 'have', 'has', 'had', 'having', 'do', 'does', 'did', 'doing', 'a', 'an', 'the', 'and', 'but',
'if', 'or', 'because', 'as', 'until', 'while', 'of', 'at', 'by', 'for', 'with', 'about', 'against',
'between', 'into', 'through', 'during', 'before', 'after', 'above', 'below', 'to', 'from', 'up', 'down',
'in', 'out', 'on', 'off', 'over', 'under', 'again', 'further', 'then', 'once', 'here', 'there', 'when',
'where', 'why', 'how', 'all', 'any', 'both', 'each', 'few', 'more', 'most', 'other', 'some', 'such', 'no',
'nor', 'not', 'only', 'own', 'same', 'so', 'than', 'too', 'very', 's', 't', 'can', 'will', 'just', 'don',
"don't", 'should', "should've", 'now', 'd', 'll', 'm', 'o', 're', 've', 'y', 'ain', 'aren', "aren't",
'couldn', "couldn't", 'didn', "didn't", 'doesn', "doesn't", 'hadn', "hadn't", 'hasn', "hasn't", 'haven',
"haven't", 'isn', "isn't", 'ma', 'mightn', "mightn't", 'mustn', "mustn't", 'needn', "needn't", 'shan',
"shan't", 'shouldn', "shouldn't", 'wasn', "wasn't", 'weren', "weren't", 'won', "won't", 'wouldn',
"wouldn't"]
179
```

Κώδικας για την υλοποίηση του ερωτήματος για τις ελληνικές λέξεις:

```
# Για τα προθήματα της Ελληνικής γλώσσας.
stopwords = nltk.corpus.stopwords.words('greek')
print(stopwords)
# Εμφάνιση αριθμών προθημάτων.
print(len(stopwords))
```

Αποτελέσματα του κώδικα:

```
['αλλα', 'αν', 'αντι', 'απο', 'αυτα', 'αυτεσ', 'αυτη', 'αυτο', 'αυτοι', 'αυτοσ', 'αυτουσ', 'αυτων', 'αἱ',
'αἳ', 'αἵ', 'αὐτόσ', 'αὐτὸς', 'αὖ', 'γάρ', 'γα', 'γα^', 'γε', 'για', 'γοῦν', 'γὰρ', "δ'", 'δέ', 'δή',
'δαί', 'δαίσ', 'δαὶ', 'δαὶς', 'δε', 'δεν', "δι'", 'διά', 'διὰ', 'δὲ', 'δὴ', 'δ’', 'εαν', 'ειμαι',
'ειμαστε', 'ειναι', 'εισαι', 'εισται', 'εκεινα', 'εκεινεσ', 'εκεινη', 'εκεινο', 'εκεινοι', 'εκεινοσ',
'εκεινουσ', 'εκεινων', 'ενω', 'επ', 'επι', 'εἰ', 'εἰμί', 'εἰμὶ', 'εἰς', 'εἰσ', 'εἶ', 'εἶμι', 'εἶτε', 'ἡ',
'θα', 'ισωσ', 'κ', 'καί', 'καίτοι', 'καθ', 'και', 'κατ', 'κατά', 'κατα', 'κατὰ', 'καὶ', 'κι', 'κἀν',
'κἂν', 'μέν', 'μή', 'μήτε', 'μα', 'με', 'μεθ', 'μετ', 'μετά', 'μετα', 'μετὰ', 'μη', 'μην', 'μέν', 'μὲν',
'μὴ', 'μὴν', 'να', 'ο', 'οι', 'ομωσ', 'οπωσ', 'οσο', 'οτι', 'οἱ', 'οἳ', 'οἷς', 'οὐ', 'οὐδ', 'οὐδέ',
'οὐδείσ', 'οὐδεὶς', 'οὐδὲ', 'οὐδὲν', 'οὐκ', 'οὐχ', 'οὐχὶ', 'οὖς', 'οὔτε', 'οὕτω', 'οὕτως', 'οὕτωσ', 'οὖν',
'οὗ', 'οὗτος', 'οὗτοσ', 'παρ', 'παρά', 'παρα', 'παρὰ', 'περί', 'περὶ', 'ποια', 'ποιεσ', 'ποιο', 'ποιοι',
'ποιοσ', 'ποιουσ', 'ποιων', 'ποτε', 'που', 'ποῦ', 'προ', 'προσ', 'πρόσ', 'πρὸ', 'πρὸς', 'πωσ', 'πωσ',
'σε', 'στη', 'στην', 'στο', 'στον', 'σόσ', 'σύ', 'σύν', 'σὸς', 'σὺ', 'σὺν', 'τά', 'τήν', 'τί', 'τίς',
'τίσ', 'τα', 'ταῖς', 'τε', 'την', 'της', 'τι', 'τινα', 'τις', 'τισ', 'το', 'τοί', 'τοι', 'τοιοῦτος',
'τοιοῦτοσ', 'τον', 'τοτε', 'του', 'τούσ', 'τοὺς', 'τοῖς', 'τοῦ', 'των', 'τό', 'τόν', 'τότε', 'τὰ', 'τὰς',
'τὴν', 'τὸ', 'τὸν', 'τῆς', 'τῆσ', 'τῇ', 'τῶν', 'τῷ', 'ωσ', "ἀλλ'", 'ἀλλά', 'ἀλλὰ', 'ἀλλ’', 'ἀπ', 'ἀπό',
'ἀπὸ', 'ἀφ', 'ἂν', 'ἃ', 'ἄλλος', 'ἄλλοσ', 'ἐν', 'ἄρα', 'ἅμα', 'ἐάν', 'ἐγώ', 'ἐγὼ', 'ἐκ', 'ἐμόσ', 'ἐμὸς',
'ἐν', 'ἐξ', 'ἐπί', 'ἐπεὶ', 'ἐπὶ', 'ἐστι', 'ἐφ', 'ἐὰν', 'ἑαυτοῦ', 'ἔτι', 'ἡ', 'ἦ', 'ἦ', 'ἤ', 'ἤ', 'ἧς',
'ἵνα', 'ὁ', 'ὃ', 'ὃν', 'ὃς', 'ὁ', 'ὅδε', 'ὅθεν', 'ὅπερ', 'ὅς', 'ὅσ', 'ὅστις', 'ὅστισ', 'ὅτε', 'ὅτι',
'ὑμόσ', 'ὑπ', 'ὑπέρ', 'ὑπό', 'ὑπὲρ', 'ὑπὸ', 'ὡς', 'ὡσ', 'ὥς', 'ὥστε', 'ὦ', 'ᾧ']
265
```

Συμπεράσματα:
Παραπάνω εμφανίζονται όλα τα stopwords για την Αγγλική και την Ελληνική γλώσσα. Είναι 179 για την Αγγλική γλώσσα και 265 για την Ελληνική.

Συνάρτηση για τον καθαρισμό κειμένου:

```
# Συνάρτηση για να καθαρίσουμε ένα κείμενο απο σημεία στίξης και προθήματα.
def clean_text(text):
    # Ορίζουμε ξανά τα stopwords.
    stopwords = nltk.corpus.stopwords.words('english')
    stopwords = stopwords + nltk.corpus.stopwords.words('greek')
    # Άδεια λίστα ώστα να βάλουμε τα "καθαρά" tokens.
    cleaned_tokens = []
    for token in text:
        # Εδώ ελέγχουμε ότι η κάθε λέξη δεν έιναι ούτε σημείο στίξης ούτε πρόθημα.
        if token not in string.punctuation and token not in stopwords:
            # Εδώ προσθέτουμε στην λέξη στη λίστα cleaned_tokens.
            cleaned_tokens.append(token)
    # Εκτύπωση του τελικού αποτελέσματος.
    print(cleaned_tokens)
```

Ερώτηση 8:
Κώδικας για την υλοποίηση του ερωτήματος:

```
# Βάζουμε στην συνάρτηση τις πρώτες 200 λέξεις του
# "Sense and Sensibility και εκτυπώνουμε το αποτέλεσμα.
clean_text(text2[0:200])
# Κάνουμε το ίδιο και με δικές μας προτάσεις.
greek_text = "Σήμερα δέν έπαιζαν τά παιδιά στο πάρκο."
english_text = "The children weren't playing in the park today."
clean_text(nltk.word_tokenize(greek_text))
clean_text(nltk.word_tokenize(english_text))
```

Αποτελέσματα της εκτέλεσης του κώδικα:

```
['Sense', 'Sensibility', 'Jane', 'Austen', '1811', 'CHAPTER', '1', 'The', 'family', 'Dashwood', 'long',
'settled', 'Sussex', 'Their', 'estate', 'large', 'residence', 'Norland', 'Park', 'centre', 'property',
'many', 'generations', 'lived', 'respectable', 'manner', 'engage', 'general', 'good', 'opinion',
'surrounding', 'acquaintance', 'The', 'late', 'owner', 'estate', 'single', 'man', 'lived', 'advanced',
'age', 'many', 'years', 'life', 'constant', 'companion', 'housekeeper', 'sister', 'But', 'death',
'happened', 'ten', 'years', 'produced', 'great', 'alteration', 'home', 'supply', 'loss', 'invited',
'received', 'house', 'family', 'nephew', 'Mr', 'Henry', 'Dashwood', 'legal', 'inheritor', 'Norland',
'estate', 'person', 'intended', 'bequeath', 'In', 'society', 'nephew', 'niece', 'children', 'old',
'Gentleman', 'days', 'comfortably', 'spent', 'His', 'attachment', 'increased', 'The', 'constant']
['Σήμερα', 'δέν', 'έπαιζαν', 'παιδιά', 'πάρκο']
['The', 'children', "n't", 'playing', 'park', 'today']
```

Συμπεράσματα:
Η συνάρτηση λειτουργεί σχετικά σωστά διαγράφοντας από τα κείμενα μας τα σημεία στίξης και τα προθήματα με εξαίρεση τις αποστρόφους(επειδή χρησιμοποιούμε το word_tokenizer για να χωρίσουμε τις προτάσεις μας σε tokens, με split πάλι θα είχαμε θέμα επειδή δέν θα γινόταν διαχωρισμος στις λέξεις που είχαν διπλα σημείο στίξη π.χ. τελεία , θαυμαστικό) και τις λέξεις με κεφαλαία γράμματα δέν τις λαμβάνει ώς stopword, αυτό μπορεί να λυθεί κάνοντας μια απλή κανονικοποίηση στο κείμενο μας ή και μέσα στην συνάρτηση. Λειτουργεί σωστά και για Ελληνικά και Αγγλικά.

Κάνουμε μία μικρή αλλαγή στην συνάρτηση ώστε να λειτουργεί για την επόμενη ερώτηση:

```
def clean_text(text):
    # Ορίζουμε ξανά τα stopwords.
    stopwords = nltk.corpus.stopwords.words('english')
    stopwords = stopwords + nltk.corpus.stopwords.words('greek')
    # Άδεια λίστα ώστα να βάλουμε τα "καθαρά" tokens.
    cleaned_tokens = []
    for token in text:
        # Εδώ ελέγχουμε ότι η κάθε λέξη δεν έιναι ούτε σημείο στίξης ούτε πρόθημα.
        if token not in string.punctuation and token not in stopwords:
            # Εδώ προσθέτουμε στην λέξη στη λίστα cleaned_tokens.
            cleaned_tokens.append(token)
    # Εκτύπωση του τελικού αποτελέσματος.
    return cleaned_tokens
```

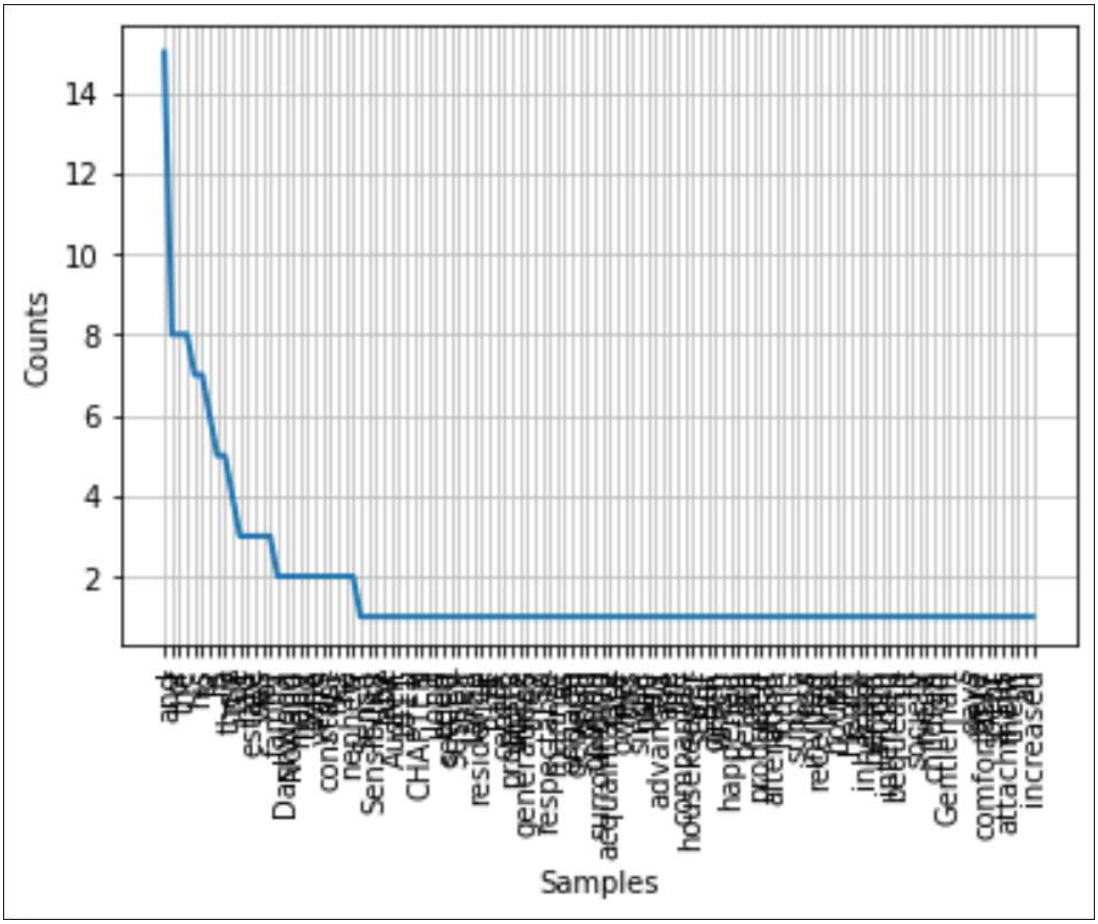
Ερώτηση 9:

Κώδικας για την υλοποίηση του ερωτήματος:

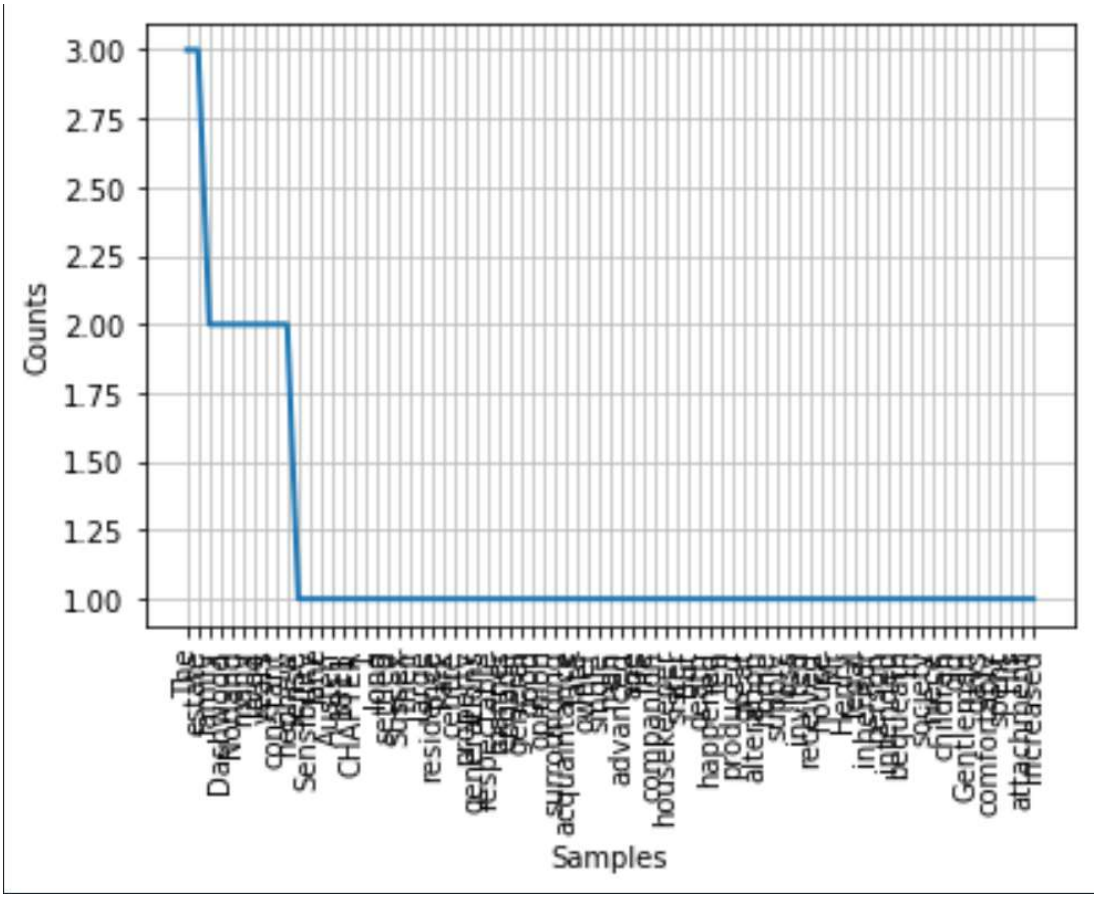
```
# Για το αρχικό κείμενο.
fdist2 = FreqDist(text2[0:200])
print(fdist2)
fdist2.most_common()
fdist2.plot()
# Για το καθαρό κείμενο.
fdist2 = FreqDist(clean_text(text2[0:200]))
print(fdist2)
fdist2.most_common()
fdist2.plot()
```

Αποτελέσματα της εκτέλεσης του κώδικα:

Για το αρχικό κείμενο(πρώτες 200 λέξεις):



Για το “καθαρό” κείμενο(Οι 200 λέξεις περασμένες απο την συνάρτηση clean_text):



Συμπεράσματα:

Το πρώτο γράφημα αν και έχει περισσότερες λέξεις δεν μπορεί να αντιπροσωπεύσει πραγματικά το νοήμα που έχει το κείμενο καθώς περιέχει μέσα στις λέξεις του τα σημεία στίξης και τα προθήματα βγάζοντας μια όχι τόσο καθαρή εικόνα. Αντιθέτως στο γράφημα για το καθαρό κείμενο μας εμφανίζει τις λέξεις τις οποίες συνεισφέρουν στον νόημα του κειμένου. Και στις δύο περιπτώσεις βέβαια έχουμε πολύ μικρό δείγμα λέξεων για να καταλάβουμε το νόημα ενός ολόκληρου βιβλίου.