МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ “ЛЬВІВСЬКА ПОЛІТЕХНІКА”
ІНСТИТУТ КОМП’ЮТЕРНИХ НАУК ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра “Системи автоматизованого проектування”

Звіт

до лабораторної роботи №4

на тему: “ ВИВЧЕННЯ БІБЛІОТЕКИ ПРИКЛАДНИХ ПРОГРАМ NLTK, ДЛЯ ОПРАЦЮВАННЯ ТЕКСТІВ ПРИРОДНОЮ МОВОЮ.
ДОСТУП ТА РОБОТА З ЛЕКСИЧНИМИ РЕСУРСАМИ..”

з дисципліни “Комп’ютерна лінгвістика”

Виконала:
студентка групи ПРЛм-11
Зварич О.І.
Прийняв:
викладач
Дупак Б.П.

Львів-2015

1. Мета.

- Вивчення основ програмування на мові Python.
- Вивчення методів доступу та роботи з лексичним ресурсами.
- Семантичний словник англійської мови WordNet.

2. Теоретичні відомості.

1. Поняття функції та модуля.

При програмуванні часто необхідно частину програми виконати (використати) декілька разів. Наприклад, потрібно написати програму, яка здійснює утворення множини з однини іменників і вона буде виконуватись в різних місцях програми. Швидше ніж повторювати той самий код декілька разів і більш ефективно і надійно організувати цю роботу через функцію. Функція - це програмна конструкція, яку можна викликати з одним або більше вхідними параметрами, і отримувати результат на виході. Визначаємо функцію, використовуючи ключове слово *def* далі потрібно дати назву функції і визначити вхідні параметри, після двокрапки записується тіло функції. Ключове слово *return* використовується для відображення значення, яке ми хочемо отримати на виході функції.

2. Генерація випадкового тексту за допомогою біграмів.

Умовний частотний розподіл можна використати для побудови таблиці біграмів (пар слів). Функція NLTK bigrams() , як аргумент бере список слів і повертає список послідовних пар слів.

3. Лексичні ресурси NLTK.

Лексичний ресурс або просто словник це набір слів та/або словосполучень, які асоціюються з такою інформацією, як частина мови та опис значення. Лексичні ресурси є вторинними по відношенню до текстів і зазвичай створюються і вдосконалюються з використанням текстів.

4. WordNet – лексична база даних англійської мови

WordNet, це семантично орієнтований словник англійської мови, подібний до традиційних тезаурусів але з більш багатою структурою. У *WordNet* слова групуються у набори синонімів – синсети, кожен із своїм визначенням і зв'язками з іншими синсетами. *WordNet 3.0* розповсюджується разом з NLTK і містить 155287 слів та 117659 синсетів. Хоча *WordNet* розроблявся для психолінгвістики - цей словник широко використовується в NLP та в задачах інформаційного пошуку. Розробки для інших мов проводяться на основі документації, яка наведена у http://www.globalwordnet.org/.

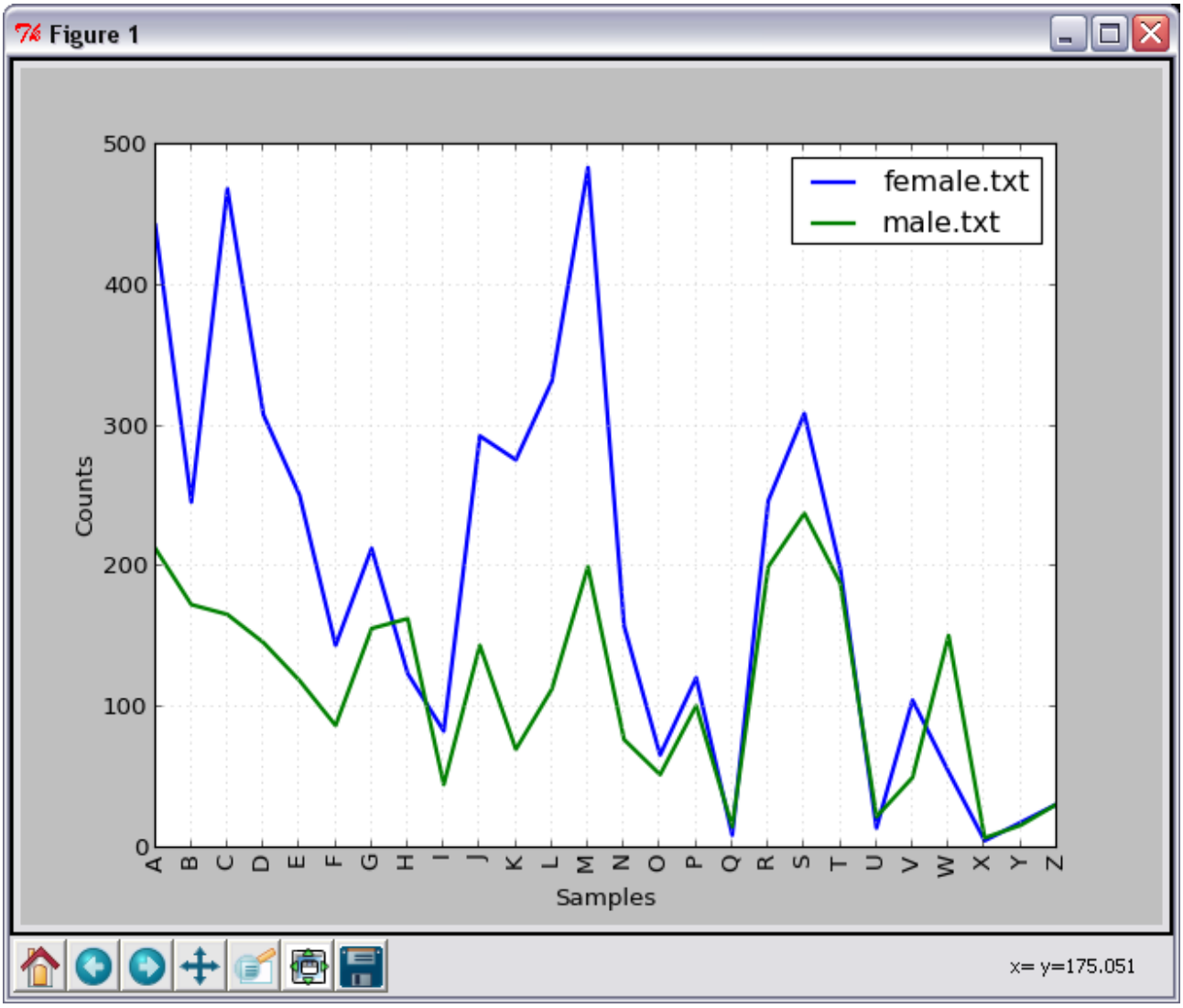
Варіант 4:

2. Використовуючи компаративний словник знайти близькі слова для німецької, італійської та англійської мов. Чи можуть отримані результати використовуватися для здійснення перекладу?

```
>>> import nltk
>>> from nltk.corpus import swadesh
>>> swadesh.fileids()
['be', 'bg', 'bs', 'ca', 'cs', 'cu', 'de', 'en', 'es', 'fr', 'hr', 'it', 'la', '
mk', 'nl', 'pl', 'pt', 'ro', 'ru', 'sk', 'sl', 'sr', 'sw', 'uk']
>>> de2en2it = swadesh.entries(['de', 'en', 'it'])
>>> de2en2it
[('ich', 'I', 'io'), ('du, Sie', 'you (singular), thou', 'tu, Lei'), ('er', 'he'
, 'lui, egli'), ('wir', 'we', 'noi'), ('ihr, Sie', 'you (plural)', 'voi'), ('sie
', 'they', 'loro, essi'), ('dieses', 'this', 'questo'), ('jenes', 'that', 'quell
o'), ('hier', 'here', 'qui, qua'), ('dort', 'there', 'l\xc3\xa0'), ('wer', 'who'
, 'chi'), ('was', 'what', 'che'), ('wo', 'where', 'dove'), ('wann', 'when', 'qua
ndo'), ('wie', 'how', 'come'), ('nicht', 'not', 'non'), ('alle', 'all', 'tutto')
, ('viele', 'many', 'molti'), ('einige', 'some', 'alcuni'), ('wenige', 'few', 'p
ochi'), ('andere', 'other', 'altro'), ('eins', 'one', 'uno'), ('zwei', 'two', 'd
ue'), ('drei', 'three', 'tre'), ('vier', 'four', 'quattro'), ('f\xc3\xbcnf', 'fi
ve', 'cinque'), ('gro\xc3\x9f', 'big', 'grande'), ('lang', 'long', 'lungo'), ('b
reit, weit', 'wide', 'largo'), ('dick', 'thick', 'spesso'), ('schwer', 'heavy',
'pesante'), ('klein', 'small', 'piccolo'), ('kurz', 'short', 'corto'), ('eng', '
narrow', 'stretto'), ('d\xc3\xbcnn', 'thin', 'sottile'), ('Frau', 'woman', 'donn
a'), ('Mann', 'man (adult male)', 'uomo'), ('Mensch', 'man (human being)', 'uomo
'), ('Kind', 'child', 'bambino'), ('Frau, Ehefrau', 'wife', 'moglie'), ('Mann, E
hemann', 'husband', 'marito'), ('Mutter', 'mother', 'madre'), ('Vater', 'father'
```

3. Побудувати умовний частотний розподіл для корпусу імен. Знайти які перші літери частіше використовуються в чоловічих та жіночих іменах.

```
import nltk
names = nltk.corpus.names
names.fileids()
cfd = nltk.ConditionalFreqDist(
    (fileid, name[0])
    for fileid in names.fileids()
    for name in names.words(fileid))
cfd.plot()
```



6. Визначити функцію `supergloss(s)` , яка буде приймати синсет s як аргумент і повертати стрічку в якій будуть поєднані всі описи всіх значень синсету s та описи всіх гіпернімів та гіпонімів s.

```
import nltk
from nltk.corpus import wordnet as wn
def supergloss(s):
    a='DEFINITION: '+str(wn.synset(s).definition) + '\n'  +'HYPERNYMS: '+
    str(wn.synset(s).hypernyms())+ '\n' + 'HYPERNYMS: '+str(wn.synset(s).hyponyms())
    return a
print supergloss('city.n.01')
```

```
>>>
DEFINITION: a large and densely populated urban area; may include several indepe
ndent administrative districts
HYPERNYMS: [Synset('municipality.n.01')]
HYPERNYMS: [Synset('state_capital.n.01'), Synset('provincial_capital.n.01'), Syn
set('national_capital.n.01')]
>>>
```

9. Модифікувати програму генерації випадкового тексту для виконання наступного: тренування програми на текстах двох різних жанрів та генерації тексту об’єднаного жанру.

```
import nltk
def generate_texts(cfdist, word, num=50):
    for i in range(num):
        print word,
        word=cfdist[word].max()

text = nltk.corpus.brown.words(categories='news')
bigrams = nltk.bigrams(text)
cfd = nltk.ConditionalFreqDist(bigrams)
print cfd['good']
generate_texts(cfd, 'good')
def generate_texts(cfdist, word, num=50):
    for i in range(num):
        print word,
        word=cfdist[word].max()

text = nltk.corpus.brown.words(categories='humor')
bigrams = nltk.bigrams(text)
cfd = nltk.ConditionalFreqDist(bigrams)
print cfd['good']
generate_texts(cfd, 'good')
def generate_texts(cfdist, word, num=50):
    for i in range(num):
        print word,
        word=cfdist[word].max()

text = nltk.corpus.brown.words(categories='humor') and nltk.corpus.brown.words(c
bigrams = nltk.bigrams(text)
cfd = nltk.ConditionalFreqDist(bigrams)
print cfd['good']
generate_texts(cfd, 'good')
```

```
>>>
<FreqDist: 'of': 3, "'": 2, ',': 2, '.': 2, 'as': 2, 'condition': 2, 'feeling': 2,
'is': 2, 'news': 2, 'a': 1, ...>
good of the first time . The President Kennedy , and the first time . The President
Kennedy , and the first time . The President Kennedy , and the first time . The Pres
ident Kennedy , and the first time . The President Kennedy , and the first time <Fre
qDist: ',': 2, 'to': 2, "'": 1, 'enough': 1, 'for': 1, 'fortune': 1, 'girl': 1, 're
ason': 1, 'times': 1>
good to the other day when the other day when the other day when the other day when
the other day when the other day when the other day when the other day when the othe
r day when the other day when the other day when the other day when <FreqDist: 'of':
3, "'": 2, ',': 2, '.': 2, 'as': 2, 'condition': 2, 'feeling': 2, 'is': 2, 'news':
2, 'a': 1, ...>
good of the first time . The President Kennedy , and the first time . The President
Kennedy , and the first time . The President Kennedy , and the first time . The Pres
ident Kennedy , and the first time . The President Kennedy , and the first time
>>>
```

13.Полісемія - це явище коли одне слово має декілька значень (іменник dog має 7 значень, кількість яких визначити можна як `len(wn.synsets('dog', 'n'))`). Знайдіть середнє значення полісемії для прислівників.

```
import nltk
from nltk.corpus import wordnet as wn
d=[]
for i in wn.all_synsets('r'):
    for j in i.lemma_names:
        d.append(j)
p=len(set(d))
print p
q=0
for i in set(d):
    q=q+len(wn.synsets(i,'r'))
print q
poly_adv = float(q)/p
print poly_adv
```

```
4481
5616
1.25329167597
```

17. Використовуючи один з методів визначення подібності слів побудуйте відсортований по спаданню список значень подібності для наступних пар слів: monk-oracle, cemetery-woodland, food-rooster, coast-hill, forest-graveyard, crane-implement, journey-car, coast-shore, asylum-madhouse, magician-wizard, midday-noon, furnace-stove, food-fruit, bird-cock.

```
import nltk
from nltk.corpus import wordnet as wn
monk = wn.synset('monk.n.01')
oracle = wn.synset('oracle.n.01')
p=monk.path_similarity(oracle)
print 'monk-oracle'
print p
cemetery = wn.synset('cemetery.n.01')
woodland = wn.synset('woodland.n.01')
p1=cemetery.path_similarity(woodland)
print 'cemetery-woodland'
print p1
food = wn.synset('food.n.01')
rooster=wn.synset('rooster.n.01')
p2=food.path_similarity(rooster)
print 'food-rooster'
print p2
coast = wn.synset('coast.n.01')
hill = wn.synset('hill.n.01')
p3=coast.path_similarity(hill)
print 'coast-hill'
print p3
forest = wn.synset('forest.n.01')
graveyard = wn.synset('graveyard.n.01')
p4=forest.path_similarity(graveyard)
print 'forest-graveyard'
print p4
crane = wn.synset('crane.v.01')
implement = wn.synset('implement.v.01')
p5=crane.path_similarity(implement)
print 'crane-implement'
print p5
journey = wn.synset('journey.n.01')
car = wn.synset('car.n.01')
p6=journey.path_similarity(car)
print 'journey-car'
print p6
coast = wn.synset('coast.n.01')
shore = wn.synset('shore.n.01')
p7=coast.path_similarity(shore)
```

```
print 'coast-shore'
print p7
asylum = wn.synset('asylum.n.01')
madhouse = wn.synset('madhouse.n.01')
p8=asylum.path_similarity(madhouse)
print 'asylum-madhouse'
print p8
magician = wn.synset('magician.n.01')
wizard = wn.synset('wizard.n.01')
p9=magician.path_similarity(wizard)
print 'magician-wizard'
print p9
midday = wn.synset('midday.n.01')
noon = wn.synset('noon.n.01')
p10=midday.path_similarity(noon)
print 'midday-noon'
print p10
furnace = wn.synset('furnace.n.01')
stove = wn.synset('stove.n.01')
p11=furnace.path_similarity(stove)
print 'furnace-stove'
print p11
food = wn.synset('food.n.01')
fruit = wn.synset('fruit.n.01')
p12=food.path_similarity(fruit)
print 'food-fruit'
print p12
bird = wn.synset('bird.n.01')
cock = wn.synset('cock.n.01')
p13=bird.path_similarity(cock)
print 'bird-cock'
print p13
s=[]
s.append(p)
s.append(p1)
s.append(p2)
s.append(p3)
s.append(p4)
s.append(p5)
s.append(p6)
```

```
s.append(p7)
s.append(p8)
s.append(p9)
s.append(p10)
s.append(p11)
s.append(p12)
s.append(p13)
print s
s.sort()
print 'sortovane po zrostanniu'
print s
s.reverse()
print 'sortovane po spadanniu'
print s
```

```
>>>
monk-oracle
0.125
cemetery-woodland
0.111111111111
food-rooster
0.0625
coast-hill
0.2
forest-graveyard
0.0714285714286
crane-implement
0.125
journey-car
0.05
coast-shore
0.5
asylum-madhouse
0.125
magician-wizard
0.166666666667
midday-noon
1.0
furnace-stove
0.0769230769231
food-fruit
0.0909090909091
bird-cock
0.0625
[0.125, 0.1111111111111111, 0.0625, 0.2, 0.07142857142857142, 0.125, 0.05, 0.5,
0.125, 0.16666666666666666, 1.0, 0.07692307692307693, 0.09090909090909091, 0.062
5]
sortovane po zrostanniu
[0.05, 0.0625, 0.0625, 0.07142857142857142, 0.07692307692307693, 0.0909090909090
9091, 0.1111111111111111, 0.125, 0.125, 0.125, 0.16666666666666666, 0.2, 0.5, 1.
0]
sortovane po spadanniu
[1.0, 0.5, 0.2, 0.16666666666666666, 0.125, 0.125, 0.125, 0.1111111111111111, 0.
09090909090909091, 0.07692307692307693, 0.07142857142857142, 0.0625, 0.0625, 0.0
5]
>>>
```

Висновок: я ознайомилась з основами програмування на мові Python.Освоїла методів доступу та роботи з лексичним ресурсами. Та опанувала ази роботи з семантичним словником англійської мови WordNet.