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
Гарбуз Л.В.
Прийняв:
викладач
Дупак Б.П.

Львів 2015

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

Тексти програм на мові *Python*.

Варіант – 3

1.Дослідити зв'язки голонім-меронім для іменників. Знайти іменники для демонстрації наступних зв'язків: member_meronyms(), part_meronyms(), substance_meronyms(), member_holonyms(), part_holonyms(), та substance_holonyms().

```
import nltk
from nltk.corpus import wordnet as wn
for w in ["dog", "water", "car"]:
    s = wn.synset(w+ ".n.01")
    if (s is not None):
        print "member_meronym(" + w + ")=", s.member_meronyms()
        print "part_meronym(" + w + ")=", s.part_meronyms()
        print "substance_meronym(" + w + ")=", s.substance_meronyms()
        print "member_holonym(" + w + ")=", s.member_holonyms()
        print "part_holonym(" + w + ")=", s.part_holonyms()
        print "substance_holonym(" + w + ")=", s.substance_holonyms()
```

```
member_meronym(dog)= []
part_meronym(dog)= [Synset('flag.n.07')]
substance_meronym(dog)= []
member_holonym(dog)= [Synset('pack.n.06'), Synset('canis.n.01')]
part_holonym(dog)= []
substance_holonym(dog)= []
member_meronym(water)= []
part_meronym(water)= []
substance_meronym(water)= [Synset('oxygen.n.01'), Synset('hydrogen.n.01')]
member_holonym(water)= []
part_holonym(water)= []
substance_holonym(water)= [Synset('tear.n.01'), Synset('perspiration.n.01'), Synset('
snowflake.n.01'), Synset('ice_crystal.n.01'), Synset('ice.n.01'), Synset('body_of_wat
er.n.01')]
member_meronym(car)= []
part_meronym(car)= [Synset('gasoline_engine.n.01'), Synset('car_mirror.n.01'), Synset
('third_gear.n.01'), Synset('hood.n.09'), Synset('automobile_engine.n.01'), Synset('g
rille.n.02'), Synset('automobile_horn.n.01'), Synset('rear_window.n.01'), Synset('car
_window.n.01'), Synset('floorboard.n.02'), Synset('accelerator.n.01'), Synset('tail_f
in.n.02'), Synset('window.n.02'), Synset('reverse.n.02'), Synset('glove_compartment.n
.01'), Synset('first_gear.n.01'), Synset('buffer.n.06'), Synset('car_door.n.01'), Syn
set('roof.n.02'), Synset('auto_accessory.n.01'), Synset('car_seat.n.01'), Synset('hig
h_gear.n.01'), Synset('fender.n.01'), Synset('stabilizer_bar.n.01'), Synset('bumper.n
.02'), Synset('air_bag.n.01'), Synset('sunroof.n.01'), Synset('luggage_compartment.n.
01'), Synset('running_board.n.01')]
substance_meronym(car)= []
member_holonym(car)= []
part_holonym(car)= []
substance_holonym(car)= []
```

4.Здійснити аналіз словника вимов. Знайти скільки різних слів він містить. Який відсоток слів з цього словника можуть мати різну вимову?

```
import nltk
from nltk.corpus import cmudict
entries=cmudict.entries()
len(entries)
words=[w for w, pron in entries]
len(words)
fdist=nltk.FreqDist(words)
s=[w for w in words if fdist[w]>1]
percent = (len(s)* 100)/len(words)
print len(entries)
print(percent)
```

```
>>>
133737
14
>>>
```

5. Який відсоток синсетів іменників не мають гіпонімів? До всіх синсетів можна доступитися за допомогою wn.all_synsets('n').

```
>>> from nltk.corpus import wordnet as wn
>>> all = wn.all_synsets('n')
>>> l_all = 0
>>> awh = 0
>>> for syn in all:
        if syn.hyponyms() == []:
                awh += 1
        l_all += 1


>>> l_all
82115
>>> awh
65422
>>> from __future__ import division
>>> awh/l_all*100
79.67119283931072
```

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='adventure')
bigrams = nltk.bigrams(text)
cfd = nltk.ConditionalFreqDist(bigrams)
print cfd['time']
generate_texts(cfd, 'time')
def generate_texts(cfdist, word, num=50):
    for i in range(num):
        print word,
        word=cfdist[word].max()

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

```
>>>
<FreqDist: '.': 19, ',': 16, 'to': 16, 'he': 8, 'I': 5, 'before': 4, 'for': 4, "
'''": 3, 'of': 3, 'that': 3, ...>
time . `` I was a little girls were the door . `` I was a little girls were the
door . `` I was a little girls were the door . `` I was a little girls were the
door . `` I was a little girls were the <FreqDist: ',': 20, 'to': 15, 'he': 9, '
.': 4, 'I': 4, "'": 3, 'in': 3, 'of': 3, 'and': 2, 'as': 2, ...>
time , and the same time , and the same time , and the same time , and the same
time , and the same time , and the same time , and the same time , and the same
time , and the same time , and the same <FreqDist: ',': 36, 'to': 31, '.': 23, '
he': 17, 'I': 9, "'": 6, 'of': 6, 'before': 5, 'for': 5, 'and': 4, ...>
time , and the door . `` I was a little girls were the door . `` I was a little
girls were the door . `` I was a little girls were the door . `` I was a little
girls were the door . `` I was a
```

12. 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 дієслів.

```
import nltk
from nltk.corpus import wordnet as wn
d=[]
for i in wn.all_synsets('v'):
    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,'v'))
print q
poly_v = float(q)/p
print poly_v


>>>
11531
25214
2.18662735235
>>>
```

16.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, shore-woodland, monk-slave, coast-forest, lad-wizard, chord-smile, glass-magician, rooster-voyage, noon-string.

```
from nltk.corpus import wordnet as wn
monk = wn.synset('monk.n.01')
oracle = wn.synset('oracle.n.01')
a=monk.path_similarity(oracle)
print 'monk-oracle'
print a
cemetery = wn.synset('cemetery.n.01')
woodland = wn.synset('woodland.n.01')
b=cemetery.path_similarity(woodland)
print 'cemetery-woodland'
print b
food = wn.synset('food.n.01')
rooster=wn.synset('rooster.n.01')
c=food.path_similarity(rooster)
print 'food-rooster'
print c
coast = wn.synset('coast.n.01')
hill = wn.synset('hill.n.01')
d=coast.path_similarity(hill)
print 'coast-hill'
print d
forest = wn.synset('forest.n.01')
graveyard = wn.synset('graveyard.n.01')
e=forest.path_similarity(graveyard)
print 'forest-graveyard'
print e
shore = wn.synset('crane.v.01')
woodland = wn.synset('implement.v.01')
f=shore.path_similarity(woodland)
print 'shore-woodland'
print f
monk = wn.synset('journey.n.01')
slave = wn.synset('car.n.01')
g=monk.path_similarity(slave)
print 'monk-slave'
print g
coast = wn.synset('coast.n.01')
forest = wn.synset('shore.n.01')
h=coast.path_similarity(forest)
print 'coast-forest'
```

```
j=chord.path_similarity(smile)
print 'chord-smile'
print j
glass = wn.synset('midday.n.01')
magician = wn.synset('noon.n.01')
k=glass.path_similarity(magician)
print 'glass-magician'
print k
rooster = wn.synset('furnace.n.01')
voyage = wn.synset('stove.n.01')
l=rooster.path_similarity(voyage)
print 'rooster-voyage'
print l
noon = wn.synset('food.n.01')
string = wn.synset('fruit.n.01')
m=noon.path_similarity(string)
print 'noon-string'
print m
s=[]
s.append(a)
s.append(b)
s.append(c)
s.append(d)
s.append(e)
s.append(f)
s.append(g)
s.append(h)
s.append(i)
s.append(j)
s.append(k)
s.append(l)
s.append(m)
print s
s.sort()
print 'po zrostanniu'
print s
s.reverse()
print 'po spadanniu'
print s
```

```
>>>
monk-oracle
0.125
cemetery-woodland
0.111111111111
food-rooster
0.0625
coast-hill
0.2
forest-graveyard
0.0714285714286
shore-woodland
0.125
monk-slave
0.05
coast-forest
0.5
lad-wizard
0.125
chord-smile
0.166666666667
glass-magician
1.0
rooster-voyage
0.0769230769231
noon-string
0.0909090909091
[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]
po zrostanniu
[0.05, 0.0625, 0.07142857142857142, 0.07692307692307693, 0.09090909090909091, 0.
1111111111111111, 0.125, 0.125, 0.125, 0.16666666666666666, 0.2, 0.5, 1.0]
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, 0.05]
```

Висновок: на цій лабораторній роботі я навчилася програмувати на мові Python. Також ознайомилася з методами доступу та роботи з лексичним ресурсами і семантичним словником англійської мови WordNet.