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

Кафедра “Системи автоматизованого проектування”

Звіт

до лабораторної роботи №9

на тему: ***ВИВЧЕННЯ БІБЛІОТЕКИ ПРИКЛАДНИХ ПРОГРАМ NLTK, ДЛЯ ОПРАЦЮВАННЯ ТЕКСТІВ ПРИРОДНОЮ МОВОЮ.***

АВТОМАТИЧНИЙ МОРФОЛОГІЧНИЙ АНАЛІЗ (частина1).

з дисципліни “Комп’ютерна лінгвістика”

Виконала:

студентка групи ПРЛм-11

Звари О. І

Прийняв:

Дупак Б.П

Львів-2015

МЕТА РОБОТА

- Вивчення основ програмування на мові *Python*.
- Ознайомлення з автоматичним морфологічним аналізом в NLTK.

1. Токенізувати та здійснити морфологічний аналіз наступного речення: *They wind back the clock, while we chase after the wind.* Які відмінності у вимові слів пов'язані з їх морфологічними характеристиками.

```
import nltk
text = nltk.word_tokenize("They wind back the clock, while we chase after the wi
print nltk.pos_tag(text)
 [('They', 'PRP'), ('wind', 'VBP'), ('back', 'RB'), ('the', 'DT'), ('clock', 'NN'
 ), (',', ','), ('while', 'IN'), ('we', 'PRP'), ('chase', 'VBP'), ('after', 'IN')
 , ('the', 'DT'), ('wind', 'NN')]
```

2. Опрацювати всі приклади з методичних вказівок по роботі зі словниками. Що станеться, якщо доступитися до неіснуючого запису звичайного словника та словника по замовчуванню?

```
#>>> pos={}
#>>> pos
#{}
#>>> pos['colorless']='ADJ'
#>>> pos
#{'colorless': 'ADJ'}
#>>> pos['ideas']='N'
#>>> pos['sleep']='V'
#>>> pos['furiously']='ADV'
#>>> pos
#{'furiously': 'ADV', 'sleep': 'V', 'ideas': 'N', 'colorless': 'ADJ'}
#>>> pos['ideas']
#'N'
#>>> pos['colorless']
#'ADJ'
#>>> pos['green']
#Traceback (most recent call last):
#  File "<pyshell#28>", line 1, in <module>
#    pos['green']
#KeyError: 'green'
#>>> list(pos)
#['furiously', 'sleep', 'ideas', 'colorless']
#>>> sorted(pos)
#['colorless', 'furiously', 'ideas', 'sleep']
#>>> [w for w in pos if w.endswith('s')]
#['ideas', 'colorless']
#>>> for word in sorted(pos):
#         print word+':', pos[word]
#colorless: ADJ
#furiously: ADV
#ideas: N
#sleep: V
#>>> pos.keys()
#['furiously', 'sleep', 'ideas', 'colorless']
#>>> pos.values()
```

```
#['ADV', 'V', 'N', 'ADJ']
#>>> pos.items()
#[('furiously', 'ADV'), ('sleep', 'V'), ('ideas', 'N'), ('colorless', 'ADJ')]
#>>> for key, val in sorted(pos.items()):
#         print key+':', val
#colorless: ADJ
#furiously: ADV
#ideas: N
#sleep: V
#>>>
#>>> pos={'colorless':'ADJ', 'ideas':'N', 'sleep':'V', 'furiously':'ADV'}
#>>> pos=dict(colorelss='ADJ', ideas='N', sleep='V', furiously='ADV')
#>>> frequency=nltk.defaultdict(int)
#>>> frequency['colorless']=4
#>>> frequency['ideas']
#0
#>>> pos=nltk.defaultdict(list)
#>>> pos['sleep']=['N','V']
#>>> pos['ideas']
#[]
#>>> pos['ideas']
#[]
#>>> pos=nltk.defaultdict(lambda:'N')
#>>> pos['colorless']='ADJ'
#>>> pos['blog']
#'N'
#>>> pos.items()
#[('blog', 'N'), ('colorless', 'ADJ')]
#>>> counts=nltk.defaultdict(int)
#>>> from nltk.corpus import brown
#>>> for (word, tag) in brown.tagged_words(categories='news'):
#         counts[tag]+=1
#>>> counts['N']
#0
#>>> list(counts)
#['BE', 'BEZ-HL', 'NP$', 'WQL', 'AT-TL', 'BEDZ*', 'WDT', 'JJ', 'NR-HL', 'AP$', 'RP',
'WPS+BEZ', 'JJ-NC', '(', 'PPSS+BER', ',', 'VBN-TL-HL', 'HVD-HL', 'PPSS+BEM', 'NPS-HL',
'RB', 'FW-PP$-NC', 'JJ-HL', 'NNS', 'WRB', 'MD-TL', 'NN-NC', 'DOD*', 'NN$', 'PPLS', ')-HL',
'BEZ*', 'RB-HL', 'NNS$', 'NPS-TL', 'NNS-HL', 'FW-IN+NN-TL', '--', 'BER-TL', 'OD', 'PP$$',
'CC-TL', 'FW-NN-TL', 'NP-TL-HL', 'AP-TL', 'PPSS+MD', 'FW-JJ', 'FW-DT', 'BER*', 'FW-
WDT', 'NPS', 'DTI', 'BEN', 'BEM', 'EX+BEZ', 'HV', 'BEG', 'BED', 'HVD', 'BEZ', 'DTX', 'FW-
VB-NC', 'VBZ', 'DTS', 'RB-TL', 'VB-TL', 'NNS-TL', 'FW-CC', 'CS-HL', 'NP$-TL', 'FW-CD',
'ABN-HL', 'IN-HL', 'JJT-HL', 'BED*', 'BEDZ', 'NN-TL-HL', 'PN', 'JJR-HL', 'FW-AT-TL',
'PPSS+HVD', 'VBD-HL', 'MD-HL', 'NNS-TL-HL', 'DTI-HL', 'EX', 'VBN-HL', 'NNS$-HL',
'PPSS-HL', 'MD', 'BE-HL', 'TO-TL', 'NN-HL', 'VBZ-HL', 'NR$-TL', 'DT$', 'WP$', 'N', 'MD+HV',
'TO-HL', 'PPS+BEZ', 'DT-HL', 'CD$', 'VBG', 'VBD', 'VBN-TL', 'DOZ*', 'VBN', 'DOD', 'UH-TL',
'DOZ', 'NR-TL', 'AP-HL', 'AT-HL', '.', 'FW-AT', 'NN', '(-HL', 'MD*-HL', '*', 'WPS', 'WPO', 'FW-
NNS', 'NP', 'JJR-NC', 'NR', ':', 'BER-HL', 'MD*', '``', ':-HL', 'RP-HL', 'CC', 'PP$-TL',
'WDT+BEZ', 'CD-HL', 'NPS$-TL', 'CD', 'DT+BEZ', ',-HL', 'OD-HL', 'PPS+MD', 'CS', 'NN$-HL',
'NP-TL', 'QL-TL', 'DO*', 'PPS+BEZ-HL', 'VB-HL', 'DO-HL', 'HVN', 'JJT', 'JJS', 'JJR', 'HVG',
'HVZ', 'PN+HVZ', 'NNS$-TL', 'CC-HL', 'JJ-TL', 'HVZ*', 'VBG-TL', 'DO', 'FW-JJ-TL', 'FW-*',
'NP+BEZ', 'NP-HL', 'NPS$', 'NN-TL', 'PPSS', 'NR$', """, 'BER', 'FW-VB', 'PN-HL', 'CD-TL',
```

'BEDZ-HL', 'DT', 'VBD-TL', 'PN$', 'VB+PPO', ')', 'VBG-HL', 'PPO', 'PPL', 'PPS', 'TO', 'RB$', 'FW-IN+NN', 'UH', 'VB', 'OD-TL', 'FW-IN', 'PP$', 'RBT', 'ABL', 'RBR', 'ABN', 'AP', 'PPSS+HV', 'AT', 'JJS-TL', 'IN', 'ABX', '*-HL', 'FW-AT-HL', 'HVD*', '"', 'JJR-TL', 'RB+BEZ', 'NN$-TL', 'FW-IN-TL', 'QLP', 'IN-TL', 'FW-NN', 'FW-IN+AT-TL', 'PPS+HVZ', 'QL', '.-HL']
#>>> >>> from operator import itemgetter
#>>> sorted(counts.items(), key=itemgetter(1), reverse=True)
#[('NN', 13162), ('IN', 10616), ('AT', 8893), ('NP', 6866), (',', 5133), ('NNS', 5066), ('.', 4452), ('JJ', 4392), ('CC', 2664), ('VBD', 2524), ('NN-TL', 2486), ('VB', 2440), ('VBN', 2269), ('RB', 2166), ('CD', 2020), ('CS', 1509), ('VBG', 1398), ('TO', 1237), ('PPS', 1056), ('PP$', 1051), ('MD', 1031), ('AP', 923), ('NP-TL', 741), ('``', 732), ('BEZ', 730), ('BEDZ', 716), ('"', 702), ('JJ-TL', 689), ('PPSS', 602), ('DT', 589), ('BE', 525), ('VBZ', 519), ('NR', 495), ('RP', 482), ('QL', 468), ('PPO', 412), ('WPS', 395), ('NNS-TL', 344), ('WDT', 343), ('WRB', 328), ('BER', 328), ('OD', 309), ('HVZ', 301), ('--', 300), ('NP$', 279), ('HV', 265), ('HVD', 262), ('*', 256), ('BED', 252), ('NPS', 215), ('BEN', 212), ('NN$', 210), ('DTI', 205), ('NP-HL', 186), ('ABN', 183), ('NN-HL', 171), ('IN-TL', 164), ('EX', 161), (')', 151), ('(', 148), ('JJR', 145), (':', 137), ('DTS', 136), ('JJT', 100), ('CD-TL', 96), ('NNS-HL', 92), ('PN', 89), ('RBR', 88), ('VBN-TL', 87), ('ABX', 73), ('NN$-TL', 69), ('IN-HL', 65), ('DOD', 64), ('DO', 63), ('BEG', 57), (',-HL', 55), ('VBN-HL', 53), ('AT-TL', 50), ('NNS$', 50), ('CD-HL', 50), ('JJS', 49), ('JJ-HL', 46), ('CC-TL', 46), ('"', 46), ('MD*', 43), ('VBZ-HL', 39), ('PPL', 36), ('PPSS+MD', 31), ('PPS+BEZ', 31), ('OD-TL', 30), ('DOZ', 26), ('VB-HL', 25), ('NR$', 24), ('WP$', 22), ('FW-NN', 22), ('PPLS', 21), ('ABL', 21), ('PPSS+BER', 20), (')-HL', 20), ('(-HL', 20), ('NNS$-TL', 20), ('.-HL', 20), ('PPSS+HV', 19), ('PPSS+BEM', 18), ('HVN', 18), ('DO*', 17), ('NPS$', 17), ('FW-NN-TL', 16), ('DOD*', 15), ('RB-HL', 15), ('NPS-TL', 15), ('VBG-TL', 15), ('NR-TL', 14), ('AT-HL', 14), ('HVG', 14), ('FW-IN', 14), ('BEM', 13), ('DOZ*', 13), ('NN-TL-HL', 12), (':-HL', 12), ('DT+BEZ', 12), ('FW-JJ-TL', 12), ('VBG-HL', 12), ('UH', 12), ('QLP', 12), ('NP$-TL', 11), ('WPO', 9), ('BEZ*', 8), ('DTX', 8), ('RB-TL', 8), ('VB-TL', 8), ('PPS+MD', 8), ('AP-HL', 7), ('CC-HL', 7), ('FW-AT-TL', 6), ('VBD-HL', 6), ('TO-HL', 6), ('MD-HL', 5), ('RBT', 5), ('BER*', 4), ('JJR-HL', 4), ('RP-HL', 4), ('JJR-TL', 4), ('PPS+HVZ', 4), ('BEZ-HL', 3), ('BEDZ*', 3), ('NPS-HL', 3), ('NN-NC', 3), ('PP$$', 3), ('FW-JJ', 3), ('FW-AT', 3), ('NR-HL', 2), ('WPS+BEZ', 2), ('EX+BEZ', 2), ('JJT-HL', 2), ('DTI-HL', 2), ('NNS$-HL', 2), ('CD$', 2), ('FW-NNS', 2), ('NN$-HL', 2), ('FW-IN+NN', 2), ('JJS-TL', 2), ('HVD*', 2), ('WQL', 1), ('AP$', 1), ('JJ-NC', 1), ('VBN-TL-HL', 1), ('HVD-HL', 1), ('FW-PP$-NC', 1), ('MD-TL', 1), ('FW-IN+NN-TL', 1), ('BER-TL', 1), ('NP-TL-HL', 1), ('AP-TL', 1), ('FW-DT', 1), ('FW-WDT', 1), ('FW-VB-NC', 1), ('FW-CC', 1), ('CS-HL', 1), ('FW-CD', 1), ('ABN-HL', 1), ('BED*', 1), ('PPSS+HVD', 1), ('NNS-TL-HL', 1), ('PPSS-HL', 1), ('BE-HL', 1), ('TO-TL', 1), ('NR$-TL', 1), ('DT$', 1), ('MD+HV', 1), ('DT-HL', 1), ('UH-TL', 1), ('MD*-HL', 1), ('JJR-NC', 1), ('BER-HL', 1), ('PP$-TL', 1), ('WDT+BEZ', 1), ('NPS$-TL', 1), ('OD-HL', 1), ('QL-TL', 1), ('PPS+BEZ-HL', 1), ('DO-HL', 1), ('PN+HVZ', 1), ('HVZ*', 1), ('FW-*', 1), ('NP+BEZ', 1), ('FW-VB', 1), ('PN-HL', 1), ('BEDZ-HL', 1), ('VBD-TL', 1), ('PN$', 1), ('VB+PPO', 1), ('RB$', 1), ('*-HL', 1), ('FW-AT-HL', 1), ('RB+BEZ', 1), ('FW-IN-TL', 1), ('FW-IN+AT-TL', 1), ('N', 0)]
#>>> [t for t, c in sorted(counts.items(), key=itemgetter(1), reverse=True)]
#['NN', 'IN', 'AT', 'NP', ',', 'NNS', '.', 'JJ', 'CC', 'VBD', 'NN-TL', 'VB', 'VBN', 'RB', 'CD', 'CS', 'VBG', 'TO', 'PPS', 'PP$', 'MD', 'AP', 'NP-TL', '``', 'BEZ', 'BEDZ', '"', 'JJ-TL', 'PPSS', 'DT', 'BE', 'VBZ', 'NR', 'RP', 'QL', 'PPO', 'WPS', 'NNS-TL', 'WDT', 'WRB', 'BER', 'OD', 'HVZ', '--', 'NP$', 'HV', 'HVD', '*', 'BED', 'NPS', 'BEN', 'NN$', 'DTI', 'NP-HL', 'ABN', 'NN-HL', 'IN-TL', 'EX', ')', '(', 'JJR', ':', 'DTS', 'JJT', 'CD-TL', 'NNS-HL', 'PN', 'RBR', 'VBN-TL', 'ABX', 'NN$-TL', 'IN-HL', 'DOD', 'DO', 'BEG', ',-HL', 'VBN-HL', 'AT-TL', 'NNS$', 'CD-HL', 'JJS', 'JJ-HL', 'CC-TL', '"', 'MD*', 'VBZ-HL', 'PPL', 'PPSS+MD', 'PPS+BEZ', 'OD-TL', 'DOZ', 'VB-HL', 'NR$', 'WP$', 'FW-NN', 'PPLS', 'ABL', 'PPSS+BER', ')-HL', '(-HL', 'NNS$-TL', '.-HL', 'PPSS+HV', 'PPSS+BEM', 'HVN', 'DO*', 'NPS$', 'FW-NN-TL', 'DOD*', 'RB-HL', 'NPS-TL', 'VBG-TL', 'NR-TL', 'AT-HL', 'HVG', 'FW-IN', 'BEM', 'DOZ*', 'NN-TL-HL', ':-HL', 'DT+BEZ', 'FW-JJ-TL', 'VBG-HL', 'UH', 'QLP', 'NP$-TL', 'WPO', 'BEZ*', 'DTX', 'RB-TL', 'VB-TL', 'PPS+MD', 'AP-HL', 'CC-HL', 'FW-AT-TL', 'VBD-HL', 'TO-HL', 'MD-HL', 'RBT', 'BER*', 'JJR-HL', 'RP-HL', 'JJR-TL', 'PPS+HVZ',

'BEZ-HL', 'BEDZ*', 'NPS-HL', 'NN-NC', 'PP$$', 'FW-JJ', 'FW-AT', 'NR-HL', 'WPS+BEZ', 'EX+BEZ', 'JJT-HL', 'DTI-HL', 'NNS$-HL', 'CD$', 'FW-NNS', 'NN$-HL', 'FW-IN+NN', 'JJS-TL', 'HVD*', 'WQL', 'AP$', 'JJ-NC', 'VBN-TL-HL', 'HVD-HL', 'FW-PP$-NC', 'MD-TL', 'FW-IN+NN-TL', 'BER-TL', 'NP-TL-HL', 'AP-TL', 'FW-DT', 'FW-WDT', 'FW-VB-NC', 'FW-CC', 'CS-HL', 'FW-CD', 'ABN-HL', 'BED*', 'PPSS+HVD', 'NNS-TL-HL', 'PPSS-HL', 'BE-HL', 'TO-TL', 'NR$-TL', 'DT$', 'MD+HV', 'DT-HL', 'UH-TL', 'MD*-HL', 'JJR-NC', 'BER-HL', 'PP$-TL', 'WDT+BEZ', 'NPS$-TL', 'OD-HL', 'QL-TL', 'PPS+BEZ-HL', 'DO-HL', 'PN+HVZ', 'HVZ*', 'FW-*', 'NP+BEZ', 'FW-VB', 'PN-HL', 'BEDZ-HL', 'VBD-TL', 'PN$', 'VB+PPO', 'RB$', '*-HL', 'FW-AT-HL', 'RB+BEZ', 'FW-IN-TL', 'FW-IN+AT-TL', 'N']
#>>> pair=('NP', 8336)
#>>> pair[1]
#8336
#>>> itemgetter(1)(pair)
#8336
#>>> last_letters=nltk.defaultdict(list)
#>>> for word in words:
key=word[-2:]
last_letters[key].append(word)
#>>> last_letters['ly']
#['abactinally', 'abandonedly', 'abasedly', 'abashedly', 'abashlessly',...]
#>>> anagrams=nltk.defaultdict(list)
#>>> words=nltk.corpus.words.words('en')
#>>> for word in words:
key=''.join(sorted(word))
anagrams[key].append(word)
#>>> anagrams['aeilnrt']
#['entrail', 'latrine', 'ratline', 'reliant', 'retinal', 'trenail']
#>>> pos=nltk.defaultdict(lambda: nltk.defaultdict(int))
#>>> brown_news_tagged=brown.tagged_words(categories='news', simplify_tags=True)
#>>> for ((w1, t1), (w2, t2)) in nltk.ibigrams(brown_news_tagged):
pos[(t1, w2)][t2]+=1
#>>> pos[('DET', 'right')]
#defaultdict(<type 'int'>, {'ADV': 3, 'ADJ': 9, 'N': 4})
#>>> counts=nltk.defaultdict(int)
#>>> for word in nltk.corpus.gutenberg.words('milton-paradise.txt'):
counts[word]+=1
#>>> [key for (key, value) in counts.items() if value ==32]
#['brought', 'Him', 'virtue', 'Against', 'There', 'thine', 'King', 'mortal', 'every', 'been']
#>>> pos={'colorless':'ADJ', 'ideas':'N', 'sleep':'V', 'furiously':'ADV'}
#>>> pos2=dict((value,key) for (key, value) in pos.items())
#>>> pos2['N']
#'ideas'
#print 'Yakshcho dostupatys do neisnuiuchoho zapysu zvychainoho slovnyka, to pomulka, iakshcho do slovnyka po zamovchuvanniu, to slovo avtomatychno dodaietsia do slovnyka'
#5. Створити два словники Що станеться зі словниками після виконання команди d1.update(d2)
#d1={'I':'PRO', 'go':'V', 'to':'TO', 'university':'N', 'every':'DET', 'day':'N'}
#d2={'This':'DET', 'is':'V', 'autumn':'N'}
#d1.update(d2)
#print d1
#print d2
#print "d1 ob'iednurtsia z d2, d2 zalyshaietsia nezminnym"

4. Спробуйте видалити запис зі словника d, використовуючи del d['abc'].

```
>>> d = {'abc': 'N', 'colorless': 'ADJ', 'ideas': 'N', 'sleep': 'V', 'furiously'
: 'ADV'}
>>> del d['abc']
>>> d
{'sleep': 'V', 'ideas': 'N', 'furiously': 'ADV', 'colorless': 'ADJ'}
>>>
```

7. Використовуючи sorted() та set() отримайте відсортований список всіх тегів корпуса Brown без їх дублювання.

```
import nltk, re, pprint
from nltk.corpus import brown
def tag_list (tagged_words):
    tags=[]
    for (w,t) in tagged_words:
        tags.append(t)
    tags_list=set(tags)
    return pprint.pprint (sorted(tags_list))
print tag_list(nltk.corpus.brown.tagged_words())
```

```
["''",
 "'''",
 '(',
 '(-HL',
 ')',
 ')-HL',
 '*',
 '*-HL',
 '*-NC',
 '*-TL',
 ',',
 ',-HL',
 ',-NC',
 ',-TL',
 '--',
 '---HL',
 '.',
 '.-HL',
 '.-NC',
 '.-TL',
 ':',
 ':-HL',
```

11. Напишіть програму, яка обробить *Brown Corpus* і допоможе відповісти на наступне запитання: які теги для маркування іменників найчастіше використовуються і що вони означають.

```
import nltk
brown=nltk.corpus.brown.tagged_words()
def findtag(tags, text):
    tag_list=[]
    for tag in text:
        if tag[1].startswith(tags):
            tag_list+=[tag[1]]
    fd=nltk.FreqDist(tag_list)
    print 'Noun tags', (tags,len(set(tag_list)))
    print 'All:', fd.keys()
    return 'The most frequent:', fd.keys()[:10]
print findtag('N',brown)
```

```
Noun tags ('N', 60)
All: ['NN', 'NNS', 'NP', 'NN-TL', 'NP-TL', 'NP$', 'NNS-TL', 'NR', 'NN$', 'NN-HL'
, 'NPS', 'NNS-HL', 'NP-HL', 'NN$-TL', 'NR-TL', 'NNS$', 'NIL', 'NP$-TL', 'NN-TL-H
L', 'NN-NC', 'NNS$-TL', 'NPS-TL', 'NR$', 'NPS$', 'NN+BEZ', 'NNS-NC', 'NP+BEZ', '
NN$-HL', 'NRS', 'NP-NC', 'NNS-TL-HL', 'NR$-TL', 'NR-HL', 'NP$-HL', 'NPS-HL', 'NP
-TL-HL', 'NP+HVZ', 'NN+HVZ', 'NR-TL-HL', 'NNS$-HL', 'NR-NC', 'NN-TL-NC', 'NNS-TL
-NC', 'NP+BEZ-NC', 'NPS$-TL', 'NN+BEZ-TL', 'NN+MD', 'NNS$-NC', 'NNS+MD', 'NP+MD'
, 'NPS-NC', 'NN+HVD-TL', 'NN+HVZ-TL', 'NN+IN', 'NN+NN-NC', 'NNS$-TL-HL', 'NP+HVZ
-NC', 'NPS$-HL', 'NR+MD', 'NRS-TL']
('The most frequent:', ['NN', 'NNS', 'NP', 'NN-TL', 'NP-TL', 'NP$', 'NNS-TL', 'N
R', 'NN$', 'NN-HL'])
```

12. Напишіть програму для збору статистичних даних по розмічених корпусах і відповіді на наступне запитання: який відсоток типів слів (types) завжди маркуються тими самими тегами.

```
import nltk
from nltk.corpus import brown
brown_news_tagged = brown.tagged_words(categories='news', simplify_tags=True)
data = nltk.ConditionalFreqDist((word.lower(),tag) for (word,tag) in brown_news_tag
result=[]
for word in data.conditions():
    if len(data[word])==1:
        tags=data[word].keys()
        result.append(word)
print len(result)
print len(brown_news_tagged)
print len(result)*100/len(brown_news_tagged)
```

```
>>>
11995
100554
11
>>>
```

19. Напишіть програми для знаходження слів та словосполучень згідно відповідних їм тегів для відповіді на наступне питання: які послідовності слів маркуються як IN + DET + NN.

```
import nltk
from nltk.corpus import brown
def process(sentence):
    for (w1,t1), (w2,t2), (w3,t3) in nltk.trigrams(sentence):
        if (t1 == 'IN' and t2 == 'DT' and t3 =='NN'):
            return w1, w2, w3
for tagged_sent in brown.tagged_sents():
    print process(tagged_sent)
```

```
None
('on', 'this', 'side')
None
None
None
None
('In', 'this', 'manner')
None
None
None
None
None
None
None
None
None
None
None
None
None
None
('of', 'this', 'veteran')
None
None
```

21. Написати програму побудови словника, записами якого будуть набори словників. Використовуючи створений словник, збережіть у ньому набори можливих тегів, які зустрічаються після заданого слова з певним тегом, наприклад $word_i \rightarrow tag_i \rightarrow tag_{i+1}$.

```
import nltk
from nltk.corpus import brown
pos = nltk.defaultdict(lambda: nltk.defaultdict(int))
brown_news_tagged = brown.tagged_words(categories='news', simplify_tags=True)
for ((w1, t1), (w2, t2)) in nltk.ibigrams(brown_news_tagged):
    pos[(w1,t1)][t2] += 1
print pos[('this','DET')]
```

```
>>>
defaultdict(<type 'int'>, {'ADV': 2, '': 1, '``': 1, ':': 2, 'VD': 1, 'VG': 2, '
ADJ': 18, 'DET': 1, ',': 1, 'N': 176, "''": 3, 'P': 3, 'NUM': 3, 'V': 12, 'NP':
3, 'VBZ': 3, '.': 3, 'CNJ': 4, 'MOD': 11})
>>>
```

Висновок: під час виконання цієї лабораторної роботи я ознайомилась з автоматичним морфологічним аналізом в NLTK.