МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ “ЛЬВІВСЬКА ПОЛІТЕХНІКА”

Лабораторна робота № 9

ВИВЧЕННЯ БІБЛІОТЕКИ ПРИКЛАДНИХ ПРОГРАМ NLTK, ДЛЯ ОПРАЦЮВАННЯ ТЕКСТІВ ПРИРОДНОЮ МОВОЮ. АВТОМАТИЧНИЙ МОРФОЛОГІЧНИЙ АНАЛІЗ (частина1).

Виконала:
студентка групи ПРЛм-12
Іваськів М.Є.

Прийняв:
Дупак Б.П.

Львів 2015

МЕТА РОБОТИ

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

КОРОТКІ ТЕОРЕТИЧНІ ВІДОМОСТІ

Процес класифікації слів за їх приналежністю до частини мови і їх відповідне маркування називається морфологічним аналізом (tagging, POS tagging). В загальному для назв цих груп слів вживаються терміни – класи слів, лексичні категорії, частини мови.

Перелік тегів, який використовується для цієї специфічної задачі називається набором тегів. Автоматичний морфологічний аналіз це дуже важливий та цінний етап опрацювання текстів природною мовою, результати якого мають широке застосування

Морфологічний аналізатор (**POS-tagger)**, це програма, яка обробляє послідовність слів і ставить у відповідність до кожного з них, відповідний тег (тег відповідає певному набору морфологічних характеристик.

Здійснивши простий аналіз розподілу слів у тексті, можна висунути гіпотезу про приналежність цього слова до певної лексичної категорії або поставити йому у відповідність певний тег. Програма морфологічного аналізу може коректно ідентифікувати теги слів при врахуванні їх контексту в реченні.

Автоматичний морфологічний аналіз також допомагає передбачити частину мови попередньо невідомих слів.

За домовленістю в NLTK промарковані слова (tokens) представляються з використанням типу даних – кортеж, Кортеж містить слово і тег. Для створення таких спеціальних кортежів з стандартної стрічки промаркованих слів потрібно використовувати функцію str2tuple().

Деякі корпуси тексті, які розповсюджуються разом з NLTK містять морфологічну розмітку (промарковані всі слова).

Використовуючи засоби Python можно коректно доступитись до морфологічно розмічених корпусів текстів. Якщо корпус містить морфологічну розмітку то потрібно використовувати метод *tagged_words()* . Детальну інформацію про набір тегів корпуса можна знайти у файлі README, який зберігається разом з корпусами.

Якщо корпус поділений (сегментований) на окремі речення то можна використовувати метод tagged_sents() для доступу до промаркованих окремих речень.

Використовуючи графічне застосування nltk.app.concordance() можна здійснювати побудову конкордансів з врахуванням морфологічних характеристик слів.

Частотний розподіл містить інформацію про частоти пар слово-тег. Якщо вважати слово – умовою, а тег – подією, то можна побудувати умовний частотний розподіл для таких умова-подія пар. В результаті побудови умовного частотного розподілу можна дізнатися впорядкований за частотою набір тегів для певного слова:

Якщо змінити порядок елементів в парах слово тег і вважати теги – умовою та слова подією, то можна встановити слова, які найчастіше маркуються певним тегом.

Для аналізу ширшого контексту потрібно знаходити слова , які відповідають деякій послідовності тегів та слів (наприклад "<Verb> to <Verb>"). Для вирішення цієї задачі потрібно спочатку в кожному реченні виділити послідовності з трьох слів, і перевірити ці слова на відповідність до заданої умови. Якщо умова задовольняється то послідовність слів виводиться на екран.

Деякі слова в корпусі маркуються різними тегами, ці слова є неоднозначні. Знайшовши такі слова та проаналізувавши їх вживання в тексті можна зрозуміти особливості їх морфологічного аналізу.

Промарковані слова представляються у вигляді(word, tag), де слово асоціюється з тегом, який відповідає певним морфологічним характеристикам. Автоматичний морфологічний аналіз можна розглядати , як задачу пошуку відповідного тега для слова. Найпростіший спосіб

збереження та обробки таких відповідностей (відображень) в Python це використання такого типу даних, як словник.

На відміну від стрічки і списку, де використовується *len()* для визначення цілого значення, яке відповідає максимальному індексу, аналогічно поступити зі словниками не можна. У випадку не великих словників вміст словника можна переглянути просто ввівши його ім'я. В результаті на екран буде виведено пари ключ-значення. Порядок цих пар відрізняється від порядку, в якому формувався словник. Це відбулося тому, що словник не є послідовністю, а є відповідністю. Ключі у відповідності не є впорядковані. Для знаходження ключів словник можна конвертувати у список — або використовувати словник як параметр sorted(), або обробляючи словник в for циклі.

Використовуючи методи keys(), values(), items() властиві словникам можна отримувати доступ до ключів, значень, та до пар ключ:значення як до окремих списків. Можна відсортувати кортежі, за їх першими елементами (якщо перші елементи однакові то сортування відбувається за другими елементами).

ВИКОНАННЯ ЗАВДАНЬ:

Завдання 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 = 'They wind back the clock, while we chase after the wind'
tok = nltk.word_tokenize(text)
morf = nltk.pos_tag(tok)
print tok
print morf
# "wind back" - дієслово |ˈwaɪnd| ,
# "wind" - іменник |wɪnd|
>>>
['They', 'wind', 'back', 'the', 'clock', ',', 'while', 'we', 'chase', 'after', '
the', 'wind']
[('They', 'PRP'), ('wind', 'VBP'), ('back', 'RB'), ('the', 'DT'), ('clock', 'NN'
), (',', ','), ('while', 'IN'), ('we', 'PRP'), ('chase', 'VBP'), ('after', 'IN')
, ('the', 'DT'), ('wind', 'NN')]
```

Завдання 3. Опрацювати всі приклади з методичних вказівок по роботі зі словниками. Що станеться, якщо доступитися до неіснуючого запису звичайного словника та словника по замовчуванню?

```
>>>
{}
{'colorless': 'ADJ'}
{'furiously': 'ADV', 'sleep': 'V', 'ideas': 'N', 'colorless': 'ADJ'}
N
ADJ
['furiously', 'sleep', 'ideas', 'colorless']
['colorless', 'furiously', 'ideas', 'sleep']
['ideas', 'colorless']
colorless: ADJ
furiously: ADV
ideas: N
sleep: V
['furiously', 'sleep', 'ideas', 'colorless']
['ADV', 'V', 'N', 'ADJ']
[('furiously', 'ADV'), ('sleep', 'V'), ('ideas', 'N'), ('colorless', 'ADJ')]
colorless: ADJ
furiously: ADV
ideas: N
sleep: V
{'furiously': 'ADV', 'sleep': 'V', 'ideas': 'N', 'colorless': 'ADJ'}
{'furiously': 'ADV', 'sleep': 'V', 'ideas': 'N', 'colorelss': 'ADJ'}
0
[]
N
[('blog', 'N'), ('colorless', 'ADJ')]
0
['NN', 'IN', 'AT', 'NP', ',', 'NNS', '.', 'JJ', 'CC', 'VBD', 'NN-TL', 'VB', 'VBN
', 'RB', 'CD', 'CS', 'VBG', 'TO', 'PPS', 'PP$', 'MD', 'AP', 'NP-TL', '``', 'BEZ'
, 'BEDZ', "''", 'JJ-TL', 'PPSS', 'DT', 'BE', 'VBZ', 'NR', 'RP', 'QL', 'PPO', 'WP
S', 'NNS-TL', 'WDT', 'WRB', 'BER', 'OD', 'HVZ', '--', 'NP$', '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$', 'CD-HL'
, 'JJS', 'JJ-HL', 'CC-TL', "'", 'MD*', 'VBZ-HL', 'PPL', 'PPSS+MD', 'PPS+BEZ', 'O
D-TL', 'DOZ', 'VB-HL', 'NR$', '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
```

```
import nltk
pos={}
print pos
pos['colorless']='ADJ'
print pos
pos['ideas']='N'
pos['sleep']='V'
pos['furiously']='ADV'
print pos
print pos['ideas']
print pos['colorless']
print list(pos)
print sorted(pos)
a = [w for w in pos if w.endswith('s')]
print a
for word in sorted(pos):
    print word+':', pos[word]
print pos.keys()
print pos.values()
print pos.items()
for key, val in sorted(pos.items()):
    print key+':', val

pos={'colorless':'ADJ', 'ideas':'N', 'sleep':'V', 'furiously':'ADV'}
print pos
pos=dict(colorelss='ADJ', ideas='N', sleep='V', furiously='ADV')
print pos
frequency=nltk.defaultdict(int)
frequency['colorless']=4
print frequency['ideas']
pos=nltk.defaultdict(list)
pos['sleep']=['N','V']
print pos['ideas']
pos=nltk.defaultdict(lambda:'N')
pos['colorless']='ADJ'
print pos['blog']
print pos.items()
counts=nltk.defaultdict(int)
from nltk.corpus import brown
for (word, tag) in brown.tagged_words(categories='news'):
```

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

```
d = {'I':'PRO', 'go':'V','to':'TO','cinema':'N','every':"DET", 'week':'N'}
print d
del d['I']
print d
del d['go']
print d
```

```
>>>
{'week': 'N', 'cinema': 'N', 'I': 'PRO', 'to': 'TO', 'every': 'DET', 'go': 'V'
{'week': 'N', 'cinema': 'N', 'to': 'TO', 'every': 'DET', 'go': 'V'}
{'week': 'N', 'cinema': 'N', 'to': 'TO', 'every': 'DET'}
```

Завдання 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',
 ':-TL',
 'ABL',
 'ABN',
 'ABN-HL',
 'ABN-NC',
 'ABN-TL',
 'ABX',
 'AP',
 'AP$',
```

Завдання 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 findt(tags, text):
        tag_list=[]
        for tag in text:
                if tag[1].startswith(tags):
                        tag_list+=[tag[1]]
        fd=nltk.FreqDist(tag_list)
        print 'tags for nouns', (tags,len(set(tag_list)))
        print 'all tags:', fd.keys()
        return 'most frequent tags:', fd.keys()[:15]
print findt('N',brown)
>>>
tags for nouns ('N', 60)
all tags: ['NN', 'NNS', 'NP', 'NN-TL', 'NP-TL', 'NP$', 'NNS-TL', 'NR', 'NN$', 'N
N-HL', 'NPS', 'NNS-HL', 'NP-HL', 'NN$-TL', 'NR-TL', 'NNS$', 'NIL', 'NP$-TL', 'NN
-TL-HL', 'NN-NC', 'NNS$-TL', 'NPS-TL', 'NR$', 'NPS$', 'NN+BEZ', 'NNS-NC', 'NP+BE
Z', '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', 'N
NS-TL-NC', 'NP+BEZ-NC', 'NPS$-TL', 'NN+BEZ-TL', 'NN+MD', 'NNS$-NC', 'NNS+MD', 'N
P+MD', 'NPS-NC', 'NN+HVD-TL', 'NN+HVZ-TL', 'NN+IN', 'NN+NN-NC', 'NNS$-TL-HL', 'N
P+HVZ-NC', 'NPS$-HL', 'NR+MD', 'NRS-TL']
('most frequent tags:', ['NN', 'NNS', 'NP', 'NN-TL', 'NP-TL', 'NP$', 'NNS-TL', '
NR', 'NN$', 'NN-HL', 'NPS', 'NNS-HL', 'NP-HL', 'NN$-TL', 'NR-TL'])
```

Завдання 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
browntag = brown.tagged_words(categories='religion', simplify_tags=True)
data = nltk.ConditionalFreqDist((word.lower(),tag) for (word,tag) in browntag)
res=[]
for word in data.conditions():
        if len(data[word])==1:
                tags=data[word].keys()
                res.append(word)
print len(res)
print len(browntag)
print 'the percentage is:', len(res)*100/len(browntag), '%'
>>>
5499
39399
the percentage is: 13 %
```

Завдання 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 proc(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 proc(tagged_sent)

None
('of', 'this', 'trend')
('of', 'this', 'phase')
None
None
None
None
None
None
None
None
None
('of', 'this', 'phase')
```

Завдання 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
dic=nltk.defaultdict(lambda: nltk.defaultdict(int))
browntag = brown.tagged_words(categories='fiction', simplify_tags=True)
for ((w1, t1), (w2, t2)) in nltk.ibigrams(browntag):
    dic[(w1, t1)][t2]+=1
print dic[('room','N')]
>>>
defaultdict(<type 'int'>, {'ADV': 4, 'VD': 1, 'TO': 1, 'ADJ': 1, 'PRO': 1,
16, '.': 13, "''": 1, 'P': 10, 'EX': 1, 'V': 5, 'N': 1, 'CNJ': 7, 'MOD': 1}
```

Висновок.

У ході виконання лабораторної роботи Ввивчила основи програмування на мові *Python* та ознайомилась із автоматичним морфологічним аналізом у NLTK.