

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»
Кафедра САПР

ЗВІТ

до лабораторної роботи № 12

на тему:

ВИВЧЕННЯ БІБЛІОТЕКИ ПРИКЛАДНИХ ПРОГРАМ NLTK, ДЛЯ
ОПРАЦЮВАННЯ ТЕКСТІВ ПРИРОДНОЮ МОВОЮ.

АВТОМАТИЧНИЙ СИНТАКСИЧНИЙ АНАЛІЗ (частина 2)

з дисципліни “Комп’ютерна лінгвістика”

Виконала:

Студентка групи ПРЛм-12

Рибчак Х. В.

Перевірив:

Асистент кафедри САПР

Дупак Б. П.

Львів 2015

МЕТА РОБОТИ

Вивчення основ програмування на мові Python. Ознайомлення з автоматичним синтаксичним аналізом в NLTK.

КОРОТКІ ТЕОРЕТИЧНІ ВІДОМОСТІ

Well-Formed Substring Tables

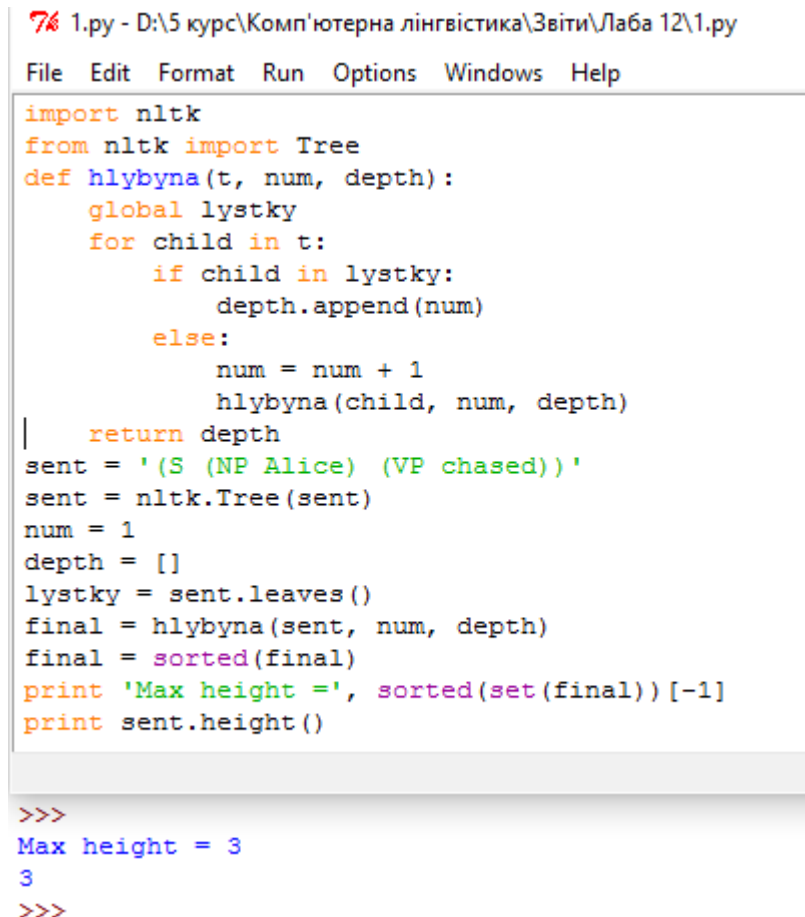
Прості синтаксичні аналізатори, які розглядалися в попередній лабораторній роботі мають ряд недоліків, які накладають значні обмеження як на ефективність так і взагалі на можливість отримання результатів синтаксичного аналізу. Для вирішення цих проблем використовуються алгоритми що базуються на динамічному програмуванні. Динамічне програмування передбачає збереження проміжних результатів та їх використання при необхідності що дозволяє значно підвищити ефективність роботи різноманітних алгоритмів. Застосування динамічного програмування для синтаксичного аналізу дозволить зберегти часткові рішення при аналізі та використовувати ці рішення для одержання загальних (повних, завершених) результатів синтаксичного аналізу. Одним з алгоритмів що базується на динамічному програмуванні є алгоритм аналізу за допомогою схем (**chart parser**).

Динамічне програмування дозволяє при синтаксичному аналізі речення речення I shot an elephant in my pajamas, будувати PP in my pajamas тільки один раз. Як тільки цей складник побудовано він зберігається в таблиці і ним можна скористатися при потребі як безпосереднім складником при побудові NP або VP. Таку таблицю називають таблицею закономірностей підстрічок (**well-formed substring table WFST**). Підстрічкою вважається послідовність слів в межах одного речення. Розглянемо побудову такої таблиці на основі стратегії знизу-вверх в якій будуть послідовно збережені всі можливі безпосередні складники.

ТЕКСТИ ПРОГРАМ НА МОВІ PYTHON

ВАРІАНТ №8

1. Написати рекурсивну функцію для перегляду дерева, яка визначає його глибину. Дерево з одного вузла має глибину рівну нулю. (глибина піддерева це максимальна глибина його дітей плюс один).



```
7% 1.py - D:\5 курс\Комп'ютерна лінгвістика\Звіти\Лаба 12\1.py
File Edit Format Run Options Windows Help

import nltk
from nltk import Tree
def hlybyna(t, num, depth):
    global lystky
    for child in t:
        if child in lystky:
            depth.append(num)
        else:
            num = num + 1
            hlybyna(child, num, depth)
    return depth
sent = '(S (NP Alice) (VP chased))'
sent = nltk.Tree(sent)
num = 1
depth = []
lystky = sent.leaves()
final = hlybyna(sent, num, depth)
final = sorted(final)
print 'Max height =', sorted(set(final))[-1]
print sent.height()

>>>
Max height = 3
3
>>>
```

Рис. 1. Текст програми №1.

3. Chart parser додає але ніколи не видаляє дуги з chart. Чому?
Chart parser — метод динмічного програмування, у якому не запам'ятовують проміжні результати.

5. Вибрати декілька (2) загальних дієслова та напишіть програми для вирішення наступних задач: Пошук дієслів в корпусі Prepositional Phrase Attachment Corpus `nltk.corpus.ppattach`. Пошук всіх випадків вживання дієслова з двома різними PP в яких перший іменник, або другий іменник або прийменник залишаються незмінними. Розробити правила CFG граматики для врахування цих випадків.

File Edit Format Run Options Windows Help

```

import nltk
from nltk.corpus import ppattach
entries = nltk.corpus.ppattach.attachments('training')
table=nltk.defaultdict(lambda:nltk.defaultdict(set))
for entry in entries:
    key=entry.noun1 + '-' + entry.prep + '-' + entry.noun2
    table[key][entry.attachment].add(entry.verb)
for key in sorted(table):
    if len(table[key])>1:
        print key, 'N', sorted(table[key]['N']), 'V:', sorted(table[key]['V'])

```

Ln: 11 Col: 0

```

services-for-customers N ['offering'] V: ['provide']
shareholder-in-bank N ['become'] V: ['become']
stake-in-Airlines N ['acquiring', 'buy', 'raise'] V: ['buy']
stake-in-Mixte N ['bring'] V: ['boost']
stake-in-Rally N ['hold'] V: ['had']
stake-in-company N ['bought', 'building', 'built', 'buying', 'give', 'hold', 'obtaining', 'own',
'owns', 'raised', 'take'] V: ['accumulating', 'had', 'has', 'holds', 'own']
stake-in-concern N ['acquires', 'lowered'] V: ['retaining']
stake-in-unit N ['sell'] V: ['acquire']
stake-in-venture N ['has', 'hold', 'holds'] V: ['held']
suit-against-Keating N ['press'] V: ['brought']
swings-in-market N ['cause', 'create'] V: ['cause']
system-for-city N ['design'] V: ['design']
system-in-Pakistan N ['operate'] V: ['operate']
time-for-Congress N ['is'] V: ['buy', 'buys']
venture-with-Co. N ['started'] V: ['started']
ventures-with-companies N ['established'] V: ['form']
verdict-in-case N ['is', 'won'] V: ['won']
volatility-in-stocks N ['ignoring'] V: ['see']
vote-on-issue N ['allow'] V: ['prevent']
way-for-declines N ['open'] V: ['pave']
writer-in-York N ['is'] V: ['is']

```

Рис. 2. Текст програми №5.

7. Використовуючи позиції в дереві побудувати список підметів перших 100 речень корпусу Penn treebank; для спрощення представлення результатів підмети представляти як піддерева з глибиною не більше 2.

```
78 /py - D:/5 курс/Комп'ютерна лінгвістика/Звіти/Лаба 12//py
File Edit Format Run Options Windows Help

import nltk
from nltk.corpus import treebank
penn = treebank.parsed_sents()[ :30]
def filter(tree):
    child_nodes = [child.node for child in tree
                    if isinstance (child, nltk.Tree) and len(tree) <2]
    return tree.node=='NP-SBJ'
result = [subtree for tree in penn for subtree in tree.subtrees(filter)]
print result

>>>
[Tree('NP-SBJ', [Tree('NP', [Tree('NNP', ['Pierre']), Tree('NNP', ['Vinken'])]),
Tree(',', ['']), Tree('ADJP', [Tree('NP', [Tree('CD', ['61']), Tree('NNS', ['y
ears'])]), Tree('JJ', ['old'])]), Tree(',', ['']), Tree('NP-SBJ', [Tree('NNP'
, ['Mr.']), Tree('NNP', ['Vinken'])]), Tree('NP-SBJ', [Tree('-NONE-', ['*-1'])])
, Tree('NP-SBJ', [Tree('NP', [Tree('NP', [Tree('DT', ['A']), Tree('NN', ['form'
])]), Tree('PP', [Tree('IN', ['of']), Tree('NP', [Tree('NN', ['asbestos'])])])]),
Tree('RRC', [Tree('ADVP-TMP', [Tree('RB', ['once'])]), Tree('VP', [Tree('VBN',
['used']), Tree('NP', [Tree('-NONE-', ['*'])]), Tree('S-CLR', [Tree('NP-SBJ', [T
ree('-NONE-', ['*'])]), Tree('VP', [Tree('TO', ['to']), Tree('VP', [Tree('VB', [
'make']), Tree('NP', [Tree('NNP', ['Kent']), Tree('NN', ['cigarette']), Tree('NN
S', ['filters'])])])])])])]), Tree('NP-SBJ', [Tree('-NONE-', ['*'])]), Tree('NP-
P-SBJ', [Tree('NNS', ['researchers'])]), Tree('NP-SBJ', [Tree('NP', [Tree('DT',
['The']), Tree('NN', ['asbestos']), Tree('NN', ['fiber'])]), Tree(',', ['']), T
ree('NP', [Tree('NN', ['crocidolite'])]), Tree(',', ['']), Tree('NP-SBJ', [Tr
ee('PRP', ['it'])]), Tree('NP-SBJ', [Tree('NP', [Tree('RB', ['even']), Tree('JJ'
, ['brief']), Tree('NNS', ['exposures'])]), Tree('PP', [Tree('TO', ['to']), Tree
```

Рис. 3. Текст програми №7.

12. Розробити програму обробки дерев корпусу Treebank nltk.corpus.treebank , яка вилучить всі правила з кожного з дерев за допомогою Tree.productions(). Правилами, які зустрічаються тільки один раз можна знехтувати. Правила з однаковими лівими частинами та подібними правими частинами об'єднати для отримання еквівалентного але більш компактного набору правил.

File Edit Format Run Options Windows Help

```
import nltk
from nltk.corpus import treebank
t = treebank.parsed_sents('wsj_0002.mrg')[0]
print t
for i in t.productions():
    print i
```

```
PP -> IN NP
IN -> 'of'
NP -> NNP NNP NNP NNP
NNP -> 'Consolidated'
NNP -> 'Gold'
NNP -> 'Fields'
NNP -> 'PLC'
, -> ','
VP -> VBD VP
VBD -> 'was'
VP -> VBN S
VBN -> 'named'
S -> NP-SBJ NP-PRD
NP-SBJ -> -NONE-
-NONE- -> '*-1'
NP-PRD -> NP PP
NP -> DT JJ NN
DT -> 'a'
JJ -> 'nonexecutive'
NN -> 'director'
PP -> IN NP
IN -> 'of'
NP -> DT JJ JJ NN
DT -> 'this'
JJ -> 'British'
JJ -> 'industrial'
NN -> 'conglomerate'
. -> '.'
```

Рис. 4. Текст програми №12.

ВИСНОВОК

У цій лабораторній роботі я вивчила основи структурного програмування мовою Python та знайомилася з автоматичним синтаксичним аналізом в NLTK.