МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ “ЛЬВІВСЬКА ПОЛІТЕХНІКА”

Лабораторна робота № 12

ВИВЧЕННЯ БІБЛІОТЕКИ ПРИКЛАДНИХ ПРОГРАМ NLTK, ДЛЯ ОПРАЦЮВАННЯ ТЕКСТІВ ПРИРОДНОЮ МОВОЮ. АВТОМАТИЧНИЙ СИНТАКСИЧНИЙ АНАЛІЗ (частина2).

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 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**).

Більш загально, в комірку *(i, j)* записується *A* якщо існує правило $A \rightarrow B\ C$, і знайдено нетермінал *B* в комірці *(i, k)* та *C* в комірці *(k, j)*.

Програма побудови WFST це практично простий chart parser, який має декілька недоліків. Перший, таблиця яку ми отримали це не є окреме дерево розбору, а скоріше метод розпізнавання чи речення породжується (допускається) граматикою, а не програма його синтаксичного аналізу. Другий, програма потребує щоб правила граматики, де в правій частині знаходяться не термінали були бінарними. Звичайно, будь-яку контекстно-вільну граматику можна перетворити у таку форму (нормальну форму Хомського), але зручніше працювати без таких додаткових обмежень. Третій, підхід знизу вверх, відзначається великою надлишковістю, коли будує складники (пропонує розмістити не термінальні символи в комірках), які не передбачені граматикою.

Залежності і граматика залежностей

Граматики, які будуються на основі фразової структури речення, описують як слова та їх послідовності об'єднуються (поєднуються) у складники (безпосередні складники). Альтернативний, або доповнюючий підхід, який називають граматикою залежностей, полягає у встановленні взаємоз'язків між окремими словами. Між парами слів у реченні встановлюється бінарний асиметричний зв'язок, який вказує на основне слово та залежне. Основним словом в реченні звичайно вважається дієслово (присудок) і інші слова, або безпосередньо або через інші зв'язки залежать від нього.

Дерево залежностей представляється у вигляді поміченого орієнтованого графа, у якому вузлами є лексичні одиниці а помічені дуги представляють відношення залежностей між основними і та залежними словами.

ВИКОНАННЯ ЗАВДАНЬ

Завдання 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)

```
import nltk
def treeverse(t):
    try:
        t.node
    except AttributeError:
        print t,
    else:
        print '(', t.node,
        l=0
        for child in t:
            l=l+1
            print l
            treeverse(child)
            print ')',
t = nltk.Tree('(S (NP We) (VP love (NP Ukraine)))')
b = treeverse(t)
print b
>>>
( S 1
( NP 1
We ) ) 2
( VP 1
love ) 2
( NP 1
Ukraine ) ) ) None
```

Завдання 3. chart parser додає, але ніколи не видаляє дуги з chart. Чому?
Тому що це динамічне програмування і воно запам’ятовує проміжні результати.

Завдання 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.

```
import nltk
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'])
%-below-level N: ['left'] V: ['be']
%-from-year N: ['was'] V: ['declined', 'dropped', 'fell', 'grew', 'increased',
'plunged', 'rose', 'was']
%-in-August N: ['was'] V: ['climbed', 'fell', 'leaping', 'rising', 'rose']
%-in-September N: ['increased'] V: ['climbed', 'declined', 'dropped', 'edged',
'fell', 'grew', 'plunged', 'rose', 'slipped']
%-in-week N: ['declined'] V: ['was']
%-to-% N: ['add', 'added', 'backed', 'be', 'cut', 'go', 'grow', 'increased', 'i
ncreasing', 'is', 'offer', 'plummet', 'reduce', 'rejected', 'rise', 'risen', 's
haved', 'wants', 'yield', 'zapping'] V: ['fell', 'rise', 'slipped']
%-to-million N: ['declining'] V: ['advanced', 'climbed', 'cutting', 'declined',
'declining', 'dived', 'dropped', 'edged', 'fell', 'gained', 'grew', 'increased'
, 'jump', 'jumped', 'plunged', 'rising', 'rose', 'slid', 'slipped', 'soared', '
tumbled']
1-to-21 N: ['dropped'] V: ['dropped']
1-to-33 N: ['gained'] V: ['dropped', 'fell', 'jumped']
1-to-4 N: ['added'] V: ['gained']
1-to-47 N: ['jumped'] V: ['added', 'rose']
1-to-point N: ['ended'] V: ['fell', 'rose']
3-to-17 N: ['lost'] V: ['lost']
500,000-in-fines N: ['paid'] V: ['paid']
6.9-on-scale N: ['registered'] V: ['registered']
access-to-AZT N: ['had'] V: ['had']
access-to-arena N: ['permits'] V: ['lack']
activity-in-part N: ['showed'] V: ['attributed']
agreement-in-principle N: ['reached'] V: ['reached']
agreement-with-Inc. N: ['announced', 'signed'] V: ['signed']
agreement-with-creditors N: ['reached'] V: ['nearing']
agreement-with-regulators N: ['presages', 'reach'] V: ['reach']
aid-to-Contras N: ['renewing'] V: ['renewing']
alliance-with-GM N: ['discussing', 'wrapping'] V: ['forge', 'have', 'negotiatin
```

Завдання 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.

```
import nltk
from nltk.corpus import treebank
treetrain=treebank.parsed_sents()[:100]
def listp(tree):
    cild=[child.node for child in tree
          if isinstance(child,nltk.Tree) and len(tree)<2]
    return tree.node=='NP-SBJ'
a=[subtree for tree in treetrain for subtree in tree.subtrees(listp)]
print a
>>>
[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('N
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
('NP', [Tree('PRP', ['it'])])])]), Tree('NP-SBJ', [Tree('-NONE-', ['*T*-1'])]),
Tree('NP-SBJ', [Tree('NNS', ['researchers'])]), Tree('NP-SBJ', [Tree('NP', [Tree
('NNP', ['Lorillard']), Tree('NNP', ['Inc.'])]), Tree(',', [',']), Tree('NP', [T
ree('NP', [Tree('DT', ['the']), Tree('NN', ['unit'])]), Tree('PP', [Tree('IN', [
'of']), Tree('NP', [Tree('ADJP', [Tree('JJ', ['New']), Tree('JJ', ['York-based']
)]), Tree('NNP', ['Loews']), Tree('NNP', ['Corp.'])])])]), Tree('SBAR', [Tree('WHN
P-2', [Tree('WDT', ['that'])]), Tree('S', [Tree('NP-SBJ', [Tree('-NONE-', ['*T*-
2'])]), Tree('VP', [Tree('VBZ', ['makes']), Tree('NP', [Tree('NNP', ['Kent']), T
ree('NNS', ['cigarettes'])])])])])]), Tree(',', [','])]), Tree('NP-SBJ', [Tree('
-NONE-', ['*T*-2'])]), Tree('NP-SBJ', [Tree('DT', ['the']), Tree('JJS', ['latest
']), Tree('NNS', ['results'])]), Tree('NP-SBJ', [Tree('-NONE-', ['*'])]), Tree('
NP-SBJ', [Tree('DT', ['A']), Tree('NNP', ['Lorillard']), Tree('NN', ['spokewoman
'])]), Tree('NP-SBJ', [Tree('DT', ['This'])]), Tree('NP-SBJ', [Tree('PRP', ['We'
```

Завдання 12. Розробити програму обробки дерев корпуса 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.

```
import nltk
from nltk.corpus import treebank
t = treebank.parsed_sents('wsj_0002.mrg')[0]
print t
prod = t.productions()
print prod
fdist = nltk.FreqDist(prod)
fd=fdist.keys()
print fd
for i in fdist.keys():
    if fdist[i]>0:
        print i
```

```
>>>
(S
  (NP-SBJ-1
    (NP (NNP Rudolph) (NNP Agnew))
    (, ,)
    (UCP
      (ADJP (NP (CD 55) (NNS years)) (JJ old))
      (CC and)
      (NP
        (NP (JJ former) (NN chairman))
        (PP
          (IN of)
          (NP (NNP Consolidated) (NNP Gold) (NNP Fields) (NNP PLC)))))
    (, ,))
  (VP
    (VBD was)
    (VP
      (VBN named)
      (S
        (NP-SBJ (-NONE- *-1))
        (NP-PRD
          (NP (DT a) (JJ nonexecutive) (NN director))
          (PP
            (IN of)
            (NP
              (DT this)
              (JJ British)
              (JJ industrial)
              (NN conglomerate)))))))
  (. .))
[S -> NP-SBJ-1 VP ., NP-SBJ-1 -> NP , UCP ,, NP -> NNP NNP, NNP -> 'Rudolph', NN
P -> 'Agnew', , -> ',', UCP -> ADJP CC NP, ADJP -> NP JJ, NP -> CD NNS, CD -> '5
5', NNS -> 'years', JJ -> 'old', CC -> 'and', NP -> NP PP, NP -> JJ NN, JJ -> 'f
ormer', NN -> 'chairman', PP -> IN NP, IN -> 'of', NP -> NNP NNP NNP NNP, NNP ->
 'Consolidated', NNP -> 'Gold', NNP -> 'Fields', NNP -> 'PLC', , -> ',', VP -> V
BD 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 -> 'none
xecutive', NN -> 'director', PP -> IN NP, IN -> 'of', NP -> DT JJ JJ NN, DT -> '
this', JJ -> 'British', JJ -> 'industrial', NN -> 'conglomerate', . -> '.']
[, -> ',', IN -> 'of', PP -> IN NP, -NONE- -> '*-1', . -> '.', ADJP -> NP JJ, CC
```

ВИСНОВОК

Під час виконання лабораторної роботи я ознайомилась з автоматичним синтаксичним аналізом в NLTK chart parser, дізналася про динамічне програмування, граматику залежностей.