МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ “ЛЬВІВСЬКА ПОЛІТЕХНІКА”

ІНСТИТУТ КОМП’ЮТЕРНИХ НАУК ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

### *Кафедра “Системи автоматизованого проектування”*



Звіт

до лабораторної роботи №12

на тему: «Вивчення бібліотеки прикладних програм NLTK для опрацювання текстів природною мовою. Автоматичний синтаксичний аналіз (частина2)»

з дисципліни “Комп’ютерна лінгвістика”

Виконала:

студентка групи ПРЛм-11

Павлів І.О.

Прийняв:

викладач

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.

Розглянемо речення I shot an elephant in my pajamas як вхідні дані. Дане речення можна представити у вигляді, який показано на рис.1 і така структура називається Chart Data Structure.



**Рис. 1.** Структура речення: словами речення промарковані дуги графа**.**

В таблицю WFST, записуються позиції слів шляхом заповнення комірок трикутної матриці в якій по вертикалі записуються початкові позиції підстрічок а по горизонталі кінцеві позиції (таким чином слову *shot* буде відповідати комірка з координатами (1, 2)). Для спрощення такого представлення, припускається що кожному слову відповідає тільки одна унікальна лексична категорія (тег морфологічних характеристик) і саме вона зберігається в комірці матриці(в комірці (1, 2) міститься V). Більш загально, якщо вхідної стрічкою є стрічка *a*1*a*2 ... *a*n, і граматика містить правило вигляду *A* → *a*i, тоді *A* додається до коміркиl (*i*-1, *i*).

**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. На рис представлено такий граф і напрям стрілок вказує на залежні слова.

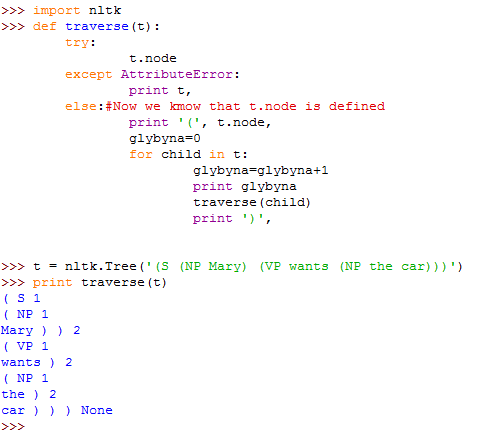


**Рис. 3.** Графічне представлення дерева залежностей

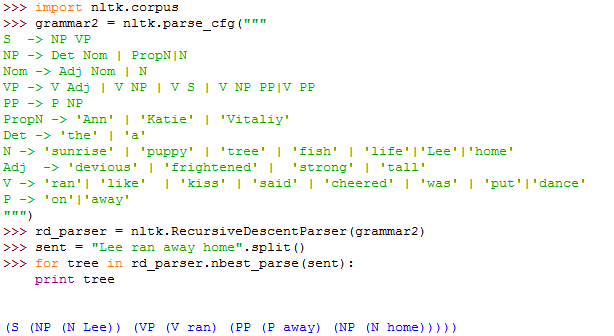
Кожна з дуг на рис. є помічена позначеннями типу відношень, які встановлені між основним та залежним словом. Наприклад , *I* це SBJ (підмет), *shot* (основне слово речення), *in* - NMOD (модифікатор іменника *elephant*). В граматиці залежностей зв’язки міх членами речення можуть бути представлені за допомогою типів залежностей.

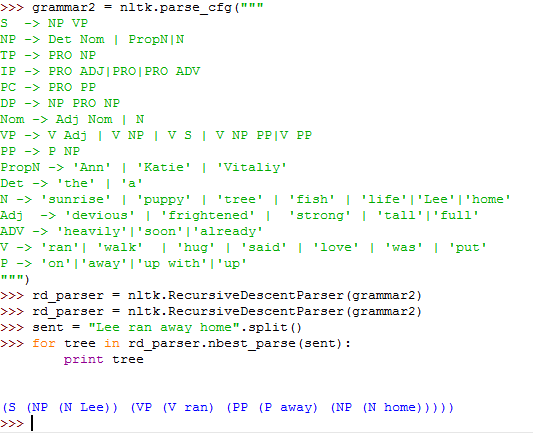
**Варіант №13**

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)



1. Розширити граматику grammar2 з попередньої лабораторної роботи правилами які розділяють прийменники як перехідні, неперехідні та такі що вимагають PP доповнення. На основі цих правил побудуйте дерево розбору для речення Lee ran away home, використовуючи аналізатор рекурсивного спуску.



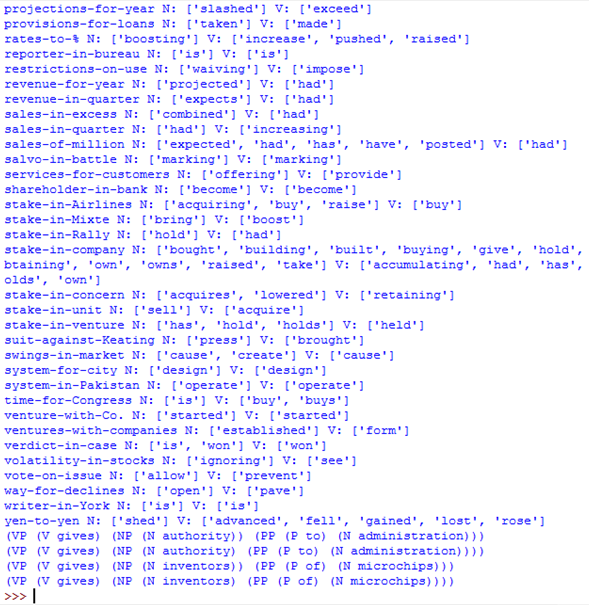


1. Вибрати декілька (2) загальних дієслова та напишіть програми для вирішення наступних задач:

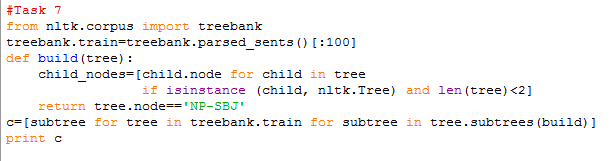
Пошук дієслів в корпусі Prepositional Phrase Attachment Corpus nltk.corpus.ppattach. Пошук всіх випадків вживання дієслова з двома різними РР в яких перший іменник, або другий іменник або прийменник залишаються незмінними

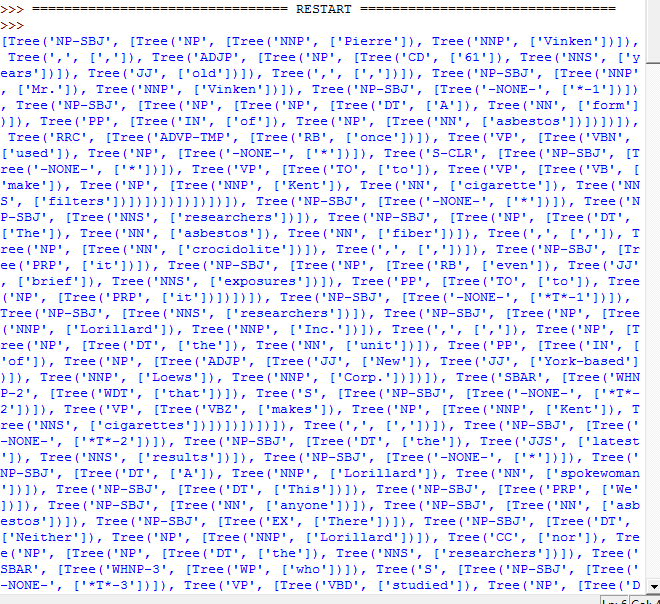
Розробити правила CFG граматики для врахування цих випадків.



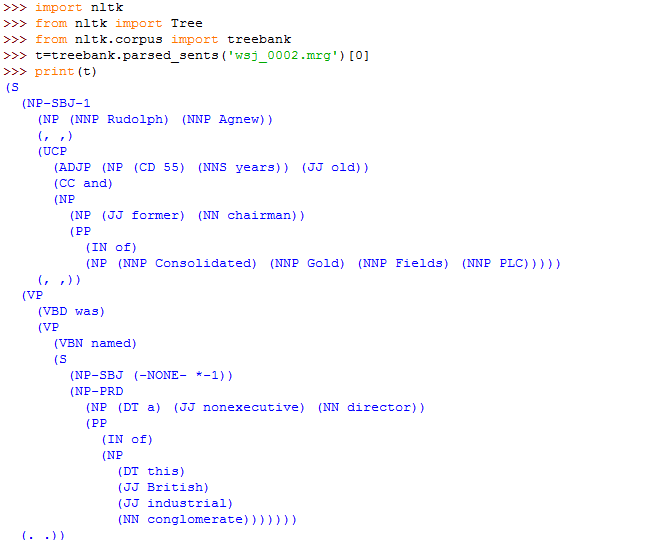


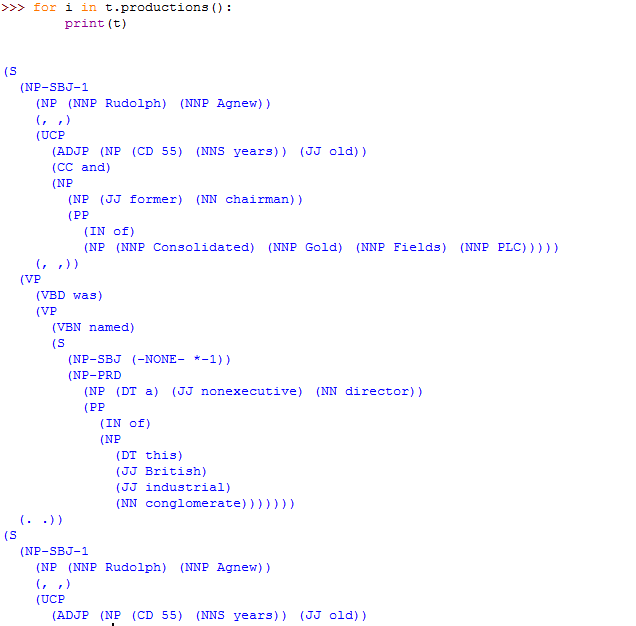
1. Використовуючи позиції в дереві побудувати список підметів перших 100 речень корпусу Penn treebank; для спрощення представлення результатів підмети представляти як піддерева з глибиною не більше 2.





1. 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.





**Висновок:**

на цій лабораторній роботі я ознайомлення з автоматичним синтаксичним аналізом в NLTK chart parser, дізналася про динамічне програмування, граматику залежностей.