**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ “ЛЬВІВСЬКА ПОЛІТЕХНІКА”**

**іНСТИТУТ КОМП’ютерних НАУК та ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ**

### Кафедра “Системи автоматизованого проектування”



Звіт до лабораторної роботи №3

З дисципліни «Комп’ютерна лінгвістика»

на тему

«ВИВЧЕННЯ БІБЛІОТЕКИ ПРИКЛАДНИХ ПРОГРАМ NLTK, ДЛЯ ОПРАЦЮ­ВАННЯ ТЕКСТІВ ПРИРОДНОЮ МОВОЮ. ДОСТУП ТА РОБОТА З ЛЕКСИЧНИМИ РЕСУРСАМИ»

Виконала

Ст.гр.ПРЛм-12

Гречух Ю.

Перевірив

Старший викладач кафедри САПР

Дупак П.

Львів-2015**Мета роботи -** вивчення основ програмування на мові *Python*, вивчення методів доступу та роботи з лексичним ресурсами, семантичний словник англійської мови WordNet.

**Теоретичні відомості**

1. ***Поняття функції та модуля***.

Функція - це програмна конструкція, яку можна викликати з одним або більше вхідними параметрами, і отримувати результат на виході. Визначаємо функцію, використовуючи ключове слово *def* далі потрібно дати назву функції і визначити вхідні параметри, після двокрапки записується тіло функції. Ключове слово *return* використовується для відображення значення, яке ми хочемо отримати на виході функції..

Множина змінних і функцій збережених у файлі називаються в Python – модулем. Множина пов’язаних між собою модулів називають – пакетом. Програма обробки корпуса Brown це є приклад модуля, а множина програм для роботи зі всіма корпусами це є приклад пакету. NLTK це множина пакетів, яку називають бібліотекою.

1. ***Генерація випадкового тексту за допомогою біграмів.***

Умовний частотний розподіл можна використати для побудови таблиці біграмів (пар слів). Функція NLTK bigrams() , як аргумент бере список слів і повертає список послідовних пар слів.

>>> sent = ['In', 'the', 'beginning', 'God', 'created', 'the', 'heaven',

... 'and', 'the', 'earth', '.']

>>> nltk.bigrams(sent)

[('In', 'the'), ('the', 'beginning'), ('beginning', 'God'), ('God', 'created'),

('created', 'the'), ('the', 'heaven'), ('heaven', 'and'), ('and', 'the'),

('the', 'earth'), ('earth', '.')]

В наступному прикладі кожне слово розглядається, як умова і для кожного з них будується частотний розподіл по словам, які слідують після нього. Функція generate\_model() містить простий цикл для генерації тексту. Коли ця функція викликається то одним з її аргументів є слово – початковий контекст (у прикладі 'living') . В циклі поточне значення змінної word виводиться на екран і її значення замінюється на слово, яке найчастіше є наступним словом (max()) . На наступному кроці циклу вже це слово буде наступним контекстом.

1. ***Лексичні ресурси NLTK.***

Лексичний ресурс або просто словник це набір слів тa/або словосполучень, які асоціюються з такою інформацією, як частина мови та опис значення. Лексичні ресурси є вторинними по відношенню до текстів і зазвичай створюються і вдосконалюються з використанням текстів.

Найпростіший словник це відсортований список слів. Досконаліші словники містять складну структуру записів та зв’язків між ними. В цій лабораторні роботі будуть розглянуті лексичні ресурси, які розповсюджуються разом з NLTK.

* 1. ***Корпуси слів***

NLTK розповсюджується з деякими корпусами, які насправді є списками слів. Корпус words це файл з Unix, який використовується для перевірки правопису. Цей список можна використати для знаходження незвичних та написаних з помилками слів в корпусі текстів.

* 1. ***Словник із позначенням вимови.***

Більш багатим лінгвістичним ресурсом може бути словник де кожному слову поставлена у відповідність певна інформація. NLTK включає CMU Pronouncing Dictionary американського варіанту англійської , який розроблений для використання в синтезаторах мови.

1. ***WordNet – лексична база даних англійської мови***

*WordNet,* це семантично орієнтований словник англійської мови, подібний до традиційних тезаурусів але з більш багатою структурою. У *WordNet* слова групуються у набори синонімів – синсети, кожен із своїм визначеннямі зв’язками з іншими синсетами. *WordNet 3.0* розповсюджується разом з NLTK і містить 155287 слів та 117659 синсетів. Хоча *WordNet* розроблявся для психолінгвістики - цей словник широко використовується в NLP та в задачах інформаційного пошуку. Розробки для інших мов проводяться на основі документації, яка наведена у <http://www.globalwordnet.org/>.

Для одержання значення слова потрібно вибрати до якої частини мови воно належить. *WordNet* містить чотири словники (іменники, дієслова, прикметники, прислівники ). Знайдемо слово *motorcar* у словнику іменників:

>>> from nltk.corpus import wordnet as wn

>>> wn.synsets('motorcar')

[Synset('car.n.01')]

Слова в синсет об’єднані за спільним значенням, яке є однакове для всіх слів. В синсеті вказується текстовий опис цього значення та приклад вживання слів з синсету.

Для уникнення двозначності слова з син сету можна ідентифікувати як car.n.01.automobile, car.n.01.motorcar . Такі пари, як синсет і слово називають лемою.

* 1. ***Ієрархія в WordNet***

Синсети відповідають абстрактним поняттям, які можуть мати або не мати відповід­них слів. Ці поняття зв’язуються разом в ієрархії. Деякі поняття Entity, State, Event – є загаль­ними і їх називають унікальними початковими поняттями. Інші є більше специфічними. Части­на ієрархії понять наведена на рис.4. Лінії між вузлами вказують на зв’язки (гіперонім\гіпонім), пунктирна лінія вказує, що *artefact* не є безпосереднім гіперонімом *motorcar*.

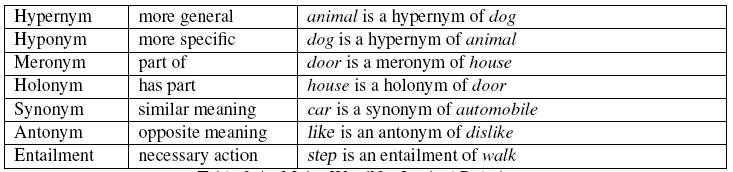
*WordNet* дозволяє легко переміщатися між поняттями. Наприклад для поняття *motorcar* ми можемо переглянути поняття, які є більш специфічними (гіпонім).

Аналогічно можна піднятися по ієрархії і переглянути більш широкі поняття ніж motorcar(гіпероніми). Деякі слова мають декілька шляхів вверх, ці слова можуть класифікуватися більш ніж одним способом. Від car.n.01 до entity.n.01 є два шляхи оскільки wheeled\_vehicle.n.01 може розглядатися як vehicle та container.

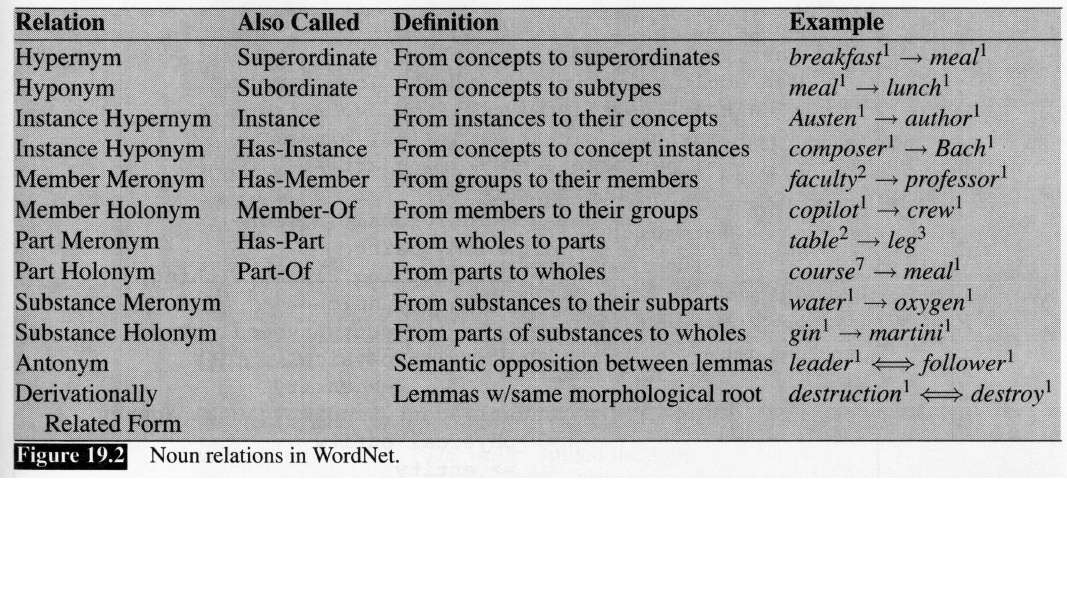
* 1. ***Лексичні зв’язки в WordNet.***

Таблиця 1 містить список найбільш важливих типів зв’язків, які реалізовані у WordNet. Таблиця 2 містить повний список зв’язків іменників.

Таблиця 1



Таблиця 2



Гіперніми та гіпоніми називають лексичними зв’язками тому що вони пов’язують один син сет з іншим. Ці два зв’язки вказують на рух вверх-вниз в ієрархії «is-a». Інший можливий шлях в ієрархії WordNet це від предмету до його складових (меронім), або до поняття яке містить предмет в собі (голоніми). Наприклад, частини дерева – стовбур, крона та ін. part\_meronyms() . Речовина з якого дерево зроблено включає heartwood та sapwood; - substance\_meronyms(). Багато дерев утворюють ліс - member\_holonyms()

Для дієслів характерні зв’язки подібні до наступних:

>>> wn.synset('walk.v.01').entailments()

[Synset('step.v.01')]

>>> wn.synset('eat.v.01').entailments()

[Synset('swallow.v.01'), Synset('chew.v.01')]

>>> wn.synset('tease.v.03').entailments()

[Synset('arouse.v.07'), Synset('disappoint.v.01')]

Дія walking передбачає дію stepping - walking спричиняє (entails) stepping.

Ще один тип зв’язків представлений в WordNet це антоніми:

>>> wn.lemma('supply.n.02.supply').antonyms()

***4.4. Оцінка подібності в WordNet***

Синсети зв’язані між собою складною мережею лексичних зв’язків. Для певного синсету можна переглянути зв’язки у WordNet і знайти синсети, які з ним зв’язані за змістом. Інформація про семантичні взаємозв’язки між словами цінна при класифікації текстів.

Кожен синсет має один або більше шляхів за яким він зводиться до ключового поняття entity.n.01. Два синсети можуть мати спільне ключове поняття і ми нижче за ієрархією це ключове поняття тим ближчі між собою ці два синсети.

Семантична подібність двох понять пов’язана з довжиною шляху між цими поняттями в *WordNet* . Пакет *wordnet* містить багато засобів для здійснення таких вимірювань (Leacock-Chodorow, Wu-Palmer, Resnik, Jiang-Conrath, Lin ). Наприклад *path\_similarity* (присвоює значення від 0 до 1) базується на найкоротшому шляху, який поєднує поняття за ієрархією гіперонімів (-1 означає що шлях (спільний гіперонім) не знайдено).

**Практичні завдання**

**Завдання 1:** Дослідити зв’язки голонім-меронім для іменників. Знайти іменники для демонстрації наступних зв’язків: member\_meronyms(), part\_meronyms(), substance\_meronyms(), member\_holonyms(), part\_holonyms(), та substance\_holonyms().

*Програма:*

import nltk

from nltk.corpus import wordnet as wn

print('Part meronyms of "library" are:')

print(wn.synset('library.n.03').part\_meronyms())

print('Member meronyms of "family" are:')

print(wn.synset('family.n.02').member\_meronyms())

print('Substance meronyms of "water" are:')

print(wn.synset('water.n.01').substance\_meronyms())

print('Part holonyms of "atom" are:')

print(wn.synset('atom.n.01').part\_holonyms())

print('Member holonyms of "letter" are:')

print(wn.synset('letter.n.02').member\_holonyms())

print('Substance holonyms of "oxygen" are:')

print(wn.synset('oxygen.n.01').substance\_holonyms())

*Результат:*

Part meronyms of "library" are:

[Synset('carrel.n.02'), Synset('reading\_room.n.01'), Synset('stacks.n.02')]

Member meronyms of "family" are:

[Synset('child.n.02'), Synset('parent.n.01'), Synset('sibling.n.01')]

Substance meronyms of "water" are:

[Synset('hydrogen.n.01'), Synset('oxygen.n.01')]

Part holonyms of "atom" are:

[Synset('chemical\_element.n.01'), Synset('molecule.n.01')]

Member holonyms of "letter" are:

[Synset('alphabet.n.01')]

Substance holonyms of "oxygen" are:

[Synset('air.n.01'), Synset('ozone.n.01'), Synset('water.n.01')]

**Завдання 5:** Який відсоток синсетів іменників не мають гіпонімів? До всіх синсетів можна доступитися за допомогою wn.all\_synsets('n').

*Програма:*

import nltk

from nltk.corpus import wordnet as wn

k=0

n=0

for x in wn.all\_synsets('n'):

hyp=x.hyponyms();

if not hyp:

k=k+1;

n=n+1

print('Total noun synsets:', n)

print('Without hyponims:', k)

print('Percentage without hyponims:', k/n\*100, ‘%’)

*Результат:*

Total noun synsets: 82115

Without hyponims: 65422

Percentage without hyponims: 79.67119283931072 %

**Завдання 4:** Здійснити аналіз словника вимов. Знайти скільки різних слів він містить. Який відсоток слів з цього словника можуть мати різну вимову?

*Програма:*

import nltk

from nltk.corpus import wordnet as wn

from nltk import FreqDist

entries = nltk.corpus.cmudict.entries()

words=[]

for word, pron in entries:

words.append(word)

k=len(entries)

n=len(set([w.lower() for w in words]))

print('Number of entries:', k)

print('Number of original words:', n)

print('Words with multiple variants of pronunciation:', 100-n/k\*100, '%')

*Результат:*

Number of entries: 133737

Number of original words: 123455

Words with multiple variants of pronunciation: 7.688223902136286 %

**Завдання 9:** Модифікувати програму генерації випадкового тексту для виконання наступного: тренування програми на текстах двох різних жанрів та генерації тексту об’єднаного жанру

*Програма:*

import nltk

from nltk.corpus import state\_union

from nltk.corpus import gutenberg

def generate\_model(cfdist, word, num=10):

for i in range(num):

print (word),

word = cfdist[word].max()

text1 = nltk.corpus.state\_union.words('1963-Johnson.txt')

text2 = nltk.corpus.gutenberg.words('chesterton-ball.txt')

text=text1+text2

bigrams = nltk.bigrams(text)

cfd = nltk.ConditionalFreqDist(bigrams)

generate\_model(cfd,'dress')

*Результат:*

dress

was

a

man

,

and

the

other

,

and

**Завдання 12:** Полісемія - це явище коли одне слово має декілька значень ( іменник dog має 7 значень, кількість яких визначити можна як len(wn.synsets('dog', 'n'))). Знайдіть середнє значення полісемії для дієслів.

*Програма:*

*Результат:*

**Завдання 16:** Використовуючи один з методів визначення подібності слів побудуйте відсортований по спаданню список значень подібності для наступних пар слів: monk-oracle, cemetery-woodland, food-rooster, coast-hill, forest-graveyard, shore-woodland, monk-slave, coast-forest, lad-wizard, chord-smile, glass-magician, rooster-voyage, noon-string.

*Програма:*

import nltk

from nltk.corpus import wordnet as wn

list=[]

monk=wn.synset('monk.n.01')

oracle=wn.synset('oracle.n.01')

list.append(monk.path\_similarity(oracle))

cemetery=wn.synset('cemetery.n.01')

woodland=wn.synset('woodland.n.01')

list.append(cemetery.path\_similarity(woodland))

food=wn.synset('food.n.01')

rooster=wn.synset('rooster.n.01')

list.append(food.path\_similarity(rooster))

coast=wn.synset('coast.n.01')

hill=wn.synset('hill.n.01')

list.append(coast.path\_similarity(hill))

forest=wn.synset('forest.n.01')

graveyard=wn.synset('graveyard.n.01')

list.append(forest.path\_similarity(graveyard))

shore=wn.synset('shore.n.01')

woodland=wn.synset('woodland.n.01')

list.append(shore.path\_similarity(woodland))

monk=wn.synset('monk.n.01')

slave=wn.synset('slave.n.01')

list.append(monk.path\_similarity(slave))

forest=wn.synset('forest.n.01')

coast=wn.synset('coast.n.01')

list.append(coast.path\_similarity(forest))

lad=wn.synset('lad.n.01')

wizard=wn.synset('wizard.n.01')

list.append(lad.path\_similarity(wizard))

chord=wn.synset('chord.n.01')

smile=wn.synset('smile.n.01')

list.append(chord.path\_similarity(smile))

glass=wn.synset('glass.n.01')

magician=wn.synset('magician.n.01')

list.append(glass.path\_similarity(magician))

rooster=wn.synset('rooster.n.01')

voyage=wn.synset('voyage.n.01')

list.append(rooster.path\_similarity(voyage))

noon=wn.synset('noon.n.01')

string=wn.synset('string.n.01')

list.append(noon.path\_similarity(string))

print(sorted(list))

*Результат:*

[0.041666666666666664, 0.058823529411764705, 0.0625, 0.07142857142857142, 0.09090909090909091, 0.09090909090909091, 0.1111111111111111, 0.1111111111111111, 0.125, 0.2, 0.2, 0.2, 0.2]

**Висновок:** в цій лабораторній роботі я вивчила методи доступу та роботи з лексичним ресурсами, ознайомилась з семантичним словник англійської мови WordNet.