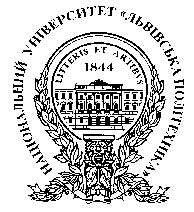
МІНІСТЕРСТВО ОСВІТИ І НАУКИ, МОЛОДІ ТА СПОРТУ УКРАЇНИ

Національний Університет “Львівська політехніка”



**Лабораторна робота № 4**

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

***ДОСТУП ТА РОБОТА З ЛЕКСИЧНИМИ РЕСУРСАМИ.***

Виконав:

студент групи ПРЛс-11

Форманюк А.А.

Прийняв:

асистент

Дупак Б.П.

Львів 2015

**МЕТА РОБОТИ**

* Вивчення основ програмування на мові *Python*.
* Вивчення методів доступу та роботи з лексичним ресурсами.
* Семантичний словник англійської мови WordNet.

**КОРОТКІ ТЕОРЕТИЧНІ ВІДОМОСТІ**

1. ***Поняття функції та модуля***.

При програмуванні часто необхідно частину програми виконати (використати) декілька разів. Наприклад, потрібно написати програму, яка здійснює утворення множини з однини іменників і вона буде виконуватись в різних місцях програми. Швидше ніж повторювати той самий код декілька разів і більш ефективно і надійно організувати цю роботу через функцію. 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. Розглянемо приклад. Функція *plural ()* отримує на вході однину іменника і формує множину на виході.

|  |
| --- |
| def plural(word):  if word.endswith('y'):  return word[:-1] + 'ies'  elif word[-1] in 'sx' or word[-2:] in ['sh', 'ch']:  return word + 'es'  elif word.endswith('an'):  return word[:-2] + 'en'  else:  return word + 's' |

|  |  |
| --- | --- |
|  | >>> plural('fairy')  'fairies'  >>> plural('woman')  'women' |

Розробляючи ту чи іншу програму протягом довшого періоду додаючи до неї нові функції і змінюючи існуючі або розробляючи декілька версій однієї програми потрібно зберігати тексти програм в окремих файлах і організовувати доступ до відповідних функцій в цих програмах. Збережемо текст останньої функції *plural ()* в окремому файлі textproc.py. тепер можна доступитися до цієї функції імпортувавши її з файлу.

>>> from textproc import plural

>>> plural('wish')

wishes

>>> plural('fan')

fen

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. Запропонований підхід генерації тексту швидко приводить до зациклювання, якого можна уникнути якщо вибирати наступні слова випадковим чином.

|  |
| --- |
| def generate\_model(cfdist, word, num=15):  for i in range(num):  print word,  word = cfdist[word].max()  text = nltk.corpus.genesis.words('english-kjv.txt')  bigrams = nltk.bigrams(text)  cfd = nltk.ConditionalFreqDist(bigrams) |

|  |  |
| --- | --- |
|  | >>> print cfd['living']  <FreqDist: 'creature': 7, 'thing': 4, 'substance': 2, ',': 1, '.': 1, 'soul': 1>  >>> generate\_model(cfd, 'living')  living creature that he said , and the land of the land of the land |

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. Наприклад, якщо визначити текст my\_text тоді vocab = sorted(set(my\_text)) побудує словник тексту my\_text, word\_freq = FreqDist(my\_text) визначить частоту кожного слова в тексті. vocab та word\_freq – приклад простих лексичних ресурсів. Так само конкорданс дає інформацію про використання слів і ця інформація може бути використана при побудові словників. Стандартна термінологія для словників (англ. мова) представлена на Рис.1. Словникова стаття містить основне слово (лему), та відповідну інформацію (частина мови значення слова).

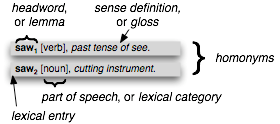
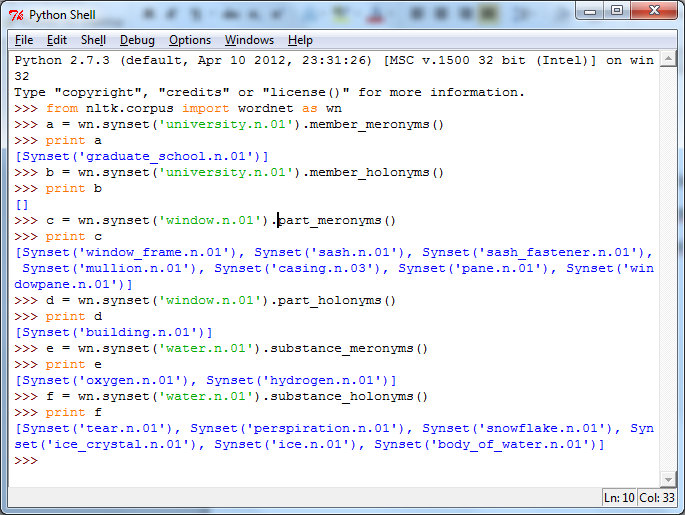


Рис.1. Термінологія англійської мови для записів словників.

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. 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().



1. Побудувати умовний частотний розподіл для корпусу імен. Знайти які перші літери частіше використовуються в чоловічих та жіночих іменах.

**import nltk**

**names = nltk.corpus.names**

**names.fileids()**

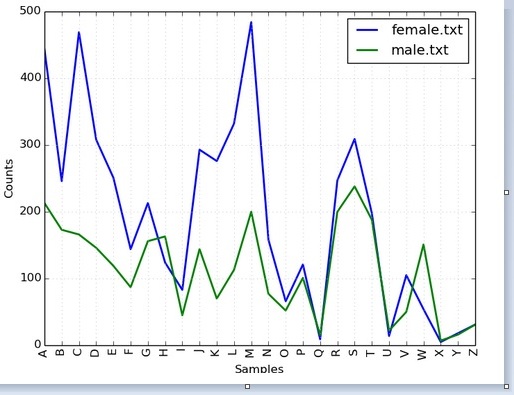
**cfd = nltk.ConditionalFreqDist(**

**(fileid, name[0])**

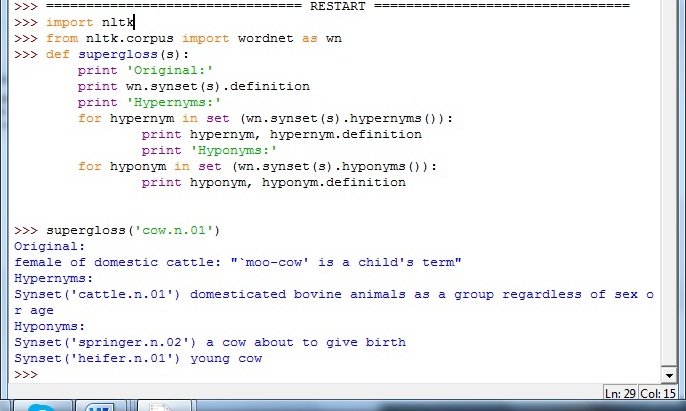
**for fileid in names.fileids()**

**for name in names.words(fileid)**

**cfd.plot()**



1. Визначити функцію supergloss(s) , яка буде приймати синсет s як аргумент і повертати стрічку в якій будуть поєднані всі описи всіх значень синсету s та описи всіх гіпернімів та гіпонімів s.



1. Модифікувати програму генерації випадкового тексту для виконання наступного: зберігати можливі наступні слова у списку та вибирати їх за допомогою random.choice() попередньо виконавши import random.

**import nltk**

**from nltk.corpus import genesis**

**import random**

**def generate\_model(cfdist,word,num=29):**

**n\_list = []**

**for i in range(num):**

**n\_list.append(word)**

**word=random.choice(cfdist[word].keys())**

**return m\_list**

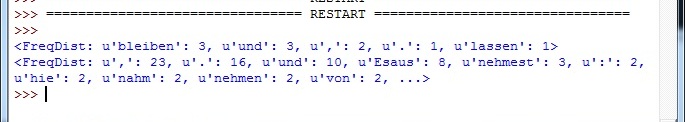
**text=nltk.corpus.genesis.words('german.txt')**

**bigrams = nltk.bigrams(text)**

**cfd=nltk.ConditionalFreqDist(bigrams)**

**print cfd ['leben']**

**print cfd ['Frau']**



1. 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 іменників.

**from nltk.corpus import wordnet as wn**

**d = []**

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

**for j in i.lemma\_names:**

**d.append(j)**

**num = len(set(d))**

**print num**

**q = 0**

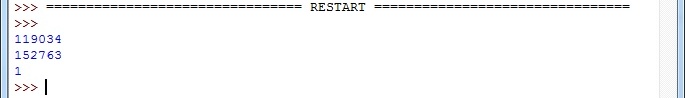
**for i in set(d):**

**q+= len(wn.synsets(i,'n'))**

**print q**

**poly = q/num**

**print poly**



1. Використовуючи один з методів визначення подібності слів побудуйте відсортований по спаданню список значень подібності для наступних пар слів: car-automobile, gem-jewel, journey-voyage, boy-lad, coast-shore, asylum-madhouse, magician-wizard, midday-noon, furnace-stove, food-fruit, bird-cock.

**import nltk**

**from nltk.corpus import wordnet as wn**

**def similarity(word1, word2):**

**synset1 = wn.synset(word1 + '.n.01')**

**synset2 = wn.synset(word2 + '.n.01')**

**return synset1.path\_similarity(synset2)**

**list = []**

**list.append(similarity('car', 'automobile'))**

**list.append(similarity('jem', 'jewel'))**

**list.append(similarity('journey', 'voyage'))**

**list.append(similarity('boy', 'lad'))**

**list.append(similarity('coast', 'shore'))**

**list.append(similarity('asylum', 'madhouse'))**

**list.append(similarity('magician', 'wizard'))**

**list.append(similarity('midday', 'noon'))**

**list.append(similarity('furnace', 'stove'))**

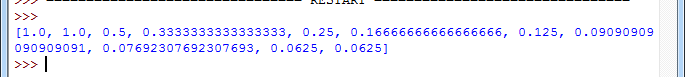
**list.append(similarity('food', 'fruit'))**

**list.append(similarity('bird', 'cock'))**

**a=sorted(list)**

**a.reverse()**

**print a**

****

**Висновок:** на цій лабораторній роботі я вивчив бібліотоку прикладних програм nltk. Ознайомився з теоретичними відомостями. Навчився працювати з методами доступу та роботи до лексичних ресурсів.