**Завдання (Варіант 16)**

**2. Проаналізувати таблицю 4.**

| **Linguistic Object** | **Maps From** | **Maps To** |
| --- | --- | --- |
| Document Index | Word | List of pages (where word is found) |
| Thesaurus | Word sense | List of synonyms |
| Dictionary | Headword | Entry (part-of-speech, sense definitions, etymology) |
| Comparative Wordlist | Gloss term | Cognates (list of words, one per language) |
| Morph Analyzer | Surface form | Morphological analysis (list of component morphemes) |

У даній таблиці представлено лінгвістичні об’єкти та відповідності, яких вони потребують. До основних лінгвістичних даних належать індексовані документи, тезаурус, словник, порівняльний список слів та морфологічний аналізатор.

|  |  |  |
| --- | --- | --- |
| **Лінгвістичний об'єкт** | **Позначуване** | **Позначене** |
| Картка платника податків | Номер картки | Дані про особу |
| Словник антонімів | Слово | Антоніми до слова |
| Словник термінів | Слово | Визначення терміну |
| Список неправильних дієслів | Дієслово | Форми past tense та past participle. |

**3. Опрацювати приклади з методичних вказівок по роботі зі словниками. Що станеться, якщо доступитися до неіснуючого запису звичайного словника та словника по замовчуванню?**

*Звичайний словник*

>>> pos = {'colorless': 'ADJ', 'ideas': 'N', 'sleep': 'V', 'furiously': 'ADV'}

>>> pos

{'furiously': 'ADV', 'sleep': 'V', 'ideas': 'N', 'colorless': 'ADJ'}

>>> pos['night']

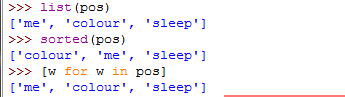
Traceback (most recent call last):

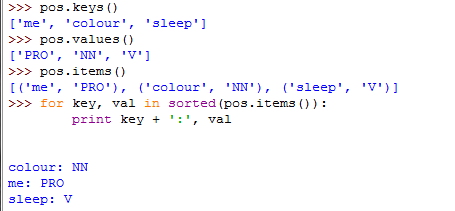
File "<pyshell#5>", line 1, in <module>

pos['night']

KeyError: 'night'

Якщо доступатися до неіснуючого запису звичайного словника, то вибиває помилку.

До словника можна застосовувати функції list() та sorted(), а також ітерацію 



Функція keys() виводить значення ключів, values() - значень, items() - елементів

**Словник за замовчуванням**

>>> pos = nltk.defaultdict(list)

>>> pos['blog']

[] #пуста множина, бо не вказано ніяких параметрів для значення

>>> pos = nltk.defaultdict(lambda: 'V') #доступаюся до неіснуючого запису словника

>>> pos['google'] #автоматично додається

'V'

У випадку зі словником за замовчуванням слово автоматично додається до словника.

**7. Використовуючи sorted() та set()отримайте відсортований список всіх тегів корпуса Brown без їх дублювання.**

>>> import nltk

>>> text=nltk.corpus.brown.tagged\_words()

>>> tags=sorted(set(tag for (word,tag) in text))

>>> tags[:30]

["'", "''", '(', '(-HL', ')', ')-HL', '\*', '\*-HL', '\*-NC', '\*-TL', ',', ',-HL', ',-NC', ',-TL', '--', '---HL', '.', '.-HL', '.-NC', '.-TL', ':', ':-HL', ':-TL', 'ABL', 'ABN', 'ABN-HL', 'ABN-NC', 'ABN-TL', 'ABX', 'AP']

**10. Напишіть програму, яка обробить *Brown Corpus* і допоможе відповісти на наступне запитання: які теги найчастіше зустрічаються (створити список 20 тегів, які мають максимальну частоту);**

>>> import nltk

>>> text=nltk.corpus.brown.tagged\_words()

>>> tags=[t for (w,t) in text]

>>> fd = nltk.FreqDist(tags)

>>> fd.keys()[:20]

['NN', 'IN', 'AT', 'JJ', '.', ',', 'NNS', 'CC', 'RB', 'NP', 'VB', 'VBN', 'VBD', 'CS', 'PPS', 'VBG', 'PP$', 'TO', 'PPSS', 'CD']

>>> nltk.help.brown\_tagset('IN')

IN: preposition

of in for by considering to on among at through with under into

regarding than since despite according per before toward against as

after during including between without except upon out over ...

**12.Напишіть програму для збору статистичних даних по розмічених корпусах і відповіді на наступне запитання: який відсоток типів слів (types) завжди маркуються тими самими тегами.**

import nltk

from nltk.corpus import brown

brown\_fiction\_tags=brown.tagged\_words(categories='fiction',simplify\_tags=True)

data=nltk.ConditionalFreqDist((word.lower(),tag) for (word,tag) in brown\_fiction\_tags)

result=[]

for word in data.conditions():

if len(data[word])==1:

tags=data[word].keys()

result.append(word)

print len(result)

print len (brown\_fiction\_tags)

print len(result)\*100/len(brown\_fiction\_tags)

>>>

7914

68488

11

>>>

**20. Напишіть програму для знаходження всіх слів, які маркуються як QL в корпусі Brown і зустрічаються перед наступними дієсловами adore, love, like, prefer.**

>>> def process(sentence):

for (w1, t1), (w2, t2) in nltk.bigrams(sentence):

if (t1 =='QL' and w2 == 'like') or (t1 =='QL' and w2 == 'adore') or (t1 =='QL' and w2 == 'love') or (t1 =='QL' and w2=='prefer'):

print w1, w2

>>> for tagged\_sent in brown.tagged\_sents():

process(tagged\_sent)

quite like

much like

much like

so like

quite like

quite like

**21. Написати програму побудови словника, записами якого будуть набори словників. Використовуючи створений словник, збережіть у ньому набори можливих тегів, які зустрічаються після заданого слова з певним тегом, наприклад wordi → tagi → tagi+1.**

>>> import nltk

>>> from nltk.corpus import brown

>>> pos = nltk.defaultdict(lambda: nltk.defaultdict(int))

>>> btags = brown.tagged\_words(simplify\_tags=True)

>>> for ((w1, t1), (w2, t2)) in nltk.ibigrams(btags):

pos[(w1, t1)][t2] += 1

>>> pos [('curly', 'ADJ')]

defaultdict(<type 'int'>, {'ADJ': 2, 'N': 2})\_

>>> pos [('spring', 'N')]

defaultdict(<type 'int'>, {'ADV': 2, 'ADJ': 1, 'VD': 3, 'WH': 5, '(': 1, 'PRO': 2, ',': 10, 'VN': 1, '.': 16, "''": 1, 'P': 25, 'UH': 1, 'V': 1, 'N': 21, 'CNJ': 11})