МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ "ЛЬВІВСЬКА ПОЛІТЕХНІКА"
ІНСТИТУТ КОМП'ЮТЕРНИХ НАУК ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра "Системи автоматизованого проектування"

Звіт

до лабораторної роботи №10

на тему: ВИВЧЕННЯ БІБЛІОТЕКИ ПРИКЛАДНИХ ПРОГРАМ NLTK, ДЛЯ ОПРАЦЮВАННЯ ТЕКСТІВ ПРИРОДНОЮ МОВОЮ. АВТОМАТИЧНИЙ МОРФОЛОГІЧНИЙ АНАЛІЗ (частина2).

з дисципліни "Комп'ютерна лінгвістика"

Виконала:
студентка групи ПРЛм-11
Гарбуз Л.В.
Прийняв:
викладач
Дупак Б.П.

Львів 2015

Мета роботи: вивчення основ програмування на мові Python. Ознайомлення з автоматичним морфологічним аналізом в NLTK.

Тексти програм на мові *Python*.

Варіант – 3

1. Здійсніть тренування юніграм аналізатора на основі частини корпуса, який відповідає першій або другій літері прізвища студента та виконайте аналіз тексту з частини корпуса, яка відповідає першій або другій літері імені студента. Результати поясніть. Чому для деяких слів не встановлені теги.

```
import nltk
from nltk.corpus import brown
brown_tagged_sents_for_training=brown.tagged_sents(categories='editorial')
unigram_tagger=nltk.UnigramTagger(brown_tagged_sents_for_training)
brown_sents_for_test=brown.sents(categories='government')
for sent_index in range(0,10):
    sent=brown_sents_for_test[sent_index]
    result=unigram_tagger.tag(sent)
print result
```

```
>>>
[('The', 'AT'), ('Small', 'JJ'), ('Business', 'NN-TL'), ('Administration', 'NN-T
L'), ('(', '('), ('SBA', None), (')', ')'), ('provides', 'VBZ'), ('guidance', 'N
N'), ('and', 'CC'), ('advice', 'NN'), ('on', 'IN'), ('sources', 'NNS'), ('of', '
IN'), ('technical', 'JJ'), ('information', 'NN'), ('relating', None), ('to', 'TO
'), ('small', 'JJ'), ('business', 'NN'), ('management', 'NN'), ('and', 'CC'), ('
research', 'NN'), ('and', 'CC'), ('development', 'NN'), ('of', 'IN'), ('products
', None), ('.', '.')]
>>>
```

2. Прочитати файл допомого про морфологічний аналізатор на основі афіксів (help(nltk.AffixTagger)). Напишіть програму, яка викликає аналізатор на основі афіксів в циклі, з різними значеннями довжини афіксів і мінімальними довжинами слів. При яких значеннях можна отримати кращі результати.

```
import nltk
from nltk.corpus import brown
brown_tag=brown.tagged_sents(categories='romance')
sent=brown.sents(categories='belles_lettres')[1]
affix_size=[-1,-2,-3]
stem_size=[2,3]
for i in affix_size:
    for j in stem_size:
        tagger=nltk.AffixTagger(brown_tag, affix_length=i, min_stem_length=j)
        analyzed=tagger.tag(sent)
        evaluation=tagger.evaluate(brown_tag)
        print'Analyzed sentence'
        print analyzed
        print 'Tagger', tagger, 'affix_size=',i,'min_stem_length=',j
        print 'Evaluation:',evaluation
        print
```

```
>>>
Analyzed sentence
[('They', 'RB'), ('have', 'AT'), ('also', 'IN'), ('led', 'VBD'), ('the', 'AT'),
('nation', 'NN'), ('in', None), ('the', 'AT'), ('direction', 'NN'), ('of', None)
, ('a', None), ('welfare', 'AT'), ('state', 'AT'), ('.', None)]
Tagger <AffixTagger: size=37> affix_size= -1 min_stem_length= 2
Evaluation: 0.206078089743

Analyzed sentence
[('They', 'RB'), ('have', 'NN'), ('also', 'IN'), ('led', None), ('the', None), (
'nation', 'NN'), ('in', None), ('the', None), ('direction', 'NN'), ('of', None),
 ('a', None), ('welfare', 'NN'), ('state', 'NN'), ('.', None)]
Tagger <AffixTagger: size=35> affix_size= -1 min_stem_length= 3
Evaluation: 0.170217931507

Analyzed sentence
[('They', 'PPSS'), ('have', 'HV'), ('also', 'RB'), ('led', None), ('the', None),
 ('nation', 'NN'), ('in', None), ('the', None), ('direction', 'NN'), ('of', None
), ('a', None), ('welfare', 'BED'), ('state', 'JJ'), ('.', None)]
Tagger <AffixTagger: size=262> affix_size= -2 min_stem_length= 2
Evaluation: 0.241324155265

Analyzed sentence
[('They', None), ('have', None), ('also', None), ('led', None), ('the', None), (
'nation', 'NN'), ('in', None), ('the', None), ('direction', 'NN'), ('of', None),
 ('a', None), ('welfare', 'EX'), ('state', 'JJ'), ('.', None)]
Tagger <AffixTagger: size=226> affix_size= -2 min_stem_length= 3
Evaluation: 0.171817428808

Analyzed sentence
[('They', None), ('have', None), ('also', None), ('led', None), ('the', None), (
'nation', 'NN'), ('in', None), ('the', None), ('direction', 'NN'), ('of', None),
 ('a', None), ('welfare', 'VB'), ('state', 'JJ'), ('.', None)]
Tagger <AffixTagger: size=1023> affix_size= -3 min_stem_length= 2
Evaluation: 0.206692182457

Analyzed sentence
[('They', None), ('have', None), ('also', None), ('led', None), ('the', None), (
'nation', 'NN'), ('in', None), ('the', None), ('direction', 'NN'), ('of', None),
 ('a', None), ('welfare', 'NN'), ('state', None), ('.', None)]
Tagger <AffixTagger: size=840> affix_size= -3 min_stem_length= 3
Evaluation: 0.146782439805

>>>
```

3. Здійсніть тренування біграм аналізатора на частинах корпуса з вправи 3.1 без backoff аналізатора. Перевірте його роботу. Що відбулося з продуктивністю аналізатора? Чому?

```
import nltk
from nltk.corpus import brown
brown_tagged_sents_for_training=brown.tagged_sents(categories='reviews')
bigram_tagger=nltk.BigramTagger(brown_tagged_sents_for_training)
brown_sents_for_test=brown.sents(categories='learned')
for sent_index in range(0,10):
    sent=brown_sents_for_test[sent_index]
    result=bigram_tagger.tag(sent)
    print result
```

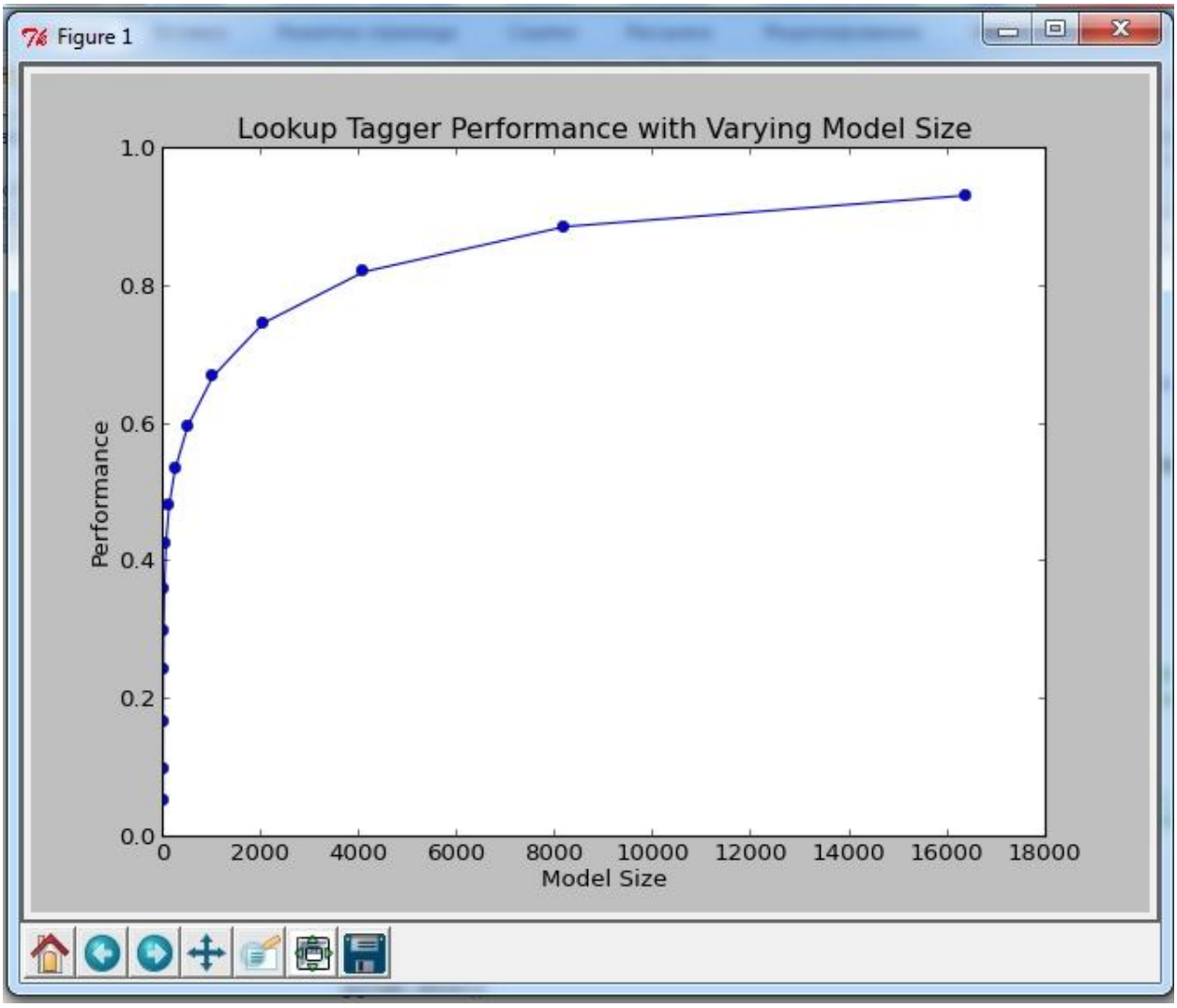
```
>>>
[('1', None), ('.', None)]
[('Introduction', None)]
[('It', 'PPS'), ('has', 'HVZ'), ('recently', 'RB'), ('become', 'VBN'), ('practical', None), ('to', None), ('use', None), ('the',
None), ('radio', None), ('emission', None), ('of', None), ('the', None), ('moon', None), ('and', None), ('planets', None), ('as',
None), ('a', None), ('new', None), ('source', None), ('of', None), ('information', None), ('about', None), ('these', None), ('bod
ies', None), ('and', None), ('their', None), ('atmospheres', None), ('.', None)]
[('The', 'AT'), ('results', 'NNS'), ('of', 'IN'), ('present', 'JJ'), ('observations', None), ('of', None), ('the', None), ('therm
al', None), ('radio', None), ('emission', None), ('of', None), ('the', None), ('moon', None), ('are', None), ('consistent', None)
, ('with', None), ('the', None), ('very', None), ('low', None), ('thermal', None), ('conductivity', None), ('of', None), ('the',
None), ('surface', None), ('layer', None), ('which', None), ('was', None), ('derived', None), ('from', None), ('the', None), ('va
riation', None), ('in', None), ('the', None), ('infrared', None), ('emission', None), ('during', None), ('eclipses', None), ('(',
None), ('e.g.', None), (',', None), ('Garstung', None), (',', None), ('1958', None), (')', None), ('.', None)]
[('When', 'WRB'), ('sufficiently', None), ('accurate', None), ('and', None), ('complete', None), ('measurements', None), ('are',
None), ('available', None), (',', None), ('it', None), ('will', None), ('be', None), ('possible', None), ('to', None), ('set', No
ne), ('limits', None), ('on', None), ('the', None), ('thermal', None), ('and', None), ('electrical', None), ('characteristics', N
one), ('of', None), ('the', None), ('surface', None), ('and', None), ('subsurface', None), ('materials', None), ('of', None), ('t
he', None), ('moon', None), ('.', None)]
[('Observations', None), ('of', None), ('the', None), ('radio', None), ('emission', None), ('of', None), ('a', None), ('planet',
None), ('which', None), ('has', None), ('an', None), ('extensive', None), ('atmosphere', None), ('will', None), ('probe', None),
('the', None), ('atmosphere', None), ('to', None), ('a', None), ('greater', None), ('extent', None), ('than', None), ('those', No
ne), ('using', None), ('shorter', None), ('wave', None), ('lengths', None), ('and', None), ('should', None), ('in', None), ('some
', None), ('cases', None), ('give', None), ('otherwise', None), ('unobtainable', None), ('information', None), ('about', None), (
'the', None), ('characteristics', None), ('of', None), ('the', None), ('solid', None), ('surface', None), ('.', None)]
[('Radio', 'NN'), ('observations', None), ('of', None), ('Venus', None), ('and', None), ('Jupiter', None), ('have', None), ('alre
ady', None), ('supplied', None), ('unexpected', None), ('experimental', None), ('data', None), ('on', None), ('the', None), ('phy
sical', None), ('conditions', None), ('of', None), ('these', None), ('planets', None), ('.', None)]
[('The', 'AT'), ('observed', None), ('intensity', None), ('of', None), ('the', None), ('radio', None), ('emission', None), ('of',
None), ('Venus', None), ('is', None), ('much', None), ('higher', None), ('than', None), ('the', None), ('expected', None), ('ther
mal', None), ('intensity', None), (',', None), ('although', None), ('the', None), ('spectrum', None), ('indicated', None), ('by',
None), ('measurements', None), ('at', None), ('wave', None), ('lengths', None), ('near', None), ('3', None), ('cm', None), ('and'
, None), ('10', None), ('cm', None), ('is', None), ('like', None), ('that', None), ('of', None), ('a', None), ('black', None), ('
body', None), ('at', None), ('about', None), ('600-degrees', None), ('.', None)]
[('This', 'DT'), ('result', None), ('suggests', None), ('a', None), ('very', None), ('high', None), ('temperature', None), ('at',
None), ('the', None), ('solid', None), ('surface', None), ('of', None), ('the', None), ('planet', None), (',', None), ('although'
, None), ('there', None), ('is', None), ('the', None), ('possibility', None), ('that', None), ('the', None), ('observed', None),
('radiation', None), ('may', None), ('be', None), ('a', None), ('combination', None), ('of', None), ('both', None), ('thermal', N
one), ('and', None), ('non-thermal', None), ('components', None), ('and', None), ('that', None), ('the', None), ('observed', None
), ('spectrum', None), ('is', None), ('that', None), ('of', None), ('a', None), ('black', None), ('body', None), ('merely', None)
, ('by', None), ('coincidence', None), ('.', None)]
[('For', 'IN'), ('the', 'AT'), ('case', 'NN'), ('of', 'IN'), ('Jupiter', None), (',', None), ('the', None), ('radio', None), ('em
ission', None), ('spectrum', None), ('is', None), ('definitely', None), ('not', None), ('like', None), ('the', None), ('spectrum'
, None), ('of', None), ('a', None), ('black-body', None), ('radiator', None), (',', None), ('and', None), ('it', None), ('seems',
None), ('very', None), ('likely', None), ('that', None), ('the', None), ('radiation', None), ('reaching', None), ('the', None), (
'earth', None), ('is', None), ('a', None), ('combination', None), ('of', None), ('thermal', None), ('radiation', None), ('from',
None), ('the', None), ('atmosphere', None), ('and', None), ('non-thermal', None), ('components', None), ('.', None)]
```

4. Дослідити наступні проблеми. що виникають при роботі з аналізатором на основі підстановок: що відбудеться з продуктивністю аналізатора, якщо опустити backoff аналізатор (дослідити на частині броунівського корпусу, яка відповідає першій або другій літері прізвища студента); на основі рис.1. та відповідного фрагмента програми встановити точку максимальної продуктивності незважаючи на розмір списку (об’єм оперативної пам’яті) і точку достатньої продуктивності при мінімальному розмірі списку.

```
>>> import nltk
>>> from nltk.corpus import brown
>>> def performance(cfd,wordlist):
    it = dict((word,cfd[word].max()) for word in wordlist)
    baseline_tagger=nltk.UnigramTagger(model=it)
    return baseline_tagger.evaluate(brown.tagged_sents(categories='hobbies'))


>>> def display():
    import pylab
    words_by_freq=list(nltk.FreqDist(brown.words(categories='hobbies')))
    cfd=nltk.ConditionalFreqDist(brown.tagged_words(categories='hobbies'))
    sizes=2**pylab.arange(15)
    perfs=[performance(cfd,words_by_freq[:size]) for size in sizes]
    pylab.plot(sizes,perfs,'-bo')
    pylab.title('Lookup Tagger Performance with Varying Model Size')
    pylab.xlabel('Model Size')
    pylab.ylabel('Performance')
    pylab.show()

>>> display()
```



5. Знайдіть розмічені корпуси текстів для інших мов які вивчаєте або володієте (українська, польська, німецька, російська, італійська, японська). Здійсніть тренування та оцініть продуктивність роботи різних аналізаторів та комбінацій різних аналізаторів. Точність роботи аналізаторів порівняйте з точністю роботи аналізаторів для англійських корпусів. Результати поясніть.

```
import nltk
def taggers(tagged_sents,data_koef):
    size=int(len(tagged_sents)*data_koef)
    train_sents=tagged_sents[:size]
    test_sents=tagged_sents[size:]
    t0=nltk.DefaultTagger('NN')
    t1=nltk.UnigramTagger(train_sents,backoff=t0)
    t2=nltk.BigramTagger(train_sents,backoff=t1)
    t3=nltk.TrigramTagger(train_sents,backoff=t2)
    return t3.evaluate(test_sents)
def taggers1(tagged_sents,data_koef):
    size=int(len(tagged_sents)*data_koef)
    train_sents=tagged_sents[:size]
    test_sents=tagged_sents[size:]
    t0=nltk.DefaultTagger('NN')
    t1=nltk.UnigramTagger(train_sents,backoff=t0)
    t2=nltk.BigramTagger(train_sents,backoff=t1)
    return t2.evaluate(test_sents)
def taggers2(tagged_sents,data_koef):
    size=int(len(tagged_sents)*data_koef)
    train_sents=tagged_sents[:size]
    test_sents=tagged_sents[size:]
    t0=nltk.DefaultTagger('NN')
    t1=nltk.UnigramTagger(train_sents,backoff=t0)
    t2=nltk.BigramTagger(train_sents, cutoff=2,backoff=t1)
    t3=nltk.TrigramTagger(train_sents,cutoff=2,backoff=t2)
    return t3.evaluate(test_sents)
print 'English'
tagged_sents=nltk.corpus.brown.tagged_sents(categories='humor')
koef=0.9
print 'DefaulTagger+UnigramTagger+BigramTagger'
print (str(koef*100)+'%',taggers(tagged_sents,koef))
print 'DefaulTagger+UnigramTagger+BigramTagger+TrigramTagger'
print (str(koef*100)+'%',taggers1(tagged_sents,koef))
print 'DefaulTagger+UnigramTagger+BigramTagger+TrigramTagger with cutoff'
print (str(koef*100)+'%',taggers2(tagged_sents,koef))
print 'German'
tagged_sents=nltk.corpus.floresta.tagged_sents()
koef=0.9
print 'DefaulTagger+UnigramTagger+BigramTagger'
print (str(koef*100)+'%',taggers(tagged_sents,koef))
print 'DefaulTagger+UnigramTagger+BigramTagger+TrigramTagger'
print (str(koef*100)+'%',taggers1(tagged_sents,koef))
print 'DefaulTagger+UnigramTagger+BigramTagger+TrigramTagger with cutoff'
print (str(koef*100)+'%',taggers2(tagged_sents,koef))


>>>
English
DefaulTagger+UnigramTagger+BigramTagger
('90.0%', 0.7549591598599766)
DefaulTagger+UnigramTagger+BigramTagger+TrigramTagger
('90.0%', 0.7569039284325165)
DefaulTagger+UnigramTagger+BigramTagger+TrigramTagger with cutoff
('90.0%', 0.7576818358615325)
German
DefaulTagger+UnigramTagger+BigramTagger
('90.0%', 0.7882029697182276)
DefaulTagger+UnigramTagger+BigramTagger+TrigramTagger
('90.0%', 0.7911843797497954)
DefaulTagger+UnigramTagger+BigramTagger+TrigramTagger with cutoff
('90.0%', 0.7843446743832574)
>>>
```

6. Створити аналізатор по замовчуванню та набір юніграм і n-грам аналізаторів. Використовуючи backoff здійсніть тренування аналізаторів на частині корпуса з вправи 2. Дослідіть три різні комбінації поєднання цих аналізаторів. Перевірте точність роботи аналізаторів. Визначіть комбінацію

аналізаторів з максимальною точністю аналізу. Змініть розмір даних на яких проводилось тренування. Повторіть експерименти для змінених даних для тренування. Результати порівняйти і пояснити.

```
import nltk
def taggers(tagged_sents,data_koef):
    size=int(len(tagged_sents)*data_koef)
    train_sents=tagged_sents[:size]
    test_sents=tagged_sents[size:]
    t0=nltk.DefaultTagger('NN')
    t1=nltk.UnigramTagger(train_sents,backoff=t0)
    t2=nltk.BigramTagger(train_sents,backoff=t1)
    t3=nltk.TrigramTagger(train_sents,backoff=t2)
    return t3.evaluate(test_sents)
def taggers1(tagged_sents,data_koef):
    size=int(len(tagged_sents)*data_koef)
    train_sents=tagged_sents[:size]
    test_sents=tagged_sents[size:]
    t0=nltk.DefaultTagger('NN')
    t1=nltk.UnigramTagger(train_sents,backoff=t0)
    t2=nltk.BigramTagger(train_sents,backoff=t1)
    return t2.evaluate(test_sents)
def taggers2(tagged_sents,data_koef):
    size=int(len(tagged_sents)*data_koef)
    train_sents=tagged_sents[:size]
    test_sents=tagged_sents[size:]
    t0=nltk.DefaultTagger('NN')
    t1=nltk.UnigramTagger(train_sents,backoff=t0)
    t2=nltk.BigramTagger(train_sents, cutoff=2,backoff=t1)
    t3=nltk.TrigramTagger(train_sents,cutoff=2,backoff=t2)
    return t3.evaluate(test_sents)
tagged_sents=nltk.corpus.brown.tagged_sents(categories='science_fiction')
for koef in [0.5,0.6,0.7,0.8,0.9]:
    print 'DefaulTagger+UnigramTagger+BigramTagger'
    print (str(koef*100)+'%',taggers(tagged_sents,koef))
    print 'DefaulTagger+UnigramTagger+BigramTagger+TrigramTagger'
    print (str(koef*100)+'%',taggers1(tagged_sents,koef))
    print 'DefaulTagger+UnigramTagger+BigramTagger+TrigramTagger with cutoff'
    print (str(koef*100)+'%',taggers2(tagged_sents,koef))
```

```
>>>
DefaulTagger+UnigramTagger+BigramTagger
('50.0%', 0.7645580226225388)
DefaulTagger+UnigramTagger+BigramTagger+TrigramTagger
('50.0%', 0.763720150816925)
DefaulTagger+UnigramTagger+BigramTagger+TrigramTagger with cutoff
('50.0%', 0.7631615696131825)
DefaulTagger+UnigramTagger+BigramTagger
('60.0%', 0.7668845315904139)
DefaulTagger+UnigramTagger+BigramTagger+TrigramTagger
('60.0%', 0.7673872967990615)
DefaulTagger+UnigramTagger+BigramTagger+TrigramTagger with cutoff
('60.0%', 0.7658790011731188)
DefaulTagger+UnigramTagger+BigramTagger
('70.0%', 0.763355592654424)
DefaulTagger+UnigramTagger+BigramTagger+TrigramTagger
('70.0%', 0.7639816360601002)
DefaulTagger+UnigramTagger+BigramTagger+TrigramTagger with cutoff
('70.0%', 0.7623121869782972)
DefaulTagger+UnigramTagger+BigramTagger
('80.0%', 0.8025412087912088)
DefaulTagger+UnigramTagger+BigramTagger+TrigramTagger
('80.0%', 0.8035714285714286)
DefaulTagger+UnigramTagger+BigramTagger+TrigramTagger with cutoff
('80.0%', 0.8008241758241759)
DefaulTagger+UnigramTagger+BigramTagger
('90.0%', 0.8541266794625719)
DefaulTagger+UnigramTagger+BigramTagger+TrigramTagger
('90.0%', 0.8579654510556622)
DefaulTagger+UnigramTagger+BigramTagger+TrigramTagger with cutoff
('90.0%', 0.8515674984005118)
>>>
```

7. Прочитати стрічку документування функції demo Brill аналізатора. Здійснити експерименти з різними значення параметрів цієї функції. Встановити який взаємозв'язок є між часом тренування (навчання аналізатора) і точністю його роботи.

```
>>> import nltk
>>> nltk.tag.brill.demo(num_sents=1000, max_rules=100)
Loading tagged data...
Done loading.
Training unigram tagger:
    [accuracy: 0.800229]
Training bigram tagger:
    [accuracy: 0.805561]
Training Brill tagger on 800 sentences...
Finding initial useful rules...
    Found 4370 useful rules.

           B      |
   S   F   r   O  |        Score = Fixed - Broken
   c   i   o   t  |  R     Fixed = num tags changed incorrect -> correct
   o   x   k   h  |  u     Broken = num tags changed correct -> incorrect
   r   e   e   e  |  l     Other = num tags changed incorrect -> incorrect
   e   d   n   r  |  e
------------------+-------------------------------------------------------
   7   7   0   0  | WDT -> IN if the tag of words i+1...i+2 is 'NNP'
   5   7   2   0  | WDT -> IN if the tag of the following word is 'DT'
   5   6   1   0  | IN -> RB if the text of words i+1...i+2 is 'as'
   4   4   0   0  | RB -> IN if the tag of the preceding word is 'RB',
               |   and the tag of the following word is 'PRP'
   4   4   0   0  | VBP -> VB if the text of words i-2...i-1 is "n't"
   3   3   0   0  | RB -> IN if the tag of the preceding word is 'NN',
               |   and the tag of the following word is 'DT'
   3   4   1   0  | WDT -> IN if the tag of words i+1...i+2 is 'NNS'

Brill accuracy: 0.807084
Done; rules and errors saved to rules.yaml and errors.out.
>>> nltk.tag.brill.demo(num_sents=100, max_rules=50)
Loading tagged data...
Done loading.
Training unigram tagger:
    [accuracy: 0.688337]
Training bigram tagger:
    [accuracy: 0.690249]
Training Brill tagger on 80 sentences...
Finding initial useful rules...
    Found 78 useful rules.

           B      |
   S   F   r   O  |        Score = Fixed - Broken
   c   i   o   t  |  R     Fixed = num tags changed incorrect -> correct
   o   x   k   h  |  u     Broken = num tags changed correct -> incorrect
   r   e   e   e  |  l     Other = num tags changed incorrect -> incorrect
   e   d   n   r  |  e
------------------+-------------------------------------------------------

Brill accuracy: 0.690249
Done; rules and errors saved to rules.yaml and errors.out.
```

```
>>> nltk.tag.brill.demo(num_sents=2000, max_rules=200,train=0.4)
Loading tagged data...
Done loading.
Training unigram tagger:
    [accuracy: 0.802379]
Training bigram tagger:
    [accuracy: 0.806001]
Training Brill tagger on 800 sentences...
Finding initial useful rules...
    Found 4370 useful rules.

           B      |
   S   F   r   O  |        Score = Fixed - Broken
   c   i   o   t  |  R     Fixed = num tags changed incorrect -> correct
   o   x   k   h  |  u     Broken = num tags changed correct -> incorrect
   r   e   e   e  |  l     Other = num tags changed incorrect -> incorrect
   e   d   n   r  |  e
------------------+-------------------------------------------------------
   7   7   0   0  | WDT -> IN if the tag of words i+1...i+2 is 'NNP'
   5   7   2   0  | WDT -> IN if the tag of the following word is 'DT'
   5   6   1   0  | IN -> RB if the text of words i+1...i+2 is 'as'
   4   4   0   0  | RB -> IN if the tag of the preceding word is 'RB',
               |   and the tag of the following word is 'PRP'
   4   4   0   0  | VBP -> VB if the text of words i-2...i-1 is "n't"
   3   3   0   0  | RB -> IN if the tag of the preceding word is 'NN',
               |   and the tag of the following word is 'DT'
   3   4   1   0  | WDT -> IN if the tag of words i+1...i+2 is 'NNS'

Brill accuracy: 0.807594
Done; rules and errors saved to rules.yaml and errors.out.
```

```
>>> nltk.tag.brill.demo(num_sents=3000, max_rules=200,train=0.8)
Loading tagged data...
Done loading.
Training unigram tagger:
    [accuracy: 0.873951]
Training bigram tagger:
    [accuracy: 0.882669]
Training Brill tagger on 2400 sentences...
Finding initial useful rules...
    Found 14607 useful rules.

           B      |
   S   F   r   O  |        Score = Fixed - Broken
   c   i   o   t  |  R     Fixed = num tags changed incorrect -> correct
   o   x   k   h  |  u     Broken = num tags changed correct -> incorrect
   r   e   e   e  |  l     Other = num tags changed incorrect -> incorrect
   e   d   n   r  |  e
------------------+-------------------------------------------------------
  19  25   6   0  | WDT -> IN if the tag of words i+1...i+2 is 'DT'
  13  13   0   0  | WDT -> IN if the tag of the preceding word is
                  |   'NN', and the tag of the following word is 'PRP'
  12  21   9   0  | IN -> RB if the text of word i+2 is 'as'
  11  12   1   0  | WDT -> IN if the tag of the preceding word is
                  |   'NN', and the tag of the following word is 'NNP'
   9  12   3   0  | WDT -> IN if the tag of words i+1...i+2 is 'NNS'
   7   7   0   0  | RBR -> JJR if the tag of the following word is
                  |   'NN'
   7   7   0   0  | WDT -> IN if the tag of the preceding word is
                  |   'NNS', and the tag of the following word is
                  |   'PRP'
   4   4   0   0  | VBN -> VBD if the tag of the preceding word is
                  |   'NN', and the tag of the following word is 'DT'
   4   5   1   0  | VBP -> VB if the tag of words i-3...i-1 is 'MD'
   4   4   0   0  | RP -> IN if the text of the following word is 'of'
   4   4   0   0  | VBP -> VB if the text of words i-2...i-1 is "n't"
   3   8   5   1  | RBR -> JJR if the tag of words i+1...i+2 is 'CD'
   3   3   0   0  | RBR -> JJR if the tag of the following word is
                  |   'NNS'
   3   3   0   0  | RBR -> JJR if the tag of word i-2 is 'VBD'
   3   3   0   3  | WDT -> DT if the tag of the following word is 'NN'
   3   3   0   0  | IN -> RB if the text of the preceding word is
                  |   'month', and the text of the following word is
                  |   '.'
   3   4   1   0  | IN -> WDT if the text of the preceding word is
                  |   'in', and the text of the following word is
                  |   'the'
   3   3   0   0  | JJ -> NNP if the text of the following word is
                  |   'Union'
   3   3   0   0  | NN -> JJ if the text of the following word is
                  |   'branch'
   3   3   0   0  | NNS -> NN if the text of the preceding word is
                  |   'one'
   3   3   0   0  | VBN -> VBD if the text of the preceding word is
                  |   'also', and the text of the following word is
                  |   'that'

Brill accuracy: 0.885684
Done; rules and errors saved to rules.yaml and errors.out.
>>>
```

Чим більше речень, тим точність виконання буде більшою. Якщо встановлений вищий показник точності, то час тренування програми більший, а продукривність аналізаторів вища.

Висновок: на цій лабораторній роботі я продовжила ознайомлення з автоматичним морфологічним аналізом в NLTK.