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

Кафедра "Системи автоматизованого проектування"

Лабораторна робота №10
на тему:

«ВИВЧЕННЯ БІБЛІОТЕКИ ПРИКЛАДНИХ ПРОГРАМ NLTK, ДЛЯ ОПРАЦЮВАННЯ ТЕКСТІВ ПРИРОДНОЮ МОВОЮ. АВТОМАТИЧНИЙ МОРФОЛОГІЧНИЙ АНАЛІЗ (частина2)»

Виконала:
ст. гр. ПРЛм-11
Неїжмак О.А.
Прийняв:
Дупак Б.П.

Львів-2015

МЕТА РОБОТА

- Вивчення основ програмування на мові *Python*.
- Ознайомлення з автоматичним морфологічним аналізом в NLTK.

Варіант 10

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_training = brown.tagged_sents(categories='adventure')
brown_sents_training = brown.sents(categories='adventure')
unigram_tagger = nltk.UnigramTagger(brown_tagged_sents_training)
brown_tagged_sents = brown.tagged_sents(categories='science_fiction')
brown_sents = brown.sents(categories='science_fiction')
for sentn in range (0,5):
    sent=brown_sents[sentn]
    print unigram_tagger.tag(sent)
print unigram_tagger.evaluate(brown_tagged_sents)
```

```
>>>
[('Now', 'RB'), ('that', 'CS'), ('he', 'PPS'), ('knew', 'VBD'), ('himself', 'PPL'), ('to', 'TO')
, ('be', 'BE'), ('self', None), ('he', 'PPS'), ('was', 'BEDZ'), ('free', 'JJ'), ('to', 'TO'), ('
grok', None), ('ever', 'RB'), ('closer', 'RBR'), ('to', 'TO'), ('his', 'PP$'), ('brothers', 'NNS
'), (',', ','), ('merge', None), ('without', 'IN'), ('let', 'VB'), ('.', '.')]
[("Self's", None), ('integrity', None), ('was', 'BEDZ'), ('and', 'CC'), ('is', 'BEZ'), ('and', '
CC'), ('ever', 'RB'), ('had', 'HVD'), ('been', 'BEN'), ('.', '.')]
[('Mike', 'NP'), ('stopped', 'VBD'), ('to', 'TO'), ('cherish', None), ('all', 'ABN'), ('his', 'P
P$'), ('brother', 'NN'), ('selves', None), (',', ','), ('the', 'AT'), ('many', 'AP'), ('threes-f
ulfilled', None), ('on', 'IN'), ('Mars', None), (',', ','), ('corporate', None), ('and', 'CC'),
('discorporate', None), (',', ','), ('the', 'AT'), ('precious', 'JJ'), ('few', 'AP'), ('on', 'IN
'), ('Earth', None), ('--', '--'), ('the', 'AT'), ('unknown', 'JJ'), ('powers', 'NNS'), ('of', '
IN'), ('three', 'CD'), ('on', 'IN'), ('Earth', None), ('that', 'CS'), ('would', 'MD'), ('be', 'B
E'), ('his', 'PP$'), ('to', 'TO'), ('merge', None), ('with', 'IN'), ('and', 'CC'), ('cherish', N
one), ('now', 'RB'), ('that', 'CS'), ('at', 'IN'), ('last', 'AP'), ('long', 'JJ'), ('waiting', '
VBG'), ('he', 'PPS'), ('grokked', None), ('and', 'CC'), ('cherished', 'VBN'), ('himself', 'PPL')
, ('.', '.')]
[('Mike', 'NP'), ('remained', 'VBD'), ('in', 'IN'), ('trance', None), (';', '.'), (';', '.')]
[('there', 'RB'), ('was', 'BEDZ'), ('much', 'AP'), ('to', 'TO'), ('grok', None), (',', ','), ('l
oose', 'JJ'), ('ends', 'NNS'), ('to', 'TO'), ('puzzle', None), ('over', 'IN'), ('and', 'CC'), ('
fit', 'VB'), ('into', 'IN'), ('his', 'PP$'), ('growing', 'VBG'), ('--', '--'), ('all', 'ABN'), (
'that', 'CS'), ('he', 'PPS'), ('had', 'HVD'), ('seen', 'VBN'), ('and', 'CC'), ('heard', 'VBD'),
('and', 'CC'), ('been', 'BEN'), ('at', 'IN'), ('the', 'AT'), ('Archangel', None), ('Foster', 'NP
'), ('Tabernacle', None), ('(', '('), ('not', '*'), ('just', 'RB'), ('cusp', None), ('when', 'WR
B'), ('he', 'PPS'), ('and', 'CC'), ('Digby', None), ('had', 'HVD'), ('come', 'VB'), ('face', 'NN
'), ('to', 'TO'), ('face', 'NN'), ('alone', 'RB'), (')', ')'), ('why', 'WRB'), ('Bishop', None),
('Senator', 'NN-TL'), ('Boone', None), ('made', 'VBD'), ('him', 'PPO'), ('warily', 'RB'), ('unea
sy', 'JJ'), (',', ','), ('how', 'WRB'), ('Miss', 'NP'), ('Dawn', 'NN'), ('Ardent', None), ('tast
ed', 'VBD'), ('like', 'CS'), ('a', 'AT'), ('water', 'NN'), ('brother', 'NN'), ('when', 'WRB'), (
'she', 'PPS'), ('was', 'BEDZ'), ('not', '*'), (',', ','), ('the', 'AT'), ('smell', 'NN'), ('of',
'IN'), ('goodness', None), ('he', 'PPS'), ('had', 'HVD'), ('incompletely', None), ('grokked', No
ne), ('in', 'IN'), ('the', 'AT'), ('jumping', 'VBG'), ('up', 'RP'), ('and', 'CC'), ('down', 'RP'
), ('and', 'CC'), ('wailing', 'VBG'), ('--', '--')]
0.796475466482
>>>
```

3.2. 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_tagged= brown.tagged_sents(categories='news')
sent=brown.sents(categories='news')[2015]
tokens = nltk.word_tokenize(text)
for affix_length in range(1,6):
    for min_stem_length in range (1,4):
        affix_tagger = nltk.AffixTagger(brown_tagged, affix_length=affix_length, min_stem_length=min_stem_length)
        print affix_tagger.tag(sent)
```

```
...
[('It', 'IN'), ('is', 'IN'), ('a', None), ('killer', 'NN'), ('sub', 'NN'), ('--'
, '--'), ('that', 'AT'), ('is', 'IN'), (',', None), ('a', None), ('hunter', 'PP$
'), ('of', 'IN'), ('enemy', 'NN'), ('subs', 'NN'), ('.', None)]
[('It', None), ('is', None), ('a', None), ('killer', 'NN'), ('sub', 'NN'), ('--'
, None), ('that', 'AT'), ('is', None), (',', None), ('a', None), ('hunter', 'PP$
'), ('of', None), ('enemy', 'NN'), ('subs', 'NN'), ('.', None)]
[('It', None), ('is', None), ('a', None), ('killer', 'NN'), ('sub', None), ('--'
, None), ('that', 'NN'), ('is', None), (',', None), ('a', None), ('hunter', 'NN'
), ('of', None), ('enemy', 'NN'), ('subs', 'NN'), ('.', None)]
[('It', None), ('is', None), ('a', None), ('killer', 'NN'), ('sub', 'NN'), ('--'
, None), ('that', 'AT'), ('is', None), (',', None), ('a', None), ('hunter', 'NN'
), ('of', None), ('enemy', 'NN'), ('subs', 'NN'), ('.', None)]
[('It', None), ('is', None), ('a', None), ('killer', 'NN'), ('sub', None), ('--'
, None), ('that', 'CS'), ('is', None), (',', None), ('a', None), ('hunter', 'NN'
), ('of', None), ('enemy', 'NN'), ('subs', 'NN'), ('.', None)]
[('It', None), ('is', None), ('a', None), ('killer', 'NN'), ('sub', None), ('--'
, None), ('that', None), ('is', None), (',', None), ('a', None), ('hunter', 'NN'
), ('of', None), ('enemy', 'NN'), ('subs', None), ('.', None)]
[('It', None), ('is', None), ('a', None), ('killer', 'VBN'), ('sub', None), ('--
', None), ('that', 'CS'), ('is', None), (',', None), ('a', None), ('hunter', 'NN
S'), ('of', None), ('enemy', 'NN'), ('subs', 'NN'), ('.', None)]
[('It', None), ('is', None), ('a', None), ('killer', 'VBN'), ('sub', None), ('--
', None), ('that', None), ('is', None), (',', None), ('a', None), ('hunter', 'NN
S'), ('of', None), ('enemy', 'NN'), ('subs', None), ('.', None)]
```

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

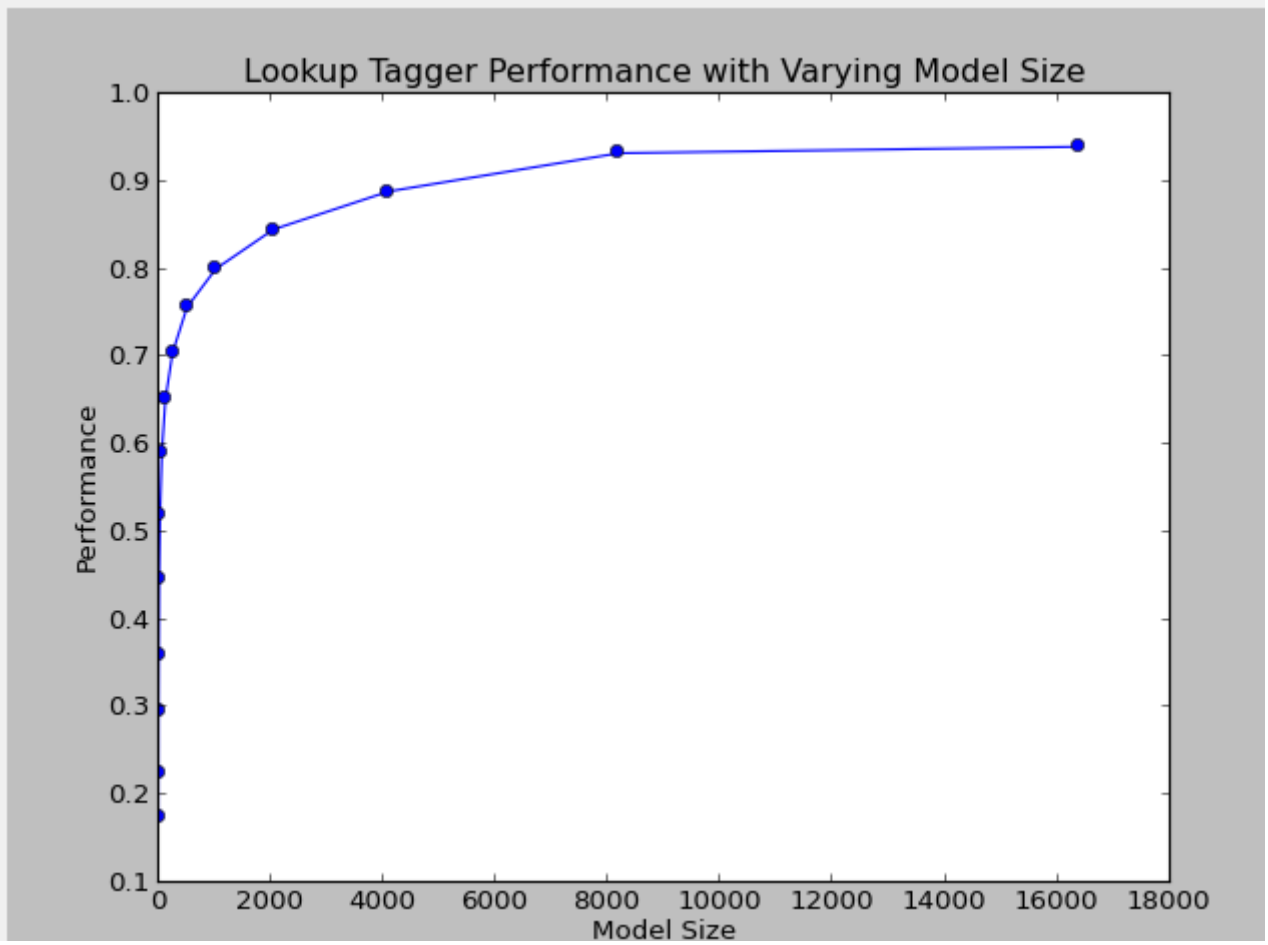
```
import nltk
from nltk.corpus import brown
brown_tagged_sents_training = brown.tagged_sents(categories='adventure')
brown_sents_training = brown.sents(categories='adventure')
bigram_tagger = nltk.BigramTagger(brown_tagged_sents_training)
brown_tagged_sents = brown.tagged_sents(categories='science_fiction')
brown_sents = brown.sents(categories='science_fiction')
for sentn in range (0,5):
    sent=brown_sents[sentn]
    print bigram_tagger.tag(sent)
print bigram_tagger.evaluate(brown_tagged_sents)
```

```
[('Now', 'RB'), ('that', 'CS'), ('he', 'PPS'), ('knew', 'VBD'), ('himself', 'PPL'), ('to', 'IN')
, ('be', None), ('self', None), ('he', None), ('was', None), ('free', None), ('to', None), ('gro
k', None), ('ever', None), ('closer', None), ('to', None), ('his', None), ('brothers', None), ('
,', None), ('merge', None), ('without', None), ('let', None), ('.', None)]
[("Self's", None), ('integrity', None), ('was', None), ('and', None), ('is', None), ('and', None
), ('ever', None), ('had', None), ('been', None), ('.', None)]
[('Mike', 'NP'), ('stopped', 'VBD'), ('to', 'TO'), ('cherish', None), ('all', None), ('his', Non
e), ('brother', None), ('selves', None), (',', None), ('the', None), ('many', None), ('threes-fu
lfilled', None), ('on', None), ('Mars', None), (',', None), ('corporate', None), ('and', None),
('discorporate', None), (',', None), ('the', None), ('precious', None), ('few', None), ('on', No
ne), ('Earth', None), ('--', None), ('the', None), ('unknown', None), ('powers', None), ('of', N
one), ('three', None), ('on', None), ('Earth', None), ('that', None), ('would', None), ('be', No
ne), ('his', None), ('to', None), ('merge', None), ('with', None), ('and', None), ('cherish', No
ne), ('now', None), ('that', None), ('at', None), ('last', None), ('long', None), ('waiting', No
ne), ('he', None), ('grokked', None), ('and', None), ('cherished', None), ('himself', None), ('.
', None)]
[('Mike', 'NP'), ('remained', 'VBD'), ('in', 'IN'), ('trance', None), (';', None), (';', None)]
[('there', 'EX'), ('was', 'BEDZ'), ('much', None), ('to', None), ('grok', None), (',', None), ('
loose', None), ('ends', None), ('to', None), ('puzzle', None), ('over', None), ('and', None), ('
fit', None), ('into', None), ('his', None), ('growing', None), ('--', None), ('all', None), ('th
at', None), ('he', None), ('had', None), ('seen', None), ('and', None), ('heard', None), ('and',
None), ('been', None), ('at', None), ('the', None), ('Archangel', None), ('Foster', None), ('Tab
ernacle', None), ('(', None), ('not', None), ('just', None), ('cusp', None), ('when', None), ('h
e', None), ('and', None), ('Digby', None), ('had', None), ('come', None), ('face', None), ('to',
None), ('face', None), ('alone', None), (')', None), ('why', None), ('Bishop', None), ('Senator'
, None), ('Boone', None), ('made', None), ('him', None), ('warily', None), ('uneasy', None), (',
', None), ('how', None), ('Miss', None), ('Dawn', None), ('Ardent', None), ('tasted', None), ('l
ike', None), ('a', None), ('water', None), ('brother', None), ('when', None), ('she', None), ('w
as', None), ('not', None), (',', None), ('the', None), ('smell', None), ('of', None), ('goodness
', None), ('he', None), ('had', None), ('incompletely', None), ('grokked', None), ('in', None),
('the', None), ('jumping', None), ('up', None), ('and', None), ('down', None), ('and', None), ('
wailing', None), ('--', None)]
0.178852798894
```

3.4. 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.

```python
import nltk
from nltk.corpus import brown
def performance(cfd, wordlist):
    lt = dict((word, cfd[word].max()) for word in wordlist)
    baseline_tagger = nltk.UnigramTagger(model=lt, backoff=nltk.DefaultTagger('NN'))
    return baseline_tagger.evaluate(brown.tagged_sents(categories='adventure'))

def display():
    import pylab
    words_by_freq = list(nltk.FreqDist(brown.words(categories='adventure')))
    cfd = nltk.ConditionalFreqDist(brown.tagged_words(categories='adventure'))
    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()
print display()
```



3.5. 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 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, backoff=t1)
    t3=nltk.TrigramTagger(train_sents, backoff=t2)
    return t3.evaluate(test_sents)

def taggers3(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='adventure')
koef=0.9
print 'DefaultTagger+UnigramTagger+BigramTagger'
print (str(koef*100)+'%', taggers1(tagged_sents, koef))
print 'DefaultTagger+UnigramTagger+BigramTagger+TrigramTagger'
print (str(koef*100)+'%', taggers2(tagged_sents, koef))
print 'DefaultTagger+UnigramTagger+BigramTagger+TrigramTagger with cutoff'
print (str(koef*100)+'%', taggers3(tagged_sents, koef))
print 'Portuguese'
tagged_sents=nltk.corpus.floresta.tagged_sents()
koef=0.9
print 'DefaultTagger+UnigramTagger+BigramTagger'
print (str(koef*100)+'%', taggers1(tagged_sents, koef))
print 'DefaultTagger+UnigramTagger+BigramTagger+TrigramTagger'
print (str(koef*100)+'%', taggers2(tagged_sents, koef))
print 'DefaultTagger+UnigramTagger+BigramTagger+TrigramTagger with cutoff'
print (str(koef*100)+'%', taggers3(tagged_sents, koef))
print 'Dutch, Spanish'
tagged_sents=nltk.corpus.conll2002.tagged_sents()
koef=0.9
print 'DefaultTagger+UnigramTagger+BigramTagger'
print (str(koef*100)+'%', taggers1(tagged_sents, koef))
print 'DefaultTagger+UnigramTagger+BigramTagger+TrigramTagger'
print (str(koef*100)+'%', taggers2(tagged_sents, koef))
print 'DefaultTagger+UnigramTagger+BigramTagger+TrigramTagger with cutoff'
print (str(koef*100)+'%', taggers3(tagged_sents, koef))
```

```
English
DefaultTagger+UnigramTagger+BigramTagger
('90.0%', 0.8523698523698524)
DefaultTagger+UnigramTagger+BigramTagger+TrigramTagger
('90.0%', 0.8513338513338513)
DefaultTagger+UnigramTagger+BigramTagger+TrigramTagger with cutoff
('90.0%', 0.8493913493913494)
Portuguese
DefaultTagger+UnigramTagger+BigramTagger
('90.0%', 0.7911843797497954)
DefaultTagger+UnigramTagger+BigramTagger+TrigramTagger
('90.0%', 0.7882029697182276)
DefaultTagger+UnigramTagger+BigramTagger+TrigramTagger with cutoff
('90.0%', 0.7843446743832574)
Dutch, Spanish
DefaultTagger+UnigramTagger+BigramTagger
('90.0%', 0.8004550348386048)
DefaultTagger+UnigramTagger+BigramTagger+TrigramTagger
('90.0%', 0.8001706380644769)
DefaultTagger+UnigramTagger+BigramTagger+TrigramTagger with cutoff
('90.0%', 0.7946249009689804)
```

3.6. Створити аналізатор по замовчуванню та набір юніграм і n-грам аналізаторів. Використовуючи backoffздійсніть тренування аналізаторів на частині корпуса з вправи 3.2. 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 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, backoff=t1)
    t3=nltk.TrigramTagger(train_sents, backoff=t2)
    return t3.evaluate(test_sents)

def taggers3(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'Brown'
tagged_sents=nltk.corpus.brown.tagged_sents(categories='news')
for koef in [0.5, 0.6, 0.7, 0.8, 0.9]:
    print 'DefaultTagger+UnigramTagger+BigramTagger'
    print (str(koef*100)+'%', taggers1(tagged_sents, koef))
    print 'DefaultTagger+UnigramTagger+BigramTagger+TrigramTagger'
    print (str(koef*100)+'%', taggers2(tagged_sents, koef))
    print 'DefaultTagger+UnigramTagger+BigramTagger+TrigramTagger with cutoff'
    print (str(koef*100)+'%', taggers3(tagged_sents, koef))
```

```
>>>
Brown
DefaultTagger+UnigramTagger+BigramTagger
('50.0%', 0.8093409730836632)
DefaultTagger+UnigramTagger+BigramTagger+TrigramTagger
('50.0%', 0.8086124401913876)
DefaultTagger+UnigramTagger+BigramTagger+TrigramTagger with cutoff
('50.0%', 0.8065843621399177)
DefaultTagger+UnigramTagger+BigramTagger
('60.0%', 0.8208995487541692)
DefaultTagger+UnigramTagger+BigramTagger+TrigramTagger
('60.0%', 0.8197714341769669)
DefaultTagger+UnigramTagger+BigramTagger+TrigramTagger with cutoff
('60.0%', 0.8178585442417108)
DefaultTagger+UnigramTagger+BigramTagger
('70.0%', 0.833153176066637)
DefaultTagger+UnigramTagger+BigramTagger+TrigramTagger
('70.0%', 0.8310866662427672)
DefaultTagger+UnigramTagger+BigramTagger+TrigramTagger with cutoff
('70.0%', 0.8287976092070961)
DefaultTagger+UnigramTagger+BigramTagger
('80.0%', 0.8355412110409942)
DefaultTagger+UnigramTagger+BigramTagger+TrigramTagger
('80.0%', 0.8338070234597043)
DefaultTagger+UnigramTagger+BigramTagger+TrigramTagger with cutoff
('80.0%', 0.8313020858422853)
DefaultTagger+UnigramTagger+BigramTagger
('90.0%', 0.8447124489185687)
DefaultTagger+UnigramTagger+BigramTagger+TrigramTagger
('90.0%', 0.8423203428685339)
DefaultTagger+UnigramTagger+BigramTagger+TrigramTagger with cutoff
('90.0%', 0.8409249476726801)
```

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

```
>>> help(nltk.tag.brill.demo)
Help on function demo in module nltk.tag.brill:

demo(num_sents=2000, max_rules=200, min_score=3, error_output='errors.out', rule_output='rules.y
aml', randomize=False, train=0.8, trace=3)
    Brill Tagger Demonstration

    :param num_sents: how many sentences of training and testing data to use
    :type num_sents: int
    :param max_rules: maximum number of rule instances to create
    :type max_rules: int
    :param min_score: the minimum score for a rule in order for it to
        be considered
    :type min_score: int
    :param error_output: the file where errors will be saved
    :type error_output: str
    :param rule_output: the file where rules will be saved
    :type rule_output: str
    :param randomize: whether the training data should be a random subset
        of the corpus
    :type randomize: bool
    :param train: the fraction of the the corpus to be used for training
        (1=all)
    :type train: float
    :param trace: the level of diagnostic tracing output to produce (0-4)
    :type trace: int
```

```
import nltk
print 'Number of sentences 3780'
nltk.tag.brill.demo(num_sents=3780, max_rules=200, min_score=3, error_output='errors.out',
                    rule_output='rules.yaml', randomize=False, train=0.8, trace=3)

print 'Number of sentences 2000'
nltk.tag.brill.demo(num_sents=2000, max_rules=200, min_score=3, error_output='errors.out',
                    rule_output='rules.yaml', randomize=False, train=0.8, trace=3)

print 'Number of sentences 1200'
nltk.tag.brill.demo(num_sents=1200, max_rules=200, min_score=3, error_output='errors.out',
                    rule_output='rules.yaml', randomize=False, train=0.8, trace=3)

print 'Number of sentences 700'
nltk.tag.brill.demo(num_sents=700, max_rules=200, min_score=3, error_output='errors.out',
                    rule_output='rules.yaml', randomize=False, train=0.8, trace=3)
```

```
Number of sentences 3780
Loading tagged data...
Done loading.
Training unigram tagger:
    [accuracy: 0.888918]
Training bigram tagger:
    [accuracy: 0.894360]
Training Brill tagger on 3024 sentences...
Finding initial useful rules...
    Found 18190 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
-----------------+-------------------------------------------------------
  24  30   6   0  | WDT -> IN if the tag of the following word is 'DT'
  16  16   0   0  | WDT -> IN if the tag of the preceding word is
                  |   'NN', and the tag of the following word is 'PRP'
  14  15   1   0  | WDT -> IN if the tag of the preceding word is
                  |   'NN', and the tag of the following word is 'NNP'
  11  17   6   0  | WDT -> IN if the tag of words i+1...i+2 is 'NNS'
  11  22  11   0  | IN -> RB if the text of word i+2 is 'as'
  10  22  12   1  | RBR -> JJR if the text of the following word is
                  |   'than'
   9   9   0   0  | RBR -> JJR if the tag of the following word is
                  |   'NN'
   8  10   2   0  | VBP -> VB if the tag of words i-3...i-1 is 'MD'
   7   7   0   0  | WDT -> IN if the tag of the preceding word is
                  |   'NNS', and the tag of the following word is
                  |   'PRP'
   6   6   0   0  | RBR -> JJR if the tag of the following word is
                  |   'NNS'
   6   6   0   0  | VBN -> VBD if the tag of the preceding word is
                  |   'NN', and the tag of the following word is 'DT'
   5   5   0   0  | RB -> JJ if the tag of the preceding word is 'DT',
                  |   and the tag of the following word is 'NN'
   5   5   0   0  | RP -> IN if the text of the following word is 'of'
```

```
                    |   '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  | RP -> IN if the text of words i-3...i-1 is
                    |   'business'
     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: 0.897133
  Done; rules and errors saved to rules.yaml and errors.out.
  Number of sentences 2000
  Loading tagged data...
  Done loading.
  Training unigram tagger:
      [accuracy: 0.832151]
  Training bigram tagger:
      [accuracy: 0.837930]
  Training Brill tagger on 1600 sentences...
  Finding initial useful rules...
      Found 9757 useful rules.
   Brill accuracy: 0.839156
   Done; rules and errors saved to rules.yaml and errors.out.
   Number of sentences 1200
   Loading tagged data...
   Done loading.
   Training unigram tagger:
       [accuracy: 0.809602]
   Training bigram tagger:
       [accuracy: 0.814864]
   Training Brill tagger on 960 sentences...
   Finding initial useful rules...
       Found 5145 useful rules.

Brill accuracy: 0.813384
Done; rules and errors saved to rules.yaml and errors.out.
Number of sentences 700
Loading tagged data...
Done loading.
Training unigram tagger:
    [accuracy: 0.797189]
Training bigram tagger:
    [accuracy: 0.798623]
Training Brill tagger on 560 sentences...
Finding initial useful rules...
    Found 2735 useful rules.

Brill accuracy: 0.799197
Done; rules and errors saved to rules.yaml and errors.out.
```

Чим більше речень, тим вища точність.

```
import nltk
import nltk
print 'Number of sentences 3780'
nltk.tag.brill.demo(num_sents=3780, max_rules=200, min_score=3, error_output='errors.out',
                    rule_output='rules.yaml', randomize=False, train=0.7, trace=3)

print 'Number of sentences 2000'
nltk.tag.brill.demo(num_sents=2000, max_rules=200, min_score=3, error_output='errors.out',
                    rule_output='rules.yaml', randomize=False, train=0.7, trace=3)

print 'Number of sentences 1200'
nltk.tag.brill.demo(num_sents=1200, max_rules=200, min_score=3, error_output='errors.out',
                    rule_output='rules.yaml', randomize=False, train=0.7, trace=3)

print 'Number of sentences 700'
nltk.tag.brill.demo(num_sents=700, max_rules=200, min_score=3, error_output='errors.out',
                    rule_output='rules.yaml', randomize=False, train=0.7, trace=3)
```

```
Number of sentences 3780
Loading tagged data...
Done loading.
Training unigram tagger:
    [accuracy: 0.878288]
Training bigram tagger:
    [accuracy: 0.884191]
Training Brill tagger on 2646 sentences...
Finding initial useful rules...
    Found 16123 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
-----------------+-------------------------------------------------------
 20  26   6   0  | WDT -> IN if the tag of the following word is 'DT'
 14  14   0   0  | WDT -> IN if the tag of the preceding word is
                 |   'NN', and the tag of the following word is 'PRP'
 12  22  10   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'
 10  15   5   0  | WDT -> IN if the tag of words i+1...i+2 is 'NNS'
  8  20  12   1  | RBR -> JJR if the text of the following word is
                 |   'than'
  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'
Brill accuracy: 0.886913
Done; rules and errors saved to rules.yaml and errors.out.
Number of sentences 2000
Loading tagged data...
Done loading.
Training unigram tagger:
    [accuracy: 0.827513]
Training bigram tagger:
    [accuracy: 0.831400]
Training Brill tagger on 1400 sentences...
Finding initial useful rules...
    Found 8316 useful rules.
```

```
Brill accuracy: 0.830064
Done; rules and errors saved to rules.yaml and errors.out.
Number of sentences 1200
Loading tagged data...
Done loading.
Training unigram tagger:
    [accuracy: 0.800194]
Training bigram tagger:
    [accuracy: 0.805472]
Training Brill tagger on 840 sentences...
Finding initial useful rules...
    Found 4583 useful rules.
```

```
Brill accuracy: 0.806333
Done; rules and errors saved to rules.yaml and errors.out.
Number of sentences 700
Loading tagged data...
Done loading.
Training unigram tagger:
    [accuracy: 0.777504]
Training bigram tagger:
    [accuracy: 0.779208]
Training Brill tagger on 489 sentences...
Finding initial useful rules...
    Found 2378 useful rules.
```

```
Brill accuracy: 0.779587
Done; rules and errors saved to rules.yaml and errors.out.
```

кількість речень	к-сть речень для тренування	к-сть речень для тестування	Точність
3780	80%	20%	0,897133
2000	80%	20%	0,839156
1200	80%	20%	0,813384
7000	80%	20%	0,799197
3780	70%	30%	0,886913
2000	70%	30%	0,830064
1200	70%	30%	0,806333
7000	70%	30%	0,779587

Висновок: під час виконання цієї лабораторної роботи, я вивчила елементи бібліотеки прикладних програм nltk, для опрацювання текстів природною мовою; автоматичний морфологічний аналіз.