МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ТАРАСА ШЕВЧЕНКА

кафедра інформаційних систем та технологій

ЗВІТ

із лабораторної роботи №3

з дисципліни «**Технології програмування об'єктів лінгвістичної предметної галузі**»

на тему: Автоматична класифікація тексту (сентимент-аналіз)

Варіант **№ 2-06**

Виконав:

Студент групи №2

Кличлієв К. С.

Перевірив:

доц. Костіков М. П.

Київ — 2022

Дані з індивідуального варіанту №2-06

Слово: radical

N = 4000

Текстовий файл (відгук): cv019_14482.txt

Метод машинного навчання: Multinomial NB

Мета роботи

Метою лабораторної роботи №3 є насамперед ознайомлення з проблемою класифікації тексту та методами машинного навчання, якими класифікація може бути здійснена. Датологічною базою ЛР№3 виступає корпус movie_reviews бібліотеки nltk, що містить 2000 відгуків - позитивних та негативних -, над яким першочергово будуть проведені наступні операції:

- створення списку всіх відгуків та частотного словника слів із корпусу;
- обрахування кількості слововживань певного слова серед усіх відгуків і окремо серед позитивних і негативних.

Також планується реалізувати класифікатор відгуків за допомогою двох методів машинного навчання - Naive Bayes та Multionomial NB - і перевірити точність цих методів.

Середовище розробки

Мова програмування: Python

Назва та версія IDE: Jupyter Notebook 6.4.11

Назва та версія ОС: Linux Ubuntu 21.04

Мова ОС: англійська

Хід роботи

1. Створюємо консольний проект мовою Python, що при запуску виводить ім’я, прізвище та групу виконавця лабораторної роботи, а також її номер:

```
print('Кличлієв Кирило\nГрупа №2\nЛабораторна робота №3')
```

```
Кличлієв Кирило
Група №2
Лабораторна робота №3
```

2. Імпортуємо в проект необхідні бібліотеки (NLTK та її корпус movie_reviews, що містить позитивні та негативні відгуки на фільми):

```
import nltk
from nltk.corpus import movie_reviews
```

3. Створюємо список відгуків та перемішуємо його за допомоги бібліотеки random:

```
import random

document = [(movie_reviews.words(file_id),category) for file_id in movie_reviews.fileids()
            for category in movie_reviews.categories(file_id)]

shuffled_doc = random.sample(document, k=len(document))
shuffled_doc
```

```
[(['starring', 'ben', 'stiller', ',', 'elizabeth', ...], 'neg'),
 (['elmore', 'leonard', 'has', 'quickly', 'become', ...], 'pos'),
 (['michael', 'crichton', 'has', 'had', 'a', 'long', ...], 'neg'),
 (['much', 'ado', 'about', 'nothing', '.', 'ah', ',', ...], 'neg'),
 (['modern', 'audiences', 'are', 'more', 'likely', 'to', ...], 'pos'),
 (['the', 'year', 'is', '1962', 'and', 'the', 'military', ...], 'pos'),
 (['who', 'knew', 'that', 'in', '16', 'years', 'eddie', ...], 'neg'),
 (['part', 'buddy', 'comedy', ',', 'part', 'fish', '-', ...], 'neg'),
 (['in', 'a', 'typical', 'cinematic', 'high', 'school', ...], 'neg'),
 (['wizards', 'is', 'an', 'animated', 'feature', 'that', ...], 'neg'),
 (['the', 'bond', 'series', 'is', 'an', 'island', 'in', ...], 'pos'),
 (['"', 'if', 'there', "'", 's', 'a', 'beast', 'in', ...], 'pos'),
 (['"', 'it', "'", 's', 'not', 'good', 'to', 'know', ...], 'pos'),
 (['woody', 'allen', 'is', 'one', 'of', 'the', 'most', ...], 'neg'),
 (['at', 'one', 'point', 'in', 'this', 'movie', 'there', ...], 'neg'),
 (['to', 'paraphrase', 'a', 'song', 'title', 'from', ...], 'pos'),
 (['note', ':', 'some', 'may', 'consider', 'portions', ...], 'pos'),
 (['ingredients', ':', 'little', 'orphan', 'boy', ',', ...], 'pos'),
 (['with', 'his', 'successful', 'books', 'and', 'movies', ...], 'neg'),
 (['getting', 'it', 'right', 'is', 'a', 'far', 'far', ...], 'pos'),
 (['the', 'army', 'comedy', 'genre', 'has', 'never', ...], 'neg'),
 (['the', 'farrelly', 'brothers', "'", 'third', 'film', ...], 'pos'),
 (['when', 'i', 'first', 'saw', 'the', 'previews', 'for', ...], 'pos'),
 (['synopsis', ':', 'valerie', ',', 'a', 'high', ...], 'neg')
```

4. Створюємо список, у який додаємо всі слова з усіх відгуків, відсортовуємо його за частотою вживання і виводимо в консоль 20 найбільш уживаних у цьому корпусі слів з їхньою частотою:

```
all_words = []
for w in movie_reviews.words():
    if w.isalpha():
        all_words.append(w.lower())

all_words = nltk.FreqDist(all_words)
print(all_words.most_common(20))
```

```
[('the', 76529), ('a', 38106), ('and', 35576), ('of', 34123), ('to', 31937), ('is', 25195), ('in', 21822), ('s', 18513), ('it', 16107), ('that', 15924), ('as', 11378), ('with', 10792), ('for', 9961), ('his', 9587), ('this', 9578), ('film', 9517), ('i', 8889), ('he', 8864), ('but', 8634), ('on', 7385)]
```

5. Знаходимо кількість вживань слова ‘radical’:

```
print(all_words['radical'])
```

```
15
```

6. Визначаємо кількість вживань слова ‘radical’ окремо серед позитивних та негативних відгуків:

```
pos_rev = movie_reviews.words(movie_reviews.fileids('pos'))
pos_rev = nltk.FreqDist(pos_rev)
print('Кількість вживань слова radical серед позитивних відгуків:', pos_rev['radical'])

neg_rev = movie_reviews.words(movie_reviews.fileids('neg'))
neg_rev = nltk.FreqDist(neg_rev)
print('Кількість вживань слова radical серед негативних відгуків:', neg_rev['radical'])
```

```
Кількість вживань слова radical серед позитивних відгуків: 12
Кількість вживань слова radical серед негативних відгуків: 3
```

7. Беремо 4000 найбільш уживаних слів у корпусі та створюємо функцію, що перевіряє їх наявність у поданому текстовому файлі (cv019_14482.txt) та повертає словник у вигляді: {'word1': True, 'word2': False, ...}. Перевіряємо створену функцію, подавши на вхід файл із папки pos. Виводимо на екран лише ті слова з 4000 найуживаніших, які є в цьому файлі:

```
word_features = list(all_words.keys())[:4000]
def find_features(d):
    words = set(d)
    features = {}
    for w in word_features:
        features[w] = (w in words)
    return features

print(find_features(movie_reviews.words('pos/cv019_14482.txt')))
```

```
{'plot': False, 'two': False, 'teen': False, 'couples': False, 'go': False, 'to': True, 'a': True, 'church': False, 'party':
False, 'drink': False, 'and': True, 'then': False, 'drive': False, 'they': True, 'get': True, 'into': True, 'an': False, 'ac
cident': False, 'one': True, 'of': True, 'the': True, 'guys': False, 'dies': False, 'but': True, 'his': True, 'girlfriend':
False, 'continues': False, 'see': True, 'him': True, 'in': True, 'her': True, 'life': True, 'has': True, 'nightmares': Fals
e, 'what': True, 's': True, 'deal': False, 'watch': False, 'movie': False, 'sorta': False, 'find': False, 'out': True, 'crit
ique': False, 'mind': True, 'fuck': False, 'for': True, 'generation': False, 'that': True, 'touches': False, 'on': True, 've
ry': False, 'cool': False, 'idea': False, 'presents': False, 'it': True, 'bad': False, 'package': False, 'which': True, 'i
s': True, 'makes': True, 'this': True, 'review': False, 'even': True, 'harder': False, 'write': False, 'since': False, 'i':
True, 'generally': False, 'applaud': False, 'films': False, 'attempt': False, 'break': False, 'mold': False, 'mess': False,
'with': True, 'your': True, 'head': False, 'such': False, 'lost': False, 'highway': False, 'memento': False, 'there': True,
'are': False, 'good': False, 'ways': True, 'making': False, 'all': True, 'types': False, 'these': True, 'folks': False, 'jus
t': False, 'didn': False, 't': True, 'snag': False, 'correctly': False, 'seem': False, 'have': False, 'taken': True, 'prett
y': False, 'neat': False, 'concept': True, 'executed': False, 'terribly': False, 'so': True, 'problems': False, 'well': Fals
e, 'its': False, 'main': False, 'problem': False, 'simply': False, 'too': False, 'jumbled': False, 'starts': False, 'off': T
rue, 'normal': False, 'downshifts': False, 'fantasy': False, 'world': False, 'you': True, 'as': True, 'audience': False, 'me
mber': False, 'no': False, 'going': False, 'dreams': False, 'characters': False, 'coming': False, 'back': False, 'from': Tru
e, 'dead': False, 'others': False, 'who': True, 'look': False, 'like': True, 'strange': False, 'apparitions': False, 'disapp
earances': False, 'looooot': False, 'chase': False, 'scenes': False, 'tons': False, 'weird': False, 'things': False, 'happe
n': False, 'most': False, 'not': False, 'explained': False, 'now': True, 'personally': False, 'don': False, 'trying': False,
'unravel': False, 'film': True, 'every': False, 'when': True, 'does': False, 'give': False, 'me': False, 'same': False, 'clu
e': False, 'over': True, 'again': True, 'kind': False, 'fed': False, 'up': True, 'after': True, 'while': True, 'biggest': Fa
lse, 'obviously': False, 'got': False, 'big': False, 'secret': False, 'hide': False, 'seems': False, 'want': True, 'complete
ly': False, 'until': False, 'final': False, 'five': False, 'minutes': False, 'do': True, 'make': True, 'entertaining': Fals
e, 'thrilling': False, 'or': False, 'engaging': False, 'meantime': False, 'really': False, 'sad': False, 'part': False, 'arr
```

8. Реалізовуємо класифікатор на методом Наївного Баєса, навчаючи модель на тренувальному наборі, що складається із 1800 випадкових відгуків із корпусу movie_reviews. Перевіряємо модель на решті відгуків із корпусу (testing_set; складається із 200 відгуків) і виводимо на екран 20 слів, що найчіткіше вказують на приналежність відгуку до позитивних чи негативних:

```
import sklearn

featuresets = [(find_features(rev), category) for (rev, category) in document]
training_set = featuresets[:1800]
testing_set = featuresets[1800:]

classifier = nltk.NaiveBayesClassifier.train(training_set)
accuracy = nltk.classify.accuracy(classifier, testing_set)*100

print(f"Точність алгоритму Наївного Баєса: {accuracy}%")
classifier.show_most_informative_features(20)
```

```
Точність алгоритму Наївного Баєса: 79.5
Most Informative Features
                annual = True              pos : neg    =     11.2 : 1.0
               idiotic = True              neg : pos    =     10.7 : 1.0
               frances = True              pos : neg    =      9.6 : 1.0
                 sucks = True              neg : pos    =      8.5 : 1.0
            accessible = True              pos : neg    =      7.9 : 1.0
              bothered = True              neg : pos    =      7.7 : 1.0
                   ugh = True              neg : pos    =      7.7 : 1.0
                 inept = True              neg : pos    =      7.4 : 1.0
             stupidity = True              neg : pos    =      7.2 : 1.0
               cunning = True              pos : neg    =      7.1 : 1.0
                turkey = True              neg : pos    =      7.0 : 1.0
          unimaginative = True              neg : pos    =      6.7 : 1.0
                wasting = True              neg : pos    =      6.7 : 1.0
          unintentional = True              neg : pos    =      6.6 : 1.0
                   lame = True              neg : pos    =      6.4 : 1.0
                regard = True              pos : neg    =      6.2 : 1.0
               singers = True              pos : neg    =      6.2 : 1.0
               unravel = True              pos : neg    =      6.2 : 1.0
             undercover = True              neg : pos    =      6.2 : 1.0
                 sexist = True              neg : pos    =      6.1 : 1.0
```

9. Створюємо аналогічну програму (до тієї, що в пункті 8), в якій проводимо тренування за допомогою методу Multinomial NB. Порівнюємо точність цієї класифікації з точністю класифікації звичайним методом наївного Баєса:

```
from nltk.classify import SklearnClassifier
from sklearn.naive_bayes import MultinomialNB

classifier = SklearnClassifier(MultinomialNB()).train(training_set)
accuracy = nltk.classify.accuracy(classifier, testing_set)*100

print(f'Точність алгоритму Multinomial NB: {accuracy}%')
```

```
Точність алгоритму Multinomial NB: 77.0%
```

Як бачимо, в цьому випадку точність алгоритму Multinomial NB трішки нижча за точність алгоритму Naive Bayes (77% проти 79.5%).

Відповідь на контрольне питання №6: методи-альтернативи наївного баєсівського алгоритму.

Для виконання ряду задачу замість Наївного Баєса можна використовувати й інші класифікаційні алгоритми машинного навчання, такі, як логістична регресія, баєсівські мережі, метод к-найближчих сусідів, метод опорних векторів тощо.

1. Логістична регресія (Logistic regression) - класифікаційний алгоритм машинного навчання, що використовується для прогнозування категорійної залежної зміни (categorical dependant variable/target). У логістичній регресії залежна змінна є бінарною (так/ні, 1/0 чи True/False). Таким чином в NLP можна класифікувати спам-листи (спам - 1, не спам - 0), а в медицині передбачити злоякісність пухлини (1 - злоякісна, 0 - доброякісна).
2. Баєсівські мережі (Bayes networks) - це різновид імовірнісної графічної моделі, яка використовує байєсівську теорему для обчислень. Їх доцільно застсовувати за наявності великих датасетів і відсутності пропущених даних (missing values) у них. Баєсівські мережі спрямовані на моделювання умовної залежності, а отже, і причинно-наслідкового зв’язку, представляючи умовну залежність ребрами в орієнтованому графі.
3. Метод опорних векторів (SVM: Support vector machine) - керований (supervised) алгоритм машинного навчання, який використовується як для класифікації, так і регресії. Суть методу полягає у знаходженні такої гіперплощини чи граничної лінії, що розділятиме текстові дані на попередньо визначені групи. Найкращою гіперплощиною вважається та, що знаходиться максимально далеко від точок даних обох класів. Вектори або точки даних, розташовані ближче до гіперплощини, називають опорними векторами, які

сильно впливають на положення та відстань оптимальної гіперплощини.

4. Метод k-найближчих сусідів (KNN: K-nearest neighbors) - керований алгоритм машинного навчанням, що класифікує новий текст, зіставляючи його з найближчими збігами в навчальних даних (training data). Оскільки сусіди мають схожу поведінку та характеристики, до них можна ставитися як до однієї групи. Подібним чином, алгоритм KNN визначає k (довільна кількість) найближчих сусідів. Модель навчена таким чином, що коли нові дані передаються через неї, вона може легко зіставляти текст із групою чи класом, до якого він належить.

Крім цього, існує три типи моделі Наївного Баєса, які можуть бути використані для різних задач:

1. Гаусівський (Gaussian NB) використовується для класифікації за наявності даних з нормальним (гаусівським) розподілом;
2. Поліноміальний (Multinomial NB) використовується для проблеми класифікації документів, тобто, наприклад, для ивзначення належності документа до категорії спорту, політики, мистецтва тощо. Характеристики, які використовує цей класифікатор, — частота слів, присутніх у документі;
3. Бернуллі (Bernoulli NB) схожий на поліноміальний НБ, однак вектори ознак (feature vectors) є бінарними показниками, наприклад, факт наявності слова в тексті (так/ні).

Висновок

Отже, у ході виконання лабораторної роботи №3 ми ознайомилися із корпусом movie_reviews бібліотеки nltk. Було створено частотний словник,

що містив усі слова із корпусу (найчастотнішими виявилися ‘the’ із 76529 вживаннями, ‘a’ із 38106 та ‘and’ із 35576). Проаналізовано кількість вживань слова за індивідуальним варіантом - *radical* - і виявлено, що серед усіх відгуків воно вжито всього 15 разів, з яких 12 - серед позитивних відгуків і 3 - серед негативних. Також було реалізовано класифікатор за методами Naive Bayes і Multinomial NB для вирішення однієї й тієї самої задачі, однак точність першого алгоритму виявилася вищою на 2.5% (79.5% проти 77%).

Додаток

Посилання на репозиторій із кодом з ЛР №3:
https://github.com/klychliiev/sentiment-analysis.git