**Міністерство освіти і науки України**

**Національний університет «Запорізька політехніка»**

Кафедра програмних засобів

**ЗВІТ**

Дисципліна «Розробка прикладних програм»

Робота №4

Тема «Оброблення природньої мови»

**Виконав варіант 19**

Студент КНТ-122 Онищенко О. А.

**Прийняли**

Викладач Дейнега Л. Ю.

2024

Мета роботи

Ознайомитись з основними інструментами оброблення природньої мови, які входять у склад бібліотеки NLTK мови програмування Python.

Навчитися розв’язувати актуальні практичні завдання у галузі оброблення природної мови за допомогою бібліотеки NLTK.

Індивідуальне завдання

Реалізувати створення програмних відповідей Українською мовою на основі виконання SQL-запитів до бази даних, розробленої в лабораторній роботі №3.

Текст файлу

NAME='data'

USER='root'

PASSWORD='1313'

HOST='localhost'

PORT='3306'

import mysql.connector

db=mysql.connector.connect(database=NAME,user=USER,password=PASSWORD,host=HOST,port=PORT)

c=db.cursor()

CLIENTS\_TABLE='finance\_client'

PAYMENTS\_TABLE='finance\_payment'

PERIODIC\_PAYMENTS\_TABLE='finance\_periodicpayment'

import nltk

while 1:

HELP\_MESSAGE='''Загальні команди:

- поможи АБО допомога: виведення цього повідомлення

- вийти АБО вихід: закриття програми

- користувачі: виведення списку користувачів

- періодичні платежі: виведення усіх запланованих платежів

- платежі: виведення історії усіх платежів

Користувацькі команди (потребують імені користувача у запиті):

- баланс: виведення балансу користувача

- ліміт АБО кредит: виведення кредитного ліміту користувача

- менеджер АБО адміністратор: виведення статусу користувача

- користувач: виведення усіх даних про користувача

'''.strip()

user\_query=input('> ')

words=nltk.word\_tokenize(user\_query.lower())

vocab=nltk.Text(words).vocab()

def check\_any(

terms:list[str],

words:list[str]=words,

):

found=False

for term in terms:

for word in words:

if term in word:

found=True

break

return found

def check\_all(

terms:list[str],

words:list[str]=words,

):

all\_good=True

for term in terms:

found=False

for word in words:

if term in word: found=True

if not found:

all\_good=False

break

return all\_good

def clean\_query(

query:list[str],

):

return [

word for word in query

if 'буд' not in word

and 'ласк' not in word

and 'про' not in word

and 'дан' not in word

and 'інф' not in word

and 'чи' not in word

]

def execute\_query(

query:str,

):

c.execute(query)

rows=c.fetchall()

return rows

def get\_client\_by\_id(

client\_id:int,

):

q=f'SELECT id,name FROM {CLIENTS\_TABLE} WHERE id={client\_id}'

return execute\_query(q)[0]

def get\_client\_by\_name(

user\_name:str,

):

q=f'SELECT name,balance,credit,manager FROM {CLIENTS\_TABLE} WHERE name="{user\_name}"'

return execute\_query(q)[0]

def get\_balance(

user\_name:str,

):

q=f'SELECT name,balance FROM {CLIENTS\_TABLE} WHERE name="{user\_name}"'

return execute\_query(q)[0]

def get\_credit(

user\_name:str,

):

q=f'SELECT name,credit FROM {CLIENTS\_TABLE} WHERE name="{user\_name}"'

return execute\_query(q)[0]

def get\_manager(

user\_name:str,

):

q=f'SELECT name,manager FROM {CLIENTS\_TABLE} WHERE name="{user\_name}"'

return execute\_query(q)[0]

if check\_any(['вих','вий']): break

elif check\_any(['пом','доп','ком']): print(HELP\_MESSAGE)

elif check\_any(['прив','віт']): print(user\_query)

elif check\_any(['користувачі']) or check\_all(['всі','кор']) or check\_all(['усі','кор']) or (check\_any(['кор']) and len(vocab)==1):

q=f'SELECT name,balance,credit,manager FROM {CLIENTS\_TABLE}'

rows=execute\_query(q)

print(f'Користувачі ({len(rows)}):')

for name,balance,credit,manager in rows:

print(f'- {name} має {balance} на рахунку, {credit} кредитного ліміту, та {"Є" if manager else "НЕ є"} менеджером')

elif check\_any(['пер']):

q=f'SELECT amount,purpose,period,next\_date,client\_id FROM {PERIODIC\_PAYMENTS\_TABLE}'

rows=execute\_query(q)

print(f'Періодичні платежі ({len(rows)}):')

for amount,purpose,period,next\_date,client\_id in rows:

client\_id,client\_name=get\_client\_by\_id(client\_id)

print(f'- {purpose} кожен {"день" if period=="Day" else "місяць" if period=="Month" else "рік"}, наступний платіж {next\_date.strftime("%d.%m.%Y")} для {client\_name}')

elif check\_any(['пл']):

q=f'SELECT timestamp,purpose,amount,client\_id,kind,operation FROM {PAYMENTS\_TABLE}'

rows=execute\_query(q)

print(f'Платежі ({len(rows)}):')

for timestamp,purpose,amount,client\_id,kind,operation in rows:

client\_id,client\_name=get\_client\_by\_id(client\_id)

print(f'- {purpose} за {timestamp.strftime("%d.%m.%Y o %H:%M:%S")} на {amount} від {client\_name}, {"одноразове" if kind=="Single" else "періодичне"} {"зняття" if operation=="Withdrawal" else "внесення"}')

elif check\_any(['бал','гро']):

stripped\_words=clean\_query([w for w in words if 'бал' not in w and 'гро' not in w])

found=False

for word in stripped\_words:

try:

name,balance=get\_balance(word)

print(f'{name} має {balance} на рахунку')

found=True

break

except: continue

if not found: print('користувача не знайдено')

elif check\_any(['кр','лім']):

stripped\_words=clean\_query([w for w in words if 'кр' not in w and 'лім' not in w])

found=False

for word in stripped\_words:

try:

name,balance=get\_credit(word)

print(f'{name} має {balance} кредитного ліміту')

found=True

break

except: continue

if not found: print('користувача не знайдено')

elif check\_any(['мен','адм']):

stripped\_words=clean\_query([w for w in words if 'мен' not in w and 'адм' not in w])

found=False

for word in stripped\_words:

try:

name,manager=get\_manager(word)

print(f'{name} {"Є" if manager else "НЕ є"} менеджером')

found=True

break

except: continue

if not found: print('користувача не знайдено')

else:

stripped\_words=clean\_query([w for w in words if 'кор' not in w])

found=False

for word in stripped\_words:

try:

name,balance,credit,is\_manager=get\_client\_by\_name(word)

print(f'{name} має {balance} на рахунку, {credit} кредитного ліміту, та {"Є" if is\_manager else "НЕ є"} менеджером')

found=True

break

except: continue

if not found: print('користувача не знайдено')

Результати виконання

Процес роботи з програмою наведено нижче:

Зображення, що містить текст, знімок екрана, Шрифт, число

Автоматично згенерований опис

Зображення, що містить текст, Шрифт, знімок екрана, інформація

Автоматично згенерований опис

Зображення, що містить текст, знімок екрана, Шрифт

Автоматично згенерований опис

Зображення, що містить текст, знімок екрана, Шрифт, число

Автоматично згенерований опис

Зображення, що містить текст, знімок екрана, Шрифт

Автоматично згенерований опис

Зображення, що містить текст, Шрифт, знімок екрана, ряд

Автоматично згенерований опис

Контрольні питання

Яким чином виконується оброблення природної мови в бібліотеці NLTK?

Бібліотека NLTK для мови програмування Python впроваджує багато способів обробки природньої мови. Основні засоби це **токенизація**, **чанкування** та **виділення** **частин мови**.

Для **токенізації** можна застосувати розділ бібліотеки nltk.tokenize та вбудовані у неї функції sent\_tokenize() для розділення тексту на речення або word\_tokenize() для розділення заданого тексту на список слів.

Для **виділення частин мови** можна застосувати функцію бібліотеки nltk під назвою pos\_tag() яка приймає список токенізованих слів тексту як параметр.

Для **чанкування** можна застосувати регулярні вирази для створення граматики як програмі потрібно виділяти чанки з тексту. Прикладом граматики чанку може бути "NP: {<DT>?<JJ>\*<NN>}". Після створення змінної граматики чанкування можна створити оброблювач чанків застосуванням функції бібліотеки nltk під назвою RegexpParser(), яка приймає граматику чанкування як аргумент. Для створення дерева чанків тепер необхідно використати метод prase об’єкту оброблювача чанків вигляду chunk\_parser.parse(text\_with\_pos\_tags). При виведенні такого об’єкту через методом draw() на екрані має з’явитися дерево чанків.

[Джерело](https://realpython.com/nltk-nlp-python/)

Які основні модулі входять до складу бібліотеки NLTK?

До складу бібліотеки оброблення природньої мови NLTK для мови програмування Python входять такі основні модулі:

* nltk.book
* nltk.chunk
* nltk.grammar
* nltk.metrics
* nltk.misc
* nltk.parse
* nltk

[Джерело](https://www.nltk.org/py-modindex.html)

Яким чином можна використати готові тексти для роботи програми на Python?

Для використання готових текстів з бібліотеки оброблення природньої мови NLTK для мови програмування Python потрібно спочатку завантажити їх викликом методу download() на об’єкт nltk. Після завантаження необхідних ресурсів можна імпортувати та використати їх включаючи відповідні назви до файлу:

from nltk.corpus import gutenberg

HOLY\_BIBLE=gutenberg.words('Bible-KJV.txt')

print(HOLY\_BIBLE)

[Джерело](https://www.geeksforgeeks.org/accessing-text-corpora-and-lexical-resources-using-nltk/)

Яким чином розбити текст на лексеми?

Для розбиття заданого тексту на лексеми (окремі слова) використовуючи бібліотеку оброблення природньої мови NLTK для мови програмування Python можна використати вбудований метод частини бібліотеки nltk.tokenize під назвою word\_tokenize(text: str):

REVELATION\_22\_UKRK='''

**Одкриттє 22:1 І показав менї чисту ріку води життя, ясну як хришталь, що виходила з престола Божого і Агнцевого.**

**Одкриттє 22:2 А посеред улицї його, та й по сей і по той бік ріки - дерево життя, що родить овощі дванайцять (раз), і що місяця свій овощ дає, а листє з дерева на сцїленнє поган.**

**Одкриттє 22:3 І вже більш не буде жодного проклону; а престол Бога і Агнця буде в ньому, і слуги Його служити муть Йому.**

**Одкриттє 22:4 І побачять лице Його, а імя Його на чолах їх.**

**Одкриттє 22:5 І ночі не буде там; і не потрібувати муть сьвічника і сьвітла сонця, бо Господь Бог осьвічує їх; і царювати муть по вічні віки.**

**Одкриттє 22:6 І рече менї: Сї слова вірні і правдиві; і Господь, Бог сьвятих пророків, післав ангела свого, показати слугам своїм, що має незабаром бути.**

**Одкриттє 22:7 Ось прийду незабаром. Блаженний, хто хоронить слова пророцтва книги сієї.**

**Одкриттє 22:8 А я Йоан, що бачив се і чув; і коли чув я, і бачив, упав я поклонитись перед ногами ангела, що менї се показував.**

**Одкриттє 22:9 І каже менї: нї, глянь, я бо слуга-товариш твій, і братів твоїх пророків, і тих, що хоронять слова книги сієї: Богу поклони ся.**

**Одкриттє 22:10 І глаголе менї: Не печатай слів пророцтва книги сієї; час бо близько.**

**Одкриттє 22:11 Хто з'обіжає, нехай ще з'обіжає, і хто поганий, нехай ще опоганюєть ся; і хто праведний, нехай ще оправдуєть ся, і хто сьвятий, нехай ще осьвячуєть ся.**

**Одкриттє 22:12 І ось, я прийду хутко, і заплата моя зо мною, щоб віддати кожному, яко ж буде дїло його.**

**Одкриттє 22:13 Я Альфа і Омега, почин і конець, Первий і Останнїй.**

**Одкриттє 22:14 Блаженні, що творять заповідї Його, щоб мали власть до дерева життя, і увійшли ворітьми в город.**

**Одкриттє 22:15 А на дворі будуть пси, і чарівники, і перелюбники, і душегубцї, і ідолські служителї, і кожен, хто любить і робить лож.**

**Одкриттє 22:16 Я Ісус післав ангела мого, сьвідкувати вам усе по церквах. Я - корінь і рід Давидів, зоря ясна і рання.**

**Одкриттє 22:17 А Дух і невіста глаголють: Прийди! і хто чує, нехай каже: Прийди! Хто жадний, нехай прийде, а хто хоче, нехай приймає воду життя дармо.**

**Одкриттє 22:18 Сьвідкую ж також кожному, хто слухає словес пророцтва книги сієї: коли хто доложить до сього, доложить йому Бог і пораз, що написані в книзї сїй.**

**Одкриттє 22:19 Коли ж хто уйме від словес книги пророцтва сього, уйме Бог часть його з книги життя, і з города сьвятого, та й з того, що написано в книзї сїй.**

**Одкриттє 22:20 Сей, що про се сьвідкує, глаголе: Так, прийду хутко! Амінь. О, прийди, Господи Ісусе!**

**Одкриттє 22:21 Благодать Господа нашого Ісуса Христа з усїма вами. Амінь.**

'''

import nltk

words=nltk.word\_tokenize(REVELATION\_22\_UKRK.replace('\n',' ').strip())

print(words)

Яким чином побудувати граматику в NLTK?

Для побудови граматики використовуючи бібліотеку оброблення природньої мови NLTK для мови програмування Python можна використати вбудований метод fromstring() частини бібліотеки під назвою nltk.CFG:

import nltk.CFG

g=CFG.fromstring("""

S->NP VP

PP->P NP

NP->Det N | NP PP

VP->V NP | VP PP

Det -> 'a' | 'the'

""")

print(g, g.start(), g.productions())

[Джерело](https://www.nltk.org/howto/grammar.html)

Що таке біграми та яким чином їх визначити у тексті?

Біграми у бібліотеці оброблення природньої мови NLTK для мови програмування Python є парами послідовних слів. Для створення списку біграм потрібно спочатку створити список токенів з вхідного рядку. Далі застосувати метод bigrams() з частини бібліотеки nltk.util таким чином:

from nltk.tokenize import word\_tokenize

from nltk.util import bigrams

text="ІСУС ХРИСТОС ГОСПОДЬ БОГ ВСЕМОГУТНІЙ ВСЕВИШНІЙ ВСЕСИЛЬНИЙ АМІНЬ СЛАВА ГОСПОДУ ІСУСУ ХРИСТУ НАВІКИ ВІЧНІ АМІНЬ"

tokens=word\_tokenize(text)

bigrams\_list=list(bigrams(tokens))

for b in bigrams\_list:

print(b)

[Джерело](https://www.geeksforgeeks.org/generate-bigrams-with-nltk/)

Що таке чанкінг та яким чином він виконується?

Чанкінг або чанкування у бібліотеці оброблення природньої мови NLTK для мови програмування Python є процесом розділення тексту на чанкки або частини в залежності від їх відповідної синтаксичної форми.

import nltk

from nltk.chunk import RegexpParser

from nltk.tokenize import word\_tokenize

text='Some example text hopefully this will work LORD JESUS please help me FATHER GOD in JESUS HOLY NAME AMEN.'

tokens=word\_tokenize(text)

pos\_tags=nltk.pos\_tag(tokens)

chunk\_patterns=r'''

NP: {<DT>?<JJ>\*<NN>}

VP: {<VB.\*><NP|PP>}

'''

chunk\_parser=RegexpParser(chunk\_patterns)

res=chunk\_parser.parse(pos\_tags)

print(res)

[Джерело](https://www.educative.io/answers/what-is-chunking-in-nltk)

Яким чином та за допомогою яких засобів виконується семантична інтерпретація?

Семантична інтерпретація заданого тексту за допомогою бібліотеки оброблення природньої мови NLTK для мови програмування Python може бути здійснена шляхом використання вбудованої граматики та оброблювача природньої мови sql0.fcfg за допомогою використання завантажувача обробників nltk.load\_parser таким чином:

from nltk import load\_parser

p=load\_parser('grammars/book\_grammars/sql0.fcfg')

q='What cities are located in Greece'

t=list(p.parse(q.split()))

a=t[0].label()['SEM']

a=[s for s in a if s]

res=' '.join(a)

print(res)

from nltk.sem import chat80

nr=chat80.sql\_query('corpora/city\_database/city.db',res)

print(', '.join([w[0] for w in nr]))

[Джерело](https://www.nltk.org/book/ch10.html)

Що таке контекстно-вільна граматика? Наведіть приклади.

Контекстно-вільна граматика (Англійською **Context-Free Grammar** або **CFG**) визначає формальну мову. У формальній природній мові граматика визначається за чіткими правилами. Зазвичай у природній мови елементи складають речення, які не залежать від контексту. Звідси і походить ім’я *контекстно-вільна*. [Джерело](https://www.freecodecamp.org/news/context-free-grammar/#:~:text=Context%20Free%20Grammars%20or%20CFGs%20define%20a%20formal%20language.%20Formal%20languages%20work%20strictly%20under%20the%20defined%20rules%20and%20their%20sentences%20are%20not%20influenced%20by%20the%20context.%20And%20that%27s%20where%20it%20gets%20the%20name%20context%20free.)

Прикладом визначення та використання контекстно-вільною граматики у мові програмування Python через використання бібліотеки оброблення природної мови NLTK може бути наступний застосунок:

[Джерело](https://www.oreilly.com/library/view/natural-language-processing/9780596803346/ch08s03.html)

Яким чином визначити частину мови слів тексту?