Навчально-науковий інститут філології

Київського національного університету імені Тараса Шевченка

Кафедра української мови та прикладної лінгвістики

**Звіт із лабораторної роботи**

з дисципліни **«**Традиційна та комп’ютерна лексикографія**»**

на тему: «**Автоматичне укладання ЧС**»

Виконала студентка 3-го курсу 1-ої групи

спеціальності 035.10 «Прикладна (комп’ютерна)

лінгвістика та англійська мова»

Ярох Анна

Київ - 2023

ЗМІСТ

[ВСТУП 3](#_Toc151592653)

[СТВОРЕННЯ ТА КОНФІГУРАЦІЯ БАЗ ДАНИХ 4](#_Toc151592654)

[ВИСНОВОК 6](#_Toc151592655)

[ВИКОРИСТАНІ ДЖЕРЕЛА 9](#_Toc151592656)

[ДОДАТОК А 10](#_Toc151592657)

# ВСТУП

**Метою лабораторної роботи** є автоматичне укладання шести частотних словників на основі двох текстових вибірок різних стилів та жанрів.

**Завдання лабораторної роботи:**

* Сформувати дві текстові вибірки різних стилів, жанрів або авторів, обсягом не менше 20500 слововживань кожна, уникаючи вибору текстів, які включають багато нумерації та цифр.
* Провести процедуру токенізації для обох текстових вибірок так, щоб їх обсяг становив 20000 слововживань.
* Створити три частотні словники для першої текстової вибірки: ЧС словоформ, ЧС лем, ЧС частин мови.
* Аналогічно, створити три частотні словники для другої текстової вибірки: ЧС словоформ, ЧС лем, ЧС частин мови.
* Підготувати звіт, який включає в себе опис вибору текстових вибірок, методи токенізації, аналіз результатів та висновки.

**Об’єктом** роботи є текстові вибірки різних стилів, що містять художні та наукові тексти.

**Матеріалом** для лабораторної роботи є дві текстові вибірки:

1. Літературний твір «Прокляте небо» Наталії Довгопол у жанрі романтичної прози та любовного роману [8].
2. Наукова монографія «Перспективи розвитку туризму в Україні та світі: управління, технології, моделі» за науковим редагуванням проф. Матвійчука Л.Ю. [7].

# СТВОРЕННЯ ТА КОНФІГУРАЦІЯ БАЗ ДАНИХ

**Етап 1: Формування текстових вибірок**

* + Було використано дві різні текстові вибірки різних стилів з мінімум 20,500 слововживань.

**Етап 2: Підготовка текстових вибірок для автоматичної обробки**

* Текст кожної вибірки було переведено у формат **txt**.
  + Застосовано фільтрацію тексту та токенізацію для кожної вибірки.
  + Створено цикл, який відраховує 20,000 слововживань у кожній вибірці.

**Етап 3:** **Автоматичне укладання ЧС**

**Зчитування та попередня обробка тексту:**

* Визначено функцію **read\_and\_preprocess**, яка приймає шлях до файлу, читає його, виконує обробку тексту (наприклад, видаляє римські числа, числа та інші виключення)
* Створено функцію **remove\_roman\_numerals** для видалення римських чисел з тексту.

**Рахування та обмеження кількості слів:**

* Визначено функцію **count\_and\_divide**, яка обмежує кількість токенів до певного значення та повертає відредагований список токенів.
* Визначено функцію **divide\_on\_samples**, яка розділяє слова на підвибірки та оновлює словник зі статистикою для кожної словоформи.

**Аналіз текстів:**

* Задано список винятків (**exceptions**), які будуть видалені з тексту.
* Визначено два шляхи до текстових файлів (**file\_path\_1** і **file\_path\_2**).
* Використовуючи функцію **read\_and\_preprocess**, оброблено текст з кожного файлу.

**Обробка та аналіз слів:**

* Визначено список додаткових винятків (**additional\_exceptions**).
* Здійснено подальшу обробку слів (розбивка на токени) з використанням регулярних виразів та бібліотеки **tokenize\_uk**.
* Проведено рахунок слів та їх розподіл на підвибірки використовуючи визначені функції.

**Автоматичне укладання ЧС:**

Автоматичне укладання ЧС реалізовано за допомогою 20 циклів для кожної вибірки з обчисленням кількості вживань кожної словоформи в 20 підвибірках.

* Створено два словники (**sample\_1\_dict** і **sample\_2\_dict**), які містять інформацію про вживані слова та їх частоту.
* Створено функцію **divide\_on\_samples**, яка обчислює частоти словоформ у різних підвибірках.
* Реалізовано функцію для додавання частот словоформ до значень словника у різних підвибірках.
* Враховано ймовірність відсутності словоформи в попередніх підвибірках.

**Формування бази даних ЧС**:

* Створено базу даних SQLite для ЧС словоформ, лем та частин мови.
* Імпортовано дані зі списку **values1** у таблицю ЧС словоформ.

**Визначення частин мови**:

* З використанням бібліотеки **pymorphy3** визначено частини мови кожного слова в текстових вибірках.
* Застосовано функцію **get\_pos\_statistics** для обробки та підрахунку частот частин мови та імпортовано дані у таблицю ЧС частин мови.

**Лематизація словоформ**:

* Застосовано лематизацію за допомогою бібліотеки **pymorphy3**.
* Обчислено абсолютні частоти лем та імпортовано дані у таблицю ЧС лем.

Функції створення таблиць та вставки даних в базу даних закоментовані, і це необхідно врахувати при подальшому використанні. Повний код наданий у додатку А.

# ВИСНОВОК

У лабораторній роботі проведено автоматичне укладання частотних словників на основі двох текстових вибірок різних стилів та жанрів. Загальний обсяг кожної вибірки складає не менше 20500 слововживань, при цьому проведена токенізація з метою обмеження обсягу до 20000 слововживань.

Для обох текстових вибірок були створені три частотні словники:

1. **Частотний словник словоформ (ЧС словоформ):** включає інформацію про частоту кожної окремої словоформи у тексті.
2. **Частотний словник лем (ЧС лем):** містить дані про частоту вживання лем (нормалізованих форм слів) у тексті.
3. **Частотний словник частин мови (ЧС частин мови):** містить інформацію про частоту вживання кожної частини мови у тексті.

Матеріалом для аналізу стали дві текстові вибірки різних стилів: літературний твір «Прокляте небо» у жанрі романтичної прози та любовного роману, а також наукова монографія «Перспективи розвитку туризму в Україні та світі: управління, технології, моделі». Обрані текстові вибірки представляють собою художні та наукові тексти відповідно.

Результатом роботи є створені три таблиці в базі даних (БД) для кожної текстової вибірки: Частотний словник словоформ (ЧС словоформ), Частотний словник лем (ЧС лем) та Частотний словник частин мови (ЧС частин мови). Кожна таблиця містить інформацію про частоту вживання відповідних словоформ, лем та частин мови у тексті.

На рис. 1 наведено зображення ЧС словоформ для першої вибірки, де відображена частота кожної словоформи у тексті. Рис. 2 демонструє структуру ЧС частин мови, відображаючи частоту вживання кожної частини мови в тексті. Окремо на рис. 3 подано зображення ЧС лем, де виділено частоту вживання нормалізованих форм слів у вибірці.

На рис. 4 наведено зображення ЧС словоформ для другої вибірки, де відображена частота кожної словоформи у тексті. Рис. 5 демонструє структуру ЧС частин мови, відображаючи частоту вживання кожної частини мови в тексті. Окремо на рис. 6 подано зображення ЧС лем, де виділено частоту вживання нормалізованих форм слів у вибірці.

Отримані результати аналізу надають важливі дані для розуміння лінгвістичних особливостей кожної вибірки, дозволяючи виявити варіативність мовлення в різних стилях та жанрах текстів. Аналіз ЧС словоформ, ЧС лем та ЧС частин мови розкриває унікальні риси мовного вживання у літературному творі «Прокляте небо» та науковій монографії «Перспективи розвитку туризму в Україні та світі».

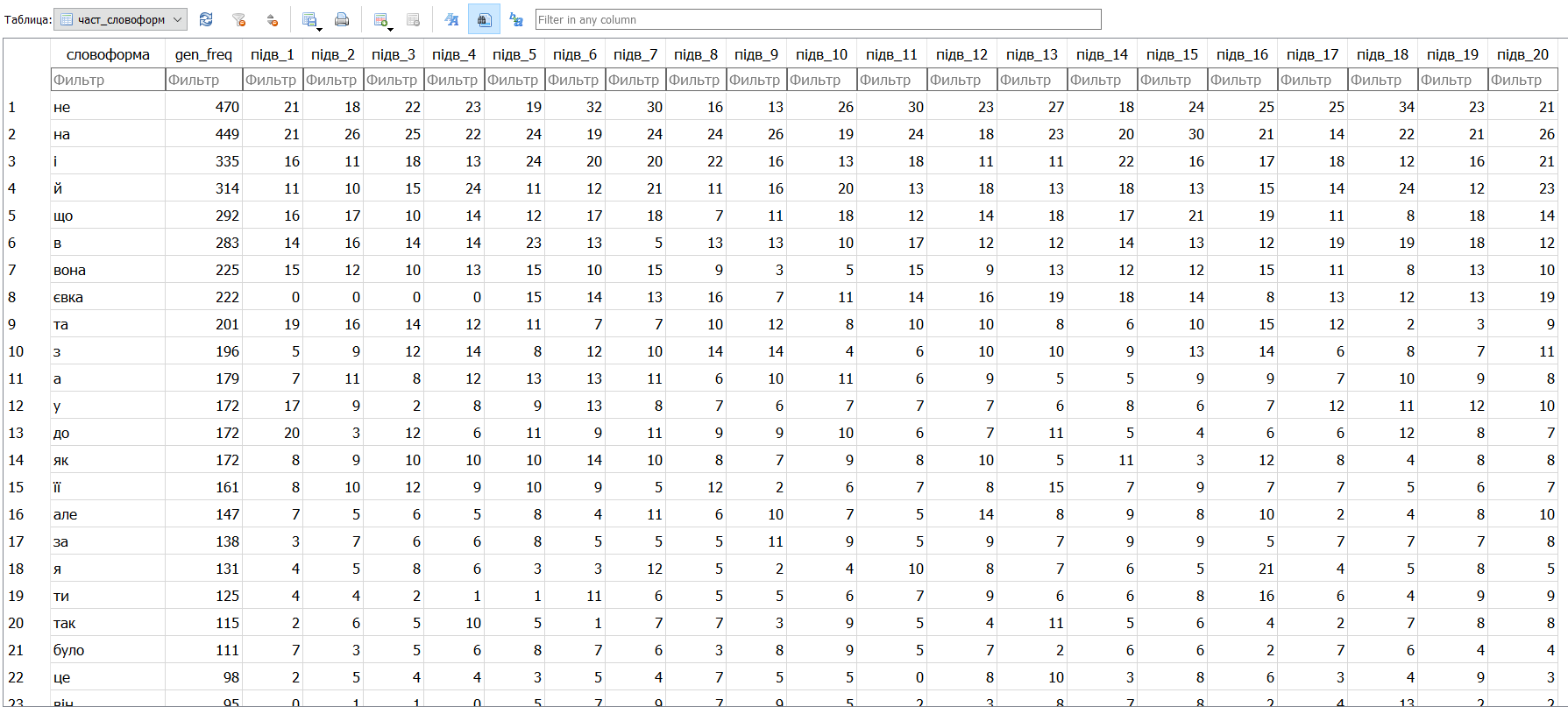


Рис. 1 «ЧС словоформ\_1»

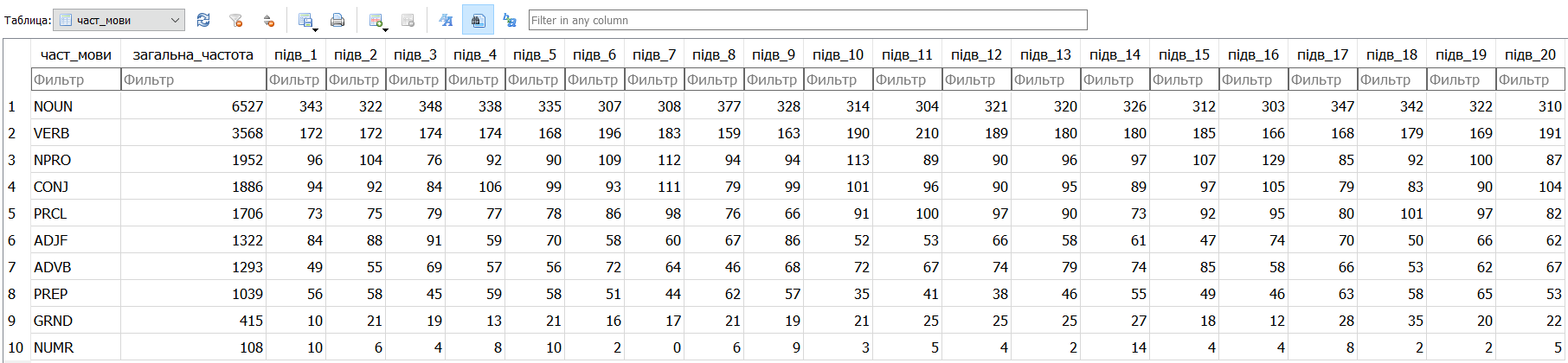


Рис. 2 «ЧС частин мови\_1»

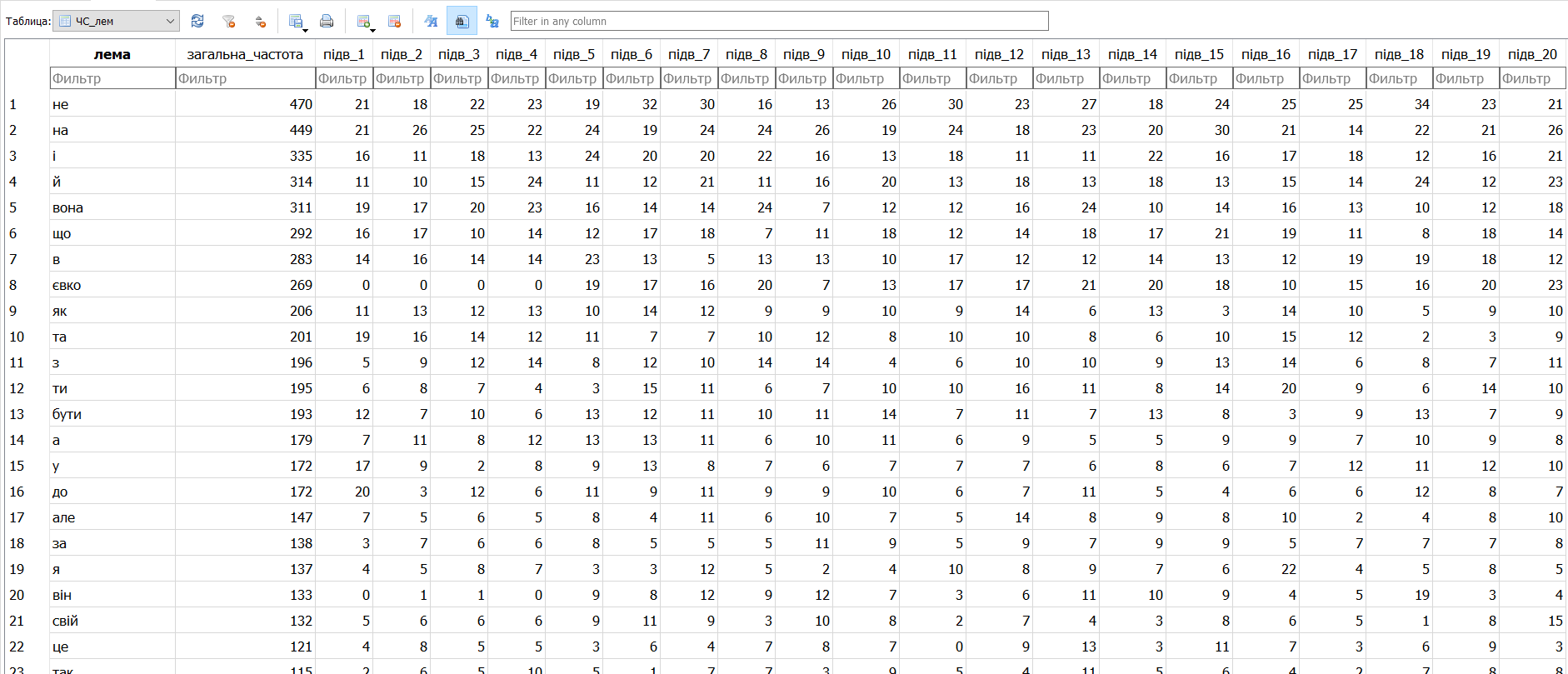


Рис. 3 «ЧС лем\_1»

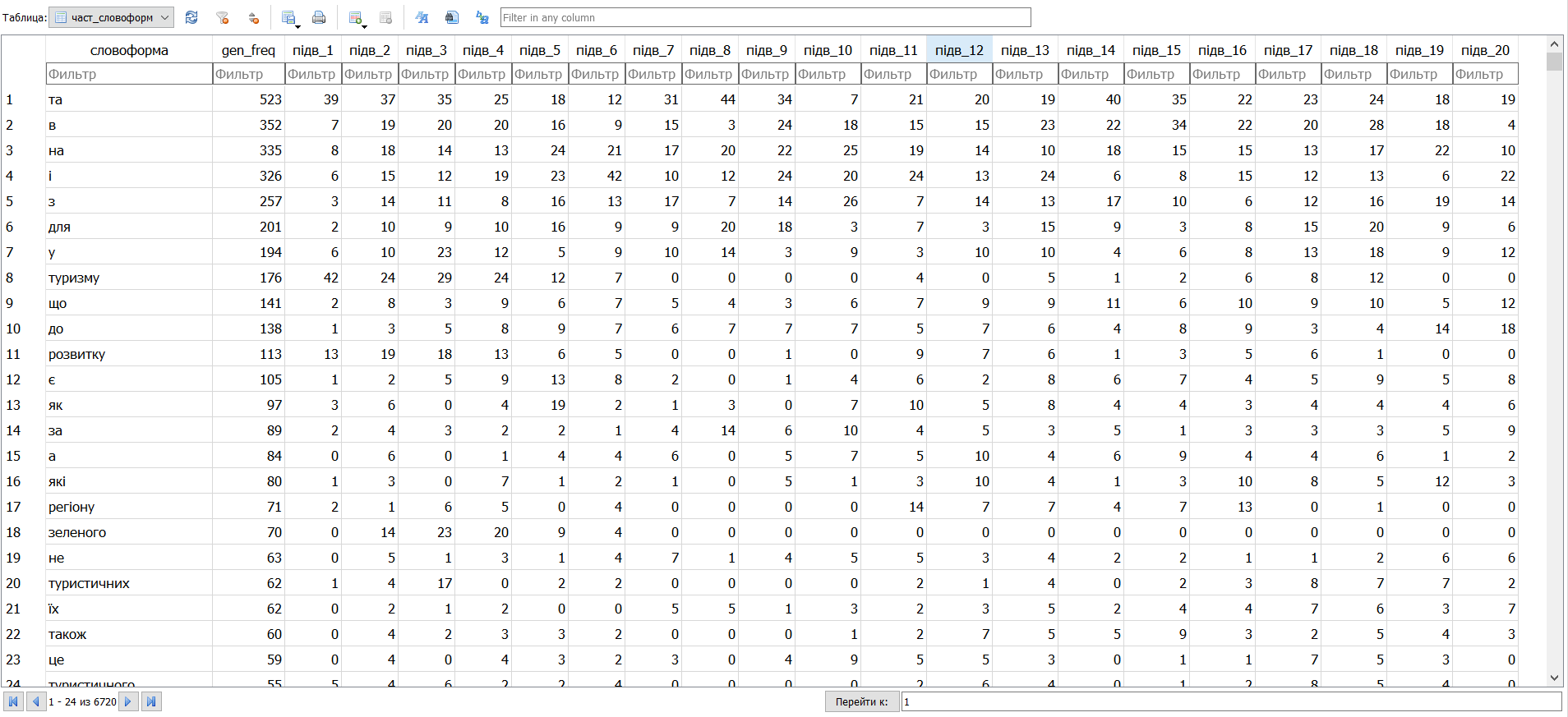


Рис. 4 «ЧС словоформ\_2»

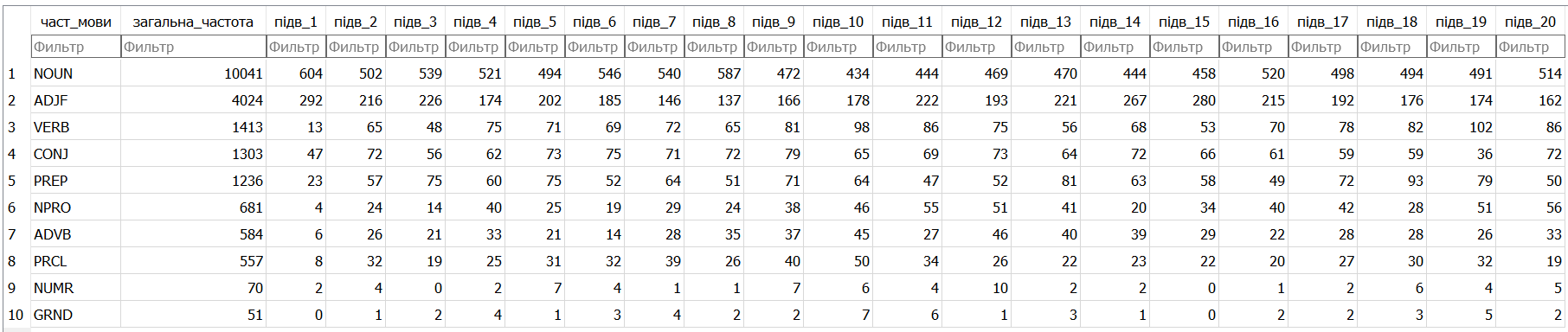


Рис. 5 «ЧС частин мови\_2»

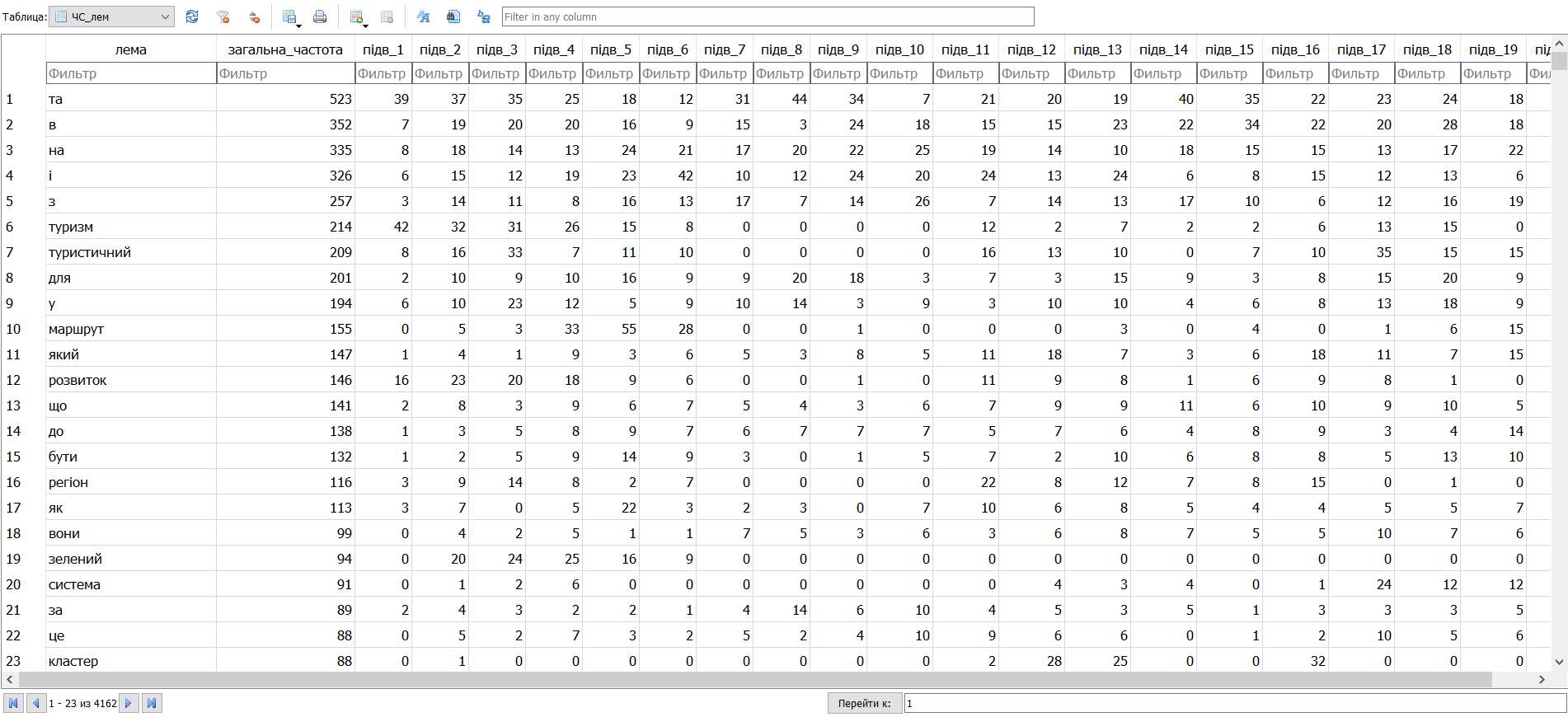


Рис. 6 «ЧС лем\_2»

# ВИКОРИСТАНІ ДЖЕРЕЛА

1. sqlite3 – модуль бібліотеки полегшеної реляційної система керування базами даних SQLite.
2. re – модуль, що забезпечує пошук регулярних виразів.
3. pymorphy 3 – морфологічний аналізатор, ВЕСУМ (для лематизації) <https://r2u.org.ua/vesum/>
4. DB Browser for SQLite - програма для створення, маніпулювання та перегляду баз даних.
5. <https://regex101.com/> - інтернет сайт для пошуку регулярних виразів у тексті.
6. <https://docs.python.org/3.7/> - документація Python 3.7
7. Перспективи розвитку туризму в Україні та світі: управління, технології, моделі: колективна монографія. Видання п’яте / за наук. ред. проф. Матвійчук Л.Ю. – Луцьк: ІВВ Луцького НТУ, 2019. – 320 с.
8. Довгопол Н.О. Прокляте небо // Видавництво книжковий клуб «Клуб Сімейного Дозвілля». – 2019. – 384 с.

# ДОДАТОК А

import re  
from tokenize\_uk import tokenize\_words  
import sqlite3  
import pymorphy3  
  
def read\_and\_preprocess(file\_path):  
 with open(file\_path, encoding="utf-8") as file:  
 text = file.read()  
 exceptions = ['вул.', 'ін.', 'ІВВ', 'матер.', 'наук.', 'Наук.', 'ред.', 'проф.', 'ISBN', 'І.', 'Є.', 'р3',  
 '«Green Ways»', 'http://volynrada.gov.ua/news/yak-rozvivali-turizm-na-volini-v-ramkakh-regionalno'  
 'yi-programi', 'НТУ', 'УДК', 'ББК', 'М.', 'АНВО', 'В.', 'УПЦ', 'Г.', 'О.', 'Л.', 'Ю.',  
 'Н.', 'М.', 'В.', 'Т.', 'К.', 'Я.', 'Л.Ю.', 'РВВ', 'ЛНТУ', 'Т.П.', 'С.', 'грн.', 'рр4', 'тис.',  
 'ПДВ', 'рис.', 'табл.', 'Рис.', 'США', 'т.п.', 'А.', 'И.', 'ГРК', 'КНТЕУ', '2-ге', 'вид.',  
 'вищ.', 'навч.', 'закл.', 'перероб.', 'допов.', 'Х.', 'Й.', '1м', 'ЗІЛ', '«Бичок»', 'посіб.',  
 'нац.', 'торг.-екон.', 'ун-т', 'кв.', '80-х', 'РАГСу', 'РАГС', 'млн.', 'обл.', 'Е.', 'Д.',  
 'СПб.', 'Изд-во', 'т.д.', 'ім.', 'ЛНУ', 'Вип.', 'т.ч.', 'П.', 'ЗАТ', 'Ф.', 'Ред.',  
 'кол.', 'НАУ', 'км.', 'г/л', 'куб.', 'м/год.', 'г/дм3', 'ПЗФ', 'з/п', 'р-ну', 'сер.',  
 'АСУ', 'ІС', 'ІДС', 'РП', 'Ін-т', 'журн.', 'всеукр.', 'Всеукр.', 'д-ра', '-ти', 'іОС', 'ДВНЗ',  
 'м/год.', 'г/дм3', 'ISO/IEC', 'ІІІ', 'ИТД', '«АртДжазКооперейшн»', 'Инфра-М', 'НДІ',  
 '"Укрпрофоздоровниця"', '"Тамед']  
 text = re.sub(r'\b(?:[А-ЯЁ]+\.)+[А-ЯЁ]+\b|\b[А-ЯЁ]+\.\b', '', text)  
 for exception in exceptions:  
 text = text.replace(exception, '')  
 text = text.lower()  
 text\_without\_digits = re.sub(r"\d", "", text)  
 filtered\_text = re.sub(r"[^\w\s'-]", "", text\_without\_digits) # видаляє всі символи, крім букв, цифр, пробілів,  
 # апострофів і тире.  
 return filtered\_text  
  
def remove\_roman\_numerals(text):  
 return re.sub(r'\b[ivxlcdm]+\b', '', text)  
  
def count\_and\_divide(tokens, exceptions, max\_tokens=20000):  
 edited\_list = []  
 count = 0  
 for token in tokens:  
 if token not in exceptions:  
 count += 1  
 edited\_list.append(token)  
 if count == max\_tokens:  
 break  
 return edited\_list, count  
  
def divide\_on\_samples(edited\_list, sample\_dict, sample\_num):  
 sample\_list = []  
 for word\_form in edited\_list:  
 if word\_form not in sample\_dict:  
 # Ініціалізуємо словоформу з нулями для 20 підсемплів  
 sample\_dict[word\_form] = [word\_form] + [0] \* 20  
 # Перевіряємо, чи є ще місце для нового підсемплу  
 if len(sample\_dict[word\_form]) <= sample\_num:  
 # Додаємо нулі, якщо потрібно  
 sample\_dict[word\_form] += [0] \* (sample\_num - len(sample\_dict[word\_form]) + 1)  
 # Збільшуємо лічильник для відповідного підсемплу  
 sample\_dict[word\_form][sample\_num] += 1  
 sample\_list.append(word\_form)  
 return sample\_list  
  
sample\_1\_dict = {}  
sample\_2\_dict = {}  
  
file\_path\_1 = "D:\\STUDIES\\lab\_slovnyk\\proklyate\_nebo.txt"  
text\_1 = read\_and\_preprocess(file\_path\_1)  
text\_without\_roman\_1 = remove\_roman\_numerals(text\_1)  
tokens\_regex\_1 = re.findall(r"\b(?:[А-ЯЁа-яёіїєґҐ']+(?:-|—)?(?:'[А-ЯЁа-яёіїєґҐ]+)?)+\b", text\_without\_roman\_1)  
tokens\_tokenize\_uk\_1 = list(tokenize\_words(text\_without\_roman\_1))  
splitted\_1 = tokens\_regex\_1 + tokens\_tokenize\_uk\_1  
splitted\_1 = [token for token in splitted\_1 if token]  
additional\_exceptions = ['л', 'с', 'м', '‘', 'га', 'км', 'кг', 'од', 'ил', 'ст', 'р', 'ок', 'осм', 'ос', "'", 'іт',  
 'інтернет-конф', 'смт', 'гр', 'ла', 'прикді', 'придніпровя', 'хіх']  
edited\_list\_1, count\_1 = count\_and\_divide(splitted\_1 + additional\_exceptions, additional\_exceptions)  
while count\_1 < 20000:  
 remaining\_text\_1 = read\_and\_preprocess(file\_path\_1)  
 remaining\_tokens\_regex\_1 = re.findall(r"\b(?:[А-ЯЁа-яёіїєґҐ']+(?:-|—)?(?:'[А-ЯЁа-яёіїєґҐ]+)?)+\b", remaining\_text\_1)  
 remaining\_tokens\_tokenize\_uk\_1 = list(tokenize\_words(remaining\_text\_1))  
 remaining\_splitted\_1 = remaining\_tokens\_regex\_1 + remaining\_tokens\_tokenize\_uk\_1  
 remaining\_splitted\_1 = [token for token in remaining\_splitted\_1 if token]  
 edited\_list\_1 = divide\_on\_samples(edited\_list\_1 + remaining\_splitted\_1 + additional\_exceptions, sample\_1\_dict, 1)  
 count\_1 = len(edited\_list\_1)  
  
file\_path\_2 = "D:\\STUDIES\\lab\_slovnyk\\monography.txt"  
text\_2 = read\_and\_preprocess(file\_path\_2)  
text\_without\_roman\_2 = remove\_roman\_numerals(text\_2)  
tokens\_regex\_2 = re.findall(r"\b(?:[А-ЯЁа-яёіїєґҐ']+(?:-|—)?(?:'[А-ЯЁа-яёіїєґҐ]+)?)+\b", text\_without\_roman\_2)  
tokens\_tokenize\_uk\_2 = list(tokenize\_words(text\_without\_roman\_2))  
splitted\_2 = tokens\_regex\_2 + tokens\_tokenize\_uk\_2  
splitted\_2 = [token for token in splitted\_2 if token]  
edited\_list\_2, count\_2 = count\_and\_divide(splitted\_2 + additional\_exceptions, additional\_exceptions)  
while count\_2 < 20000:  
 remaining\_text\_2 = read\_and\_preprocess(file\_path\_2)  
 remaining\_tokens\_regex\_2 = re.findall(r"\b(?:[А-ЯЁа-яёіїєґҐ']+(?:-|—)?(?:'[А-ЯЁа-яёіїєґҐ]+)?)+\b", remaining\_text\_2)  
 remaining\_tokens\_tokenize\_uk\_2 = list(tokenize\_words(remaining\_text\_2))  
 remaining\_splitted\_2 = remaining\_tokens\_regex\_2 + remaining\_tokens\_tokenize\_uk\_2  
 remaining\_splitted\_2 = [token for token in remaining\_splitted\_2 if token]  
 edited\_list\_2 = divide\_on\_samples(edited\_list\_2 + remaining\_splitted\_2 + additional\_exceptions, sample\_2\_dict, 2)  
 count\_2 = len(edited\_list\_2)  
  
sample\_1\_dict = {}  
sample\_2\_dict = {}  
  
for i in range(20):  
 sub\_sample\_start = i \* 1000  
 sub\_sample\_end = (i + 1) \* 1000  
 sub\_sample\_1 = edited\_list\_1[sub\_sample\_start:sub\_sample\_end]  
 sub\_sample\_counts = {}  
 for word\_form in sub\_sample\_1:  
 sub\_sample\_counts[word\_form] = sub\_sample\_counts.get(word\_form, 0) + 1  
 for word\_form, count in sub\_sample\_counts.items():  
 if word\_form not in sample\_1\_dict:  
 sample\_1\_dict[word\_form] = [word\_form] + [0] \* 20  
 sample\_1\_dict[word\_form][i + 1] = count  
  
# Прокручуємо 20 циклів для другого файлу  
for i in range(20):  
 # Витягуємо наступні 1000 слів зі списку edited\_list\_2  
 sub\_sample\_start = i \* 1000  
 sub\_sample\_end = (i + 1) \* 1000  
 sub\_sample\_2 = edited\_list\_2[sub\_sample\_start:sub\_sample\_end]  
 # Підрахуємо випадки появи кожної словоформи у підвибірці  
 sub\_sample\_counts = {}  
 for word\_form in sub\_sample\_2:  
 sub\_sample\_counts[word\_form] = sub\_sample\_counts.get(word\_form, 0) + 1  
 # Оновлюємо sample\_2\_dict підрахунками для поточної підвибірки  
 for word\_form, count in sub\_sample\_counts.items():  
 if word\_form not in sample\_2\_dict:  
 sample\_2\_dict[word\_form] = [word\_form] + [0] \* 20 # Ініціалізація нулями для 20 підзразків  
 # Оновлюємо кількість для поточної підвибірки  
 sample\_2\_dict[word\_form][i + 1] = count  
  
# Отримання списку значень (values) зі словника  
values1 = list(sample\_1\_dict.values())  
# Додаємо загальну частоту до кожного підсписку в values1  
for sublist in values1:  
 sublist.insert(1, sum(sublist[1:])) # Вставляємо загальну частоту на другу позицію  
# Сортуємо за загальною частотою в порядку спадання  
values1\_ordered = sorted(values1, key=lambda x: x[1], reverse=True)  
# Отримуємо спискок значень (values) зі словника для другої вибірки  
values2 = list(sample\_2\_dict.values())  
# Додаємо загальну частоту до кожного підсписку в values2  
for sublist in values2:  
 sublist.insert(1, sum(sublist[1:])) # Вставте загальну частоту на другу позицію  
# Сортуємо за загальною частотою в порядку спадання  
values2\_ordered = sorted(values2, key=lambda x: x[1], reverse=True)  
# Ініціалізуємо pymorphy3 для української мови  
morph = pymorphy3.MorphAnalyzer(lang='uk')  
conn = sqlite3.connect('sample\_1.db')  
cursor = conn.cursor()  
cursor.execute('''CREATE TABLE IF NOT EXISTS част\_словоформ  
 (словоформа TEXT PRIMARY KEY NOT NULL,  
 загальна\_частота INTEGER,  
 підв\_1 INTEGER,  
 підв\_2 INTEGER,  
 підв\_3 INTEGER,  
 підв\_4 INTEGER,  
 підв\_5 INTEGER,  
 підв\_6 INTEGER,  
 підв\_7 INTEGER,  
 підв\_8 INTEGER,  
 підв\_9 INTEGER,  
 підв\_10 INTEGER,  
 підв\_11 INTEGER,  
 підв\_12 INTEGER,  
 підв\_13 INTEGER,  
 підв\_14 INTEGER,  
 підв\_15 INTEGER,  
 підв\_16 INTEGER,  
 підв\_17 INTEGER,  
 підв\_18 INTEGER,  
 підв\_19 INTEGER,  
 підв\_20 INTEGER)  
 ''')  
# Додаємо дані з values1\_ordered до таблиці ЧС\_словоформ  
for i in values1\_ordered:  
 cursor.execute("""  
 INSERT INTO част\_словоформ VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)  
 """, i)  
conn.commit()  
conn.close()  
conn = sqlite3.connect('sample\_2.db')  
cursor = conn.cursor()  
cursor.execute('''CREATE TABLE IF NOT EXISTS част\_словоформ  
 (словоформа TEXT PRIMARY KEY NOT NULL,  
 загальна\_частота INTEGER,  
 підв\_1 INTEGER,  
 підв\_2 INTEGER,  
 підв\_3 INTEGER,  
 підв\_4 INTEGER,  
 підв\_5 INTEGER,  
 підв\_6 INTEGER,  
 підв\_7 INTEGER,  
 підв\_8 INTEGER,  
 підв\_9 INTEGER,  
 підв\_10 INTEGER,  
 підв\_11 INTEGER,  
 підв\_12 INTEGER,  
 підв\_13 INTEGER,  
 підв\_14 INTEGER,  
 підв\_15 INTEGER,  
 підв\_16 INTEGER,  
 підв\_17 INTEGER,  
 підв\_18 INTEGER,  
 підв\_19 INTEGER,  
 підв\_20 INTEGER)  
 ''')  
# Додаємр дані з values2\_ordered до нової таблиці част\_словоформ  
for i in values2\_ordered:  
 cursor.execute("""  
 INSERT INTO част\_словоформ VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)  
 """, i)  
conn.commit()  
conn.close()  
  
conn = sqlite3.connect('sample\_1.db')  
cursor = conn.cursor()  
  
def create\_table():  
 cursor.execute('''CREATE TABLE IF NOT EXISTS част\_мови  
 (частина\_мови TEXT PRIMARY KEY NOT NULL,  
 загальна\_частота INTEGER,  
 підв\_1 INTEGER,  
 підв\_2 INTEGER,  
 підв\_3 INTEGER,  
 підв\_4 INTEGER,  
 підв\_5 INTEGER,  
 підв\_6 INTEGER,  
 підв\_7 INTEGER,  
 підв\_8 INTEGER,  
 підв\_9 INTEGER,  
 підв\_10 INTEGER,  
 підв\_11 INTEGER,  
 підв\_12 INTEGER,  
 підв\_13 INTEGER,  
 підв\_14 INTEGER,  
 підв\_15 INTEGER,  
 підв\_16 INTEGER,  
 підв\_17 INTEGER,  
 підв\_18 INTEGER,  
 підв\_19 INTEGER,  
 підв\_20 INTEGER)  
 ''')  
create\_table()  
  
def count\_parts\_of\_speech(sample\_dict):  
 part\_of\_speech\_counts = {}  
 for word\_form, counts\_per\_subsample in sample\_dict.items():  
 for i, count in enumerate(counts\_per\_subsample[1:], start=1):  
 # Визначення частини мови за допомогою pymorphy3  
 parsed\_word = morph.parse(word\_form)[0]  
 part\_of\_speech = parsed\_word.tag.POS  
 # Перевірка, чи частина мови не є None перед оновленням лічильників  
 if part\_of\_speech is not None:  
 # Оновлення лічильників у словнику part\_of\_speech\_counts  
 if part\_of\_speech not in part\_of\_speech\_counts:  
 part\_of\_speech\_counts[part\_of\_speech] = [part\_of\_speech] + [0] \* max(20, i)  
 while len(part\_of\_speech\_counts[part\_of\_speech]) <= i:  
 part\_of\_speech\_counts[part\_of\_speech].append(0)  
 part\_of\_speech\_counts[part\_of\_speech][i] += count  
 return part\_of\_speech\_counts  
  
# Рахуємо частини мови за кожним зразком  
parts\_of\_speech\_1 = count\_parts\_of\_speech(sample\_1\_dict)  
  
def insert\_data\_into\_table(sample\_name, part\_of\_speech\_counts):  
 # Вставляємо дані в таблицю  
 for values in part\_of\_speech\_counts.values():  
 cursor.execute(f'''  
 INSERT INTO {sample\_name} VALUES {tuple(values)}  
 ''')  
  
# Викликаємо функцію insert\_data\_into\_table, щоб вставити дані в таблицю  
insert\_data\_into\_table('част\_мови', parts\_of\_speech\_1)  
cursor.execute('''SELECT \* FROM част\_мови ORDER BY загальна\_частота DESC''')  
sorted\_data = cursor.fetchall()  
cursor.execute('''DELETE FROM част\_мови''')  
# Вставка відсортованиз даних  
for row in sorted\_data:  
 cursor.execute('''INSERT INTO част\_мови VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)  
 ''', row)  
conn.commit()  
conn.close()  
  
conn = sqlite3.connect('sample\_2.db')  
cursor = conn.cursor()  
# Визначаємо функцію для створення таблиці для заданого зразка  
def create\_table():  
 cursor.execute('''CREATE TABLE IF NOT EXISTS част\_мови  
 (част\_мови TEXT,  
 загальна\_частота INTEGER,  
 підв\_1 INTEGER,  
 підв\_2 INTEGER,  
 підв\_3 INTEGER,  
 підв\_4 INTEGER,  
 підв\_5 INTEGER,  
 підв\_6 INTEGER,  
 підв\_7 INTEGER,  
 підв\_8 INTEGER,  
 підв\_9 INTEGER,  
 підв\_10 INTEGER,  
 підв\_11 INTEGER,  
 підв\_12 INTEGER,  
 підв\_13 INTEGER,  
 підв\_14 INTEGER,  
 підв\_15 INTEGER,  
 підв\_16 INTEGER,  
 підв\_17 INTEGER,  
 підв\_18 INTEGER,  
 підв\_19 INTEGER,  
 підв\_20 INTEGER)  
 ''')  
create\_table()  
  
def count\_parts\_of\_speech(sample\_dict):  
 part\_of\_speech\_counts = {}  
 for word\_form, counts\_per\_subsample in sample\_dict.items():  
 for i, count in enumerate(counts\_per\_subsample[1:], start=1):  
 # Визначення частини мови за допомогою pymorphy3  
 parsed\_word = morph.parse(word\_form)[0]  
 part\_of\_speech = parsed\_word.tag.POS  
 # Перевірка, чи частина мови не є None перед оновленням лічильників  
 if part\_of\_speech is not None:  
 # Оновлення лічильників у словнику part\_of\_speech\_counts  
 if part\_of\_speech not in part\_of\_speech\_counts:  
 part\_of\_speech\_counts[part\_of\_speech] = [part\_of\_speech] + [0] \* max(20, i)  
 # # Переконання, що довжина списку достатня  
 while len(part\_of\_speech\_counts[part\_of\_speech]) <= i:  
 part\_of\_speech\_counts[part\_of\_speech].append(0)  
 part\_of\_speech\_counts[part\_of\_speech][i] += count  
 return part\_of\_speech\_counts  
  
# Рахуємо частини мови за кожним зразком  
parts\_of\_speech\_2 = count\_parts\_of\_speech(sample\_2\_dict)  
  
def insert\_data\_into\_table(sample\_name, part\_of\_speech\_counts):  
 # Вставляємо дані в таблицю  
 for values in part\_of\_speech\_counts.values():  
 cursor.execute(f'''  
 INSERT INTO {sample\_name} VALUES {tuple(values)}  
 ''')  
# Викликаємо функцію insert\_data\_into\_table, щоб вставити дані в таблицю  
insert\_data\_into\_table('част\_мови', parts\_of\_speech\_2)  
# Запрос для відсортування таблиці част\_мови за спаданням загальної частоти  
cursor.execute('''SELECT \* FROM част\_мови ORDER BY загальна\_частота DESC''')  
sorted\_data = cursor.fetchall()  
cursor.execute('''DELETE FROM част\_мови''')  
# Вставка відсортованиз даних  
for row in sorted\_data:  
 cursor.execute('''INSERT INTO част\_мови VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)  
 ''', row)  
conn.commit()  
conn.close()  
  
conn = sqlite3.connect('sample\_1.db')  
cursor = conn.cursor()  
  
# Створення таблиці ЧС\_лем  
cursor.execute('''CREATE TABLE IF NOT EXISTS ЧС\_лем  
 (лема TEXT,  
 загальна\_частота INTEGER,  
 підв\_1 INTEGER,  
 підв\_2 INTEGER,  
 підв\_3 INTEGER,  
 підв\_4 INTEGER,  
 підв\_5 INTEGER,  
 підв\_6 INTEGER,  
 підв\_7 INTEGER,  
 підв\_8 INTEGER,  
 підв\_9 INTEGER,  
 підв\_10 INTEGER,  
 підв\_11 INTEGER,  
 підв\_12 INTEGER,  
 підв\_13 INTEGER,  
 підв\_14 INTEGER,  
 підв\_15 INTEGER,  
 підв\_16 INTEGER,  
 підв\_17 INTEGER,  
 підв\_18 INTEGER,  
 підв\_19 INTEGER,  
 підв\_20 INTEGER)  
 ''')  
  
def count\_lemmas(sample\_dict):  
 lemma\_counts = {}  
 for word\_form, counts\_per\_subsample in sample\_dict.items():  
 for i, count in enumerate(counts\_per\_subsample[1:], start=1): # використовується функція enumerate, щоб  
 # отримати індекс (i) та відповідну частоту (count) для кожної субвибірки, починаючи з першої  
 # (індекс 1).  
 # Лематизація слова  
 lemma = morph.parse(word\_form)[0].normal\_form  
 # Оновлення лічильника в словникові лем  
 if lemma not in lemma\_counts:  
 lemma\_counts[lemma] = [lemma] + [0] \* max(20, i)  
 # Переконаємось, що довжина списку достатня  
 while len(lemma\_counts[lemma]) <= i:  
 lemma\_counts[lemma].append(0)  
 lemma\_counts[lemma][i] += count  
 return lemma\_counts  
  
  
lemmas\_1 = count\_lemmas(sample\_1\_dict)  
  
for values in lemmas\_1.values():  
 cursor.execute(f'''  
 INSERT INTO ЧС\_лем VALUES {tuple(values)}  
 ''')  
  
# Запит для відсортування таблиці ЧС\_лем за спаданням загальної частоти  
cursor.execute('''SELECT \* FROM ЧС\_лем ORDER BY загальна\_частота DESC''')  
sorted\_data = cursor.fetchall()  
cursor.execute('''DELETE FROM ЧС\_лем''')  
# Вставка відсортованиз даних  
for row in sorted\_data:  
 cursor.execute('''INSERT INTO ЧС\_лем VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)  
 ''', row)  
conn.commit()  
conn.close()  
  
conn = sqlite3.connect('sample\_2.db')  
cursor = conn.cursor()  
cursor.execute('''CREATE TABLE IF NOT EXISTS ЧС\_лем  
 (лема TEXT,  
 загальна\_частота INTEGER,  
 підв\_1 INTEGER,  
 підв\_2 INTEGER,  
 підв\_3 INTEGER,  
 підв\_4 INTEGER,  
 підв\_5 INTEGER,  
 підв\_6 INTEGER,  
 підв\_7 INTEGER,  
 підв\_8 INTEGER,  
 підв\_9 INTEGER,  
 підв\_10 INTEGER,  
 підв\_11 INTEGER,  
 підв\_12 INTEGER,  
 підв\_13 INTEGER,  
 підв\_14 INTEGER,  
 підв\_15 INTEGER,  
 підв\_16 INTEGER,  
 підв\_17 INTEGER,  
 підв\_18 INTEGER,  
 підв\_19 INTEGER,  
 підв\_20 INTEGER)  
 ''')  
conn.commit()  
conn.close()  
  
def count\_lemmas(sample\_dict):  
 lemma\_counts = {}  
 for word\_form, counts\_per\_subsample in sample\_dict.items():  
 for i, count in enumerate(counts\_per\_subsample[1:], start=1):  
 lemma = morph.parse(word\_form)[0].normal\_form  
 if lemma not in lemma\_counts:  
 lemma\_counts[lemma] = [lemma] + [0] \* max(20, i)  
 while len(lemma\_counts[lemma]) <= i:  
 lemma\_counts[lemma].append(0)  
 lemma\_counts[lemma][i] += count  
 return lemma\_counts  
  
lemmas\_2 = count\_lemmas(sample\_2\_dict)  
  
conn = sqlite3.connect('sample\_2.db')  
cursor = conn.cursor()  
for values in lemmas\_2.values():  
 cursor.execute(f'''  
 INSERT INTO ЧС\_лем VALUES {tuple(values)}  
 ''')  
  
# Запит для відсортування таблиці ЧС\_лем за спаданням загальної частоти  
cursor.execute('''SELECT \* FROM ЧС\_лем ORDER BY загальна\_частота DESC''')  
sorted\_data = cursor.fetchall()  
cursor.execute('''DELETE FROM ЧС\_лем''')  
# Вставка відсортованиз даних  
for row in sorted\_data:  
 cursor.execute('''INSERT INTO ЧС\_лем VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)  
 ''', row)  
conn.commit()  
conn.close()