КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА
Факультет інформаційних технологій
Кафедра інтелектуальних технологій

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
БАКАЛАВРА
НА ТЕМУ

«Програмний модуль аналізу та обробки текстів у соціальних мережах»

Галузь знань **12 «Інформаційні технології»**

Спеціальність **122 «Комп'ютерні науки»**

Освітня програма **«Комп'ютерні науки»**

Освітній рівень: **бакалавр**

Виконав:
студент 4 курсу групи КН-42
Абрамов К.В.

Керівник:
Сорока Петро Миколайович,
кандидат фізико-математичних наук,
доцент

Випускна кваліфікаційна робота бакалавра допущена до захисту
рішенням кафедри *інтелектуальних технологій*
Протокол № ___ від «____» _____ 2023 р.
Зав. кафедри _____ доц. Іларіонов О.Є.

Київ – 2023

Анотація

Абрамов Кірілл Вікторович виконав випускну кваліфікаційну роботу на тему «Програмний модуль аналізу та обробки текстів у соціальних мережах» за спеціальністю 122 – «Комп'ютерні науки».

У кваліфікаційній роботі проведено аналіз існуючих підходів та методів обробки неструктурованих текстових даних, методів тематичного моделювання, спроектовано програмний модуль аналізу та обробки текстів у соціальних мережах, на прикладі соціальної мережі Telegram, розроблено відповідне програмне забезпечення, програмно реалізовано метод тематичного моделювання.

Ключові слова: NLP, машинне навчання, text mining, обробка тексту, тематичне моделювання, LSA, pLSA, LDA.

Summary

The degree project: «Software module for analysis and processing of texts in social networks» has been completed by **Abramov Kirill** by specialty 122 – «Computer Science».

In the graduation thesis, an analysis of existing approaches and methods of processing unstructured text data, methods of thematic modeling, a system of analysis and processing of texts in social networks was designed, using the Telegram social network as an example, appropriate software was developed, and a method of thematic modeling was implemented.

Keywords: NLP, machine learning, text mining, text processing, topic modeling, LSA, pLSA, LDA.

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

ІМЕНІ ТАРАСА ШЕВЧЕНКА

Факультет інформаційних технологій

Кафедра інтелектуальних технологій

Спеціальність 122 «Комп'ютерні науки»

ЗАТВЕРДЖУЮ
Зав. кафедри інтелектуальних технологій

к.т.н., доц. Іларіонов О.Є.
(звання, прізвище та ініціали)

(підпис)

« ___ » _____ 2023 р.

ЗАВДАННЯ

НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

Абрамову Кіріллу Вікторовичу
(прізвище, ім'я, по батькові)

1. Тема роботи: «Програмний модуль аналізу та обробки текстів у соціальних мережах»

затверджена наказом ректора від «___» 2023 року №

2. Термін виконання проекту (роботи): з 16.02.2023 до 06.06.2023

3. Вихідні дані до роботи: розробити програмний модуль аналізу та обробки текстів у соціальних мережах

4. Зміст пояснювальної записки (перелік питань, що підлягають розробці):

1) аналітичний огляд та постановка задачі для програмного модулю аналізу та обробки текстів у соціальних мережах;

2) проектні рішення для програмного модулю аналізу та обробки текстів у соціальних мережах;

3) програмна реалізація та дослідження результатів для програмного модулю аналізу та обробки текстів у соціальних мережах.

5. Перелік презентаційного матеріалу (з точним зазначенням обов'язкових презентацій):

1) Мета, об'єкт та предмет дослідження (1 слайд);

2) Актуальність теми (1 слайд);

3) Задача (1 слайд);

4) Область застосування (1 слайд);

5) Аналіз методів та технологій (1 слайд);

6) Найбільш відомі компанії, які займаються цією темою (1 слайд);

7) Тематичне моделювання (2 слайд);

8) Проектні рішення для реалізації програмного модулю (5 слайди);

9) Засоби програмної реалізації (1 слайд);

10) Програмна реалізація програмного модулю (1 слайд);

11) Аналіз результатів (3 слайд);

12) Висновок (1 слайд);

6. Консультанти з випускної кваліфікаційної роботи із зазначенням її розділів, що їх стосуються

Розділ	Консультант	Завдання видав	Завдання прийняв
1			
2			
3			

7. Дата видачі завдання 15 лютого 2023 року.

Керівник випускної кваліфікаційної роботи _____ / І.М. Сорока /
(підпис) (ініціали та прізвище)

Завдання прийняв до виконання _____ / К.В. Абрамов /
(підпис) (ініціали та прізвище)

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів випускної кваліфікаційної роботи	Термін виконання етапів	Примітка
1	Перше процентування	16.02.2023 - 01.03.2023	
2	Друге процентування	01.04.2023 - 10.04.2023	
3	Третє процентування	01.05.2023 - 10.05.2023	
4	Передзахист	19.05.2023 - 28.05.2023	
5	Перевірка готових дипломних робіт на плагіат	30.05.2023 - 06.06.2023	
6	Здача готових дипломних робіт на кафедру	07.06.2023 - 11.06.2023	
7			
8			
9			
10			

Керівник випускної кваліфікаційної роботи  / П.М. Сорока /
(підпис) (ініціали та прізвище)

Студент  / К.В. Абрамов /
(підпис) (ініціали та прізвище)

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

Data mining – інтелектуальний аналіз даних;

Natural Language Processing (NLP) – обробка природної мови;

Text mining (IAT) – інтелектуальний аналіз тексту;

LDA – Латентне розміщення Діріхле;

LSA – Латентно-семантичний аналіз;

pLSA – Імовірнісний латентно-семантичний аналіз;

IDEF0 – (Icam DEFinition) - функціональна модель системи;

ПК, РК – первинний ключ;

ЗК – зовнішній ключ;

ІПСО – інформаційно-психологічна операція;

Фейк – підробка чи імітація новин.

ЗМІСТ

ВСТУП

На сьогодні соціальні мережі - один із вагомих засобів комунікації між людьми. Величезна кількість інформації виплескується в соціальні мережі у тому числі інформації, яка б могла сигналізувати про потенційну загрозу різного характеру. Своєчасне виявлення цих даних могло б надати можливість вчасно попередити або уникнути небезпечних ситуацій таких як терористичні акти, кримінальні правопорушення та злочини, кібербулінг, кіднепінг. Користь з цього потоку інформації можна отримати лише при правильній обробці та аналізі.

Задача цієї кваліфікаційної роботи полягає в розробці програмного модулю для аналізу та обробки неструктурованих текстів у соціальних мережах на прикладі постів у соціальній мережі Telegram. Даний модуль може бути використана спецслужбами, оскільки результати аналізу можуть вказувати на загрозу. Спеціалізовані безпекові підрозділи самостійно будуть приймати рішення про важливість та актуальність даної інформації, а також про комплекс заходів реагування та пріоритетність задач. Іноді випадково написана фраза може змінити перебіг подій. Особливо актуальним це є для українських реалій. Тому що контекстний аналіз текстових повідомлень на прикладі телеграм каналів так званих «воєнкорів» та спільнот злочинних угрупувань, які знаходяться на окупованих територіях України, а також окупаційних військ Російської Федерації, може надати спецслужбам необхідну та неочікувану інформацію з боку противника.

Метою кваліфікаційної роботи є розробка програмного модулю для аналізу та обробки текстів у соціальних мережах на прикладі соціальної мережі Telegram.

Об'єктом роботи є процес аналізу та обробки текстових повідомлень у соціальних мережах на прикладі постів у соціальній мережі Telegram.

Предметом роботи виступають методи та технології аналізу та обробки неструктурованих текстових даних.

Завданнями кваліфікаційної роботи є:

- дослідження існуючих підходів методів, виявлення їх недоліків;
- розробка підходу до аналізу текстових повідомлень;
- проектування та реалізація програмного модулю аналізу текстових повідомлень;
- оцінка ефективності роботи програмного модулю.

РОЗДІЛ 1. Аналітичний огляд процесу аналізу текстових повідомлень. Постановка задачі кваліфікаційної роботи.

1.1 Опис проблеми, що вирішується, та її актуальність

На сьогоднішній день мільйони людей спілкуються в соціальних мережах. Усього активних користувачів соціальних мереж на початок липня 2022 налічувалося 4,7 мільярда, що становить 58% від населення Землі. Ця цифра невпинно зростає. Найпопулярнішими соціальними мережами є Facebook, Twitter, Youtube, Instagram, Tiktok та Telegram. Роль соціальних мереж не обмежується лише функцією спілкування та передачі інформації. Їх також використовують для розповсюдження реклами, впливу, пропаганди, фейків, емоційних вкидів та іпсо.

Повномасштабне вторгнення РФ на територію України 24 лютого 2022 року призвело до того, що багато українців почали використовувати соціальні мережі, як джерело інформації. Дослідження компанії GlobalLogic свідчить, що в Україні кількість користувачів соцмереж зросла з 60% населення у 2021 році до 76,6% у липні 2022 року. Згідно з дослідженням Київського міжнародного інституту соціології, що було проведено з 2 по 26 травня 2022 року, найпопулярнішою мережею в Україні є Telegram. Його як джерело інформації використовує 66% опитаних [2].

Через цензуру, кримінальну відповідальність за свободу слова та обмеження роботи Facebook, Instagram на території країни агресора, лідером серед месенджерів в Росії також став Telegram, наприклад лише за перші тижні березня об’єм трафіку виріс з 48% до 63% порівняно з початком лютого [3]. Суттєвою відмінністю від інших соціальних мереж для авторів контенту є можливість створення анонімних телеграм каналів та фактична відсутність цензури. Таким чином, ми маємо платформу, де люди з різними поглядами можуть писати що завгодно, при цьому ніякої відповідальності за достовірність

та правдивість цієї інформації не несуть, але при цьому інформація, яка б мала бути секретною також з’являється в глибинах телеграм каналів. Такі «витоки» можуть бути корисними, оскільки на війні випадкова фраза може стати цінною інформацією, що змінить хід військових дій. Наприклад, інформація про розташування, переміщення, кількість російських військ, про постачання, оснащення, результати обстрілів, кількість вбитих, поранених, дезертирів, захоплених у полон, а також розповсюдження фейкової (неправдивої) інформації та ІПСО (інформаційно-психологічна операція на іноземну аудиторію задля досягнення певної цілі - дезінформації) тощо. Прикладом впливу на українську аудиторію та проведення ІПСО з боку РФ, може бути факт масового створення нових Телеграм-каналів на самому початку повномасштабного вторгнення Росії, в період з 24 лютого 2022 року по кінець березня 2022 року в кількості 120 каналів. Це - системна стратегія створення локальних джерел пропаганди особливо на окупованих територіях в цей період (в окупованих містах, райцентрах тощо), з метою впливу на місцеве населення, сприяння окупації та розповсюдженню фейкової інформації. Локальним ресурсам, які змішують місцеві новини з тезами пропаганди, легше завойовувати місцеву аудиторію та здобувати її довіру [23].

1.2 Основні напрямки досліджень за останні десятиріччя

Історія розвитку аналітики текстів та неструктурованих даних тісно пов’язана з розвитком інформаційних технологій та комп’ютерних наук.

З появою перших комп’ютерів в середині XX століття увагу вчених привернула проблема аналізу та обробки текстів, але на той період акцент робився на обробці структурованих даних, таких як числа та таблиці. Розвиток NLP (обробка природної мови) у 50-х роках XX століття став основою для аналізу текстів, завдяки дослідженням методів обробки та аналізу природної мови комп’ютерами.

З розвитком ML (машинне навчання) та алгоритмів обробки текстів стали можливі більш точні та автоматизовані методи аналізу текстів. З появою та розповсюдженням Інтернету та поширенню цифрових текстів (веб-сторінки, електронні листи, соціальні мережі тощо) виникла ще більша потреба в ефективних методах та інструментах для аналізу цієї неструктурованої інформації.

В останні десятиліття розвиток глибинного навчання нейронних мереж призвів до значного прориву в аналізі текстів та неструктурованих даних.

Основні напрямки досліджень в області аналізу та обробки неструктурованих даних за останні десятиріччя:

- видобування інформації (Information Extraction): дослідження спрямовані на розробку методів та алгоритмів для автоматичного видобування структурованої інформації з невпорядкованих текстових джерел, таких як новини, наукові статті, соціальні мережі, соціальні медіа тощо. Це включає виявлення та екстрадицію іменованих сутностей, відносин між ними, подій, фактів, фейків тощо.

- класифікація та кластеризація текстів: дослідження в цій області спрямовані на розробку методів та моделей, які можуть автоматично класифікувати текстові документи за заданими категоріями або групувати їх у кластери на основі схожості, наприклад для автоматичної класифікації електронних листів, новин, статей, повідомлень у соціальних мережах.

- сентимент-аналіз: цей напрямок досліджень зосереджений на розробці методів для виявлення, аналізу та інтерпретації сентименту (тобто емоційного настрою) в текстових документах. Використовується для виявлення позитивного, негативного або нейтрального настрою відгуку користувачів, соціальних медіа повідомлень, відгуків на товари тощо.

- автоматичний переклад: дослідження в галузі машинного перекладу спрямовані на розробку моделей та алгоритмів для автоматичного перекладу

текстів з однієї мови на іншу за допомогою правил, статистичних методів або нейромереж;

- автоматична обробка природної мови (Natural Language Processing, NLP): дослідження в цій області спрямовані на розробку методів та моделей, які розуміють та обробляють природну мову людей. Це включає розпізнавання та розуміння мовних структур, синтаксичний та семантичний аналіз, генерацію тексту, виявлення спаму, побудову діалогових систем тощо;

- глибинне навчання в обробці текстів: цей напрямок досліджень зосереджений на використанні глибинного навчання (deep learning) для розв’язання завдань обробки текстів. Методи глибинного навчання, зокрема рекурентні та трансформерні нейронні мережі, дозволяють покращити результати у багатьох задачах, включаючи машинний переклад, сентимент-аналіз, відповіді на запитання тощо.

Найбільш відомі компанії, які проводили дослідження в області аналізу та обробки неструктурованих текстових даних:

Google – розробка інтелектуальних систем для аналізу текстів та обробки природної мови. Використовують алгоритми машинного навчання та глибинного навчання.

Facebook – використовують свої алгоритми машинного навчання для виявлення контексту, змісту текстових повідомлень.

IBM Watsons – аналіз тональності, кластеризація текстів, аналіз емоцій в соціальних мережах. Вони розробляють системи на основі штучного інтелекту, які допомагають розуміти та аналізувати текстові дані.

Sentiment Analysis – спеціалізуються на аналізі тональності та семантики текстових повідомлень. Пропонують інструменти та API для аналізу текстів в соціальних мережах.

Palantir Technologies – американська компанія, один з напрямків

діяльності - розробка технологій для аналізу текстів в соціальних мережах, розробка програмного забезпечення для аналізу великих об’ємів даних. Один з продуктів компанії – платформа Palantir Gotham, яка розроблена для роботи з різними типами даних, в тому числі текстових повідомлень, включаючи соціальні мережі. Дослідження компанії: розпізнавання намірів і зацікавленості, виявлення фейків та дезінформації, виявлення тематик, семантичний аналіз.

За останні роки було проведено багато досліджень по аналізу текстових повідомлень в соціальних мережах.

Аналіз сентименту в соціальних медіа повідомленнях. Дослідження зосереджені на виявленні та аналізі емоційного відтінку повідомлень в соціальних мережах. Наприклад, дослідження 2017 року «Topic Modeling of Twitter Data: A Comparison of Sentiment Analysis Techniques» порівнює різні методи аналізу тональності та тематичного моделювання для аналізу даних з Twitter [24].

Виявлення фейкових новин. Дослідження спрямовані на розробку методів для автоматичного виявлення фейкових або маніпулятивних новин у соціальних мережах. Це включає використання методів машинного навчання та аналізу тексту для виявлення характерних ознак дезінформації, маніпуляції або недостовірності. Наприклад, дослідження 2019 року «Fake News Detection on Social Media: A Data Mining Perspective» – за допомогою методів машинного навчання та аналізу даних, автори пропонують модель, розроблену на текстових ознаках та соціальній мережевій структурі для ідентифікації фейкових новин [25].

Виявлення тематик та трендів. Дослідження в цій області спрямовані на розробку методів для автоматичного виявлення тематик та трендів, які обговорюються в соціальних мережах. Це може включати виявлення головних тематик у хештегах, класифікацію повідомлень за категоріями, виявлення популярних подій або тематичних групувань.

Аналіз впливових користувачів. Дослідження спрямовані на визначення впливових користувачів у соціальних мережах на основі їхньої активності, взаємодії та контенту, який вони створюють.

Ці напрямки досліджень в області аналізу та обробки текстів постійно розвиваються і вдосконалюються за допомогою нових методів та підходів, включаючи використання тематичного моделювання, глибинного навчання, множинних джерел даних, розширених моделей мови тощо.

1.3 Аналіз існуючих підходів та методів обробки тексту

В даному дослідженні буде проаналізовано інформацію соціальних мереж на прикладі соціальної мережі Telegram.

Інформація, яка розміщується в Telegram, належить до неструктурованого типу текстових даних. Неструктуровані дані – це дані, довільні за формою, які не відповідають заздалегідь визначеній моделі даних, включають тексти, графіку, мультимедіа тощо.

Для роботи з неструктурованими даними, пошуку закономірностей з метою інтерпретації найчастіше використовують такі технології:

Data mining – інтелектуальний аналіз даних;

Natural Language Processing (NLP) – обробка природної мови;

Text mining (IAT) - інтелектуальний аналіз тексту;

Topic modeling - тематичне моделювання.

1.3.1 Text mining (IAT) - інтелектуальний аналіз тексту

Інтелектуальний аналіз тексту — це технологія штучного інтелекту (AI), яка використовує обробку природної мови (NLP) для перетворення неструктурованого тексту у нормалізовані структуровані дані, придатні для аналізу або для використання алгоритмами машинного навчання (ML).

Технологія Text Mining використовується для аналізу великих та надвеликих масивів неструктурованої інформації, що охоплює нові методи для

виконання семантичного аналізу текстів, інформаційного пошуку та управління.

Text Mining визначає факти, зв'язки та твердження, які могли би залишитись прихованими в масі великих текстових даних.

Після вилучення ця інформація перетворюється на структуровану форму, яку можна додатково проаналізувати або представити безпосередньо за допомогою кластеризованих HTML-таблиць, розумових карт, діаграм тощо. Інтелектуальний аналіз тексту використовує різноманітні методології для обробки тексту, найважливіша з яких це обробка природної мови (NLP) [1].

Програми, які реалізують ці завдання, мають деяким чином оперувати природною людською мовою і при цьому розуміти семантику тексту, що аналізується.

Ключовими завданнями Text Mining (ІАТ) є: категоризація текстів, пошук інформації, обробка змін в колекціях текстів, а також розробка засобів представлення інформації для користувача.

Категоризація документів полягає у зіставленні документів з колекції з однією або кількома групами (класами, кластерами) схожих між собою текстів (наприклад, за темою або стилем). Категоризація може відбуватися як за участю людини, так і без неї.

У першому випадку (*класифікація документів*), система ІАТ повинна віднести тексти до вже визначених (зручних для неї) класів. Для цього необхідно провести навчання з учителем, для чого користувач повинен надати системі ІАТ як перелік класів, так і зразки документів, що належать цим класам.

Другий випадок категоризації називається *кластеризацією_документів*. При цьому система ІАТ повинна сама визначити множину кластерів, за якими можуть бути розподілені тексти, — в машинному навчанні відповідне завдання називається навчанням без вчителя. У цьому випадку користувач повинен повідомити системі ІАТ кількість кластерів, на яке йому хотілося б розбити

оброблювану колекцію (передбачається, що в алгоритм програми вже закладена процедура вибору ознак) [4].

Основні етапи інтелектуального аналізу текстів наведено на рис 1.1.

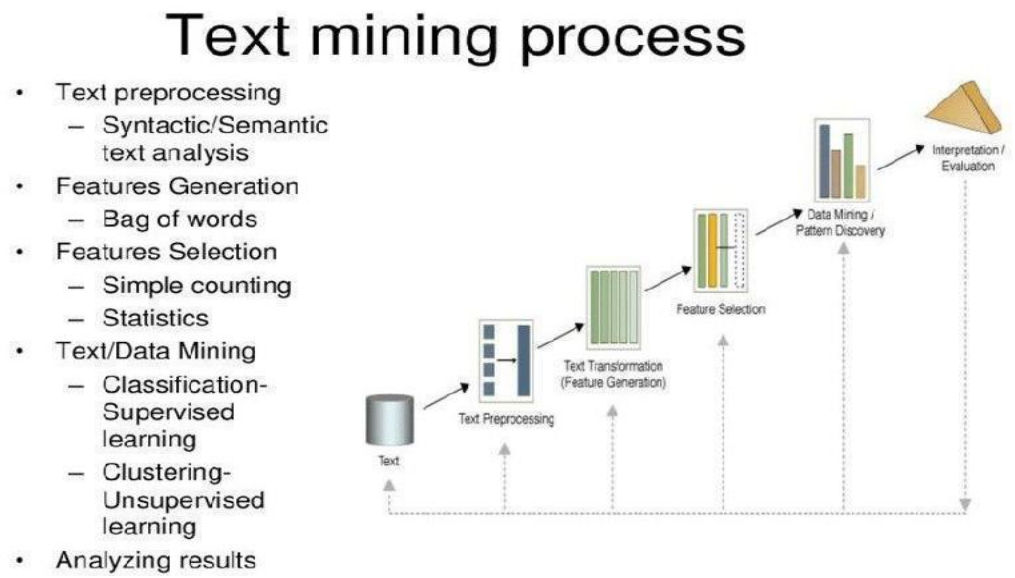

Рис 1.1 Етапи інтелектуального аналізу тексту

1.3.2 Natural Language Processing (NLP) – обробка природної мови

NLP - це один з напрямів штучного інтелекту, що займається проблемами комп'ютерного аналізу та синтезу природної мови. Використання технологій NLP та різноманітність систем, що використовують ці технології постійно зростають. Розглянемо лише ті з них, що містять у собі кібербезпековий компонент. Нижче на рис. 1.2 показано деякі з них. Виділимо три різні типи використання технологій NLP, а саме:

- програми, що використовують автоматизований збір та генерацію інформації, які можуть використовуватися для атаки або захисту людської сторони взаємодії людини та машини;
- додатки, у яких NLP може використовуватися зловмисниками для посилення атак на машини, відповідно, захисниками для посилення захисту своїх машин;

- програми, що підтримують автоматизацію деяких основних операцій із забезпечення безпеки та забезпечення відповідності вимогам [5].

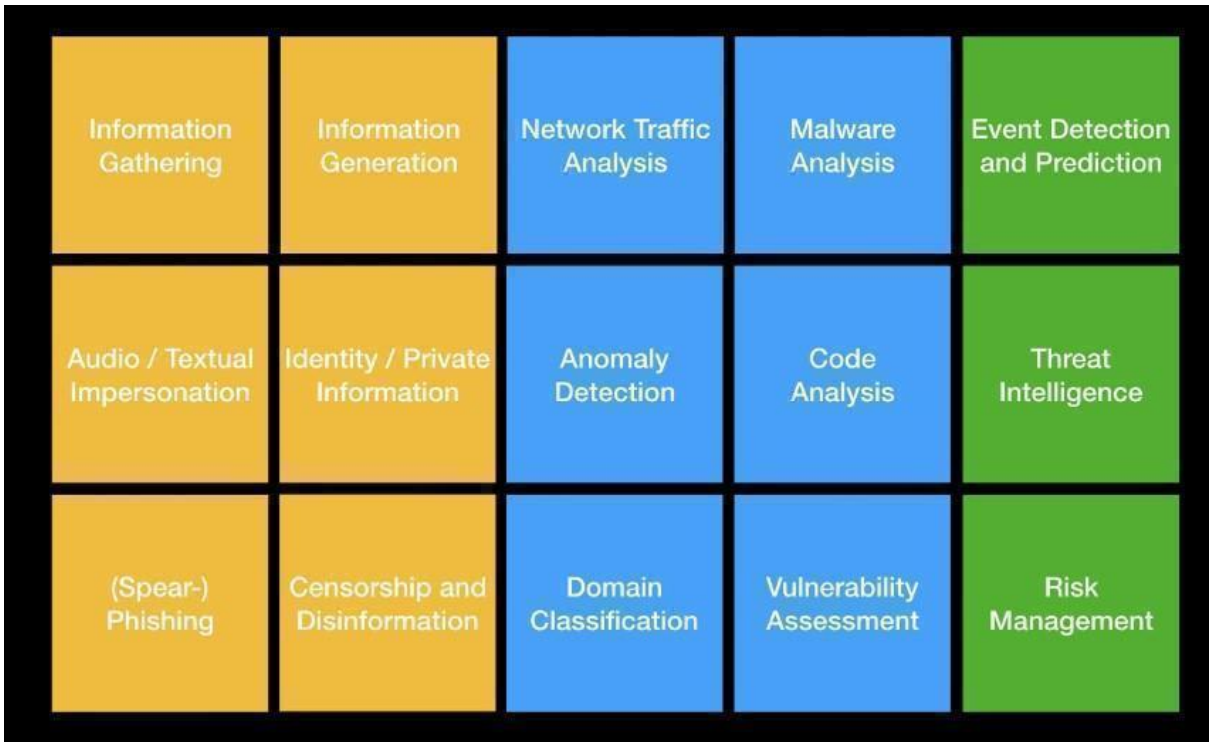

Рис 1.2 NLP у системах кібербезпеки

NLP імітує здатність людського мозку обробляти природні мови, такі як українська, англійська, іспанська тощо. NLP може робити висновки про значення текстових даних у контексті, навіть якщо документи не відповідають стандартному шаблону. Це робиться на основі семантики та граматичних відносин.

1.3.3 Тематичне моделювання

Тематичне моделювання — це інструмент аналізу тексту, який допомагає розкривати теми в тексті та знаходити групи слів, які стосуються різних тем. Конкретні слова, що часто зустрічаються в документі, вказують на теми та значення, що містяться в ньому. Це спосіб побудови моделі колекцій текстів, яка визначає до якої теми належить текст.

Аналіз теми (Topic analysis, виявлення теми, моделювання теми, виділення теми) — це техніка машинного навчання, яка організовує та розуміє великі колекції текстових даних шляхом призначення «тегів» або категорій відповідно до окремої теми чи теми кожного тексту [8].

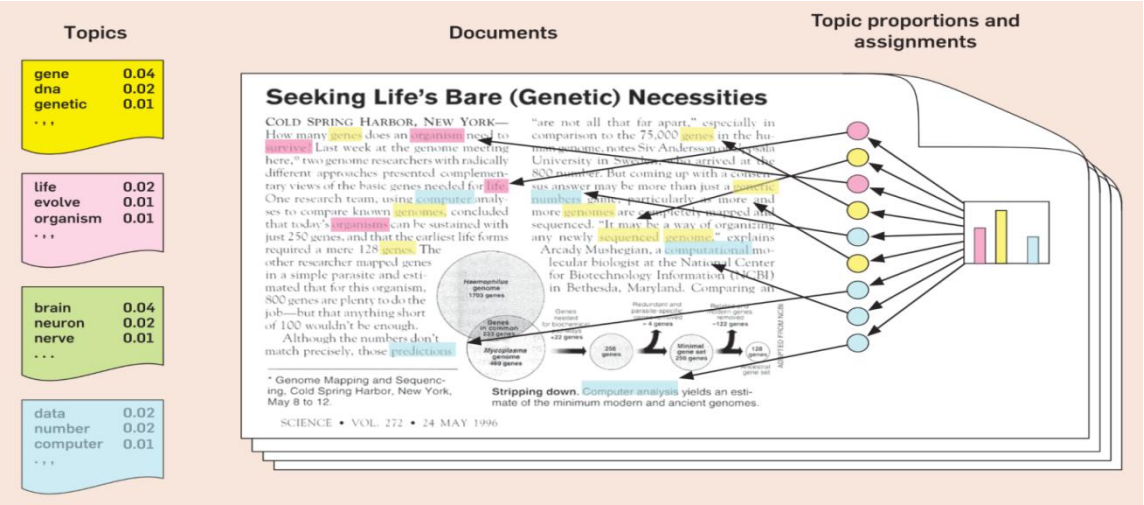

Рис 1.3 Візуалізація процесу тематичного моделювання

Двома найпоширенішими підходами до аналізу тем за допомогою машинного навчання є моделювання тем NLP і класифікація тем NLP:

- Моделювання теми (*NLP topic modeling*) — це техніка машинного навчання без учителя. Це означає, що він може виводити шаблони та кластеризувати подібні вирази без необхідності визначати теги тем чи тренувати дані заздалегідь. Цей тип алгоритму можна застосувати швидко та легко, але є недолік – вони досить неточні.
- Для класифікації тексту (*NLP topic classification*) або виділення теми з тексту необхідно знати теми тексту перед початком аналізу. тому що вам потрібно додати теги до даних, щоб навчити класифікатор тем. Хоча є додатковий крок, класифікатори тем окупаються в довгостроковій перспективі, і вони набагато точніші, ніж методи кластеризації.

1.4 Постановка задачі

Метою проекту є розробка програмного модулю аналізу та обробки текстів у соціальних мережах. Тому головне завдання дослідження полягає у тому, щоб спроектувати та програмно реалізувати модуль аналізу повідомлень у соціальних мережах, використовуючи існуючі методи та технології, враховуючи їх недоліки, що не дають розв'язати цю задачу в повному обсязі.

Функціональні вимоги (functional requirements) визначають функціональність ПЗ, яку розробники повинні побудувати, щоб користувачі змогли виконати свої завдання в рамках своїх індивідуальних вимог. Іноді вони називаються вимогами поведінки (behavioral requirements).

Функціональні вимоги програмного модулю:

1. Реєстрація користувачів, наявність різних ролей, аутентифікація та авторизація.
2. Можливість вибору телеграм каналів, задання часового періоду пошуку.
3. Програмний модуль повинен класифікувати повідомлення за даними темами.
4. Можливість перегляду та збереження результатів аналізу

Нефункціональні вимоги описують цілі та атрибути якості. Атрибути якості (quality attributes) - це додатковий опис функцій продукту, які виражені через опис його характеристик, важливих для користувачів або розробників.

Нефункціональні вимоги:

1. Програмний модуль повинен мати легкий у використанні інтерфейс.
2. Усі дії в програмі мають виконуватись швидко.
3. Об’єму тексту та повідомлень повинен бути достатнього розміру та містити достатню кількість різних текстових повідомлень (не менше ніж 1000 прикладів).

4. Наявність документації, інструкції користувача.
5. Ефективність і стійкість до збоїв.

На вхід програмного модулю подаються текстові повідомлення - пости з соціальної мережі Телеграм. В результаті отримуємо приналежність текстів до тієї чи іншої теми. Також в процесі проаналізовані пости зберігаються у базі даних, з метою унеможливлення впливу реагування та видалення постів авторами на роботу програмного модулю. Тобто створюється для розглянутих каналів так звана «історія».

Основні етапи роботи, які необхідно виконати для розробки програмного модулю аналізу текстових повідомлень:

1. огляд сучасного стану проблеми, дослідження наявних підходів і методів;
2. постановка задачі та вимог до програмного модулю;
3. аналіз предметної області, існуючих бізнес-процесів;
4. опис проектних рішень стосовно проектування бази даних, архітектури застосувань;
5. розробка нових та використання існуючих методів та технологій для розв’язання поставленої задачі;
6. розробка методики експериментальних досліджень, створення датасету;
7. розробка програмних засобів для аналізу текстових повідомлень, проведення експериментів для вибору кращих підходів для розв’язання задачі;
8. узагальнення та порівняння результатів теоретичних і експериментальних досліджень;
9. розробка програмного модуля на основі методів, що показали найкращі результати експериментів;
10. програмна реалізація модулю;

11. оцінка результатів роботи програмного модулю.

1.5 Висновки до першого розділу

У цьому розділі описано проблему та продемонстровано її актуальність. Описано об’єкт дослідження у визначеній предметній області. Розглянуто методи та технологій аналізу та обробки неструктурованих текстових даних.

Сформулювано постановку задачі та основні завдання роботи. Визначені функціональні та нефункціональні вимоги до реалізації програмного модулю.

РОЗДІЛ 2. Дослідження методів та технологій. Проектування програмного модулю аналізу тексту

2.1 Методи та технології розв’язання завдань кваліфікаційної роботи

2.1.1 Методи попередньої обробки даних

Токенізація

Токенізація розбиває необроблений текст на невеликі фрагменти. Аналіз тексту слід починати саме з цього. Перед обробкою природної мови нам потрібно ідентифікувати слова, складові рядка символів. Ось чому токенізація є самим основним кроком для роботи з текстовими даними. Це важливо, тому що зміст тексту можна легко інтерпретувати, аналізуючи слова, присутні в тексті [9].

Токенізація розбиває неструктурований текст на слова, речення, які називають токенами. Ці токени допомагають зрозуміти контекст, розробити модель NLP, інтерпретувати значення тексту, аналізуючи послідовність слів.

Токенізація (іноді – сегментація) за реченнями – це процес поділу тексту на речення-компоненти [10]. Ідея виглядає досить простою. В англійській та деяких інших мовах ми можемо виокремлювати реченнями кожного разу, коли знаходимо певний знак пунктуації – точку. Але навіть в англійській це завдання нетривіальне, тому що точка використовується і в скороченнях. Таблиця скорочень може допомогти під час обробки тексту, щоб уникнути неправильного задання меж речення. У більшості випадків для цього використовуються бібліотеки, наприклад пакет NLTK [14].

Токенізація за словами – це процес поділу речень на слова-компоненти.

Очищення від стоп слів

Слова, які зазвичай відфільтровуються перед обробкою природної мови, називаються стоп-словами. Насправді це найпоширеніші слова у будь-якій мові (такі як артиклі, прийменники, займенники, спілки тощо), і вони не додають

багато інформації до тексту. Приклади кількох стоп-слів в англійській мові: "the", "a", "an", "so", "what".

Стоп-слів багато в будь-якій людській мові. Видаляючи ці слова, ми видаляємо низькорівневу інформацію з нашого тексту, щоб приділити більше уваги важливій інформації. Можна сказати, що видалення таких слів не тягне за собою жодних негативних наслідків для моделі, яку ми навчаємо для нашого завдання.

Видалення стоп-слів безперечно зменшує розмір набору даних і, отже, скорочує час навчання через меншу кількість токенів, що беруть участь у навчанні [11].

Ми не завжди видаляємо стоп-слова. Видалення стоп-слів сильно залежить від завдання, яке ми виконуємо, та мети, яку ми хочемо досягти. Наприклад, якщо ми навчаємо модель, яка може виконувати завдання аналізу настроїв, ми можемо не видаляти стоп-слова.

NLTK має встановлений список стоп-слів (словник). Перші 100 слів з цього списку для російської мови можна побачити на рис 2.1. Цей список можна доповнювати та розширювати залежно від контексту задачі.

```
['и', 'в', 'во', 'не', 'что', 'он', 'на', 'я', 'с', 'со', 'как', 'а', 'то', 'все', 'она', 'так', 'его', 'но', 'да',
'ты', 'к', 'у', 'же', 'вы', 'за', 'бы', 'по', 'только', 'ее', 'мне', 'было', 'вот', 'от', 'меня', 'еще', 'нет', 'о',
'из', 'ему', 'теперь', 'когда', 'даже', 'ну', 'вдруг', 'ли', 'если', 'уже', 'или', 'ни', 'быть', 'был', 'него', 'до',
'вас', 'нибудь', 'опять', 'уж', 'вам', 'ведь', 'там', 'потом', 'себя', 'ничего', 'ей', 'может', 'они', 'тут', 'где',
'есть', 'надо', 'ней', 'для', 'мы', 'тебя', 'их', 'чем', 'была', 'сам', 'чтоб', 'без', 'будто', 'чего', 'раз',
'тоже', 'себе', 'под', 'будет', 'ж', 'тогда', 'кто', 'этот', 'того', 'потому', 'этого', 'какой', 'совсем', 'ним',
'здесь', 'этом', 'один']
```

Рис 2.1 Список Стоп-слів для російської мови

Лематизація і стемінг

Зазвичай тексти містять різні граматичні форми одного і того ж слова, а також можуть зустрічатися однокореневі слова. Лематизація і стемінг мають на меті привести всі словоформи до однієї, нормальної словникової форми.

Лематизація і стемінг – це окремі випадки нормалізації, які відрізняються між собою.

Стемінг – це процес, який відрізає «зайве» від кореня слів, що часто призводить до втрати словотворчих суфіксів. Тобто стемінг – це усунення придатків до кореня, тобто відділення суфікса, приставки, закінчення.

Лематизація – це процес, який використовує словник і морфологічний аналіз, щоб в результаті привести слово до його канонічної форми – леми.

Відмінність в тому, що стемінг працює без знання контексту і, відповідно, не розуміє різницю між словами, які мають різний зміст залежно від частини мови. На рис 2.2 зображено приклад виконання лематизації та стемінгу для англійської мови.

Word	Stemming	Lemmatization
information	inform	information
informative	inform	informative
computers	comput	computer
feet	feet	foot

Рис 2.2 Порівняння стемінгу та лематизації для англійської мови

Embedding

Це підхід для представлення слів і документів. Word Embedding або Word Vector — це цифрове векторне відображення, яке представляє слово в просторі нижчої вимірності. Це дозволяє словам зі схожим значенням мати подібне представлення.

Два основні різні підходи до ембедінгу слів:

1. GloVe: Це ще один метод створення word embeddings. У цьому методі ми беремо корпус і повторюємо його та отримуємо спільне входження кожного слова з іншими словами в корпусі. Через це ми отримуємо матрицю спільного входження. Слова, які зустрічаються поруч одне з одним, отримують значення 1, якщо вони розташовані на відстані одного слова, то 1/2, якщо на відстані два слова, то 1/3 і так далі.

2. Word to Vec: У Word2Vec кожному слову присвоюється вектор. Ми починаємо або з випадкового вектора, або з одного гарячого вектора (one-hot). One-Hot вектор: представлення, де лише один біт у векторі дорівнює одиниці. Якщо в корпусі є 500 слів, то довжина вектора буде 500. Після призначення векторів кожному слову ми беремо розмір вікна та повторюємо весь корпус. Під час такої обробки, використовуються два методи Continuous Bowl of Words (CBOW) та Skip Gram. Skip-gram - один з методів машинного навчання без вчителя, що використовується для пошуку найбільш споріднених слів для даного слова. Skip-gram [12] використовується для передбачення контекстного слова для даного цільового слова. Тут цільове слово подається на вхід, тоді як контекстні слова є результатом.

Переваги методу:

1. Навчання без учителя, отже може працювати над будь-яким поданим необробленим текстом.
2. Вимагає менше пам’яті порівняно з іншими словами для векторних представлень.

2.1.2 Методи тематичного моделювання

Одним з основних способів отримання знань з текстових колекцій є тематичне моделювання — спосіб побудови моделі колекції текстових документів, яка визначає, до яких тем відноситься кожен із документів. Тематичне моделювання вирішує класичне завдання аналізу текстів – як створити імовірнісну модель великої колекції текстів, яку можна буде використовувати, наприклад, для інформаційного пошуку (information retrieval), класифікації чи, як в нашому випадку, для розмітки контенту [6,13].

Усі тематичні моделі базуються на однаковому базовому припущенні: кожен документ (в нашому випадку - пост) складається з суміші тем і кожна тема складається з набору слів. Тобто маємо наступні припущення:

- порядок документів у колекції не має значення;

- порядок слів у документі не має значення, документ – мішок слів;
- слова, що зустрічаються часто в більшості документів, не важливі для визначення тематики.

Прихований семантичний аналіз (LSA) є одним з основоположних методів тематичного моделювання. Основна ідея полягає в тому, щоб взяти матрицю того, що ми маємо — документів і термінів — і розкласти її на окремі матриці документ-тема та матриця тема-термін [11].

Базовими тематичними моделями є:

- Імовірнісний прихований семантичний аналіз (probabilistic latent semantic analysis, PLSA);
- Модель прихованого розміщення Діріхле (latent Dirichlet allocation, LDA) [6,13].

PLSA - це статистична модель аналізу автоматизованої індексації документів. Графічна ймовірнісна модель зображена на рис 2.3. Дана модель є подальшим розвитком **прихованого-семантичного аналізу (LSA)** та заснована на запровадженні прихованих змінних - тематик текстових документів. Хоча ця модель і вважається поліпшенням латентно-семантичного аналізу, все ж таки вона має суттєві недоліки. Для PLSA характерне перенавчання, а також неоднозначність результатів, пов’язані з великою кількістю імовірнісних параметрів, на які не накладаються обмеження регуляризації. Також модель не виокремлює нетематичні слова. Більшість недоліків може бути усунена застосуванням семплування, регуляризації та розрідження, що призводить до створення великого сімейства алгоритмів з урахуванням PLSA.

Імовірнісна модель появи пари «документ-слово» може бути записана трьома еквівалентними способами:

$$p(d,w) = \sum_{t \in T} p(t)p(w|t)p(d|t) = \sum_{t \in T} p(d)p(w|t)p(t|d) = \sum_{t \in T} p(w)p(t|w)p(d|t)$$

де T — множина тем;

$p(t)$ — невідомий апріорний розподіл тем у всій колекції;

$p(d)$ — апріорний розподіл на множині документів, емпірична оцінка;

$p(d) = n_d/n$, де $n = \sum_d n_d$ — сумарна довжина всіх документів;

$p(w)$ — апріорний розподіл на множині слів, емпірична оцінка

$p(w) = n_w/n$, де n_w — число входжень слова w в усі документи.

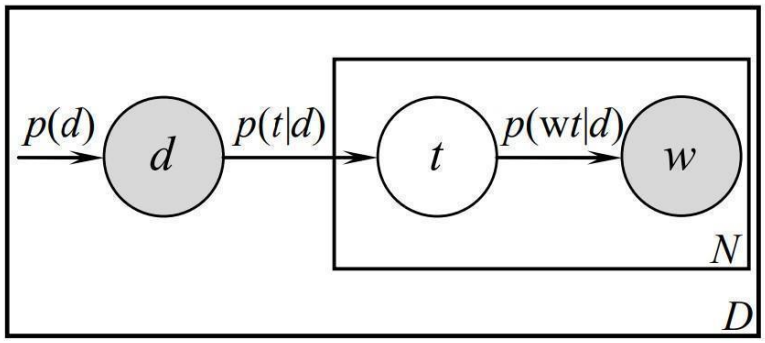

Рис 2.3 Графічна ймовірнісна модель PLSA: d - документ; w - слово; d, w - змінні, що спостерігаються; t - тема (прихована змінна); p(d) - апріорний розподіл на багатьох документах; p(w|t), p(t|d) - шукані умовні розподіли; D – колекція документів; N - довжина документа в словах

Модель прихованого розміщення Діріхле (latent Dirichlet allocation, LDA) — генеративна модель, в якій кожен документ розглядається як суміш різних тем (рис 2.4).

Ця модель схожа з **PLSA**, але відрізняється тим, що в **LDA** розподіл тем слідує розподілу Діріхле. Це дозволяє оцінювати ймовірності документів та термінів поза текстовою колекцією. Для ідентифікації параметрів моделі **LDA** за колекцією документів застосовується семплування за Гіббсом або варіаційний байєсівський висновок. Ця модель усуває основні недоліки **PLSA**, зокрема кількість параметрів не збільшується зі зростанням числа документів. Основний недолік латентного розподілу Діріхле - відсутність лінгвістичних

обґрунтувань, хоча існують її розширення, які усувають деякі її обмеження та підвищують продуктивність для конкретних завдань.

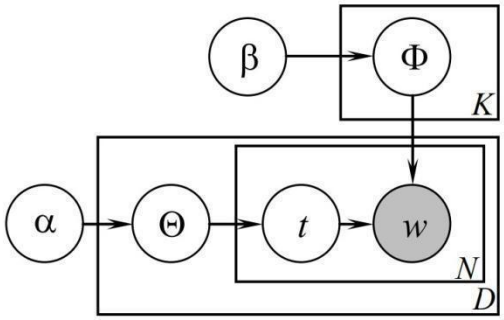

Рис 2.4 Графічна ймовірнісна модель LDA: w - слово (спостерігається змінна); t - тема (прихована змінна); D – колекція документів; N - Довжина документа в словах; K - кількість тем у колекції; Θ - розподіл тем у документі; Φ - розподіл слів у темі; α - апріорний розподіл Діріхле на параметри Θ, β - апріорний розподіл Діріхле на параметри Φ

2.2 Аналіз функцій та функціональне моделювання

2.2.1 Функціональний аналіз

Для отримання повної картини бізнес-процесів розробленого програмного модулю представимо їх у вигляді контекстної діаграми (рис 2.5).

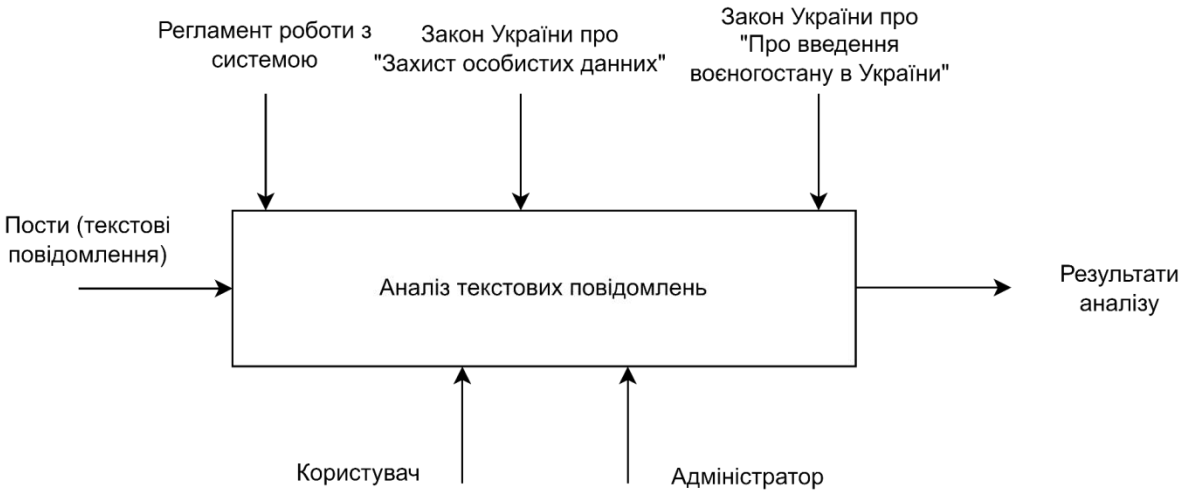

Рис 2.5 Контекстна діаграма програмного модулю

Результатом функціональної декомпозиції є розбиття функцій програмного модулю на елементарні процеси, що візуалізується

використовуючи ієрархічну модель, яка має назву дерева вузлів функціональної моделі (Node Tree Diagramming). На рис 2.6 продемонстровано результат такої декомпозиції, він буде використаний надалі для побудови функціональної моделі, в якості каркаса. Дана діаграма показує загальний склад моделі.

Головна функція програмного модулю - вершина дерева, поділяється на підфункції: клієнтські (робота з програмним модулем), програмного модуля (завантаження постів, аналіз, представлення результатів) та адміністративні (налаштування доступу користувачів).

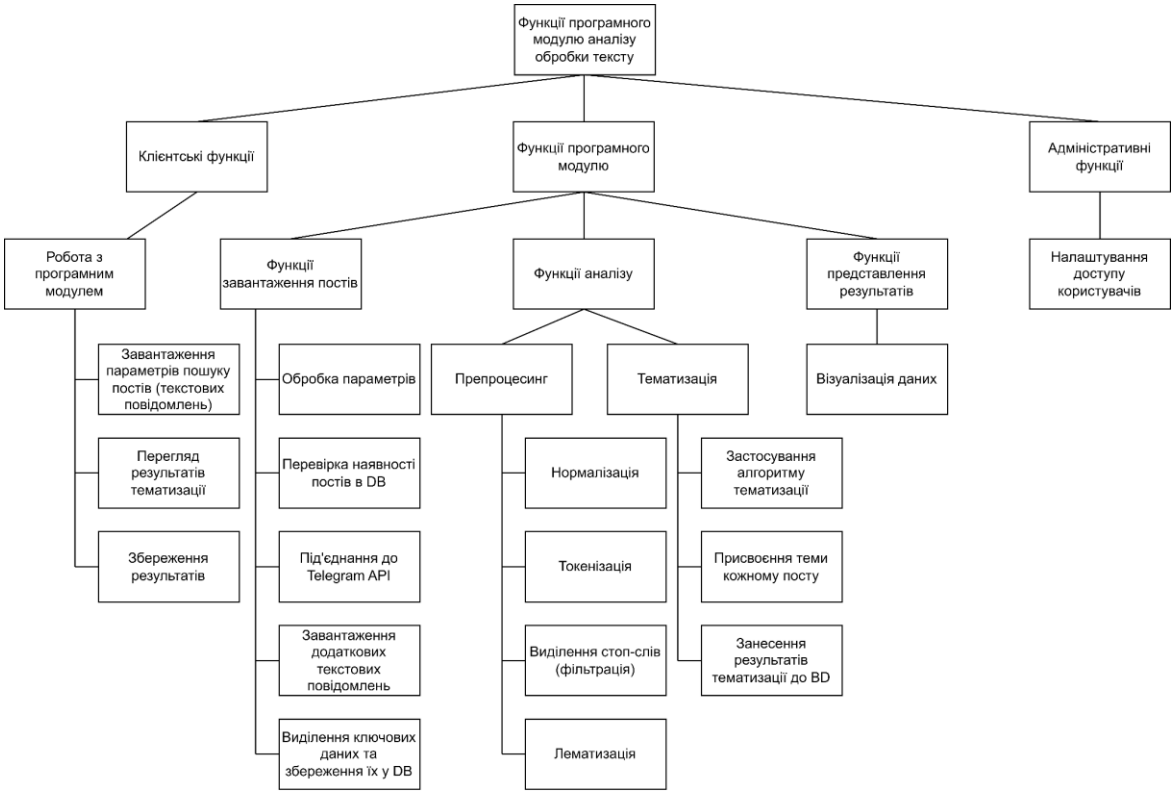

Рис 2.6 Дерево функцій програмного модулю

2.2.2 Побудова функціональної моделі програмного модулю

Сукупність діаграм IDEF0 складає функціональну модель. Функціональна модель є подальшою деталізацією (декомпозицією) контекстної діаграми (рис 2.7). Спочатку функція програмного модулю в цілому (ціль, призначення,

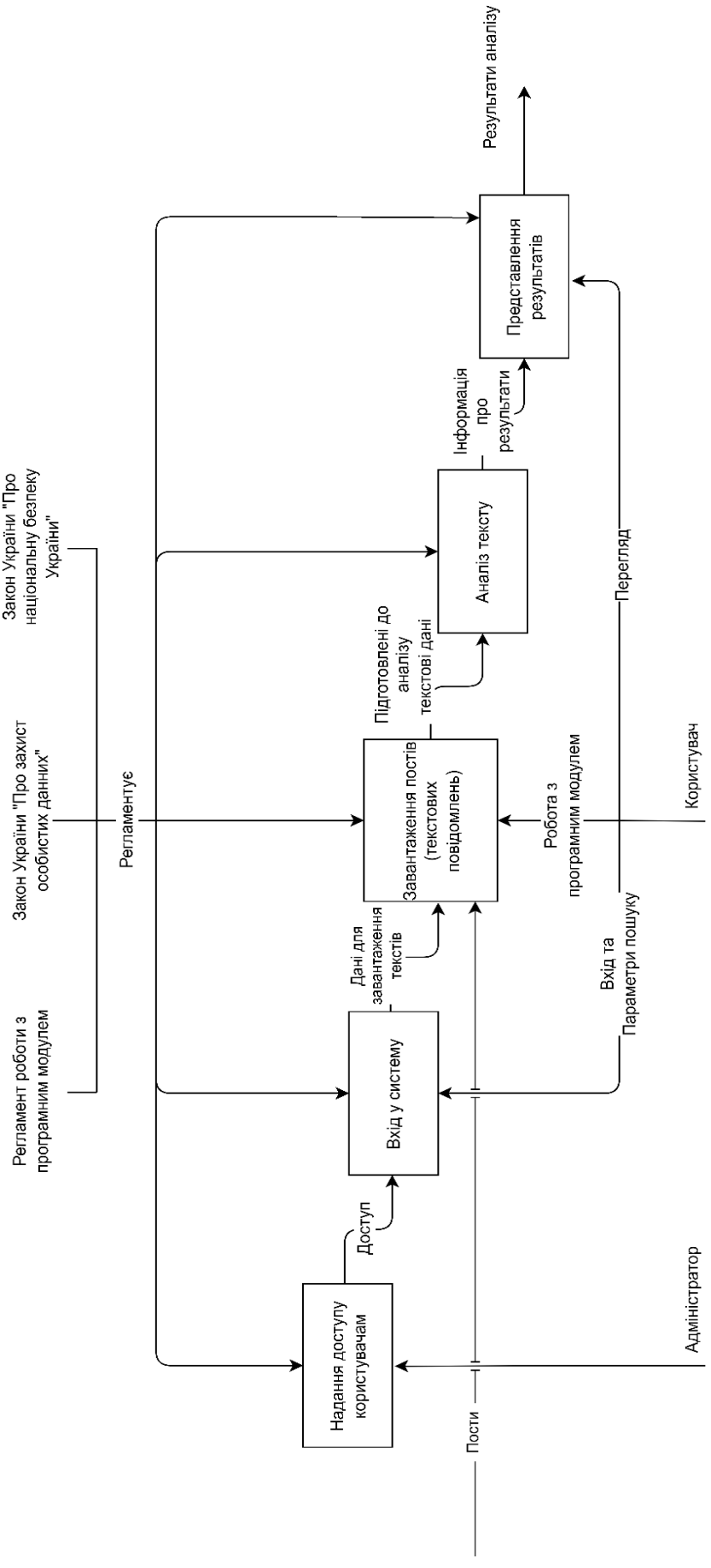

Рис 2.7 Декомпозиція контекстної діаграми «ЯК БУДЕ»

головне завдання) розбивається на декілька окремих функцій (завдань, робіт, цілей). Такі елементи називаються функціональними блоками та зображуються прямокутниками. Дуги зі стрілками демонструють зв’язки між блоками. [21]

Діаграма декомпозицій має у своєму складі 5 функціональних блоків. Першочерговим пріоритетним є виконання роботи «Надання доступу користувачам», яка виконується Адміністратором (рис 2.8). Далі Користувачі отримують доступ до «Входу систему».

Рис 2.8 Декомпозиція блоку «Вхід у систему». Діаграма «ЯК БУДЕ»

Користувач, який отримав доступ, може вводити параметри пошуку. Також після отримання результатів аналізу даний користувач зможе їх переглянути та зберегти, обравши метод збереження.

Після вводу Користувачем параметрів пошуку, ці параметри передаються до блоку «Завантаження постів». Розглянемо декомпозицію цього функціонального блоку (рис 2.9). Основні етапи Завантаження постів:

- обробка параметрів пошуку, які у блоці «Вхід у систему» задав системі користувач;

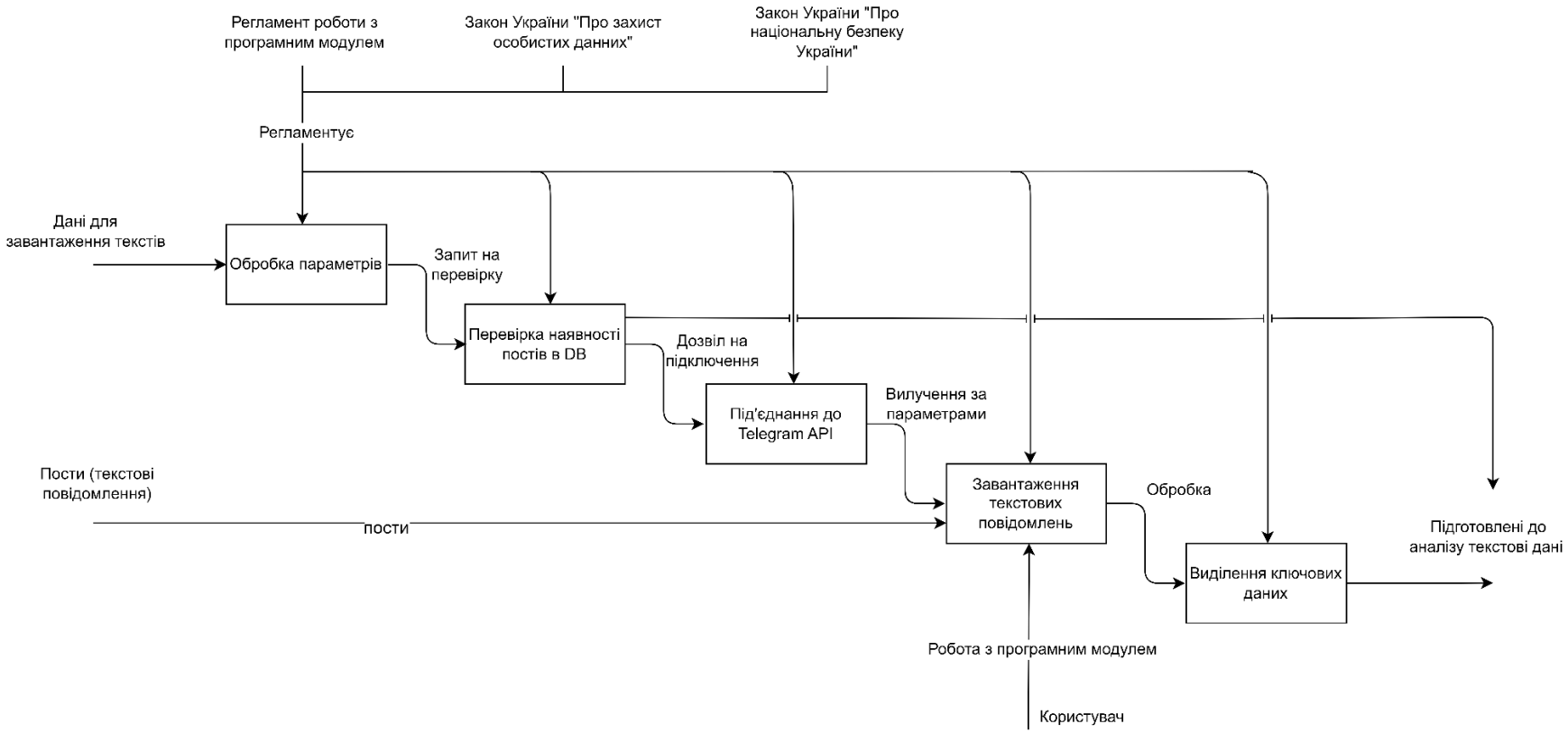

Рис 2.9 Декомпозиція процесу «Завантаження текстових повідомлень». Діаграма «ЯК БУДЕ»

- перевірка наявності таких постів у базі даних, якщо такі пости вже збережені - вони готові для аналізу;
- інакше відбувається під’єднання до Телеграм API та завантаження постів;
- далі пости обробляються - вилучаються коментарі, метадані, зображення тощо, оскільки нам необхідно зберігати лише дані про канал та безпосередній текст повідомлення та дату створення;
- відфільтровані текстові дані зберігаються у базі даних та передаються на етап аналізу.

Розглянемо блок аналіз тексту на основі якого буде розроблено окремий програмний модуль для аналізу тексту (рис 2.10). Розіб’ємо його на два підблоки, а саме на попередню обробку тексту та тематизацію.

Рис 2.10 Декомпозиція процесу «Аналіз тексту». Діаграма «ЯК БУДЕ»

Розглянемо ключові етапи препроцесингу (рис 2.11):

- нормалізація;
- токенізація;
- фільтрація ;
- лематизація.

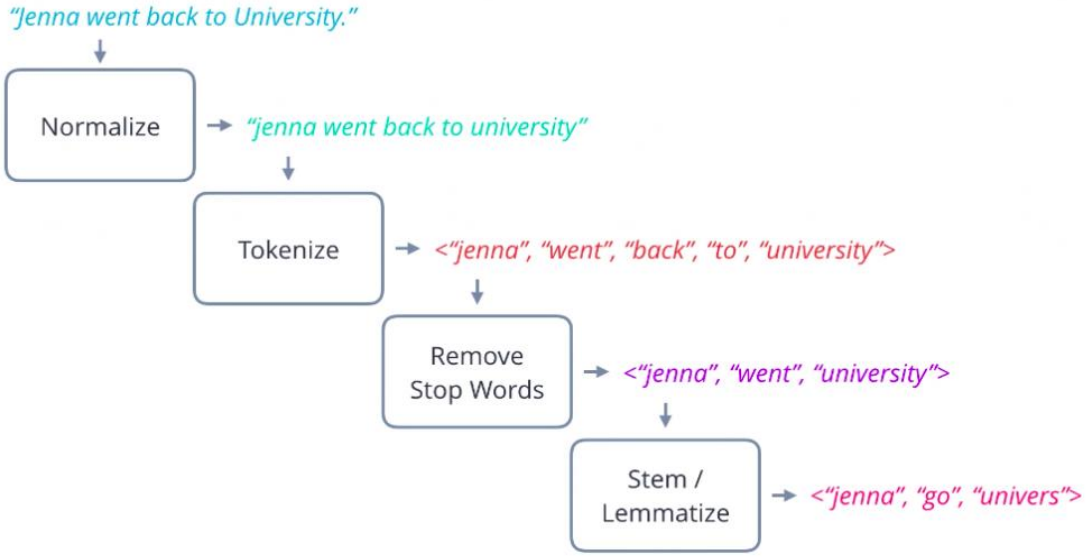

Рис 2.11 Схематичне зображення попередньої обробки тексту

Отримаємо наступну діаграму для цього блоку

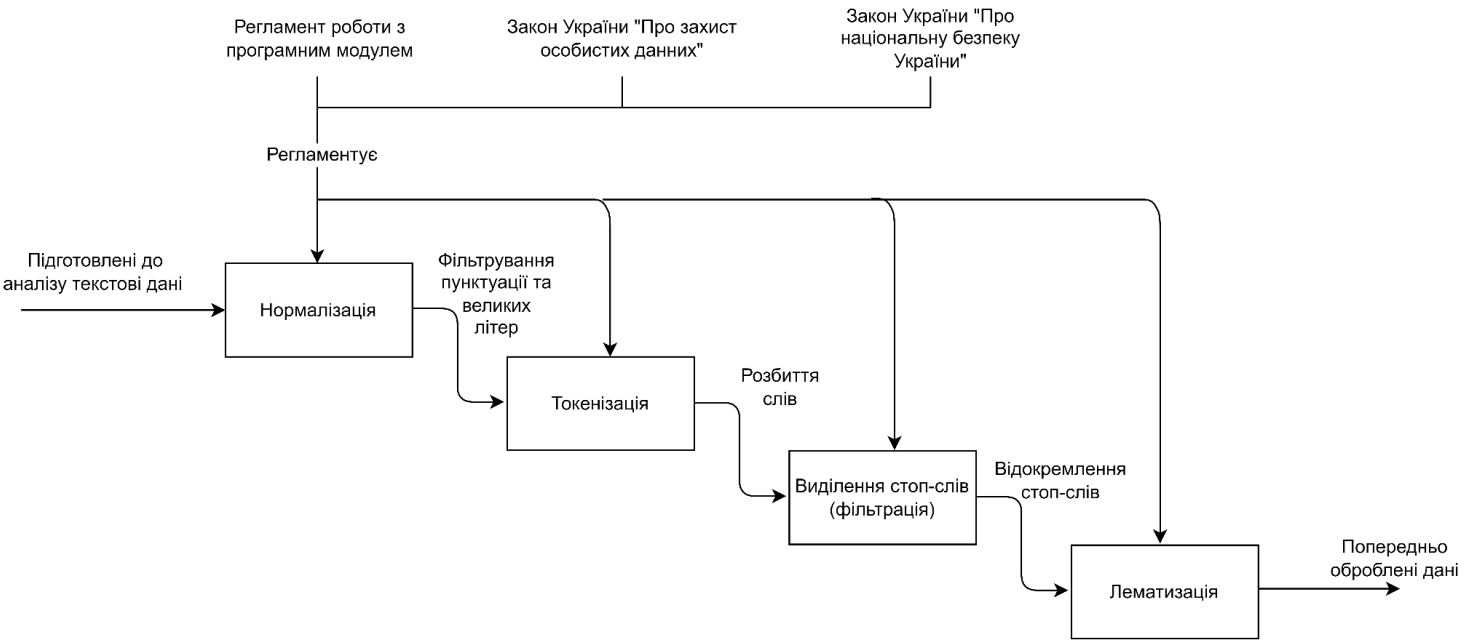

Рис 2.12 Декомпозиція процесу «Препроцесинг». Діаграма «ЯК БУДЕ»

Тематизація декомпозується на блоки: власне застосування алгоритму, присвоєння теми кожному посту, збереження результатів шляхом взаємодії з базою даних (рис 2.13). Метод, що ляже в основу алгоритму буде обрано емпіричним шляхом.

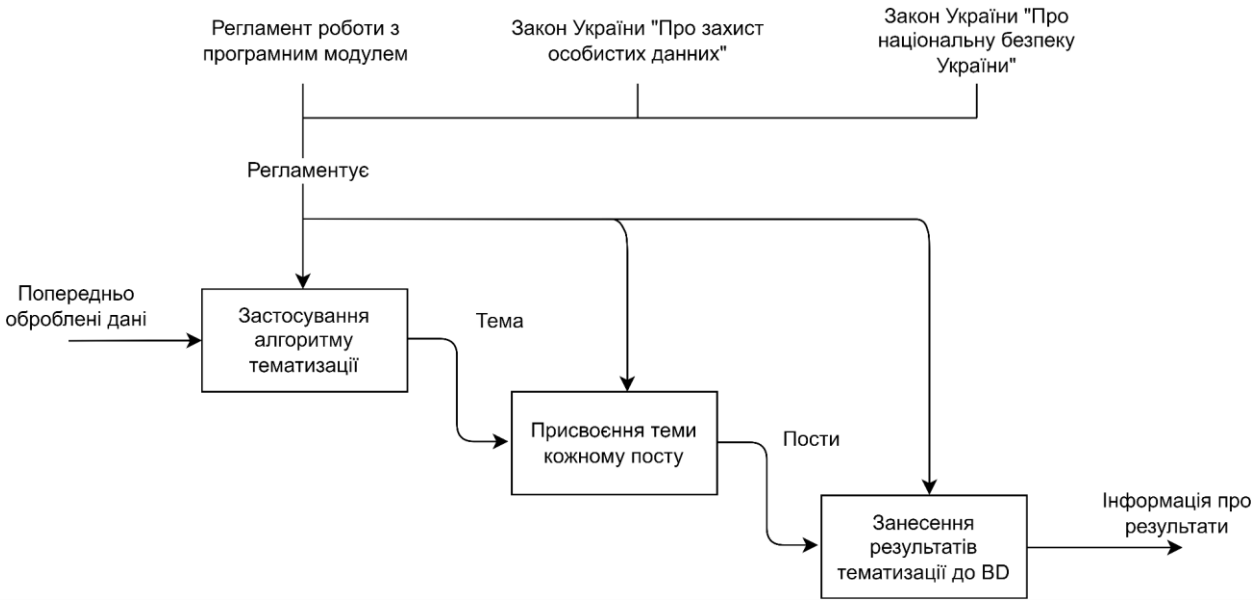

Рис 2.13 Декомпозиція процесу «Тематизація». Діаграма «ЯК БУДЕ»

2.3 Проектування програмного модулю аналізу та обробки тексту

2.3.1 Структура та архітектура програмного модулю

Наступним етапом є побудова структура розробленого програмного модулю (рис. 2.14). Дана схема відображає модель, структуру, виконувані функції й взаємозв’язок компонентів. Даний програмний модулю ділиться на модулі роботи з користувачами та адміністративний, відповідно до дерева функцій.

Рис 2.14 Структура програмного забезпечення

На структурному дереві програмного модулю знаходиться модуль користування модулем, який містить три модулі: клієнтський, адміністративний та модуль аналізу.

До клієнтського модуля входить модуль користування програмним модулем, а саме завантаження текстових повідомлень, перегляд результатів аналізу та збереження результатів.

До модуля аналізу постів входить основний функціонал програмного модулю. Він поділяється на три модулі: завантаження, аналіз та представлення результатів.

До адміністративного модуля входить модуль налаштування доступу користувачів.

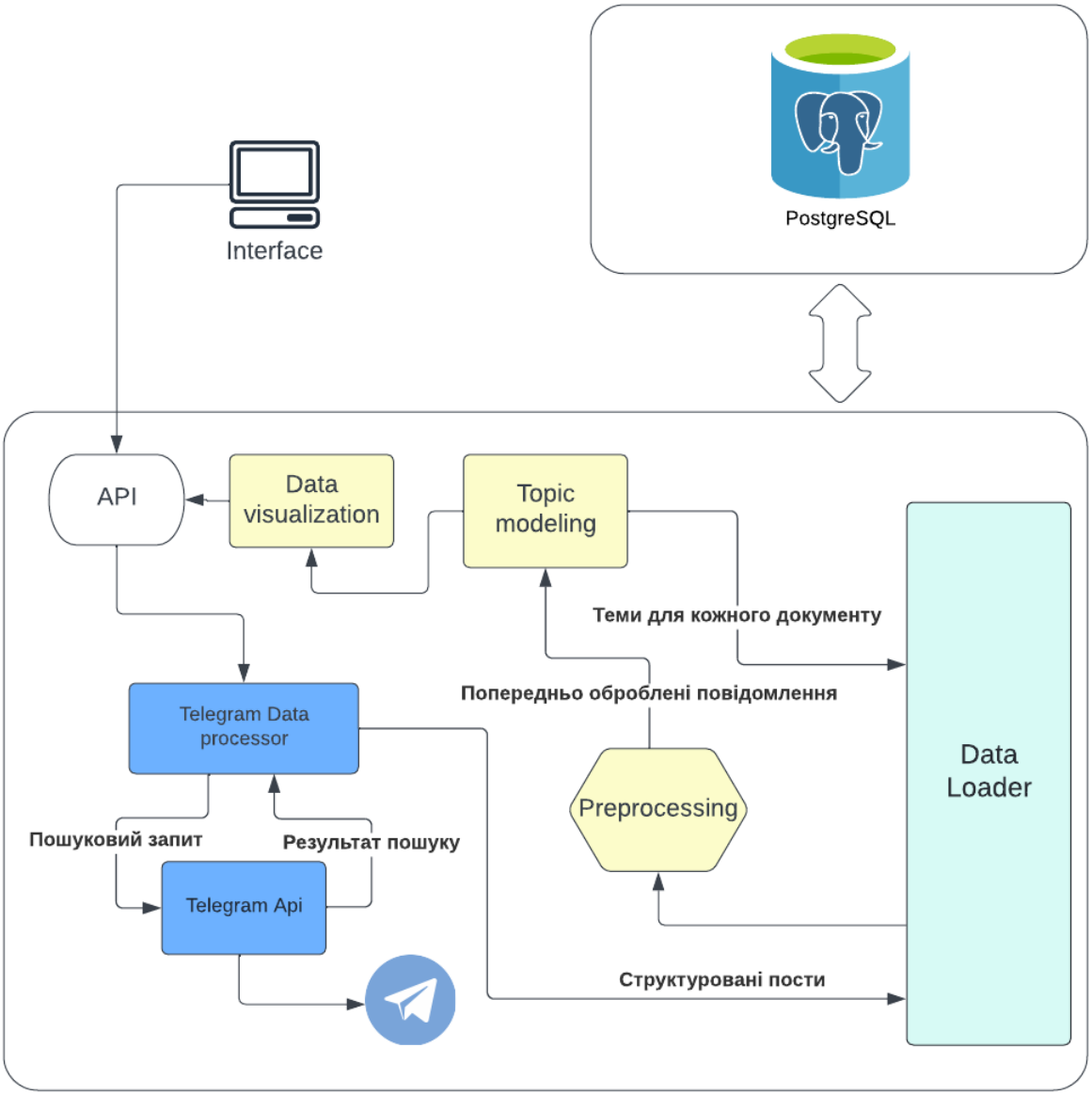

Рис 2.15 Архітектура програмного модулю

2.3.2 Проектування бази даних

Для збереження цих даних буде розроблена база даних, яка зберігатиме та видаватиме потрібну інформацію. На рис 2.16 маємо логічну ER - модель бази даних.

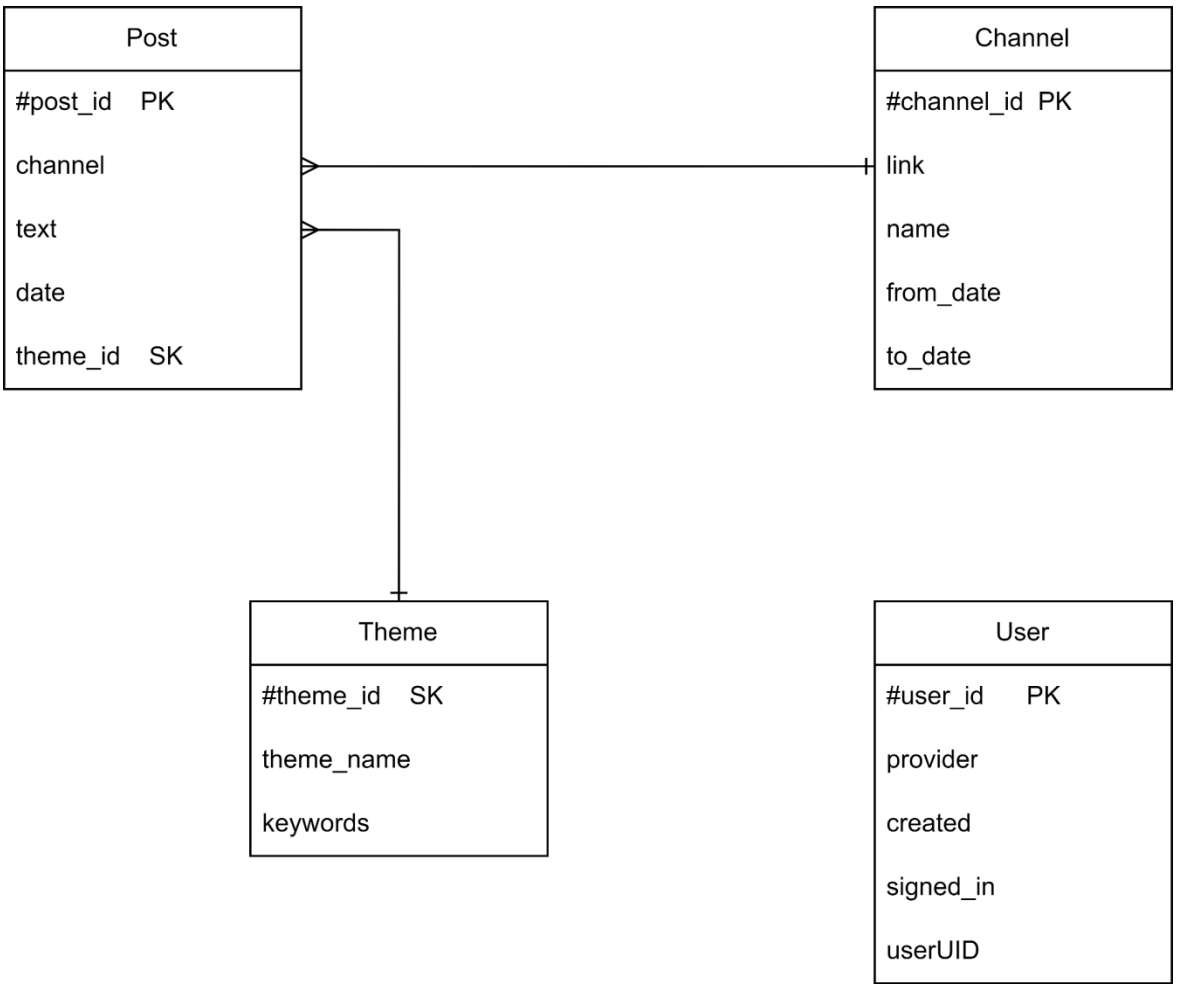

Рис 2.16 Концептуальна модель бази даних

Нехай маємо наступні сутності:

- Post (збереження текстових повідомлень - постів).
- Theme (теми).
- User (користувачі).
- Channel (телеграм - канали).

Далі детально розглянемо зв'язки між сутностями у табл. 2.1.

Таблиця 2.1

Зв'язки між сутностями

Пара	*Тип зв'язку*	*Пояснення*
Post - Theme	Один до багатьох	Один пост належить до однієї теми, але кожній темі належить багато постів
Post - Channel	Один до багатьох	Один пост має один канал в якому його розмістили, але в кожному каналі багато постів

Побудуємо фізичну модель бази даних рис 2.17.

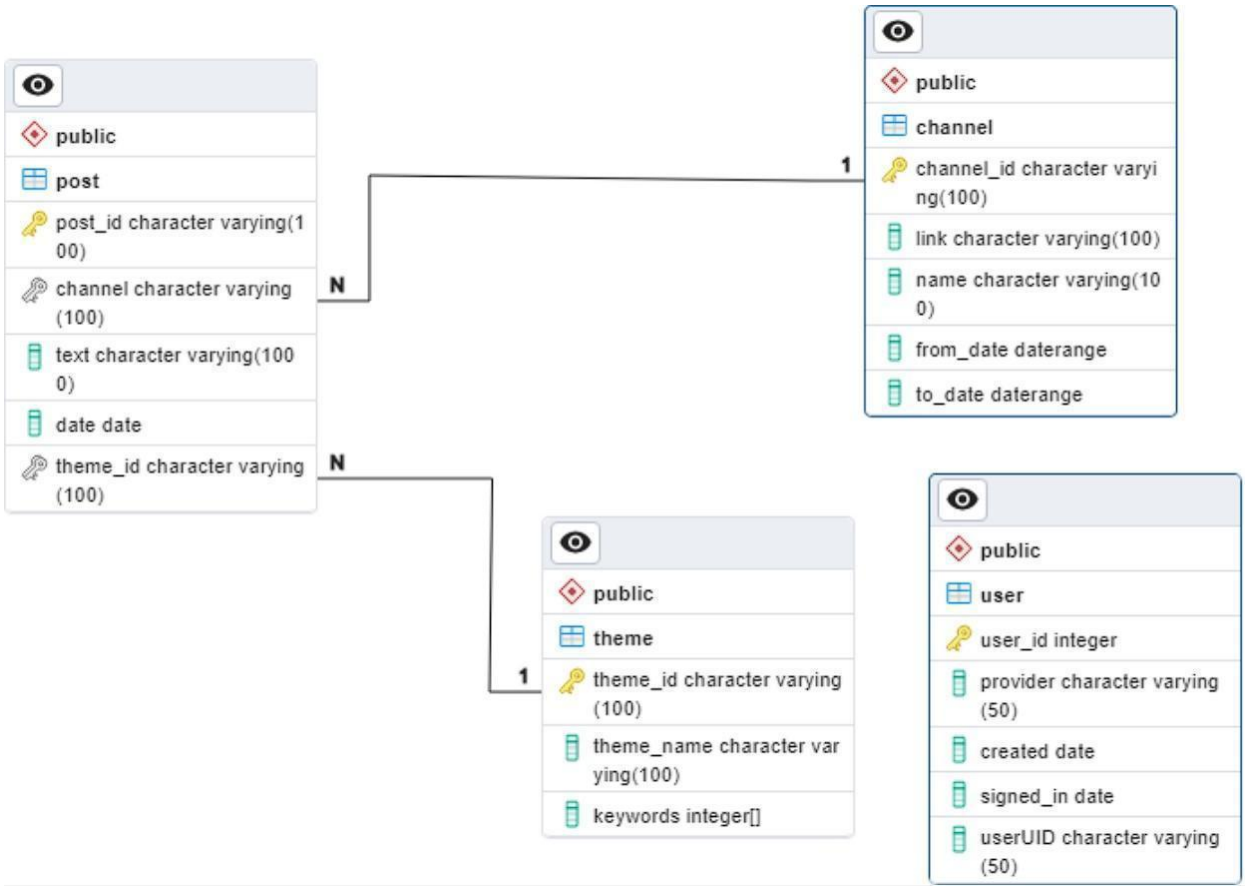

Рис 2.17 Фізична модель бази даних

У табл. 2.2 наведено структуру таблиць бази даних, а саме типи даних, види ключів та інформацію про використання полів. У табл. 2.3 наведено структуру таблиці User.

Таблиця 2.2

Структура таблиць бази даних

Елемент	Тип даних	Пояснення	Ключ
Post			
post_id	Числовий	Унікальний ідентифікатор поста	ПК*
channel_id	Числовий	Зовнішній ключ посилається на таблицю **Channel**	ЗК**
text	Символьний	Текст посту	
date	Дата	Дата публікації	
theme_id	Числовий	Зовнішній ключ посилається на таблицю **Theme**	ЗК
Theme			
theme_id	Числовий	Унікальний ідентифікатор теми	ПК
theme_name	Символьний	Назва теми	
keywords	Символьний	Хмара слів, що пов’язані з темою	
Channel			
channel_id	Числовий	Унікальний ідентифікатор каналу	ПК
channel_link	Символьний	Посилання на канал	
channel_name	Символьний	Назва каналу	
from_date	Дата	Період за який вже було	

Елемент	Тип даних	Пояснення	Ключ
to_date	Дата	проведено збереження постів	

(* первинний ключ, ** зовнішній ключ)

Таблиця 2.3

Структура таблиці User

User			
user_id	Числовий	Унікальний ідентифікатор каналу	ПК
provider	Символьний	Метод реєстрації	
created	Дата	Дата створення користувача в базі	
signed_in	Дата	Дата останнього входу	
userUID	Символьний	Хеш шифр користувача	

2.4 Висновки до другого розділу

Спроектовано програмний модуль аналізу та обробки тексту і текстових повідомлень. Проведено функціональний аналіз та побудовано дерево функцій, розроблено контекстну діаграму та декілька діаграм декомпозицій Таким чином маємо готову функціональну модель IDEF0.

Побудовано модульну архітектуру програмного модулю для відображення її внутрішніх компонентів та їх взаємозв’язок. Розроблена база даних з відповідною схемою.

РОЗДІЛ 3. Програмна реалізація, дослідження результатів

3.1 Засоби програмної реалізації

В якості IDE використовувалася платформа Visual Code, а для створення модулю аналізу тексту - Jupyter Notebook. Основна мова програмування при реалізації програмного модулю - Python. Python - це одна з найпопулярніших мов програмування в світі, яка має багато переваг, але є й свої недоліки.

Серед переваг Python можна відзначити його простоту та лаконічність, що робить його відмінним вибором для програмістів. Python також має багато різних функціональних бібліотек, які дозволяють швидко та зручно розв’язувати поставлені задачі, включаючи наукові обчислення, обробку даних, створення веб-додатків, роботу з неструктурованими даними, моделювання нейронних мереж тощо. Ще однією важливою перевагою є те, що Python є інтерпретованою мовою.

Опис бібліотек, що використовувались у розробленому програмному продукті:

NLTK (Natural Language Toolkit) – провідна платформа для створення програм NLP на Python [17]. Вона має легкі у використанні інтерфейси для багатьох мовних корпусів, а також бібліотеки для обробки текстів для класифікації, токенізації, стемінгу, розмітки, фільтрації та семантичного аналізу. Також це опен-сорсний проект, який розвивається за допомогою спільноти.

Matplotlib – це бібліотека для Python, яка надає широкі можливості для створення якісних графіків та візуалізації даних, і має багато функцій та можливостей для налаштування. За допомогою Matplotlib ви можете створювати різні типи графіків. Однією з головних переваг Matplotlib є його гнучкість та можливості налаштування. Ви можете налаштувати майже будь-який аспект графіка, такий як осі, підписи, легенди, кольори, шрифти та багато іншого, щоб створити найбільш підходящу візуалізацію для ваших даних.

Pymorphy2 - багатофункціональна бібліотека для морфологічного аналізу української мови [16]. В даній роботі використовувалася лише для лематизації.

Numpy – це пакет для наукових обчислень. Він обов'язково використовується в комплекті з matplotlib, що використовує функції numpy для числових даних та багатовимірних масивів.

Pandas - це бібліотека, що використовується matplotlib в основному для обробки та аналізу даних. Pandas надає об'єкт таблиці 2D-даних у пам'яті під назвою Dataframe.

Telegram-клієнт - це додаток, який можна встановити на свій смартфон, планшет, комп'ютер або інший пристрій і використовувати для спілкування з іншими користувачами Telegram. Клієнт Telegram дозволяє надсилати повідомлення, обмінюватися медіафайлами, створювати групи і канали, а також використовувати інші функції, доступні в Telegram. Telegram-клієнт - це додаток для кінцевих користувачів.

Telegram API - це набір програмних інтерфейсів, які розробники можуть використовувати для створення власних додатків та інтеграції з Telegram. API Telegram дозволяє розробникам створювати додатки, які можуть надсилати та отримувати повідомлення, керувати групами і каналами, отримувати дані користувачів та інші функції, доступні в Telegram. Telegram API - це набір інструментів для розробників, які хочуть інтегрувати Telegram у свої додатки.

У процесі розробки інтерфейсу було використані наступні бібліотеки:

PySide2 - це бібліотека для розробки графічних інтерфейсів користувача на мові програмування Python, яка надає доступ до функцій Qt, які забезпечують створення мультиплатформних додатків з графічним інтерфейсом. PySide2 є прямим портом бібліотеки PyQt5 для мови Python, але має відкриту ліцензію LGPL (Lesser General Public License), що дозволяє використовувати її в комерційних проектах без необхідності відкривати вихідний код.

Опис бази даних, що використовувалась у розробленому програмному продукті.

PostgreSQL - це потужна та надійна система управління базами даних, яка широко використовується в різних додатках, включаючи веб-сервіси, додатки для мобільних пристроїв, аналітичні інструменти тощо. PostgreSQL пропонує широкий набір функцій та можливостей, таких як підтримка SQL, JSON, XML, повнотекстовий пошук, реплікація та масштабування, що робить його ідеальним вибором для будь-якого проекту, який вимагає швидкої та надійної роботи з даними.

Firebase - це платформа для розробки мобільних та веб-додатків, яка надає набір інструментів та послуг для реалізації різноманітних функціональних можливостей. Вона створена компанією Google і має низку сервісів, що полегшують роботу з хмарними обчисленнями та збереженням даних. Firebase надає такі основні можливості:

- Firebase Realtime Database
- Аутентифікація користувачів:
- Збереження файлів
- Хостинг
- Аналітика та звіти.

3.2 Програмна реалізація програмного модулю

3.2.1 Специфікації розроблених програмних процедур і функцій

Для описання програмного забезпечення, було додано специфікації розроблених програмних процедур і функцій. Які в себе включають, опис параметрів основних класів програми, які включають класи для попередньої обробки даних, створення стоп-слів, лематизації та передобробки тексту, а також класи для побудови графіків та аналізу даних, забезпечуючи потужні

інструменти для обробки, візуалізації та вилучення інформації з текстового корпусу.

Клас Modelling:

- __init__:
  Функція, яка виконує ініціалізацію об'єктів та проводить передобробку даних. Вона використовує клас Preprocessing() для обробки даних, включаючи видалення пропущених значень, створення списку стоп-слів та застосування векторизації за допомогою об’єкту CountVectorizer(). Потім вона викликає методи для візуалізації та аналізу даних, такі як відображення слів і підрахунок постів за часом. Вона виконує вбудовування даних та побудову моделі;
- embedding:
  Функція, яка приймає два аргументи “size” та “data”. Створює об’єкт CountVectorizer() - це інструмент для перетворення текстових даних на матрицю лічильників токенів. Ця матриця є перетворенням текстових даних у числове уявлення, де кожен стовпець відповідає токену, а кожен рядок відповідає текстовому зразку;
- build_model:
  Функція, яка містить різні параметри, такі як будування та навчання моделі LDA, знаходить топ слова для кожної теми, знижує розмірність до 2D за допомогою t-SNE та візуалізує результати за допомогою методів об'єкта “self.plotter”;
- heatmap:
  Функція, яка будує графік та теплову карту для аналізу LDA (Latent Dirichlet Allocation) на основі текстових даних. Вона використовує об’єкт CountVectorizer() для перетворення текстів на матрицю ознак, навчає LDA модель на отриманій матриці, визначає теми та їх

кількість, а потім будує графік та теплову карту для візуалізації результатів;

- post_per_time:

  Функція, яка виконує підрахунок кількості слів та частин мови в текстових даних, ґрунтуючись на заголовках, представлених у змінній "reindexed_data". Вона використовує бібліотеку TextBlob для маркування частин мови в кожному заголовку і зберігає результати підрахунку в змінні "word_counts" (кількість слів) та "pos_counts" (кількість частин мови).

Клас Preprocessing:

- __init__:

  Функція, яка ініціалізує об'єкт та перевіряє наявність стоп-слів;

- install_nltk:

  Функція, яка виконує встановлення та завантаження необхідних ресурсів та модулів з пакету NLTK (Natural Language Toolkit). Зокрема, вона завантажує стоп-слова “stopwords”, модель визначення частин мови “averaged_perceptron_tagger”, і токенизатор “punkt”;

- set_stop_words:

  Функція, яка встановлює список стоп-слів всередині об'єкта класу, використовуючи переданий список “words”;

- create_stop_words:

  Функція, яка створює список стоп-слів шляхом поєднання визначеного списку стоп-слів для російської мови з додатковими словами. Після створення списку стоп-слів вона записує його всередині класу для подальшого використання;

- lemmatize:

  Функція, яка виконує лематизацію списку слів. Для кожного слова у списку використовується бібліотека pymorphy2 для отримання

нормальної форми (леми). Результати лематизації зберігаються в новому списку і повертаються як результат функції;

- preprocessing:
  Функція, яка виконує попередню обробку рядка тексту російською мовою. Він виконує токенізацію та приведення слів до нижнього регістру, лематизацію слів, а потім фільтрацію стоп-слів та знаків пунктуації, повертаючи оброблений текст у вигляді рядка;
- process:
  Функція, яка виконує обробку та попередню підготовку даних. Він виводить перші три рядки даних, вибирає стовпці "date" та "message" з вихідного набору даних, виконує переіндексацію текстового стовпця по стовпцю "date" та видаляє порожні значення. Потім відбувається створення стоп-слів, застосування попередньої обробки тексту та збереження оброблених даних у новий CSV-файл, а потім повертається оброблений текстовий стовпець даних.

3.2.2 Збереження даних

Для зберігання даних використовується СУБД Postgres та середовища для розробки та адміністрування PgAdmin. На рис 3.1 представлено скріншот інтерфейсу pgadmin із довільним набором колонок з таблиці Post. Також бачимо, що було створено поля для зберігання текстових повідомлень, дати публікації, ключі - посилання на таблиці Channel та Theme, що містять id каналу та теми відповідно.

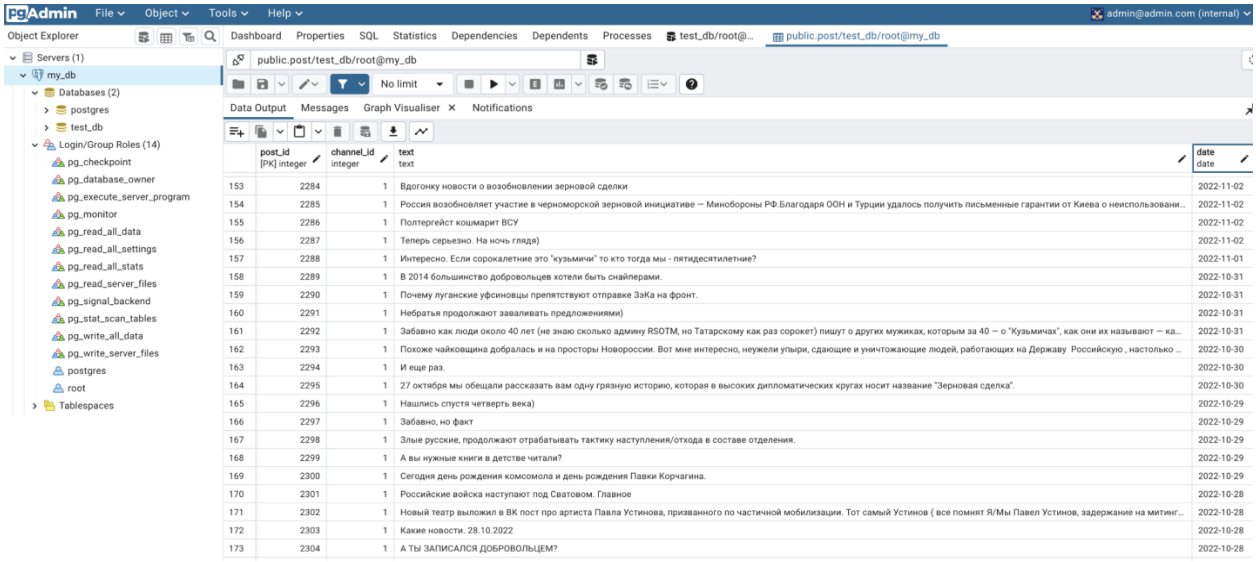

Рис 3.1 Інтерфейс PgAdmin 4

Для зберігання користувачів та виконання аутентифікації використано готове рішення на базі FireBase. На рис 3.2 наведена таблиця користувачів. Дане API має багато методів взаємодії, але на даному етапі використовувалися лише методи створення та аутентифікації користувачів. В якості провайдера використано опція логіну через email/password, але сама платформа підтримує також вхід за допомогою Google, Facebook, Yahoo акаунтів, що в майбутньому спростить реєстрацію та менеджмент акаунтів. Оскільки для блокування всіх доступів достатньо заблокувати лише один корпоративний акаунт користувача, наприклад, на платформі Google. Що є більш безпечним та сучасним підходом.

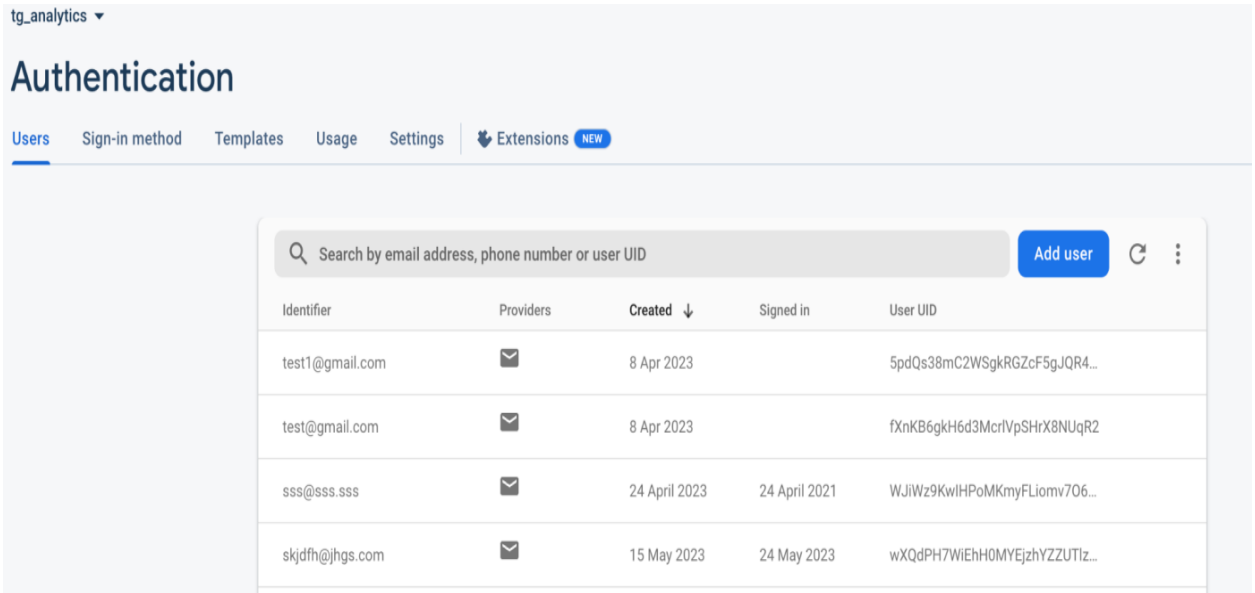

Рис 3.2 Таблиця користувачів

У п.п.1.1 було вказано, що аналіз та обробка текстів в соціальних мережах, здійснюється на прикладі соціальної мережі Telegram, як оптимальною для дослідження, через те що, цікавий для нас контент розміщуються в цій соціальній мережі частіше, через особливості аудиторії та можливість сховати обличчя за маскою аноніма.

Приєднання програмного модулю до мережі Telegram здійснюється через отримання доступу до Telegram API:

Спочатку нам потрібно зареєструвати власний додаток у сервісі Api Development tools у особистому кабінеті Telegram. Ви отримаєте токени доступу, які будуть використовуватися для аутентифікації, а саме api_id та api_hash. Після отримання токену доступу ми повинні налаштувати з'єднання з Telegram API. Для цього використовується бібліотеки або SDK (Software Development Kit) доступні для різних мов програмування,для Python це Telethon. Для підключення треба виконати автентифікацію самого додатку, для цього надаємо телефон розробника, на який приходить код з повідомленням. Ця операція виконується один раз. Після успішного підключення до Telegram API, ми можемо повноцінно працювати з Телеграм клієнтом. В Додатку В наведено код, що використовується для завантаження повідомлень з телеграм каналу. Для цього ми отримуємо від користувача посилання та період пошуку.

3.3 Демонстрація роботи програми на основі контрольного прикладу

3.3.1 Попередня обробка даних

Нехай маємо пост на рис 3.3. На основі цього прикладу розглянемо

```
'МАКРОНА ЩЕЛКНУЛИ ПО НОСУ БЕСПИЛОТЧИКИ ОБТФ «КАСКАД» МВД ДНР! \n\nПарни в районе Павловки лупят артиллерию супостата.
\nСегодня передали привет французам и Макрону и разбили «Ланцетом» хваленую французскую гаубицу «Цезарь». \nПривет
Парижу! \nВезите ещё!'
```

Рис 3.3 Приклад посту

Результати методів попередньої обробки текстів. Даний приклад разом з програмною реалізацією наведено в Додатку А. На рис 3.4 - 3.7 наведені результати виведення.

```
[['МАКРОНА', 'ЩЕЛКНУЛИ', 'ПО', 'НОСУ', 'БЕСПИЛОТЧИКИ', 'ОБТФ', '«', 'КАСКАД', '»', 'МВД', 'ДНР', '!'], ['Парни', 'в',
'районе', 'Павловки', 'лупят', 'артиллерию', 'супостата', '.'], ['Сегодня', 'передали', 'привет', 'французам', 'и',
'Макрону', 'и', 'разбили', '«', 'Ланцетом', '»', 'хваленую', 'французскую', 'гаубицу', '«', 'Цезарь', '»', '.'],
['Привет', 'Парижу', '!'], ['Везите', 'ещё', '!']]
```

Рис 3.4 Токенізація посту за реченнями

```
['МАКРОНА', 'ЩЕЛКНУЛИ', 'ПО', 'НОСУ', 'БЕСПИЛОТЧИКИ', 'ОБТФ', '«', 'КАСКАД', '»', 'МВД', 'ДНР', '!', 'Парни', 'в',
'районе', 'Павловки', 'лупят', 'артиллерию', 'супостата', '.', 'Сегодня', 'передали', 'привет', 'французам', 'и',
'Макрону', 'и', 'разбили', '«', 'Ланцетом', '»', 'хваленую', 'французскую', 'гаубицу', '«', 'Цезарь', '»', '.',
'Привет', 'Парижу', '!', 'Везите', 'ещё', '!']
```

Рис 3.5 Токенізація посту за словами

```
['макрон', 'щёлкнуть', 'по', 'нос', 'беспилотчик', 'обтф', '«', 'каскад', '»', 'мвд', 'днр', '!', 'парень', 'в',
'район', 'павловка', 'лупить', 'артиллерия', 'супостат', '.', 'сегодня', 'передать', 'привет', 'француз', 'и',
'макрон', 'и', 'разбить', '«', 'ланцет', '»', 'хвалёный', 'французский', 'гаубица', '«', 'цезарь', '»', '.',
'привет', 'париж', '!', 'везти', 'ещё', '!']
```

Рис 3.6 Лематизація

```
['макрон', 'щёлкнуть', 'нос', 'беспилотчик', 'обтф', '«', 'каскад', '»', 'мвд', 'днр', '!', 'парень', 'район',
'павловка', 'лупить', 'артиллерия', 'супостат', '.', 'сегодня', 'передать', 'привет', 'француз', 'макрон', 'разбить',
'«', 'ланцет', '»', 'хвалёный', 'французский', 'гаубица', '«', 'цезарь', '»', '.', 'привет', 'париж', '!', 'везти',
'ещё', '!']
```

Рис 3.7 Результат фільтрації від стоп-слів

На рис 3.8 та рис 3.9 подано скриншот довільно вибраних постів з датасету до процесу попередньої обробки та після нього.

```
date
2022-10-26    О ПОДГОТОВКЕ РОССИИ К СВО. ЛИЧНО Я НЕ ПОНИМАЮ,...
2022-10-26    О ВНЕЗАПНОМ РОСТЕ ОБЩЕСТВЕННОГО СТАТУСА БЛОГЕР...
2022-10-26    ВЛАДИМИР ПУТИН ЗАЯВИЛ О НЕОБХОДИМОСТИ ОБРАТНОЙ...
2022-10-26    НА ВСЮ ЛИНИЮ СОПРИКОСНОВЕНИЯ (БОЛЕЕ 1000 км) 1...
2022-10-26    АЛЕКСАНДР ХОДАКОВСКИЙ, ТЕНДЕНЦИИ СОВРЕМЕННОГО ...
2022-10-26    КИЕВ ГОТОВИТ МИНСК К ПЕРЕВОРОТУ? ГОВОРЯТ, ЧТО ...
2022-10-26    СВО, ВЛИЯНИЕ ПОГОДЫ НА ТАКТИКУ ДЕЙСТВИЙ СТОРОН...
2022-10-26    КОМАНДУЮЩИЙ СВО СУРОВИКИН ВОСХИТИЛ НАТОВСКОГО ...
2022-10-26    СНАЧАЛА ГОВОРИЛИ \nО ПОСТАВКАХ АМЕРИКАНСКИХ БР...
2022-10-26    ДОРОГИЕ ДРУЗЬЯ!\nИз-за огромного количества фе...
```

Рис 3.8 До процесу попередньої обробки тексту

date	
2022-10-26	подготовка россия сво лично понимать почему пр...
2022-10-26	внезапный рост общественный статус блогер корр...
2022-10-26	владимир путин заявить необходимость обратный ...
2022-10-26	весь линия соприкосновение 1000 км 16 « хамерс...
2022-10-26	александр ходаковский тенденция современный эт...
2022-10-26	киев готовить минск переворот говорить сценари...
2022-10-26	сво влияние погода тактика действие сторона ин...
2022-10-26	командующий сво суровикин восхитить натовский ...
2022-10-26	сначала говорить поставка американский бронежи...
2022-10-26	дорогой друг из-за огромный количество фейк ко...

Рис 3.9 Після процесу попередньої обробки тексту

3.3.2 Тематичне моделювання

Продемонструємо та порівняємо роботу LDA та pLSA на довільній вибірці розмірністю 200 текстових повідомлень. Нехай при цьому необхідно згенерувати 8 тем. Результат моделювання, використовуючи метод LDA, представлений на рис 3.10.

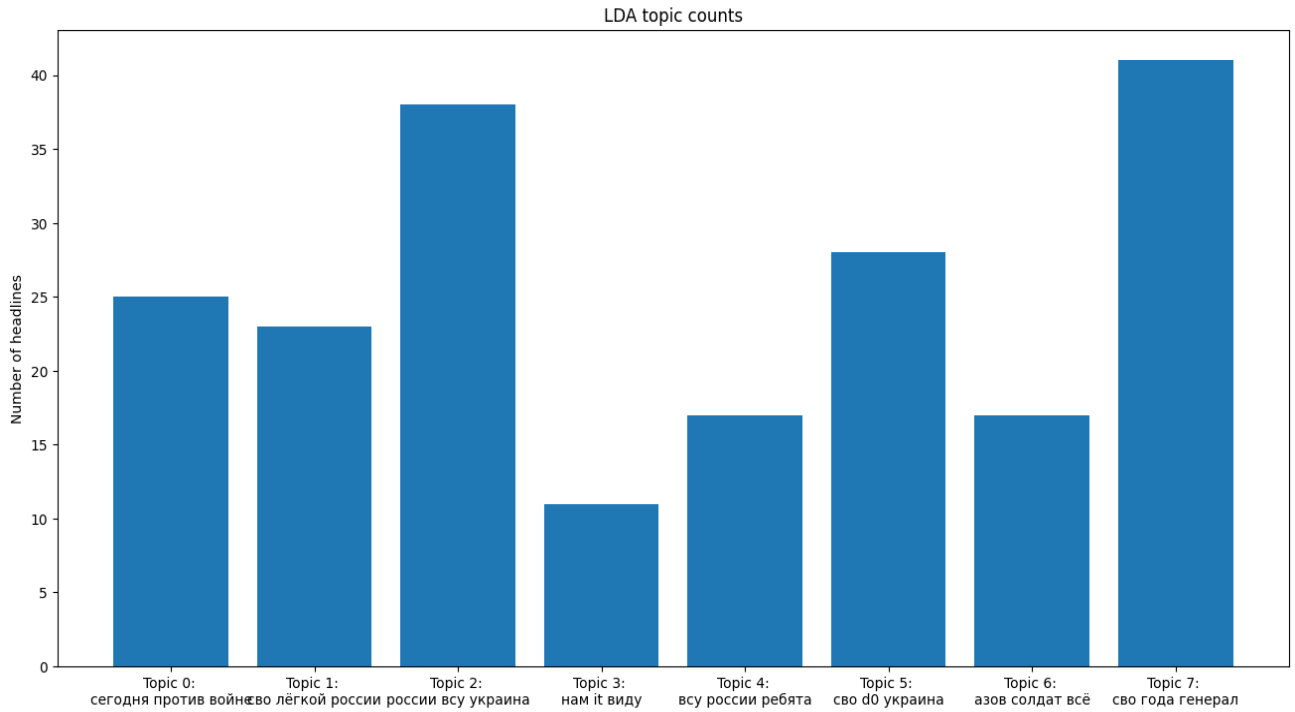

Рис 3.10 Результат моделювання, використовуючи метод LDA, згенеровано 8 тем на вибірці розмірністю 200 повідомлень

Кожна тема характеризується списком термів, в нашому випадку це безпосередньо вектор слів. При цьому вектори можуть містити спільні терми. Наприклад тема 2 характеризується набором слів «Россия, ВСУ, Украина». Для візуалізації розподілу текстів за темами використаємо бібліотечний алгоритм t-SNE (рис 3.11).

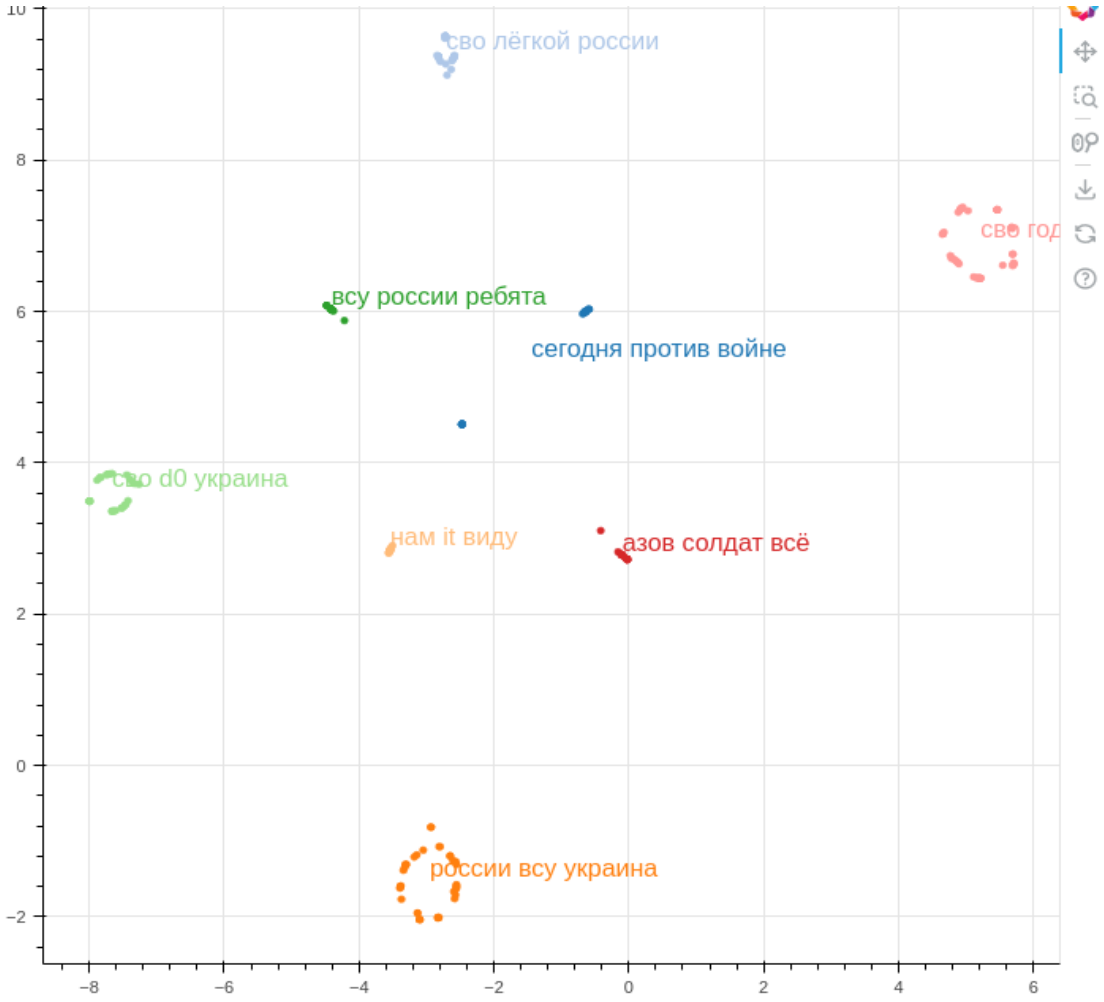

Рис 3.11 Візуалізація розподілу текстів за темами, використовуючи метод LDA

За аналогією виконаємо генерацію, використовуючи метод pLSA на рис. 3.12.

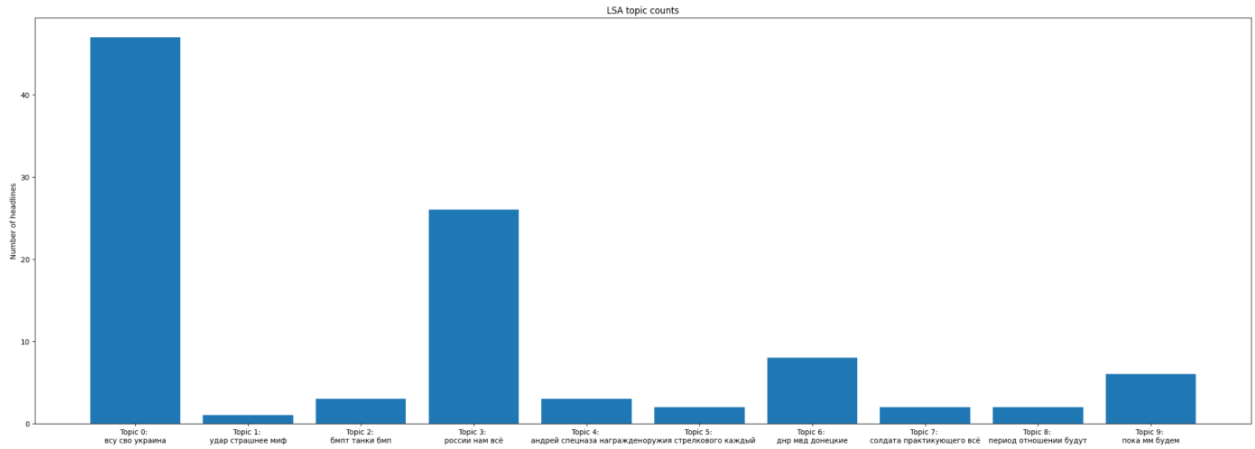

Рис 3.12 Результат моделювання, використовуючи метод pLSA, згенеровано 8 тем на вибірці розмірністю 200 повідомлень

Бачимо, що в цьому випадку pLSA тяжіє до виділення однієї теми. Ця тенденція збереглася для більшості експериментів.

Продемонструємо роботу алгоритмів при генерації тем на основі більшої вибірки. Нехай маємо 3000 повідомлень.

Результат моделювання, використовуючи pLSA (рис 3.13) та LDA (рис 3.14).

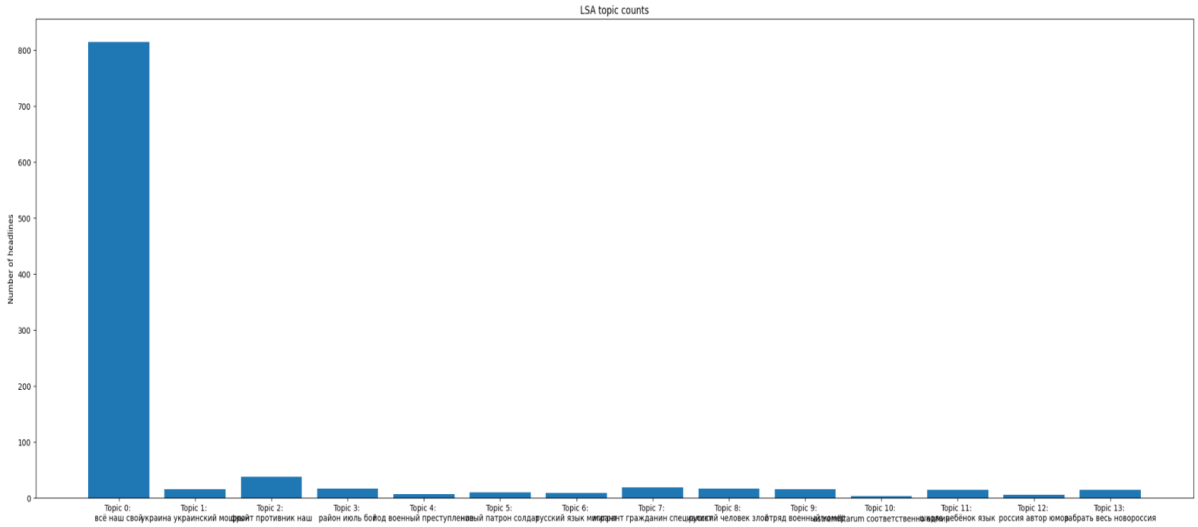

Рис 3.13 Результат моделювання, використовуючи метод pLSA, згенеровано 13 тем на вибірці розмірністю 3000 повідомлень

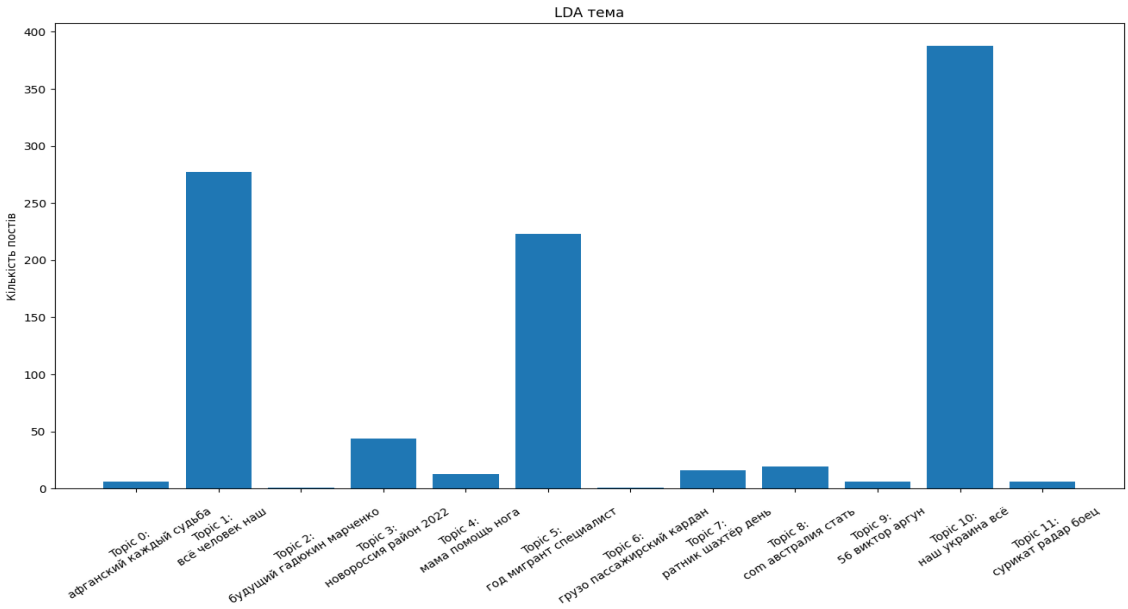

Рис 3.14 Результат моделювання, використовуючи метод LDA, згенеровано 11 тем на вибірці розмірністю 3000 повідомлень

Моделювання, використовуючи LDA, демонструє кращі результати, але бачимо, що в каналах є головна - переважна «тема», якій присвячені всі пости. Тобто існує, якась група слів, що є спільними для всіх постів та тем. Будемо вважати тему україно-російської війни спільною для всіх каналів та постів, та за допомогою стоп-слів приберемо «шум» від цієї теми з метою побачити інші підтеми.

Результат додаткової фільтрації на довільних 200 повідомленнях з використанням LDA наведено на рис 3.15.

Маємо 12 тем та наступні ключові слова до кожної теми.

'Тема 1: уйти цена оружие фронт почему',

'Тема 10: школа водитель девочка местный ребёнок',

'Тема 9: ДНР США противник Донецк позиция',

'Тема 12: сво 000 НАТО дать российский',

'Тема 11: канал российский утро РФ должный',

'Тема 5: мать SS медведь мальчик G20',

'Тема 4: первый НАТО университет Китай стать',

'Тема 7: мигрант специалист задержать её иностранный',

'Тема 6: причина новый мир цыган писать',

'Тема 2: отряд номер доброволец боевой специальность',

'Тема 3: первый стать сво армия иметь',

'Тема 8: фронт новый район противник ВСУ'

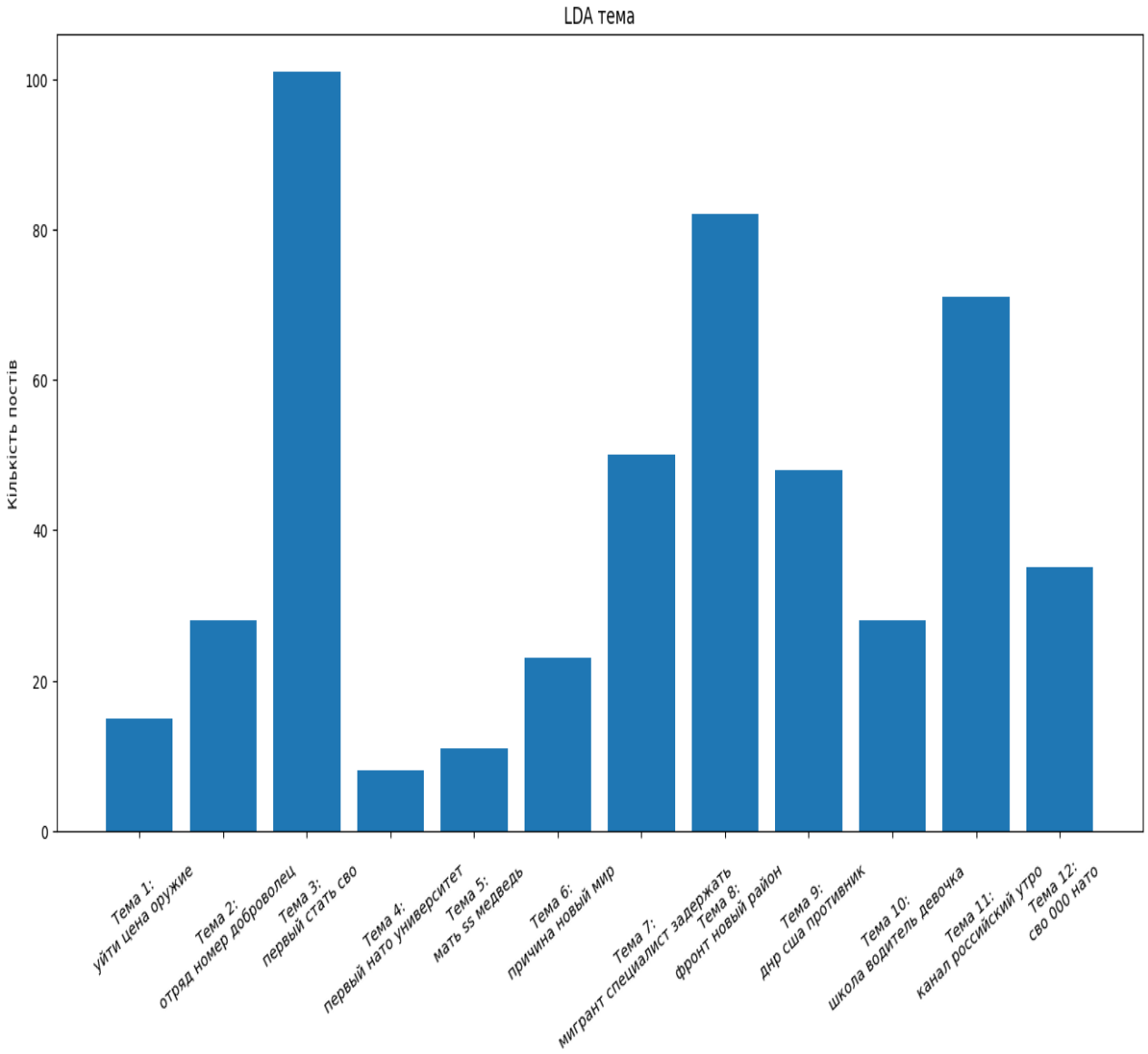

Рис 3.15 Результат моделювання, використовуючи метод LDA, з розширеним списоком стоп слів, згенеровано 12 тем на вибірці розмірністю 200 повідомлень

Використаємо отриману матрицю для того, щоб дослідити динаміку змін тем постів протягом вересня 2022 року. В період 8-10 вересня основною темою

висвітлення каналів був наступ на Донбасі та захоплення «нових районів» і цю зміну ми бачимо на рис 3.16.

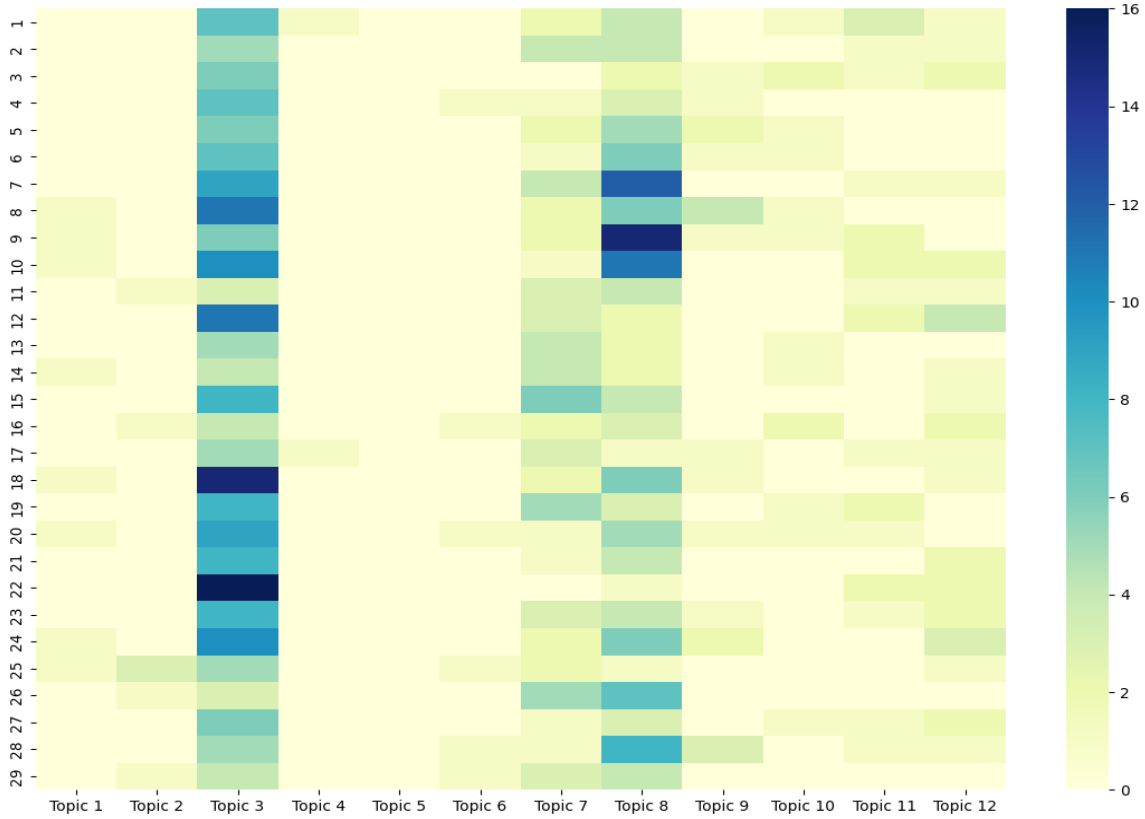

Рис 3.16 Згенерована матриця розподілу постів за темами для представлення динаміки змін тем постів протягом вересня 2022

3.4 Опис інструктивних матеріалів користувача.

З використанням фреймворка PyQt, що є програмною оболонкою для графічного фреймворку Qt було створено інтерфейс користувача. Для взаємодії з користувачем існує два типи вікон, а саме вікно для входу в систему і реєстрації та вікно безпосередньої взаємодії з системою та результатами аналізу.

3.4.1 Вхід до системи

Для роботи з програмним додатком Користувачу необхідно:

- Звернутись до адміністратора з проханням про реєстрацію.
- Отримати від адміністратора логін та пароль для входу.

- Увійти у програму з використанням логіну та паролю, як показано на рис 3.17.

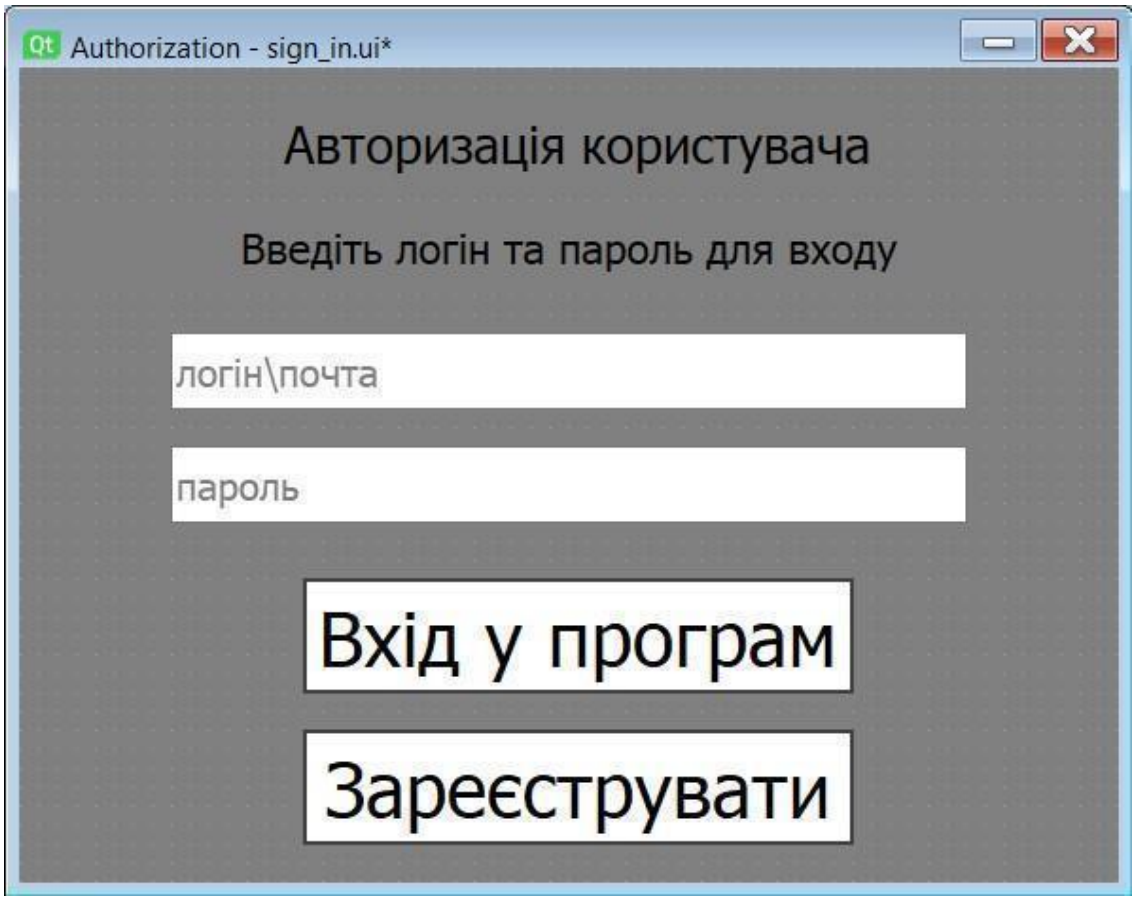

Рис 3.17 Панель авторизації для входу користувача

- Після успішної процедури ініціалізації облікового запису користувач може користуватися програмним додатком.

3.4.2 Створення запиту та отримання результату аналізу та обробки тексту

Перед подачею запиту на аналіз та обробку тексту необхідно задати параметри пошуку у програму.

- 1. Потрібно заповнити поля, як показано рис 3.18:
  - «link channel» - посилання на необхідний канал або групу;

- «from date» - з якого періоду часу, через пропуск (space);
- «to date» - по який періоду часу, через пропуск (space).

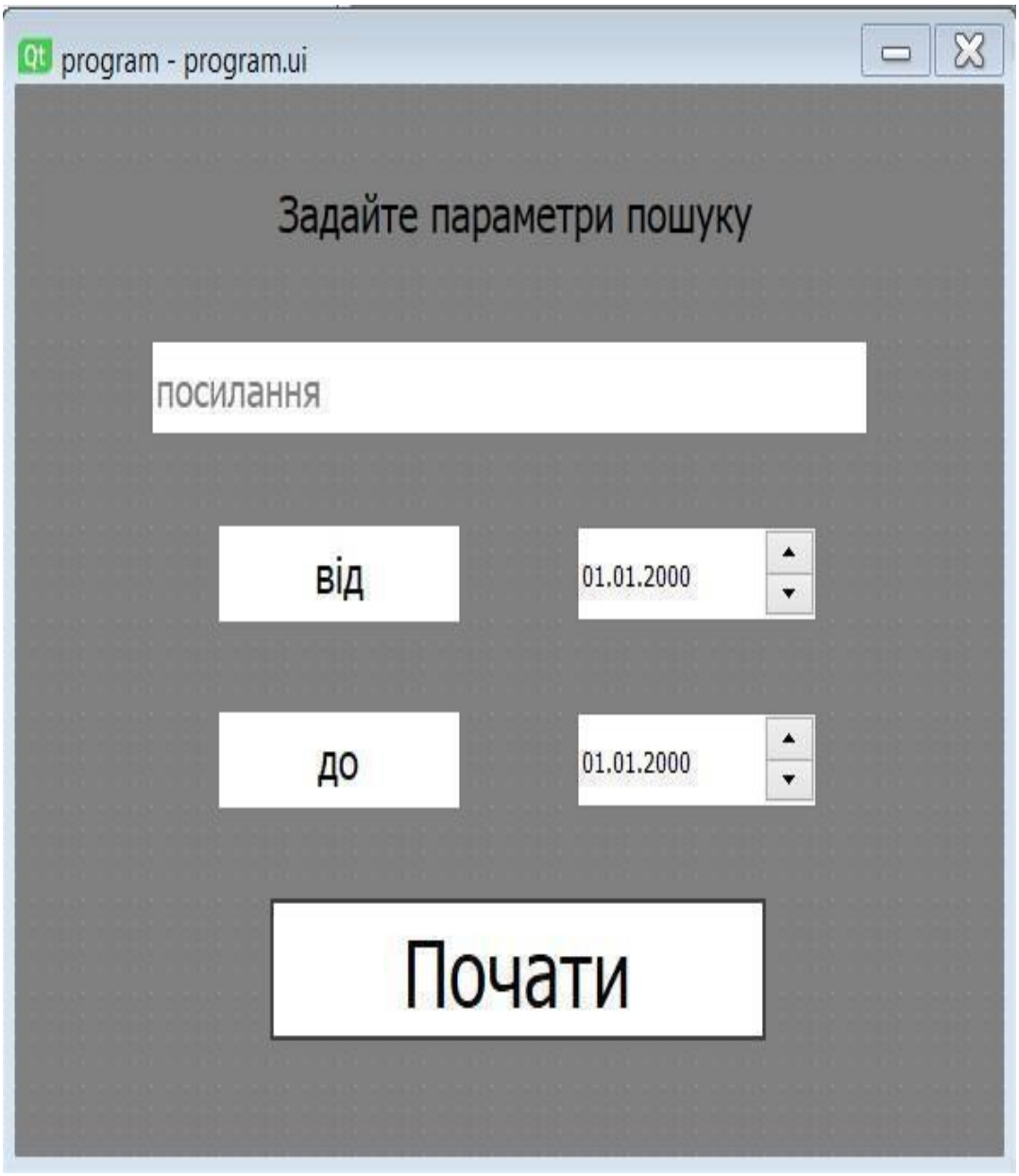

Рис 3.18 Панель для завантаження параметрів для запиту

- Натисніть кнопку «Завантажити», щоб завантажити необхідні дані.
- Відбувається автоматичний перехід на вікно результатів та завантаження цих результатів.

- На цьому кроці представлена демонстрація результатів програмного застосунку, як показано рис 3.19 і рис 3.20.

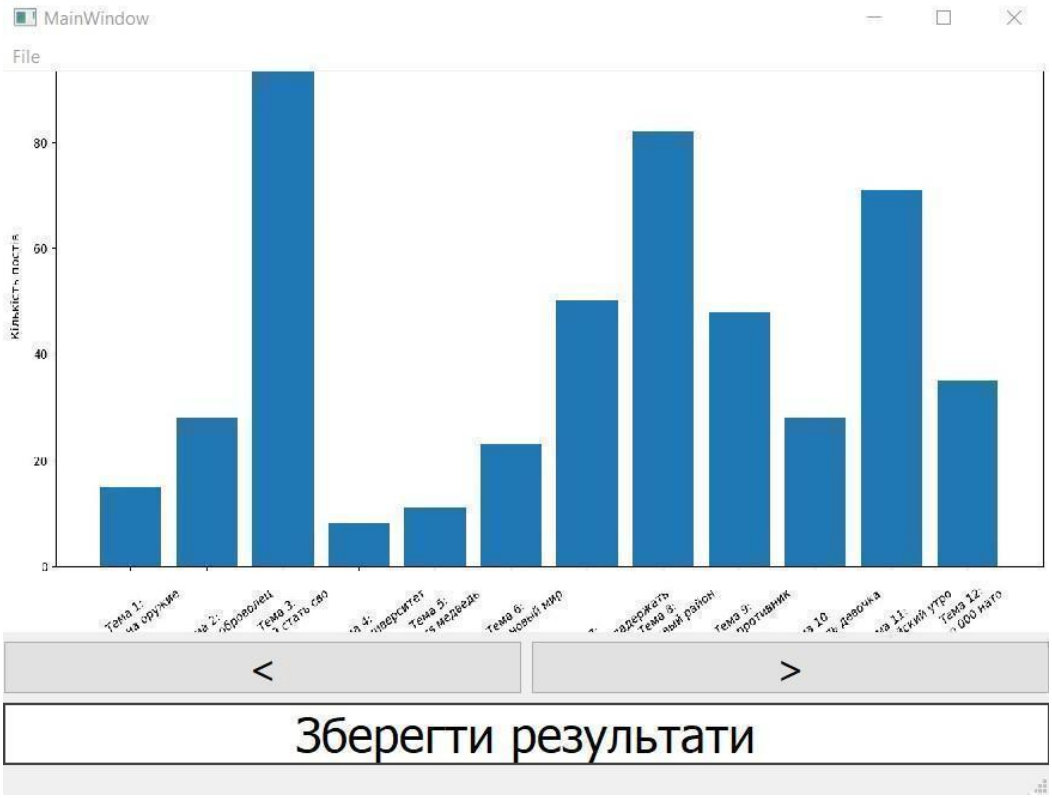

Рис 3.19 Демонстрація результатів програмного застосунку, приклад №1

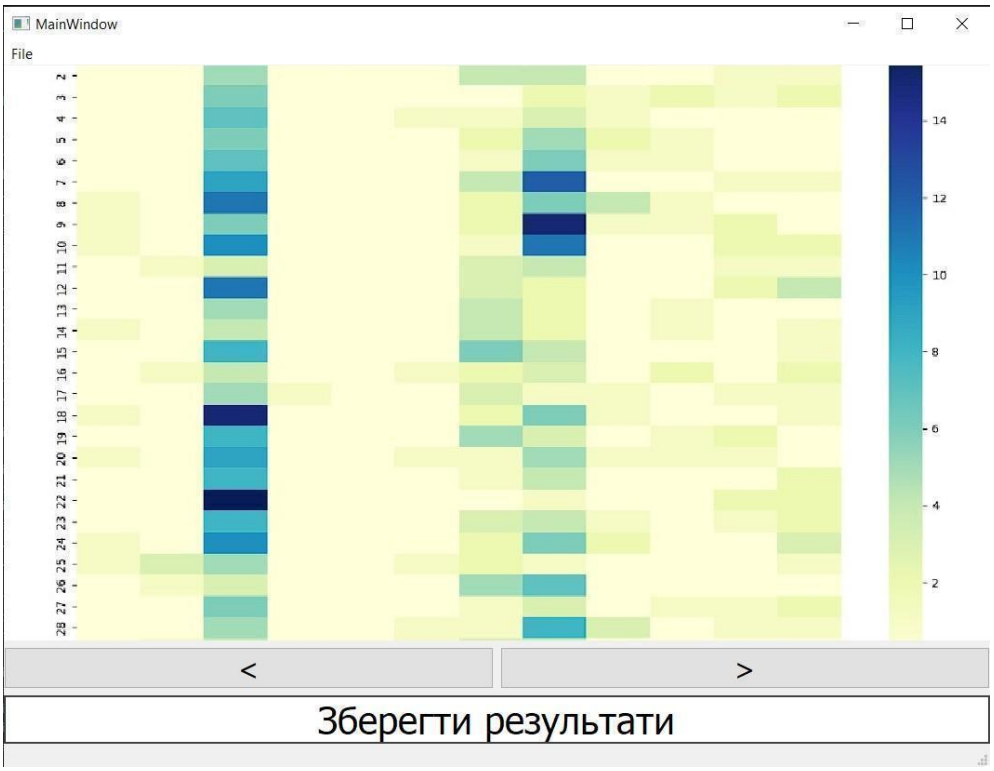

Рис 3.20 Демонстрація результатів програмного застосунку, приклад №2

- Натиснувши на кнопку «Завантажити результати», у форматі .PNG або .JPG результат можна зберегти на комп’ютер або ноутбук.

3.5 Аналіз отриманих результатів

Для реалізації модулю аналізу текстів, а саме тематизації було проведено ряд експериментів та емпіричним шляхом було підтверджено, наявність суттєвого недоліку у pLSA, а саме, що число параметрів лінійно зростає при зростанні числа документів в колекції, що може призводити до перенавчання моделі. На великому масштабі текстових повідомлень, більше 1000, pLSA практично перестає працювати. Експериментальне підтвердження - рис.3.12, рис.3.13.

Тому прийнято рішення, що для цієї задачі буде взято за основу метод LDA. Як видно з експериментів, наведених в підрозділі 3.3, рис.3.12, рис.3.13, моделювання на основі LDA видає змістовні результати та відмічається стабільність роботи при збільшенні кількості параметрів. Таким чином використовується програмна реалізація алгоритму LDA.

3.6 Висновки до третього розділу

Програмно реалізовано програмний модуль аналізу та обробки тексту у соціальних мережах на прикладі соціальної мережі Telegram, що задовольняє зазначені в першому розділі функціональні та нефункціональні вимоги.

- Реєстрація користувачів, наявність різних ролей, аутентифікація та авторизація - реалізовано інтерфейс для реєстрації та входу користувачів, існує дві ролі, а саме адміністратор та звичайний користувач, збереження та аутентифікація виконується на базі Firebase Authentication. Відповідно до існуючих стандартів безпеки пароль

користувача в базі даних не зберігається. Також Firebase Authentication автоматично блокує всі зі сторони третьої особи зміни або викрасти дані користувачів за допомогою хакерських атак [22]. Таким чином забезпечується безпечність та швидкість процедури логінізації та аутентифікації.

- Базова реалізація інтерфейсу користувача це можливість вибору телеграм каналів, задання часового періоду пошуку.
- Також, як продемонстровано у попередньому розділі користувач може переглядати та зберігати результати аналізу.
- Програмний модуль має мінімізований та доступний для інтуїтивного розуміння інтерфейс.
- Наведено інструкцію користувача (підрозділ 3.4), також в кваліфікаційній роботі наведені елементи документації, що стосуються аналізу функцій та архітектури програмного модулю.
- Швидкість роботи програми було оптимізовано, наприклад, якщо подібний запит вже колись виконувався, то крок парсингу телеграм каналів пропускається, що суттєво впливає на швидкість обробки запитів.

ВИСНОВКИ

У кваліфікаційній роботі була розглянута проблема процесу аналізу та обробки текстів у соціальних мережах на прикладі соціальної мережі Telegram. Актуальність даної проблеми обумовлена зростаючою потребою своєчасного виявлення інформації, яка б могла сигналізувати про потенційну загрозу різного характеру.

Для досягнення мети було розв'язано наступні задачі:

- дослідження існуючих підходів, методів та технологій;
- розробка підходу до аналізу текстів у соціальних мережах на прикладі соціальної мережі Телеграм;
- проектування та реалізація програмного модулю аналізу та обробки тексту.

Було зібрано текстові дані, що містять пости з каналів так званих «воєнкорів» соціальної мережі Телеграм. Для розробки програми парсингу телеграм каналів було використано Telegram клієнт Telethon.

Для програмної реалізації програмного модулю використано мову програмування Python. Для зберігання даних використовувалася СУБД PostgreSQL. Для розробки програмного модуля були використані компоненти бібліотек Telethon, Matplotlib, Pymorphy2, Numpy, Pandas, PyQt.

Отже, результатом проведеної роботи є програмний модуль аналізу та обробки тексту. У програмному модулі реалізовано основні етапи обробки тексту та виконується процедура тематичного моделювання.

Розроблений програмний модуль аналізу та обробки тексту в соціальних мережах, може бути використаний для своєчасного виявлення інформації, яка б могла сигналізувати про потенційну загрозу різного характеру. Такої інформації стає дедалі більше і більшість країн реагує на це на законодавчому рівні для запобігання цьому. Тому моніторинг, який може бути здійснений за допомогою таких моделей є важливою частиною нашого сьогодення.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. What is Text Mining, Text Analytics and Natural Language Processing? https://www.linguamatics.com/what-text-mining-text-analytics-and-natural-language-processing

2. Медіаспоживання українців в умовах повномасштабної війни. Опитування ОПОРИ, 01 червня 2022, https://www.oporaua.org/report/polit_ad/24068-mediaspozhivannia-ukrayintsiv-v-umovakh-povnomasshtabnoyi-viini-opituvannia-opori

3. https://corp.megafon.ru/press/news/federalnye_novosti/20220321-1047.html

4. Мельник А.В Інтелектуальні інтегровані системи завдання інтелектуального аналізу тексту, https://sites.google.com/a/kubg.edu.ua/integrovani-sistemi/rozpiznavanna-tekstu-text-mining/zavdanna-intelektualnogo-analizu-tekstu

5. Role and applications of NLP in Cybersecurity Ovidiu Ursachi Feb 22,2019,https://medium.com/@ursachi/role-and-applications-of-nlp-in-cybersecurity-333d9280c737

6. How to Analyze and Process Unstructured Data, https://treehousetechgroup.com/how-to-analyze-and-process-unstructured-data/

7. Spam Filtering Using Bag-of-Words Nikita Sharma May 12, 2020, https://heartbeat.comet.ml/spam-filtering-using-bag-of-words-1c5484ff07f1

8. David M. Blei. 2012. Probabilistic topic models. Commun. ACM 55, 4 (April 2012), 77–84. https://doi.org/10.1145/2133806.2133826

9. Srinivas Chakravarthy Jun 19, 2020 Tokenization for Natural Language Processing

10. How to Get Started with NLP – 6 Unique Methods to Perform Tokenization, Shubham Singh — Published On July 18, 2019 and Last Modified On August 25th, 2022

11. Chetna Khanna Feb 10, 2021, Text pre-processing: Stop words removal using different libraries

12. Distributed Representations of Words and Phrases and their Compositionality / Tomas Mikolov, Google Inc. Mountain View, 2013.

13. Blei, D.M. Latent Dirichlet Allocation / D.M. Blei, A.Y. Ng, M.I. Jordan // Journal of Machine Learning Research. — 2003. —Vol. 3. — PP. 993 — 1022.

14. Joyce Xu May 25, 2018 Topic Modeling with LSA, PLSA, LDA & lda2Vec

15. Behic Guven Oct 4, 2020, Natural Language Processing (NLP) for Beginners

16. Pymorphy2 https://pymorphy2.readthedocs.io/en/stable/

17. Natural Language Toolkit: http://www.nltk.org/1

18. Thomas Hofmann. 1999. Probabilistic latent semantic indexing. In Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval (SIGIR '99). Association for Computing Machinery, New York, NY, USA, 50–57. https://doi.org/10.1145/312624.312649

19. Christos H. Papadimitriou, Hisao Tamaki, Prabhakar Raghavan, and Santosh Vempala. 1998. Latent semantic indexing: a probabilistic analysis. In Proceedings of the seventeenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems (PODS '98). Association for Computing Machinery, New York, NY, USA, 159–168. https://doi.org/10.1145/275487.275505

20. Landauer, T. K., McNamara, D. S., Dennis, S., & Kintsch, W. (Eds.). (2007). Handbook of latent semantic analysis. Lawrence Erlbaum Associates Publishers.ISBN 9781138004191 Published June 10, 2014 by Psychology Press 544 Pages

21. https://core.ac.uk/download/pdf/11320265.pdf

22. https://firebase.google.com/docs/auth/extend-with-blocking-functions?gen=2nd

23. https://texty.org.ua/projects/108016/telehram-okupaciya-yak-rosiya-vybudovuvala-mediamerezhu-vyjshlo-potomkinske-selo/

24. https://www.researchgate.net/publication/318981549_Fake_News_Detection_on_Social_Media_A_Data_Mining_Perspective

25. https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7862202

ДОДАТКИ

Додаток А: Лістинг коду. Попередня обробка тексту

```
s ='МАКРОНА ЩЕЛКНУЛИ ПО НОСУ БЕСПИЛОТЧИКИ ОБТФ «КАСКАД» МВД ДНР! \n\nПарни в районе Павловки лупят артиллерию супостат. \nСегодня передали привет французам и Макрону и разбили «Ланцетом» хваленую французскую гаубицу «Цезарь». \nПривет Парижу! \nВезите ещё!'
```

```
'МАКРОНА ЩЕЛКНУЛИ ПО НОСУ БЕСПИЛОТЧИКИ ОБТФ «КАСКАД» МВД ДНР! \n\nПарни в районе Павловки лупят артиллерию супостат. \nСегодня передали привет французам и Макрону и разбили «Ланцетом» хваленую французскую гаубицу «Цезарь». \nПривет Парижу! \nВезите ещё!'
```

```
from nltk.tokenize import sent_tokenize, word_tokenize

print([word_tokenize(t, language='russian') for t in sent_tokenize(s)])
```

```
[['МАКРОНА', 'ЩЕЛКНУЛИ', 'ПО', 'НОСУ', 'БЕСПИЛОТЧИКИ', 'ОБТФ', '«', 'КАСКАД', '»', 'МВД', 'ДНР', '!'], ['Парни', 'в', 'районе', 'Павловки', 'лупят', 'артиллерию', 'супостат', '.'], ['Сегодня', 'передали', 'привет', 'французам', 'и', 'Макрону', 'и', 'разбили', '«', 'Ланцетом', '»', 'хваленую', 'французскую', 'гаубицу', '«', 'Цезарь', '»', '.'], ['Привет', 'Парижу', '!'], ['Везите', 'ещё', '!']]
```

```
import pymorphy2
morph = pymorphy2.MorphAnalyzer()


def lemmatize(words):
    res = list()
    for word in words:
        p = morph.parse(word)[0]
        res.append(p.normal_form)

    return res
```

```
l = lemmatize(arr2)
print(l)
```

['макрон', 'щёлкнуть', 'по', 'нос', 'беспилотчик', 'обтф', '«', 'каскад', '»', 'мвд', 'днр', '!', 'парень', 'в', 'район', 'павловка', 'лупить', 'артиллерия', 'супостат', '.', 'сегодня', 'передать', 'привет', 'француз', 'и', 'макрон', 'и', 'разбить', '«', 'ланцет', '»', 'хвалёный', 'французский', 'гаубица', '«', 'цезарь', '»', '.', 'привет', 'париж', '!', 'везти', 'ещё', '!']

```
import nltk
from nltk.corpus import stopwords

stop_words = stopwords.words('russian')
stop_words.extend(["это", "https", "me", "которые"])
res = []

for w in l:
   if w not in stop_words:
       res.append(w)
```

['макрон', 'щёлкнуть', 'нос', 'беспилотчик', 'обтф', '«', 'каскад', '»', 'мвд', 'днр', '!', 'парень', 'район', 'павловка', 'лупить', 'артиллерия', 'супостат', '.', 'сегодня', 'передать', 'привет', 'француз', 'макрон', 'разбить', '«', 'ланцет', '»', 'хвалёный', 'французский', 'гаубица', '«', 'цезарь', '»', '.', 'привет', 'париж', '!', 'везти', 'ещё', '!']

Додаток Б: Лістинг коду. Клас LDA model

```
class Modeling():
    def __init__(self, data):
        prep = Preprocessing()
        print("Preprocessing started")
        data = prep.process(data)
        print("Preprocessing ok")
        data = pd.read_csv("./output/new.csv")
        data  = data.dropna()

        reindexed_data = data["message"]
        reindexed_data.index = data['date']
        self.plotter = DataPlot()

        prep.create_stop_words()

        self.stop_words = prep.get_stop_words()
        count_vectorizer = CountVectorizer(stop_words=self.stop_words)
        words, word_values = get_top(n_top_words=15,
count_vectorizer=count_vectorizer, text_data=reindexed_data)
        self.stop_words.extend(words)

        prep.set_stop_words(self.stop_words)
        self.plotter.plot_top_words(words, word_values)
        self.post_per_time(reindexed_data)
        self.n_topics = 6

        print("before embeding")
        self.embedding(300, reindexed_data)
        self.build_model()


    def embedding(self, size, data):
        print(len(data))
        self.count_vectorizer =
CountVectorizer(stop_words=self.stop_words, max_features=40000)
        small_text_sample = data.sample(n=size, random_state=1).values

        self.document_term_matrix =
self.count_vectorizer.fit_transform(small_text_sample)
        print("Emdedding ok")

    def build_model(self):
        lda_model =
LatentDirichletAllocation(n_components=self.n_topics,
learning_method='online', random_state=0, verbose=0)
```

```
        lda_topic_matrix =
lda_model.fit_transform(self.document_term_matrix)
        lda_keys = get_keys(lda_topic_matrix)
        lda_categories, lda_counts = keys_to_counts(lda_keys)
        top_n_words_lda = get_top_n_words(3, lda_keys,
self.document_term_matrix, self.count_vectorizer,self.n_topics)

        print(top_n_words_lda)

        for i in range(len(top_n_words_lda)):
            print("Topic {}: ".format(i+1), top_n_words_lda[i])
        top_3_words = get_top_n_words(3, lda_keys,
self.document_term_matrix, self.count_vectorizer, self.n_topics)
        self.plotter.lda_topics(top_3_words, lda_counts,
lda_categories)


        tsne_lda_model = TSNE(n_components=2, perplexity=50,
learning_rate=100,
                              n_iter=2000, verbose=1,
random_state=0, angle=0.75)
        tsne_lda_vectors =
tsne_lda_model.fit_transform(lda_topic_matrix)

        lda_mean_topic_vectors = get_mean_topic_vectors(lda_keys,
tsne_lda_vectors, self.n_topics)

        self.plotter.clustering(self.n_topics,lda_keys,
tsne_lda_vectors, lda_mean_topic_vectors, top_3_words)

    def heatmap(self, reindexed_data):
        big_sample_size = 800
        n = 10
        big_count_vectorizer =
CountVectorizer(stop_words=self.stop_words, max_features=40000)
        big_text_sample = reindexed_data.sample(n=big_sample_size,
random_state=1).values
        big_document_term_matrix =
big_count_vectorizer.fit_transform(big_text_sample)

        big_lda_model = LatentDirichletAllocation(n_components=n,
learning_method='online', random_state=3, verbose=0)
        lda_topic_matrix =
big_lda_model.fit_transform(big_document_term_matrix)

        lda_keys = get_keys(lda_topic_matrix)
        lda_categories, lda_counts = keys_to_counts(lda_keys)
```

```
        top_3_words = get_top_n_words(3, lda_keys,
big_document_term_matrix, big_count_vectorizer,n)
        labels = ['Topic {}: \n'.format(i+1) + top_3_words[i] for i in
lda_categories]

        fig, ax = plt.subplots(figsize=(16,8))
        ax.bar(lda_categories, lda_counts);
        ax.set_xticks(lda_categories);
        ax.set_xticklabels(labels);
        ax.set_title('LDA тема');
        ax.set_ylabel('Кількість постів');
        plt.xticks(rotation = 35)

        month_topic_counts =
self.plotter.plot_heatmap_by_month(9,reindexed_data,
big_count_vectorizer, big_lda_model)
        print(month_topic_counts)

        fig, ax = plt.subplots(figsize=(14,10))
        sb.heatmap(month_topic_counts, cmap="YlGnBu", ax=ax);
        plt.show()


    def post_per_time(self, reindexed_data):
        tagged_headlines = [TextBlob(reindexed_data[i]).pos_tags for i
in range(reindexed_data.shape[0])]
        tagged_headlines_df = pd.DataFrame({'tags':tagged_headlines})

        word_counts = []
        pos_counts = {}
        for headline in tagged_headlines_df[u'tags']:
            word_counts.append(len(headline))
            for tag in headline:
                if tag[1] in pos_counts:
                    pos_counts[tag[1]] += 1
                else:
                    pos_counts[tag[1]] = 1

        print('Total number of words: ', np.sum(word_counts))
        print('Mean number of words per headline: ',
np.mean(word_counts))
        reindexed_data.index = pd.to_datetime(reindexed_data.index)
        monthly_counts = reindexed_data.resample('M').count()
        daily_counts = reindexed_data.resample('D').count()
        self.plotter.plot_post_per_time(daily_counts, monthly_counts)
```

Додаток В: Взаємодія з Telegram API

```
import configparser
import json

from datetime import datetime, timedelta

import nest_asyncio
nest_asyncio.apply()

import asyncio
from datetime import date, datetime

from telethon import TelegramClient
from telethon.errors import SessionPasswordNeededError
from telethon.tl.functions.messages import (GetHistoryRequest)
from telethon.tl.types import (
PeerChannel
)

from server.insert_posts import save_posts
from server.insert_posts import save_channel

class DateTimeEncoder(json.JSONEncoder):
    def default(self, o):
        if isinstance(o, datetime):
            return o.isoformat()
        if isinstance(o, bytes):
            return list(o)
        return json.JSONEncoder.default(self, o)

def telegram_config():
    config = configparser.ConfigParser()
    config.read("config.ini")

    api_id = config['Telegram']['api_id']
    api_hash = str(config['Telegram']['api_hash'])

    phone = config['Telegram']['phone']
    username = config['Telegram']['username']

    return dict( phone = phone, api_id = api_id, api_hash = api_hash,
username = username)

def get_client(username,api_id,api_hash):
    client = TelegramClient(username, api_id, api_hash)
    return client
```

```
async def
get_messages(phone,total_count_limit,client,link,channel_id):
    await client.start()
    print("Client Created")
    if await client.is_user_authorized() == False:
        await client.send_code_request(phone)
        try:
            await client.sign_in(phone, input('Enter the code: '))
        except SessionPasswordNeededError:
            print("Password is needed")
    me = await client.get_me()

    my_channel = await client.get_entity(link)

    offset_id = 0
    limit = 100
    all_messages = []
    total_messages = 0

    while True:
        print("Current Offset ID is:", offset_id, "; Total Messages:",
total_messages)
        history = await client(GetHistoryRequest(
            peer=my_channel,
            offset_id=offset_id,
            offset_date=None,
            add_offset=0,
            limit=limit,
            max_id=0,
            min_id=0,
            hash=0
        ))
        if not history.messages:
            break
        messages = history.messages
        for message in messages:
            all_messages.append(message.to_dict())
        offset_id = messages[len(messages) - 1].id
        total_messages = len(all_messages)
        if total_count_limit != 0 and total_messages >=
total_count_limit:
            break
    save_posts(channel_id, all_messages)

    await client.disconnect()
```

```
def data_loader(link):
    conf = telegram_config()

    channel_id = 2

    client = get_client(conf["username"],
conf["api_id"],conf["api_hash"])
    t1 = '2022-02-02'
    date_t1 = datetime.strptime(t1, '%Y-%m-%d').date()

    t2 = '2022-02-05'
    date_t2  = datetime.strptime(t2, '%Y-%m-%d').date()

    save_channel(link,"name", date_t1, date_t2)

    client.loop.run_until_complete(get_messages(conf["phone"],
10,client, link, channel_id))
```