

Всеукраїнський конкурс студентських наукових робіт
зі штучного інтелекту

Обробка природної мови
(назва напряму)

НАУКОВА РОБОТА

ГЕНЕРАЦІЯ ТА АНАЛІЗ МЕРЕЖ ПОДІЙ З НОВИНИХ КАНАЛІВ
TELEGRAM ЗА ДОПОМОГОЮ ШТУЧНОГО ІНТЕЛЕКТУ

Шифр: „Перебіг подій”

2024

ЗМІСТ

ЗМІСТ	2
ВСТУП	3
ОПИС МЕТОДОЛОГІЇ	6
ПРИКЛАД ЗАСТОСУВАННЯ	10
ВИСНОВКИ.....	20
СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ	23
ДОДАТОК А. Збирання і обробка даних із соціальної мережі Telegram на АРІ	25
ДОДАТОК В. Клієнтська частина.....	33
АНОТАЦІЯ	41

ВСТУП

Об'єкт дослідження: Аналіз новинних потоків у месенджері Telegram, з фокусом на виявленні антидержавної риторики та інших форм дезінформації.

Предмет дослідження: Розробка методології та інструментальних засобів для автоматизованого аналізу та моніторингу новинних потоків у месенджері Telegram, з особливим акцентом на українських каналах, з метою виявлення антидержавної риторики.

Мета дослідження: Розробка алгоритмів та методів для автоматизованого виявлення та категоризації подій у текстах новинних повідомлень, створення інструментів для формування та візуалізації мереж подій з урахуванням хронології та ступеня їхнього впливу, а також розробка прототипу інформаційно-аналітичної системи для моніторингу та аналізу новинних потоків у месенджері Telegram з акцентом на виявлення антидержавної риторики в Україні.

Для досягнення мети дослідження ми маємо вирішити наступні завдання:

1. Збір даних, використовуючи системи пошуку новин.
2. Виявлення подій у текстах новинних повідомлень за допомогою GenAI.
3. Ідентифікацію оригінальних подій та вилучення дублікатів.
4. Встановлення причинно-наслідкових зв'язків між подіями.
5. Аналіз та візуалізацію мережі подій.
6. Створення інтерактивних семантичних карт.

Виявлення нових подій із текстових новинних повідомлень – це актуальне завдання в області обробки природної мови, що має значний вплив на наше розуміння світу [1]. Традиційні методи дослідження зосереджуються на ідентифікації окремих подій у текстах, але не дають уявлення про взаємозв'язки між ними [2].

Застосування генеративного штучного інтелекту (GenAI) дає можливість не лише виявляти події, але й формувати каузальні мережі подій, де чітко окреслені причини та наслідки. Цей підхід розширює можливості аналізу новин, дозволяючи досліджувати динаміку та взаємопов'язаність подій [3].

Для реалізації цього підходу можуть використовуватись великі лінгвістичні моделі, такі як ChatGPT (<https://chat.openai.com/>), Llama-2 (<https://www.llama2.ai/>), Gemini (<https://gemini.google.com/app>). Ці моделі володіють потужними аналітичними можливостями, що дають змогу автоматизувати процес формування та аналізу каузальних мереж подій [4].

Подальший аналіз сформованих мереж за допомогою GenAI дозволяє виявляти ланцюжки подій, аналогічні тим, що використовуються в сценарному аналізі. Відмінність полягає в тому, що вузлами традиційних каузальних мереж є концепти, а запропонована технологія дає можливість розглядати вузлами конкретні події.

Аналіз та візуалізація каузальних мереж подій можуть здійснюватися за допомогою звичайних інструментів аналізу мереж, таких як Gephi або GraphViz [14][15]. Ці інструменти дають змогу візуалізувати зв'язки між подіями, виявляти ключові фактори та розуміти динаміку розвитку подій. Цей проект використовує написану власноруч бібліотеку на мові Flutter, яка вирішує дані завдання.

Запропонована методологія відкриває широкі можливості для дослідження та аналізу новинних потоків у Telegram. Ця методологія може бути застосована до різноманітних тем, таких як політика, економіка, бізнес, технології та інші [5][6].

Ця робота робить значний внесок у розвиток методів аналізу новинних подій. Методологія обговорена у цьому документі дозволяє автоматизувати процес формування та аналізу каузальних мереж подій, що розширює можливості розуміння динаміки та взаємозв'язків новинних потоків [7].

Наукова новизна:

1. Розроблена методологія виявлення побудови мережі і кластеризації подій на основі застосування штучного інтелекту.

2. Кластеризація подій здійснена за допомогою прихованих факторів, а саме сутностей, які відповідають саме цим подіям, що є елементами простору параметрів.

ОПИС МЕТОДОЛОГІЇ

Методологія формування та аналізу мережі подій у новинних каналах Telegram за допомогою генеративного штучного інтелекту (GenAI) складається з наступних етапів:

1. Збір даних:

1. Використання існуючих систем пошуку новин (як безкоштовних, так і платних) для збору новинних повідомлень, що відповідають заданій тематиці.
2. Фільтрування зібраних повідомлень за їх відповідністю до теми дослідження.
3. Очищення текстів повідомлень від шуму та незначущих даних.

Для пошуку й отримання актуальних новин було запропоновано наступну формулу:

$$F_{news}(t, k, C) = \{articles \mid articles.date \in [t_{start}, t_{end}] \wedge \wedge keywords(articles) \cap k \neq \emptyset \wedge articles.channel \in C\},$$

де t представляє часові параметри (дата початку та кінця), k є набором ключових слів, а C є набором ідентифікаторів каналів.

2. Виявлення подій:

- Застосування GenAI з промптом для виявлення подій у текстах новинних повідомлень.
- Формування масиву коротких позначень виявлених подій.

Для виявлення подій у новинних повідомленнях було використано таку формулу:

$$E_{detect}(text) = \{e_1, e_2, \dots, e_n \mid e_i \in extract_events(text)\}$$

Ця функція обробляє текст кожної статті для виявлення подій, де *extract_events* це функція NLP, яка здійснюється за допомогою моделей AI, таких як GenAI.

3. Ідентифікація оригінальних подій:

- Застосування GenAI з промптом 2 для ідентифікації оригінальних подій серед виявлених.
- Вилучення дублікатів та переформулювань.

4. Встановлення причинно-наслідкових зв'язків:

- Застосування GenAI з промптом для встановлення причинно-наслідкових зв'язків між оригінальними подіями.
- Формування мережі подій, де вузлами є події, а ребрами - причинно-наслідкові зв'язки.

Для знаходження причинно-наслідкових зав'язків між подіями використана наступна формула:

$$Clink(E) = \{ (e_i, e_j) \mid cause(e_i, e_j) \}$$

Для списку подій E , ця функція визначає пари подій, де e_i є причиною e_j .

5. Аналіз та візуалізація мережі:

- Аналіз мережі подій за допомогою програми.
- Виявлення ключових подій, кластерів та ланцюгів які зв'язують дані події.
- Візуалізація мережі подій у графічному форматі.

Для створення інтерактивної візуалізації мережі подій застосована наступна формула:

$$S_{map}(E, L) = \{ (e, link(e)) \mid e \in E \wedge link(e) \in L \},$$

де кожна подія e пов'язана з URL або документальним посиланням $link(e)$ зі зібраних джерел даних.

6. Створення інтерактивних семантичних карт:

- Створення відповідного програмного забезпечення, яке надасть можливість побудови інтерактивних семантичних карт на базі API.
- Додавання посилання на новину у телеграм.

7. Додаткові можливості (у майбутньому):

- Використання додаткових промптів GenAI для виявлення інших типів зв'язків між подіями (наприклад, temporal relations).
- Застосування методів машинного навчання для автоматичного аналізу мереж подій.
- Розробка інструментів для візуалізації та аналізу мереж подій в реальному часі.
- Аналіз подій використовуючи інші соціальні мережі(Twitter(X)/Facebook etc), а також новинні ресурси(New York Times API) і пошукові системи(Bind/Google)для виявлення першоджерела події.
- Розробка системи для виявлення й перевірки достовірності інформації на різних онлайн-ресурсах за допомогою моделі мови на основі довіреної мережі довготривалої пам'яті (LLM). Ця система буде інтегрована з базою даних, в якій будуть зберігатися інформація про події, які раніше визначалися як «вкиди», порівняно з нормальними подіями [9]. Під час роботи система аналізуватиме дані і використовуватиме модель мови LLM для виявлення вірогідних «вکیدів» та нормальних подій. На основі історії ресурсу буде зберігатись інформація про те, наскільки він постачає "якісні" новини. Ця інформація буде використовуватись в подальшій роботі інших алгоритмів для оцінки довіри до джерела інформації та впливу його матеріалів на загальний контекст. Такий підхід дозволить покращити якість фільтрації і виявлення "вکیدів", а також сприятиме формуванню об'єктивної картини ситуації.

Ця методологія має широкий спектр застосування в різних галузях, таких як:

- Моніторинг новин та виявлення нових тенденцій.
- Маркетингові дослідження та аналіз ринку.
- Аналіз політичних та соціальних подій.
- Розслідування злочинів та розкриття шахрайства.
- Вивчення історії та розвитку суспільства.

Запропонована методологія дозволяє автоматизувати процес формування та аналізу мереж подій, що розширює можливості розуміння динаміки розвитку новинних потоків у реальному часі [8].

Переваги дослідження Telegram-каналів:

- **Висока швидкість оновлення новин:** Telegram-канали часто оновлюються значно швидше ніж офіційні новинні сайти. Це дозволяє відслідковувати розвиток подій майже в реальному часі.
- **Неформальний стиль комунікації:** Новинні повідомлення в Telegram-каналах часто мають більш короткий та неформальний стиль, що спрощує їх обробку системами GenAI.
- **Різноманіття точок зору:** Telegram надає доступ до широкого спектру точок зору та джерел новин, включаючи як офіційні ЗМІ, так і незалежні канали. Це дозволяє аналізувати події з різних ракурсів та виявляти приховані зв'язки [12].
- **Велика кількість даних:** При правильній кластеризації кластеризації Telegram-каналів за відповідними темами (спорт/кібербезпека/новини і т.п.) можна отримати інструмент, який надає можливість швидко слідкувати за інформацією і робити її аналіз у реальному часі.

ПРИКЛАД ЗАСТОСУВАННЯ

Розглянемо все на прикладі новин про складні процеси в Українській православної церкві (УПЦ), зокрема, критику ієрархів за антиукраїнську діяльність та співпрацю з Росією.

На першому етапі користувачу потрібно задати в GUI наступні параметри:

1. Початкова дата.
2. Кінцева дата.
3. Кількість новин, яку потрібно отримати.
4. Ключові слова для пошуку новин.

Далі у вкладці ресурсів обираємо «Telegram» і канали, в яких буде проходити пошук інформації. Для прикладу виберемо такі канали як «@verkhovnaradaukrainy», «@lachentyt», «@TCH_channel».

У відповідь на запит користувача, супроводжуваний кліком на кнопку "Search", клієнтський додаток ініціює процес взаємодії з API. Даний API призначений для здійснення пошуку інформації в Telegram-каналах. Під час цього процесу використовується спеціально сформований запит, який включає в себе необхідні параметри, щоб точно визначити шукану інформацію. Центральним етапом цього процесу є формування промпта, який служить основою для пошуку. Даний промпт ретельно аналізується, і ключові слова виділяються для подальшого використання у пошуковому запиті. Це дозволяє підвищити ефективність пошуку та точність результатів, оскільки дозволяє зменшити обсяг інформації, що обробляється, до суттєвих компонентів.

У контексті дослідження ефективності системи API для пошуку інформації, був розглянутий випадок, коли API було використано для формування списку новин на основі запиту «УПЦ». Отриманий список новин є результатом даного запиту:

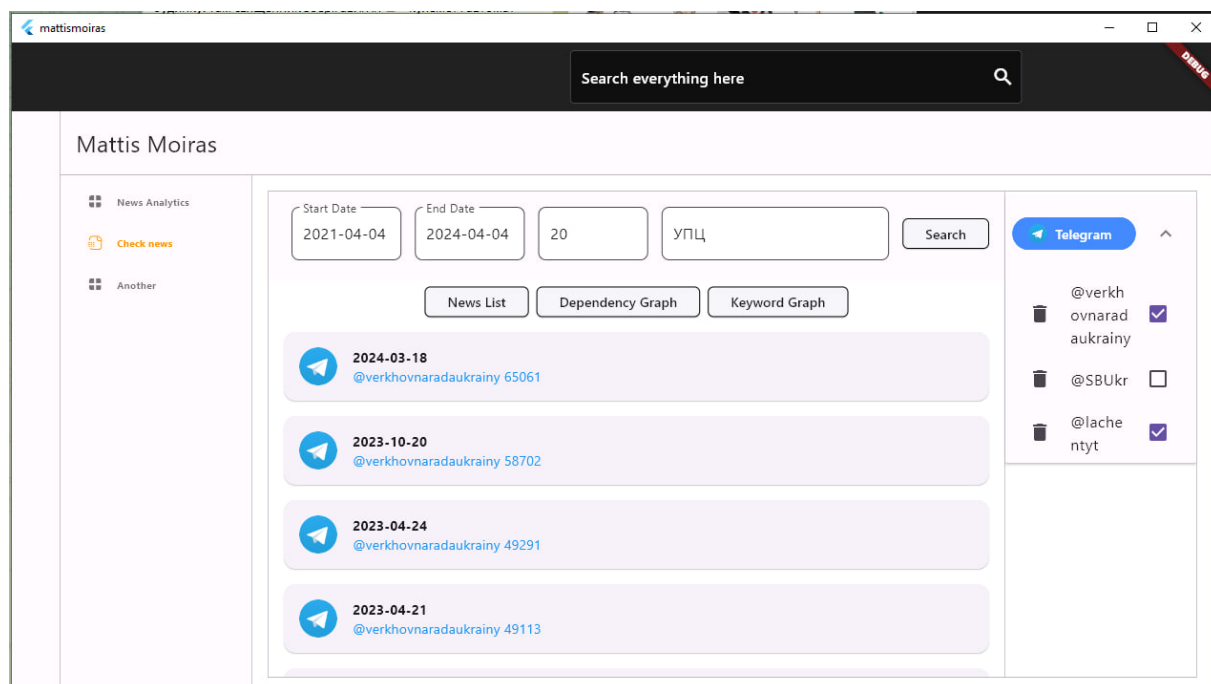


Рисунок 1. Скріншот роботи клієнтського застосунку для введених параметрів

— Комітет гуманітарної та інформаційної політики розглянув лист голови компанії «Amsterdam & Partners LLP» Роберта Р. Амстердама, який представився міжнародним юридичним радником УПЦ...

— Верховна Рада України підтримала законопроект про заборону релігійних організацій, які пов'язані з РФ! Законопроект № 8371 стосується заборони УПЦ МП і має два ключових аспекти...

— Сьогодні Комісія МКП з приймання-передачі майна Національного заповідника «Києво-Печерська лавра» продовжила свою роботу...

...

На другому етапі для кожної із знайдених новин застосовується наступний промпт 1:

В тексті описані деякі події. Детектуй їх, назви мені у вигляді нумерованого списку. Кожна подія - 4-5 слів. Ось текст:
Комітет гуманітарної та інформаційної політики розглянув лист голови компанії «Amsterdam & Partners LLP» Роберта Р. Амстердама, який представився міжнародним юридичним радником УПЦ, щодо законопроекту щодо діяльності в Україні релігійних організацій (№ 8371). Голова Комітету Микита Потураєв підкреслив: «Лист містить погрози народним депутатам України, що, у разі підтримки Закону, проти нас будуть вжиті санкції, відкриті провадження, застосовано так званий акт Магнітського». За результатами засідання, Комітет направить листа до СБУ, яка має право звернутися до ФБР США щодо встановлення підстав

діяльності Р.Р. Амстердама, втручання компанії у внутрішні справи України.

У результаті множинного виконання подібних промптів формується масив позначень подій, не всі з яких є оригінальними. Тобто можливі повтори, перекази різними словами:

- Розгляд листа в Комітеті гуманітарної та інформаційної політики.
- Представлення листа головою компанії «Amsterdam & Partners LLP» Робертом Р. Амстердамом.
- Підкреслення Микитою Потураєвим погроз народним депутатам.
- Рішення Комітету щодо направлення листа до СБУ.

...

На третьому етапі здійснюється фільтрація подій, тобто обираються оригінальні, для чого весь масив відібраних позначень подій надається системі GenAI та виконується промпт 2:

Із названих подій обери оригінальні і найважливіші. Дублікати вилучай. Ось події:

- Розгляд листа в Комітеті гуманітарної та інформаційної політики.
- Представлення листа головою компанії «Amsterdam & Partners LLP» Робертом Р. Амстердамом.
- Підкреслення Микитою Потураєвим погроз народним депутатам.
- Рішення Комітету щодо направлення листа до СБУ.

...

Відповідь системи GenAI ChatGPT після опрацювання всіх новин має вигляд:

- Розгляд листа в Комітеті гуманітарної та інформаційної політики.
- Підкреслення Микитою Потураєвим погроз народним депутатам.
- Підтримка законопроекту Верховною Радою України.
- Заборона релігійних організацій, пов'язаних з РФ.
- Законопроект № 8371 щодо заборони УПЦ МП.
- Визнання зовнішнього впливу на УПЦ МП.
- Стимування можливості церкви асоціюватися з агресором.
- Підтримка свободи вибору церкви для віруючих та громад.
- Припинення чинності розпорядження про передачу монастирю УПЦ МП майна на території заповідника.
- Наголошення на державній власності Києво-Печерської лаври та її належності українському народу.

...

На четвертому етапі API формує причинно-наслідкові зв'язки серед виявлених оригінальних подій, для чого виконується промпт 3:

Видай пари взаємопов'язаних подій за принципом причина-наслідок (твоя відповідь має виглядати так: “<причина>; <наслідок>”). Ось події:

- Розгляд листа в Комітеті гуманітарної та інформаційної політики.
- Підкреслення Микитою Потураєвим погроз народним депутатам.
- Підтримка законопроекту Верховною Радою України.
- Заборона релігійних організацій, пов'язаних з РФ.
- Законопроект № 8371 щодо заборони УПЦ МП.
- Визнання зовнішнього впливу на УПЦ МП.

...

Відповідь системи ChatGPT на цей промпт має вигляд:

- Розгляд листа в Комітеті гуманітарної та інформаційної політики;
- Підтримка законопроекту Верховною Радою України.
- Підкреслення Микитою Потураєвим погроз народним депутатам;
- Заборона релігійних організацій, пов'язаних з РФ.
- Законопроект № 8371 щодо заборони УПЦ МП;
- Визнання зовнішнього впливу на УПЦ МП.
- Стримування можливості церкви асоціюватися з агресором;
- Підтримка свободи вибору церкви для віруючих та громад.
- Припинення чинності розпорядження про передачу монастирю УПЦ МП майна на території заповідника;
- Наголошення на державній власності Києво-Печерської лаври та її належності українському народу.

...

На п'ятому етапі вся відібрана інформація з попередніх етапів надсилається на клієнтську частину, де відповідно і відображається за допомогою GUI.

Інформація передається у форматі JSON та оброблюється, де записується у $\text{Map}\langle\text{String}, \text{List}\langle\text{String}\rangle\rangle$, де першим елементом є подія-причина, а другим – список подій-наслідків [10].

Дамо визначення наступним термінам:

Семантична мережа: Граф $G = (V, E)$ де V представляє події як вузли, а E представляє семантичні зв'язки (такі як причинно-наслідкові зв'язки) як ребра між цими подіями.

Функція семантичної близькості: $s(v_i, v_j)$ кількісно оцінює семантичну близькість між двома подіями v_i та v_j , засновані на характеристиках, вилучених з тексту, що описує ці події, та їхнього контекстуального метаданих.

Після обробки інформація буде мати наступний вигляд:

"Розгляд листа в Комітеті гуманітарної та інформаційної політики":
 [
 "Підтримка законопроекту Верховною Радою України",
],
 "Підкреслення Микитою Потураєвим погроз народним депутатам":
 [
 "Заборона релігійних організацій, пов'язаних з РФ",
 "Законопроект № 8371 щодо заборони УПЦ МП",
 "Стимування можливості церкви асоціюватися з агресором"
],
 "Законопроект № 8371 щодо заборони УПЦ МП": [
 "Визнання зовнішнього впливу на УПЦ МП",
],
 "Стимування можливості церкви асоціюватися з агресором": [
 "Підтримка свободи вибору церкви для віруючих та громад",
 "Визнання зовнішнього впливу на УПЦ МП"
],
 "Припинення чинності розпорядження про передачу монастирю УПЦ МП майна на території заповідника": [
 "Наголошення на державній власності Києво-Печерської лаври та її належності українському народу",
],
 "Зазначення історичного та культурного значення Києво-Печерської лаври, включеної до списку Всесвітньої спадщини ЮНЕСКО": [
 "Пояснення припинення договірних відносин з монастирем Московського патріархату через факти порушень умов договору та його використання для пропаганди \"русского мира\" в умовах російської агресії",
],
 "Залишення ченців у лаврі та зміни їх статусу, частина з яких приєднується до ПЦУ": [
 "Проведення інвентаризації державного майна у лаврі та повернення його в державне користування",
 "Стимування можливості церкви асоціюватися з агресором"
],
 "Спроби перешкодити роботі комісії МКІП з боку представників УПЦ": [
]

"Відзначення безпідставних та цинічних протестів та закидів на адресу України з боку Москви у світлі фактів жорсткого переслідування проукраїнських релігійних громад на тимчасово окупованих територіях",
 "Наголошення на державній власності Києво-Печерської лаври та її належності українському народу"
],

Після отримання даних із вхідного джерела, програма проводить перевірку наявної карти зв'язків та формує нову карту у форматі, що відображає події у вигляді пари – «подія, кількість зв'язків». Ця процедура сприяє виявленню подій, які мають кілька зв'язків як у наслідок, так і у якості причини.

На основі цієї інформації виділяються події, які виявляються ключовими у зв'язках між подіями для покращеної і зрозумілої візуалізації даних [11]. Після відбору ключових подій у програмі здійснюється побудова графу зв'язків між ними:

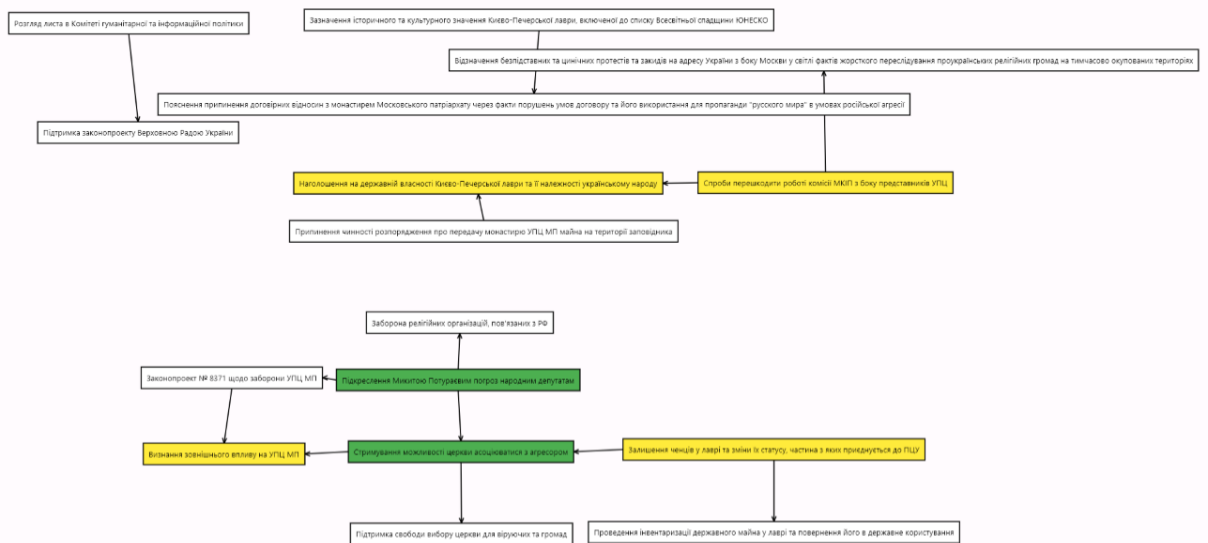


Рисунок 2. Карта зв'язку між подіями у вигляді графу

Наступним етапом є вирішення задачі кластеризації ключових слів з подій, яка потребує формування простору параметрів, у якості якого запропоновано застосовувати множину ключових слів, що відповідають подіям [13]. Для виявлення ключових слів для кожної визначеної події було відпрацьовано відповідний промпт, що включав запит і назву події:

Назви 20 ключових слів, пов'язаних з подією: Розгляд листа в Комітеті гуманітарної та інформаційної політики

Відповідь системи ChatGPT має вигляд:

- Інформаційної
- Політики
- Документ
- Доповідь
- Обговорення
- Висновок
- Рішення
- Рекомендації
- Аналіз
- Обґрунтування
- Пропозиція
- Відповідь
- Реакція
- Голосування
- Обговорення
- Презентація

...

Від API надходять дані для подальшої побудови графів на клієнтській частині. Після завершення процесу пошуку, API надсилає дані результати у форматі JSON на клієнтську частину системи. У відправленому JSON-об'єкті перший елемент визначається як подія, а другий елемент є списком ключових слів, що відображають контекст події. При отриманні цих даних програмна частина системи виконує їх обробку з метою побудови графів, що відображають взаємозв'язки між подіями.

Кластеризація може бути сформульована як оптимізаційна задача, метою якої є розділення множини вершин V на кластери таким чином, щоб події в межах кожного кластера були більш схожі одна на одну, ніж на ті, що знаходяться в інших кластерах. Формально це може бути виражено таким чином:

Ціль кластеризації: Мінімізувати внутрішньокластерні відстані, одночасно максимізуючи міжкластерні відстані.

Призначення кластера:

$$C: V \rightarrow \{1, 2, \dots, k\},$$

де $C(v)$ показує призначення кластера вершини v а k — кількість кластерів.

Внутрішньокластерна відстань:

$$dist_{intra}(C) = \sum_{i=1}^k \sum_{v, w \in C_i} d(v, w),$$

де $d(v, w) = 1 - s(v, w)$ і C_i — це множина вузлів у i -му кластері.

Міжкластерна відстань:

$$dist_{intra}(C) = \sum_{i \neq j} \min_{v \in C_i, w \in C_j} d(v, w)$$

Далі було визначено алгоритм кластеризації, який було застосований у роботі, а саме: максимізація модулярності для виявлення кластерів. Було визначено модулярність (Q) як міру сили поділу мережі на модулі (або кластери).

Математично це визначається як:

$$Q = \frac{1}{2m} \sum_{ij} S_{ij} - \left[S_{ij} - \frac{k_i \cdot k_j}{2 \cdot m} \right] \delta(C_i, C_j),$$

де S_{ij} — елемент матриці схожості S , k_i і k_j — суми ваг ребер, прикріплених до вузлів i and j , m — сума всіх ваг у мережі, C_i і C_j — спільноти вузлів i та j , а δ — кронекерівський дельта, який дорівнює 1, якщо i і j знаходяться в одній спільноті, і 0 в іншому випадку.

Для перевірки кластеризації можна використати коефіцієнт силуету, який розраховується як:

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}},$$

де $a(i)$ — середня відстань між i та всіма іншими точками у кластері, до якого належить i , а $b(i)$ — мінімальна середня відстань від i до всіх точок у будь-якому іншому кластері, членом якого i не є.

вимоги, але володіє добре структурованою архітектурою, що сприяє легкій масштабованості для вирішення завдань. Варто зазначити, що існуючі бібліотеки, такі як Gephi або GraphViz, можуть частково виконувати деякі вимоги, проте вони не забезпечують повністю задовільних рішень для конкретного контексту інтеграції з Flutter.

Після проведеного аналізу було отримано мережу, що містить 71 вузлів і 151. Зображення цієї мережі наведено на Рис. 3.

ВИСНОВКИ

У цій роботі представлено методологію формування та аналізу мереж подій у новинних Telegram-каналах за допомогою генеративного штучного інтелекту (GenAI).

Переваги методології:

- **Автоматизація:** Забезпечує автоматизоване виявлення, ідентифікацію та аналіз подій.
- **Ефективність:** Дозволяє досліджувати динаміку новинних потоків у реальному часі.
- **Гнучкість:** Може бути застосована до будь-якої тематики та будь-яких новинних Telegram-каналів.
- **Нові знання:** Дозволяє отримувати нові знання про динаміку та причинно-наслідкові зв'язки між подіями.

Результати дослідження:

- Розроблено алгоритми для автоматизованого формування мереж подій з текстів новинних повідомлень.
- Запропоновано методи візуалізації та аналізу мереж подій.
- Продемонстровано приклад застосування методології до аналізу новин, пов'язаних з діяльністю УПЦ.
- Розроблений API на мові python, та клієнтська частина на flutter, яка є багатоплатформною і може бути розгорнута як сайт, так і як програма для ОС Windows/Linux.

Запропонована методологія відкриває нові можливості для дослідження новинних потоків у Telegram. Це може бути корисно для:

- Моніторингу новин та виявлення нових тенденцій.
- Аналізу політичних та соціальних подій.
- Розслідування злочинів та розкриття шахрайства.

- Вивчення історії та розвитку суспільства.

Перспективи розвитку:

- Розробка інструментів для візуалізації та аналізу мереж подій в реальному часі.
- Застосування методів машинного навчання для автоматичного аналізу мереж подій.
- Розширення можливостей GenAI для виявлення інших типів зв'язків між подіями.

У виконанні даної роботи було досягнуто наукової новизни за допомогою розробки методології формування та аналізу мереж подій у новинних каналах Telegram. Ця методологія передбачала використання генеративного штучного інтелекту (GenAI) для автоматизованого виявлення, ідентифікації та аналізу подій у текстах новинних повідомлень. Результатом було розроблення алгоритмів та програмного забезпечення для виявлення нових подій, аналізу їх причинно-наслідкових зв'язків та візуалізації мережі подій у графічному форматі. Такий підхід виявився ефективним у вивченні динаміки новинних потоків та виявленні нових тенденцій, роблячи його цінним інструментом для дослідження політичних, соціальних подій та інших сфер.

Таким чином ми виконали наступні завдання:

1. Збір даних, використовуючи системи пошуку новин.
2. Виявлення подій у текстах новинних повідомлень за допомогою GenAI.
3. Ідентифікацію оригінальних подій та вилучення дублікатів.
4. Встановлення причинно-наслідкових зв'язків між подіями.
5. Аналіз та візуалізацію мережі подій.
6. Створення інтерактивних семантичних карт.

7. Розроблення своєї API з кросплатформною клієнтською частиною.

Запропонована методологія може стати цінним інструментом для дослідників, журналістів, аналітиків та всіх, хто цікавиться вивченням новинних потоків в Telegram.

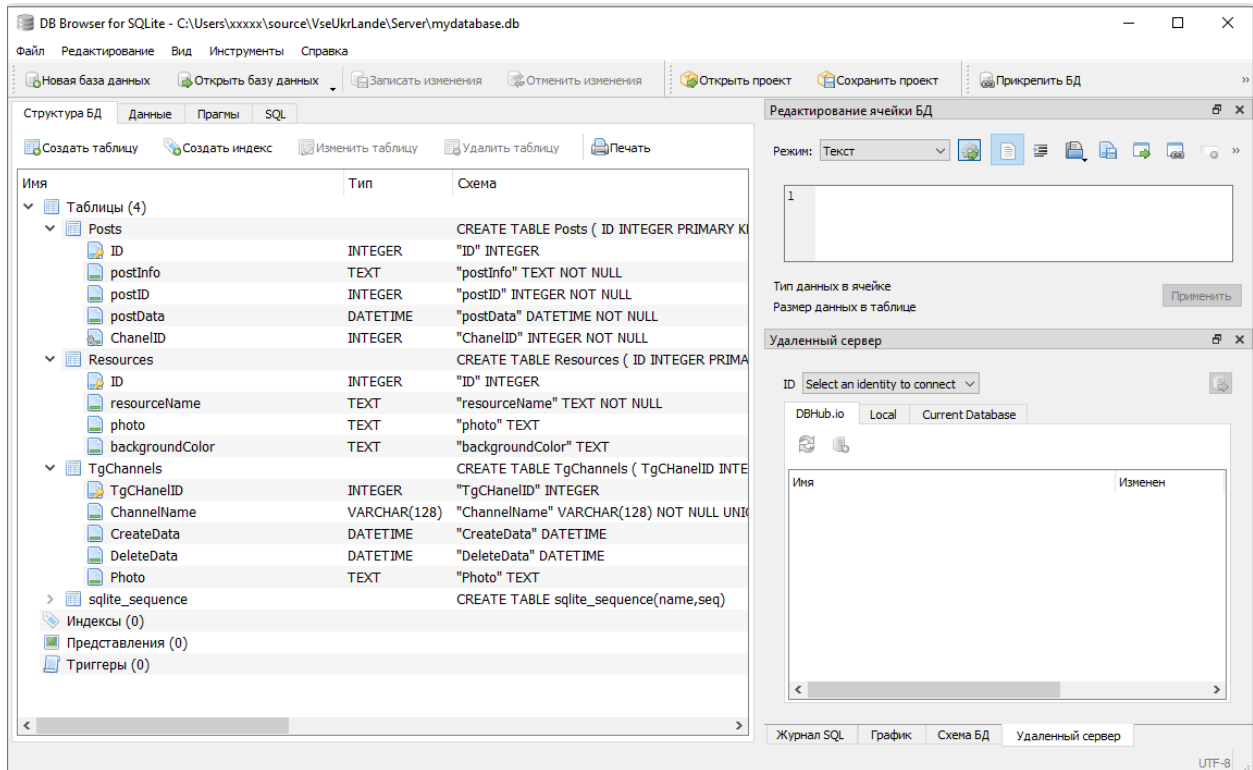
СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. S. Karimi, A. Shakery, and R. M. Verma, "Enhancement of Twitter event detection using news streams," *Natural Language Engineering*, vol. 29, no. 2, pp. 181–200, Mar. 2023, doi: <https://doi.org/10.1017/S1351324921000462>.
2. A. Zubiaga, A. Aker, K. Bontcheva, M. Liakata, and R. Procter, "Detection and resolution of rumours in social media: A survey," *ACM Computing Surveys*, vol. 51, no. 2, Art. no. 32, Feb. 2018, doi: 10.1145/3161603.
3. D. Zimbra, A. Abbasi, D. Zeng, and H. Chen, "The state-of-the-art in twitter sentiment analysis: A review and benchmark evaluation," *ACM Transactions on Management Information Systems*, vol. 9, no. 2, Art. no. 5, Apr. 2018, doi: 10.1145/3185045.
4. A. Kruspe, J. Kersten, and F. Klan, "Review article: Detection of actionable tweets in crisis events," *Natural Hazards and Earth System Sciences*, vol. 21, no. 6, pp. 1825–1845, Jun. 15, 2021, doi: 10.5194/nhess-21-1825-2021.
5. W. Tao and Y. Peng, "Differentiation and unity: A Cross-platform Comparison Analysis of Online Posts' Semantics of the Russian–Ukrainian War Based on Weibo and Twitter," *Communication and the Public*, vol. 8, no. 2, pp. 105–124, Jun. 2023, doi: 10.1177/20570473231165563.
6. D. Lande and L. Strashnoy, *GPT Semantic Networking: A Dream of the Semantic Web - The Time is Now*. Kyiv: Engineering, 2023, p. 168, ISBN: 978-966-2344-94-3. Available: <https://ssrn.com/abstract=4541673> or <http://dx.doi.org/10.2139/ssrn.4541673>.
7. D. Küçük and Fazli C.A.N., "Stance detection: A survey," *ACM Computing Surveys*, vol. 53, no. 1, Art. no. 12, Feb. 2020, doi: 10.1145/3369026.
8. W. K. Cheng et al., "A Review of Sentiment, Semantic and Event-Extraction-Based Approaches in Stock Forecasting," *Mathematics*, vol. 10, no. 14, Art. no. 2437, Jul. 2022, doi: 10.3390/math10142437.

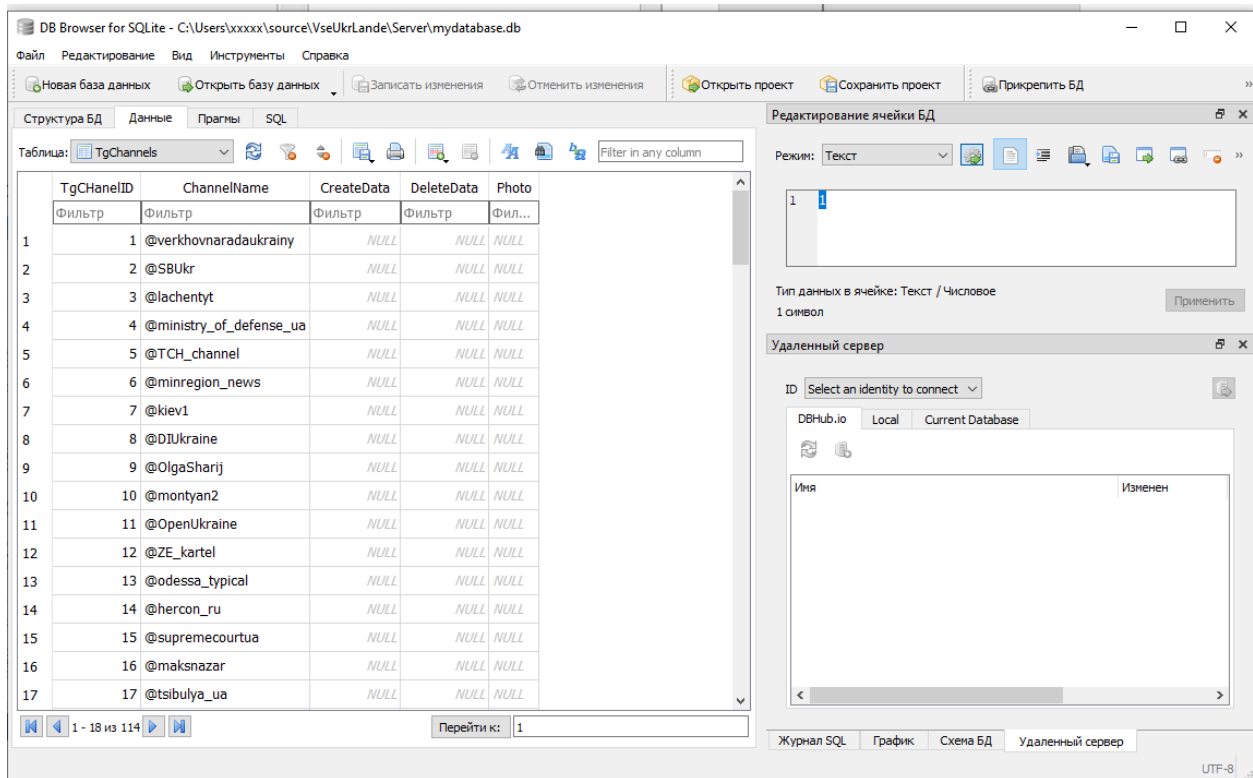
9. G. Weikum, X. L. Dong, S. Razniewski, and F. Suchanek, "Machine knowledge: Creation and curation of comprehensive knowledge bases," *Foundations and Trends in Web Science*, ISSN: 19317883, doi: 10.1561/19000000064.
10. C. Di Crescenzo et al., "HERMEVENT: A news collection for emerging-event detection," in *Proc. 17th Int. Conf. Web Intelligence, Mining and Semantics (WIMS 2017)*, Amantea, Italy, Jun. 19-22, 2017, *ACM Int. Conf. Proc. Series*, Part F129475, Art. no. a117, doi: 10.1145/3102254.3102262.
11. Z. Akhgari, M. Malekimajd, and H. Rahmani, "Sem-TED: Semantic Twitter Event Detection and Adapting with News Stories," in *Proc. 2022 8th Int. Conf. on Web Research (ICWR 2022)*, Tehran, Iran, May 11-12, 2022, pp. 61-69, doi: 10.1109/ICWR54782.2022.9786234.
12. A. V., G. Deepak, and A. Santhanavijayan, "HCODF: Hybrid Cognitive Ontology Driven Framework for Socially Relevant News Validation," in *Lecture Notes in Networks and Systems*, vol. 211 LNNS, pp. 731-739, 2021 1st Int. Conf. on Digital Technologies and Applications (ICDTA 2021), Fez, Morocco, Jan. 29-30, 2021, doi: 10.1007/978-3-030-73882-2_66.
13. P. Khare, P. Torres, and B. R. Heravi, "What just happened? A framework for social event detection and contextualisation," in *Proc. Annual Hawaii Int. Conf. on System Sciences*, vol. 2015-March, pp. 1565-1574, 48th Annual Hawaii Int. Conf. on System Sciences (HICSS 2015), Kauai, Jan. 5-8, 2015, doi: 10.1109/HICSS.2015.190.
14. K. Cherven, *Mastering Gephi Network Visualization*. Packt Publishing, 2015, 378.
15. L. M. Surhone, M. T. Tennoe, and S. F. Henssonow, *Graphviz*. VDM Publishing, 2010, p. 108.

ДОДАТОК А. Збирання і обробка даних із соціальної мережі Telegram на API

Скріншот: «Структура БД»



Скріншот: «Записані в БД канали»



Скріншот: «Запуск Flask серверу»

Код функції додавання каналів до БД:

```

@add_channel_bp.route('/add_channel', methods=['POST'])
def add_channel():
    # Обробка отриманої інформації від клієнтського застосунку
    ChannelName = request.json.get('channel')
    if ChannelName is None:
        return jsonify({'message': 'Missing channel name'}), 400

    CreateData = request.json.get('createData')
    DeleteData = request.json.get('deleteData')
    Photo = request.json.get('photo')

    # Підключення до бази даних
    IDb = get_db()
    error = None

    if not ChannelName:
        error = "ChannelName is already added to list."

    try:
        # Виконання SQL-запиту для вставки даних
        get_db().execute(
            "INSERT INTO TgChannels (ChannelName, CreateData, DeleteData, Photo) VALUES (?, ?,
?, ?)",
            (ChannelName, CreateData, DeleteData, Photo)
        )
        # Збереження змін у базі даних
        IDb.commit()
    except IDb.IntegrityError:
        # Обробка випадку, коли вже існує канал з таким ім'ям
        IDb.rollback()
        return jsonify({'message': 'Channel already exists'}), 409

    else:
        # Success, add chanel.
        return jsonify({'message': 'Channel added successfully'}), 201

```

Код функції видалення каналів з БД:

```

@delete_channel_bp.route('/delete_channel', methods=['DELETE'])
def delete_channel():
    channel_name = request.json.get('channel')
    if channel_name is None:
        return jsonify({'message': 'Missing channel name'}), 400

    try:
        # Підключення до бази даних
        db = get_db()

        # Виконання SQL-запиту для видалення каналу
        db.execute("DELETE FROM TgChannels WHERE ChannelName = ?", (channel_name,))
        db.commit()

        # Перевірка, чи було видалено хоча б один запис
        if db.rowcount > 0:
            return jsonify({'message': f'Channel {channel_name} deleted successfully'}), 200
        else:
            return jsonify({'message': f'Channel {channel_name} not found'}), 404

    except Exception as e:
        # Обробка винятку
        return jsonify({'message': f'Error: {str(e)}'}), 500

```

Код функції відправлення каналів клієнтському застосунку:

```

@get_channels_bp.route('/get_channels', methods=['POST'])
def get_channels():
    # Підключення до бази даних
    db = get_db()

    try:
        # Виконання SQL-запиту для отримання всіх каналів
        cursor = db.execute("SELECT * FROM TgChannels")
        channels = cursor.fetchall()
        # Створення списку каналів у форматі словників
        channel_list = []
        for channel in channels:
            printChanelInfo(channel)
            channel_dict = {
                'id': channel[0],
                'channelName': channel[1],
                'createData': channel[2],
                'deleteData': channel[3],
                'photo': channel[4]
            }
            channel_list.append(channel_dict)

        # Відправка даних про канали у форматі JSON
        return jsonify({'channels': channel_list}), 200

    except Exception as e:
        # Обробка винятку
        return jsonify({'message': f'Error: {str(e)}'}), 500

```

Код функції пошуку постів по запиту від клієнтської частини:

```

@search_posts_bp.route('/search_posts', methods=['POST'])
async def search_posts():
    data = request.get_json()
    channel_ids = data.get('channel_ids')
    query = data.get('query')
    start_date_str = data.get('start_date')
    end_date_str = data.get('end_date')
    sources = data.get('sources')
    limit = data.get('limit')

    if not all([channel_ids, query, start_date_str, end_date_str]):
        return jsonify({'message': 'Missing data for search'}), 400
    try:
        # Конвертація стрічки дати у об'єкт datetime
        start_date = datetime.strptime(start_date_str[:-1], '%Y-%m-
%dT%H:%M:%S.%f').replace(tzinfo=timezone.utc)
        end_date = datetime.strptime(end_date_str[:-1], '%Y-%m-
%dT%H:%M:%S.%f').replace(tzinfo=timezone.utc)

        print(start_date) # Виведе дату у форматі: 2024-03-01 00:00:00
        print(end_date) # Виведе дату у форматі: 2024-03-01 00:00:00

        posts_to_return = []

        # Запускаємо функцію і отримуємо повідомлення
        posts_to_return = await search_messages_by_channels(channel_ids, query, limit, start_date,
end_date)

        return jsonify({'posts': posts_to_return}), 200
    except Exception as e:
        # Обробка винятку
        return jsonify({'message': f'Error: {str(e)}'}), 500

```

Код функції мульти запитів до ChatGPT:

```

class MultiTalkGPT:
    def __init__(self, num, chrome_path=r"C:\Program
Files\Google\Chrome\Application\chrome.exe"):
        self.url = 'https://chat.openai.com/'
        self.chrome_path = chrome_path
        self.speakers = []
        self.driver = None

    def open_chrome(self, port):
        user_data_dir = f"remote-profile-{port}"
        chrome_cmd = f"{self.chrome_path} --remote-debugging-port={port} --user-data-
dir={user_data_dir} {self.url}"
        os.system(chrome_cmd)

    def switch_driver(self, port):
        chrome_options = webdriver.ChromeOptions()
        chrome_options.add_experimental_option("debuggerAddress", f"127.0.0.1:{port}")
        self.driver = webdriver.Chrome(options=chrome_options)

    def wait_for_login(self):
        response = ""
        while response.lower() != 'y':
            response = input("Please enter 'y' if you logged in: ")

    def start_windows(self, num):
        for i in range(num):
            port = 5801 + i
            print(port)
            threading.Thread(target=self.open_chrome, args=(port,)).start()
            time.sleep(1) # Adjust sleep duration as needed
            self.wait_for_login()
            self.speakers.append(port)

    async def ask(self, id, prompt, delay):
        print(f'Ask: {id} {prompt}')
        self.switch_driver(self.speakers[id])
        input_box = self.driver.find_element(by=By.XPATH, value=//textarea[contains(@id, "prompt-
textarea")])
        self.driver.execute_script(f"arguments[0].value = '{prompt}';", input_box)
        input_box.send_keys(Keys.RETURN)
        input_box.submit()
        await asyncio.sleep(delay)
        self.switch_driver(self.speakers[id])
        response_elements = self.driver.find_elements(by=By.CSS_SELECTOR, value='div.text-base')
        return response_elements[-1].text

    async def process_batch(self, prompts, delay):
        print(f'processing batch')
        tasks = [gpt.ask(i, prompts[i], delay) for i in range(min(len(prompts), len(self.speakers)))]
        print(tasks)
        return await asyncio.gather(*tasks)

    async def answer_async(self, prompts, delay):
        responses = []
        batch_size = len(self.speakers)

```

```

for i in range((len(prompts)+batch_size-1)//batch_size):
    try:
        curr_prompts = prompts[batch_size*i:min(batch_size*(i+1), len(prompts))]
        responses += await self.proccess_batch(curr_prompts, delay)
    except e:
        print(e)
return responses

def return_last_response(self, id):
    response_elements = self.drivers[id].find_elements(by=By.CSS_SELECTOR, value='div.text-
base')
return response_elements[-1].text

```

Функція перевірки валідності каналу:

```

async def check_channel_validity(chat_id):
    async with tg_client :
        try:
            chat = await tg_client.get_chat(chat_id)
            return True, chat.title # Повертаємо True і назву каналу, якщо канал існує
        except errors.ChannelInvalid:
            return False, "ChannelInvalid: The channel specified is invalid"
        except errors.ChannelPrivate:
            return False, "ChannelPrivate: The channel specified is private and you're not a member"
        except errors.UsernameInvalid:
            return False, "UsernameInvalid: The username specified is invalid"

```

Функція пошуку інформації по Telegram каналам:

```

async def search_messages_by_channels(channels_id, query, limit, start_date, end_date):
    channels_messages_data = []
    for chanel_id in channels_id:
        # result = loop.run_until_complete(check_channel_validity(chanel_id))
        result = await check_channel_validity(chanel_id)
        try:
            valid, title_or_error = result # wait for 10 seconds, you can change this as needed
            # return jsonify({'valid': valid, 'title': title_or_error}), 200 if valid else 400

            if not valid:
                print(f"Error checking channel '{chanel_id}': {title_or_error}")
                continue
            chanel_messages_data = await search_messages_by_info(chanel_id, query, limit, start_date,
end_date, True)
            channels_messages_data.append(chanel_messages_data)

        except asyncio.TimeoutError:
            print(f"Timeout occurred!")
    return channels_messages_data

```

Функція пошуку інформації у Telegram каналі:

```

async def search_messages_by_info(chat_id, query, limit, start_date, end_date, isChecked):
    messages_data = []

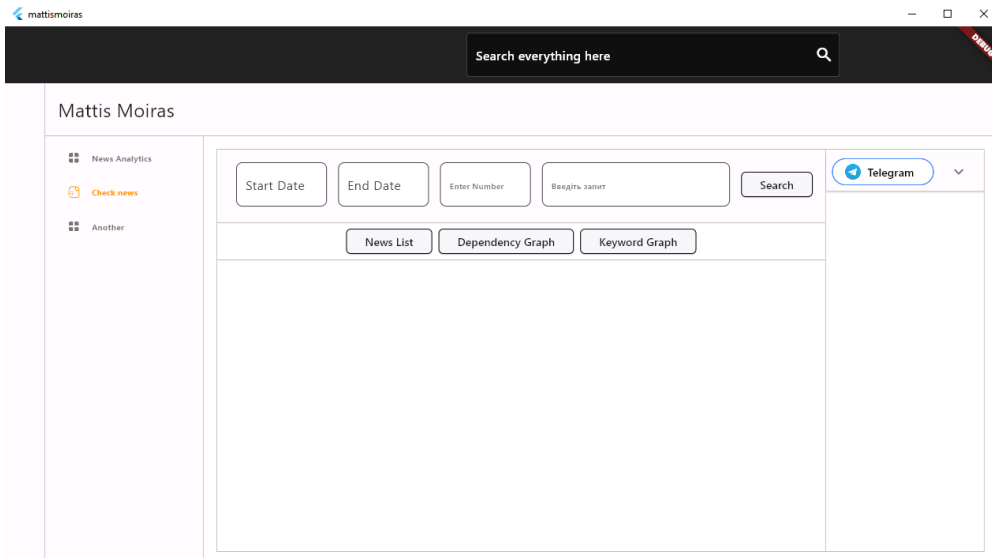
    async with tg_client:
        try:
            async for message in tg_client.search_messages(chat_id, query=query):
                message_date = message.date.astimezone(datetime.timezone.utc)
                if start_date <= message_date <= end_date:
                    if message.caption != None:
                        message_data = {
                            "channel": chat_id,
                            "messageID": message.id,
                            "text": message.caption,
                            "pub_date": message_date.strftime('%Y-%m-%d %H:%M:%S')
                        }
                        print(message_data)
                        messages_data.append(message_data)
                        if len(messages_data) >= limit: # перевіряємо, чи досягли ми ліміту
                            break # виходимо з циклу, якщо досягли ліміту
        except errors.FloodWait as e:
            print(f"Need to wait {e.x} seconds before continuing")
            await asyncio.sleep(e.x)
        except Exception as e:
            print(f"An error occurred: {e}")

    # обрізаємо список повідомлень, якщо він довший за вказаний ліміт
    return messages_data[:limit]

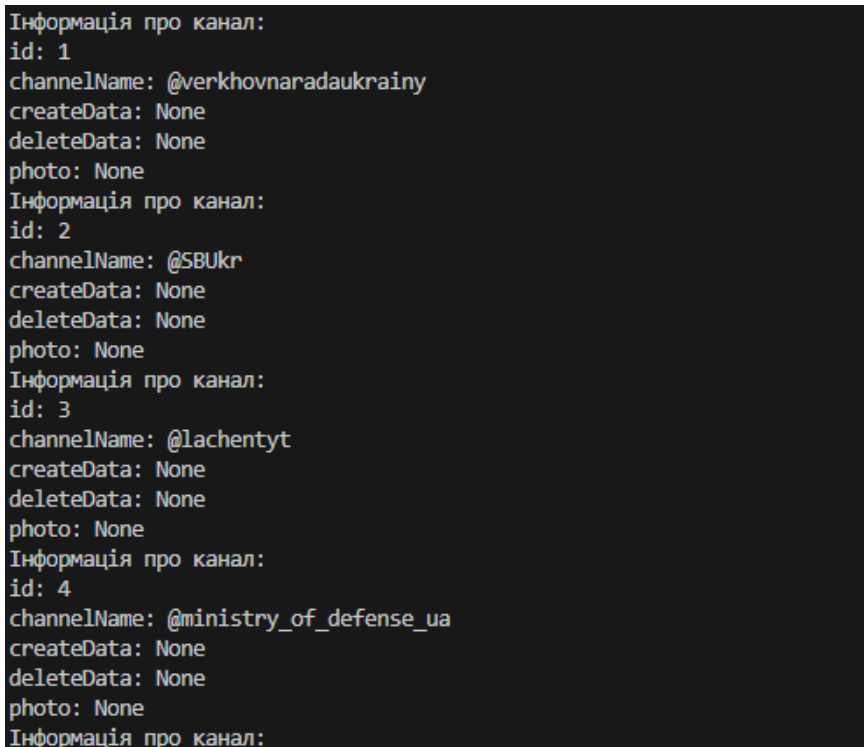
```


ДОДАТОК В. Клієнтська частина

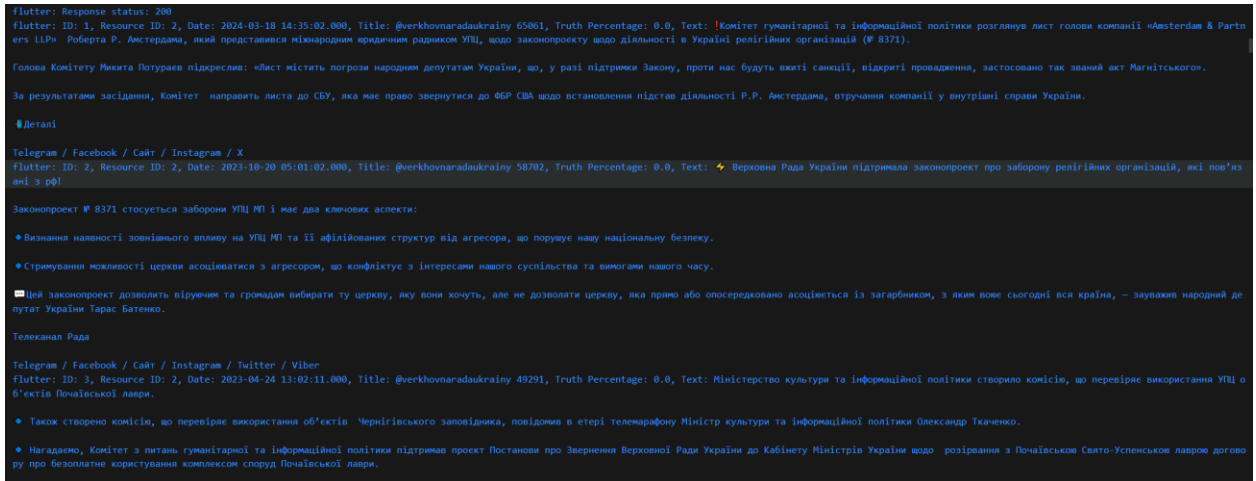
Скріншот: «Початковий екран програми “Mattis Moiras”»



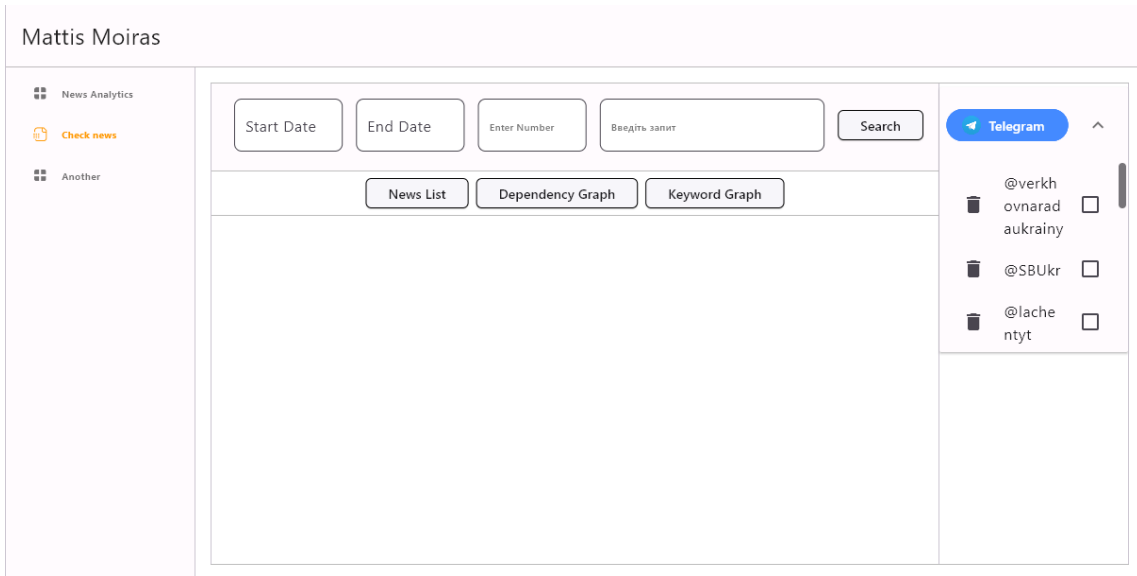
Скріншот: «Логи клієнтської програми при отриманні списку каналів»



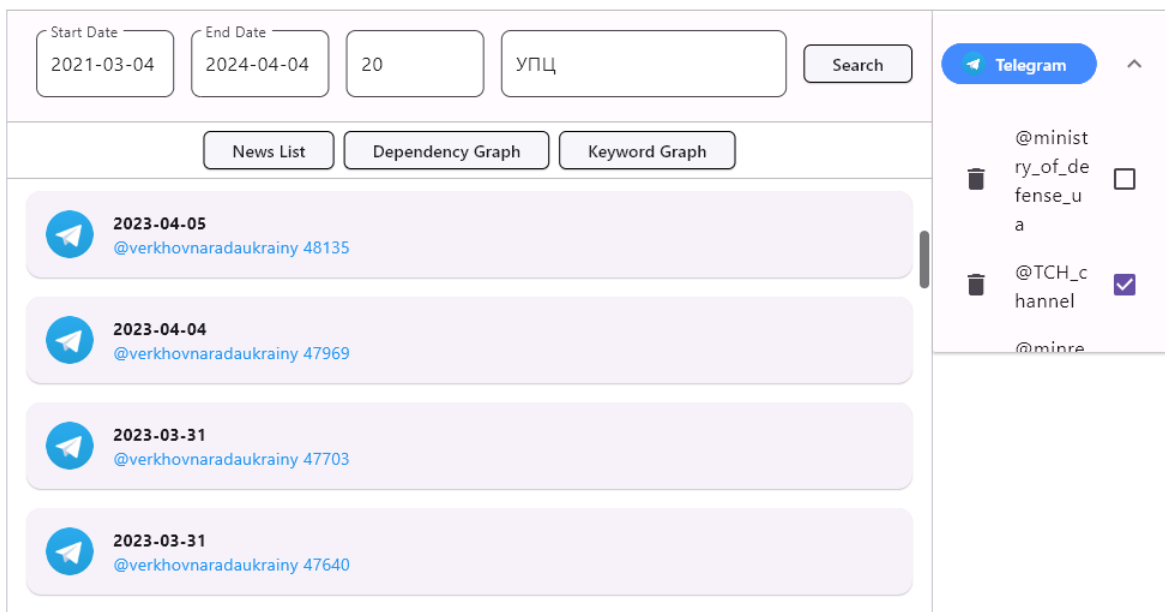
Скріншот: «Логування обробленої інформації від серверу на клієнтській програмі»



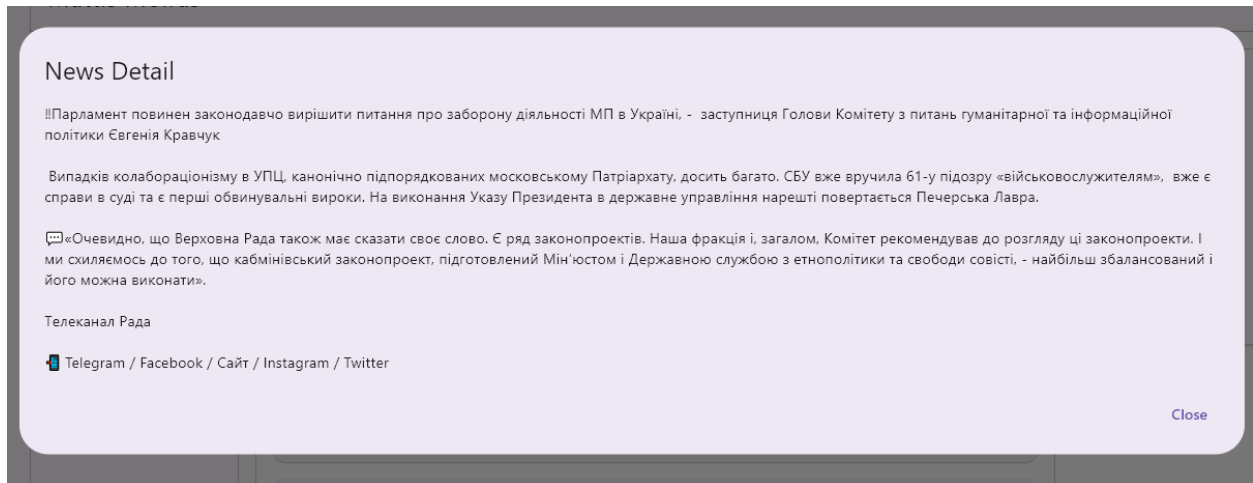
Скріншот: «Відображення списку каналів з телеграму»



Скріншот: «Відображення отриманих даних від серверу»



Скріншот: «Відображення тексту «події»»



Код основного класу відображення інтерфейсу програми

```
class MainPage extends StatefulWidget {
  @override
  _MainPageState createState() => _MainPageState();
}

class _MainPageState extends State<MainPage> {

  late Widget _selectedWidget;
  late Size screenSize;

  Future<void> _initialize() async {
    await loadDataAsync(context);
    _selectedWidget = _getWidgetForIndex(0);
  }

  @override
  void initState()
  {
    super.initState();
    _selectedWidget = _getWidgetForIndex(0);
    // Викликати асинхронну функцію завантаження даних.
    loadDataAsync(context);
  }

  CustomButtonColumn getCustomButtonColumn()
  {
    return CustomButtonColumn(
      customImageButtonList: [
        CustomImageButton(
          iconPath: 'lib/UI/Resources/HomePage/SideBar/News analytics_selected.png',
          label: 'News Analytics',
        ),
        CustomImageButton(
          iconPath: 'lib/UI/Resources/HomePage/SideBar/Breaking news.png',
          label: 'Check news',
        ),
        CustomImageButton(
          iconPath: 'lib/UI/Resources/HomePage/SideBar/News analytics_selected.png',
          label: 'Another',
        ),
      ],
    );
  }
}
```

```

    ),
  ],
  defaultColor: Colors.black54,
  selectedColor: Colors.orange,
  onTap: (int index) {
    setState(() {
      _selectedWidget = _getWidgetForIndex(index);
    });
  },
);
}

Row getSearchRow() {

return Row(
  children: [
    SizedBox(width: screenSize.width * 0.45), // Right side padding
    const Expanded(
      child: TextField(
        style: TextStyle(color: Colors.white), // Set text color to white
        decoration: InputDecoration(
          border: OutlineInputBorder(),
          hintText: 'Search everything here',
          suffixIcon: Icon(Icons.search, color: Colors.white),
          hintStyle: TextStyle(color: Colors.white),
          filled: true,
          fillColor: Colors.black54, // If you want the TextField to have a slightly lighter background
        ),
      ),
    ),
    ),
    SizedBox(width: screenSize.width * 0.15), // Right side padding
  ],
);
}

Expanded getSelectedWidget()
{
return Expanded(
  child: Container(
    padding: const EdgeInsets.all(16.0),
    color: Colors.white, // Background color for the content area
    child: _selectedWidget, // Display the selected widget here
  ),
);
}

Padding buildProgramName(String programName)
{
return Padding(
  padding: EdgeInsets.all(16.0),
  child: Text(
    programName,
    style: TextStyle(
      fontSize: 24, // Font size for the name
      color: Colors.black87, // Text color for the name
    ),
  ),
);
}

Expanded buildExpanded()

```

```

{
  return Expanded(
    child: Row(
      children: [
        Container(
          // width: 200,
          // height: 200,
          child: Column(
            crossAxisAlignment: CrossAxisAlignment.start,
            children: [
              getCustomButtonColumn(), // This should now align the buttons at the top
            ],
          ),
        ),
        const VerticalDivider(width: 1),
        getSelectedWidget(),
      ],
    ),
  );
}

```

```

@override
Widget build(BuildContext context) {
  screenSize = MediaQuery.of(context).size;
  return Scaffold(
    body: Column(
      children: [
        Container(
          padding: const EdgeInsets.symmetric(horizontal: 16.0, vertical: 8.0),
          color: Colors.black87,
          child: getSearchRow(),
        ),
        Expanded(
          child: Row(
            children: [
              const SizedBox(width: 50), // Left side padding
              const VerticalDivider(width: 1),
              Expanded(
                child: Column(
                  crossAxisAlignment: CrossAxisAlignment.start,
                  children: [
                    buildProgramName("Mattis Moiras"),
                    const HorizontalDivider(thickness: 1),
                    buildExpanded(),
                  ],
                ),
              ),
              const VerticalDivider(width: 1),
              const SizedBox(width: 5), // Right side padding
            ],
          ),
        ),
      ],
    ),
  );
}

```

```

Widget _getWidgetForIndex(int index) {
  // Return the widget that corresponds to the selected index
  switch (index) {
    case 0:
      return AnalyticsReport();
  }
}

```

```

case 1:
    return BreakingNews();
default:
    return const Text("Unknown");
}
}
}

```

Клас відображення запиту на клієнтській частині:

```

class BreakingNews extends StatelessWidget {

    Consumer<ToggleButtonWidgetState> chooseNewsByInfoRow(final submissionProvider, final
newsInfoProvider, Widget chartWidget)
    {
        return Consumer<ToggleButtonWidgetState>(
            builder: (context, toggleState, child) {
                // Use startDate and endDate from SubmissionProvider to determine the timeframe
                DateTime startDate = submissionProvider.startDate ?? DateTime.now().subtract(Duration(days:
1));
                DateTime endDate = submissionProvider.endDate ?? DateTime.now();

                // Retrieve the data for the chart based on the dates
                List<NewsCount> aggregatedData =
newsInfoProvider.getAggregatedDataForTimeframe(startDate, endDate);

                print(aggregatedData); // For debugging
                switch (toggleState.currentView) {
                    case ViewType.newsList:
                        return NewsList();
                    case ViewType.dependencyGraph:
                        return Scaffold(
                            body: InteractiveViewer(
                                boundaryMargin: EdgeInsets.all(80),
                                minScale: 0.01,
                                maxScale: 5.6,
                                child: buildDirectGraphPainterByMap(getDirectPairNews()),
                            ),
                        );
                    case ViewType.keywordGraph:
                        return Scaffold(
                            body: InteractiveViewer(
                                boundaryMargin: EdgeInsets.all(80),
                                minScale: 0.01,
                                maxScale: 5.6,
                                child: buildGraphPainterByMap(getDirectPairNews()),
                            ),
                        );
                    default:
                        return SizedBox.shrink(); // Or some default widget
                }
            },
        );
    }

    Container resourcesList(SubmissionProvider submissionProvider)
    {

```

```

return Container(
  width: 200, // The sidebar's width
  // height: 200,
  child: ListView.separated(
    itemCount: submissionProvider.resources.length,
    itemBuilder: (context, index) {
      if(submissionProvider.resources[index].isExpansionPanel)
      {
        return Padding(
          padding: const EdgeInsets.symmetric(vertical: 4.0),
          child:
            ResourceWidget(resource: submissionProvider.resources[index]),
        );
      }
      else{
        return Padding(
          padding: const EdgeInsets.symmetric(vertical: 4.0),
          child:
            ResourceWidget(resource: submissionProvider.resources[index]),
        );
      }
    },
    separatorBuilder: (context, index) => const Divider(),
  ),
);
}

@override
Widget build(BuildContext context) {
  final submissionProvider = Provider.of<SubmissionProvider>(context);
  final newsInfoProvider = Provider.of<NewsInfoProvider>(context, listen: false);

  // Use startDate and endDate from SubmissionProvider to determine the timeframe
  DateTime startDate = submissionProvider.startDate ?? DateTime.now().subtract(const
Duration(days: 1));
  DateTime endDate = submissionProvider.endDate ?? DateTime.now();

  // Retrieve the data for the chart based on the dates
  List<NewsCount> aggregatedData = newsInfoProvider.getAggregatedDataForTimeframe(startDate,
endDate);

  // Build the TimeSeriesChartWidget with the aggregated data

  Widget chartWidget = TimeSeriesChartWidget(
    aggregatedData: aggregatedData,
    resources: submissionProvider.resources,
  );

  return Container(
    // Dummy container, replace with your implementation
    child: Column(
      children: [
        const HorizontalDivider(thickness: 1), // Горизонтальна лінія зверху
        Expanded(
          child: Row(
            children: [
              const VerticalDivider(width: 1),
              Expanded(
                child: Column(
                  mainAxisAlignment: MainAxisAlignment.start, // Align items to the start of the columnz

```

```

        children: [
          Expanded(
            child: SearchArticlesScreen(), // Your search bar at the top
            flex: 2, // Adjust flex factor if necessary to allocate space efficiently
          ),
          // SizedBox(height: 0), // Minimal space between widgets
          const HorizontalDivider(thickness: 1), // Горизонтальна лінія зверху
          Expanded(
            child: ToggleButtonWidget(),
            flex: 1,
          ),
          const HorizontalDivider(thickness: 1), // Горизонтальна лінія зверху
          Expanded(
            child: chooseNewsByInfoRow(submissionProvider, newsInfoProvider, chartWidget), //
Content of news or chart
            flex: 8, // Adjust flex factor if necessary
          ),
        ],
      ),
    ),
    const VerticalDivider(width: 1), // This visually separates the sidebar from the main content
    resourcesList(submissionProvider), // Список ресурсів
    const VerticalDivider(width: 1),
  ],
),
),
const HorizontalDivider(thickness: 1), // Горизонтальна лінія знизу
],
),
);
}
}

```


АНОТАЦІЯ

наукової роботи під шифром: “Перебіг подій”

Актуальність роботи.

В умовах інформаційної війни та зростання ролі Telegram як джерела новин, виникає гостра потреба в ефективних методах аналізу та моніторингу новинних потоків у цій платформі. Telegram здобуває все більшу популярність, а з нею зростає і ризик поширення дезінформації, пропаганди та антидержавної риторики. Традиційні методи аналізу даних не справляються з обсягом інформації, що генерується в Telegram, тому потрібні нові підходи, які б дозволили дослідникам, журналістам, аналітикам та державним органам отримувати знання про динаміку новинних потоків, виявляти нові тенденції, досліджувати причинно-наслідкові зв'язки між подіями та боротися з дезінформацією.

Мета роботи – розробка методології та інструментальних засобів для автоматизованого формування та аналізу мереж подій у новинних Telegram-каналах, з акцентом на виявлення та дослідження антидержавної риторики УПЦ в Україні.

Завдання полягає у розв'язанні часткових поставлених задач:

1. Розробка алгоритмів для автоматизованого виявлення та категоризації подій з текстів новинних повідомлень.
2. Створення методів для формування та візуалізації мереж подій, що враховують хронологію, типи подій, та ступінь їхнього впливу.
3. Розробка прототипу інформаційно-аналітичної системи, що використовує моніторинг Telegram-каналів УПЦ в Україні для:
 - Виявлення та ранжування новинних повідомлень, що містять антидержавну риторику.
 - Автоматизованого аналізу та візуалізації динаміки та причинно-наслідкових зв'язків подій.

Результати роботи можна використовувати для автоматизованого аналізу новинних потоків у Telegram, виявлення дезінформації, пропаганди та антидержавної риторики, а також дослідження соціальних процесів на цій платформі. Розроблені алгоритми та методи автоматизують виявлення подій з текстів та формування мереж подій. Прототип системи можна застосувати для боротьби з іншими інформаційними загрозами, а результати допоможуть удосконалити інструменти аналізу даних.

Наукова новизна:

1. Розроблена методологія виявлення побудови мережі і кластеризації подій на основі застосування штучного інтелекту.
2. Кластеризація подій здійснена за допомогою прихованих факторів, а саме сутностей, які відповідають саме цим подіям, що є елементами простору параметрів.

Наукова робота: 41 с., 3 рис., 15 джерел.

Ключові слова: GENAI, ВИЯВЛЕННЯ ПОДІЙ, ЕКСТРАГУВАННЯ ЗВ'ЯЗКІВ, МЕРЕЖА ПОДІЙ, КАУЗАЛЬНА МЕРЕЖА, ВІЗУАЛІЗАЦІЯ МЕРЕЖ, TELEGRAM, АНАЛІЗ НОВИНИХ ПОТОКІВ, МОНІТОРИНГ НОВИН, API.