МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій
(повне найменування факультету)

Кафедра «Системний аналіз та обчислювальна математика»
(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)
Магістратура
(ступінь вищої освіти)

на тему Прогнозування динаміки пошукових запитів із використанням гібридної моделі Transformer—GRU для задач SEO-оптимізації
(назва теми)

Виконав(ла): студент(ка) 2 курсу, групи КНТз-814м
Спеціальності 124 Системний аналіз
(код і найменування спеціальності)

Освітня програма (спеціалізація)
Інтелектуальні технології та прийняття рішень у складних системах
ЧОРНИЙ Ю. І.
(ПРІЗВИЩЕ та ініціали)

Керівник БАКУРОВА А.В.
(ПРІЗВИЩЕ та ініціали)

Рецензент ОЧЕРЕТІН Д. В.
(ПРІЗВИЩЕ та ініціали)

2025

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

Кафедра «Системний аналіз та обчислювальна математика»

Ступінь вищої освіти Магістр

Спеціальність 124 Системний аналіз

(код і найменування)

Освітня програма (спеціалізація) Інтелектуальні технології та прийняття рішень у складних системах

(назва освітньої програми (спеціалізації)

ЗАТВЕРДЖУЮ
Завідувач кафедри Терещенко Еліна Валентинівна
«__25_____» ___листопада_____2025 року

ЗАВДАННЯ

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

ЧОРНИЙ Юрій Іванович

(ПРІЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Прогнозування динаміки пошукових запитів із використанням гібридної моделі Transformer—GRU для задач SEO-оптимізації
керівник проєкту (роботи)
професор, д.е.н., професор кафедри системного аналізу та обчислювальної математики_БАКУРОВА Анна Володимирівна

(науковий ступінь, вчене звання, ПРІЗВИЩЕ, ім'я, по батькові)

затверджені наказом закладу вищої освіти від «10» листопада 2025 року № 506

2. Строк подання студентом проєкту (роботи) 18.12.2025

3. Вихідні дані до проєкту (роботи) Часові ряди індексу популярності ключових слів Google Trends, що належать до одного семантичного кластера (товарної або тематичної категорії)

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

- Виконати збір та підготовку даних з Google Trends (очистка, нормалізація, формування навчальної та тестової вибірок, перевірка стаціонарності).
- Проаналізувати базові моделі прогнозування часових рядів (Naïve, Seasonal Naïve, Ridge Regression з лаговими ознаками, Random Forest з лаговими ознаками та нейромережева модель LSTM) та на основі відомих з літератури властивостей порівняти їх із гібридною моделлю Transformer—GRU.
- Розробити та реалізувати гібридну модель T-GRU (Transformer—GRU з Dense-шаром) і повний програмний конвеєр її застосування у складі комплексу

TrendsAI (CLI-інтерфейс, база даних SQLite, модулі збору, підготовки даних, навчання та формування звітів).

- Реалізувати модуль автоматичного семантичного відбору ключових слів на основі моделей типу Sentence-BERT (Transformers) та інтегрувати його в програмний комплекс TrendsAI.
- Виконати оцінювання якості прогнозів за метриками RMSE, MAE, R^2, MAPE, sMAPE; проаналізувати результати та обґрунтувати доцільність використання гібридної моделі T-GRU у завданнях SEO-прогнозування.
- Підготувати пояснювальну записку, таблиці, графіки, звітні матеріали та інструкцію з відтворення експериментів.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, кількість слайдів, плакатів) Презентація 10 слайдів

6. Консультанти розділів проєкту (роботи)

Розділ	ПРІЗВИЩЕ, ініціали та посада Консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
Нормоконтроль	Широкорад Д.В.	22.12.2025	24.12.2025

7. Дата видачі завдання «01» вересня 2025 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Аналітичний огляд наукової літератури	30.06.2025	виконано
2	Збір та підготовка даних	28.08.2025	виконано
3	Побудова моделей прогнозування	25.10.2025	виконано
4	Аналіз результатів	30.11.2025	виконано
5	Оформлення роботи	18.12.2025	виконано

Студент(ка) _____ **Юрій ЧОРНИЙ**
(підпис) (Ім'я ПРІЗВИЩЕ)

Керівник проєкту (роботи) _____ **Анна БАКУРОВА**
(підпис) (Ім'я ПРІЗВИЩЕ)

РЕФЕРАТ

Дипломна робота: 96 стор., 13 рис., 13 табл., 26 джерел.

У дипломній роботі розглянуто задачу прогнозування динаміки пошукових запитів як часових рядів на основі даних сервісу Google Trends у контексті SEO-оптимізації. Актуальність дослідження зумовлена необхідністю підвищення точності прогнозів за наявності сезонності, нестаціонарності та складних часових залежностей.

У роботі наведено огляд сучасних методів прогнозування часових рядів, зокрема статистичних моделей та підходів машинного і глибинного навчання. Проаналізовано підходи до прогнозування часових рядів, зокрема статистичні моделі (ARIMA, Prophet) та методи глибинного навчання (LSTM). Для експериментального порівняння в межах програмного комплексу реалізовано базові моделі Naïve і Seasonal Naïve, а також Ridge Regression та Random Forest з лаговими ознаками і нейромережеву модель LSTM.

Описано процес збору, очищення та підготовки даних з Google Trends, включаючи нормалізацію та формування навчальних і тестових вибірок. Запропоновано гібридну модель Transformer—GRU, яка поєднує механізм уваги архітектури Transformer для врахування глобальних залежностей та можливості GRU для моделювання локальної часової динаміки. Проведено експериментальну оцінку ефективності запропонованої моделі на реальних даних пошукових запитів.

Якість прогнозів оцінювалася за метриками RMSE, MAE, R^2, MAPE та sMAPE. Отримані результати підтверджують доцільність використання гібридної моделі Transformer—GRU для задач SEO-оптимізації.

ЧАСОВІ РЯДИ, ПРОГНОЗУВАННЯ, GOOGLE TRENDS, МАШИННЕ НАВЧАННЯ, ГЛИБИННЕ НАВЧАННЯ, НЕЙРОННІ МЕРЕЖІ, TRANSFORMER, GRU, LSTM, ARIMA, PROPHET, SEO, ЦИФРОВИЙ МАРКЕТИНГ

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

MAE — Mean Absolute Error — середня абсолютна похибка.

RMSE — Root Mean Squared Error — середньоквадратична похибка.

MAPE — Mean Absolute Percentage Error — середня відносна абсолютна похибка.

sMAPE — Symmetric Mean Absolute Percentage Error — симетрична середня відносна абсолютна похибка.

R^2 — коефіцієнт детермінації.

DM-тест — тест Дієболда—Маріано для порівняння точності прогнозів.

T-GRU — гібридна модель Transformer—GRU.

ARIMA — AutoRegressive Integrated Moving Average — авторегресійна інтегрована модель ковзного середнього.

LSTM — Long Short-Term Memory — рекурентна нейромережева архітектура з довгою короткочасною пам’яттю.

GRU — Gated Recurrent Unit — рекурентна нейромережева архітектура з керованими вентилями.

SEO — Search Engine Optimization — пошукова оптимізація.

SBERT — Sentence-BERT — модель для семантичного порівняння текстів.

Embedding — векторне подання тексту (sentence embedding).

Cosine similarity — косинусна подібність.

SQLite — вбудована реляційна база даних.

RobustScaler — робастний метод масштабування (за медіаною та IQR).

Huber Loss — робастна функція втрат для регресії.

Adam — метод стохастичної оптимізації.

Early Stopping — метод зупинки навчання для запобігання перенавчанню.

TensorFlow — фреймворк для глибинного навчання.

Keras — високорівневий API для побудови нейромереж.

ВСТУП

Сучасний розвиток цифрових технологій та інтернет-маркетингу зумовлює потребу у використанні інтелектуальних методів аналізу та прогнозування динаміки пошукових запитів. Інформаційна насиченість ринку, мінливість поведінки споживачів і сезонність попиту вимагають побудови моделей, здатних оперативно реагувати на зміни трендів. Саме тому прогнозування часових рядів, що відображають інтерес користувачів у пошукових системах, набуває особливого значення для ефективного планування SEO-стратегій та управління трафіком [1].

Традиційні статистичні підходи, зокрема моделі ARIMA [2], демонструють обмежену ефективність у випадках, коли часові ряди мають складну структуру залежностей або проявляють нестаціонарність. З іншого боку, нейронні мережі типу LSTM [3] і GRU [4], здатні враховувати нелінійні взаємозв’язки, але поступаються у врахуванні довгострокових залежностей. Застосування архітектури Transformer [5], що ґрунтується на механізмі уваги, відкриває нові можливості у сфері прогнозування часових рядів, забезпечуючи гнучке моделювання контекстних зв’язків.

У зв’язку з цим актуальним є дослідження гібридних підходів, які поєднують переваги рекурентних мереж та трансформерів. Одним із таких рішень є модель Transformer—GRU (T-GRU), що поєднує здатність GRU моделювати короткострокові залежності з ефективністю механізму уваги у виявленні глобальних трендів.

Об’єкт дослідження — процес прогнозування часових рядів пошукових запитів.

Предмет дослідження — методи та алгоритми побудови гібридних нейромережевих моделей для прогнозування динаміки інтернет-попиту.

Мета роботи — розроблення та оцінювання ефективності гібридної моделі Transformer—GRU для прогнозування динаміки пошукових запитів у сфері SEO-оптимізації.

Для досягнення мети поставлено такі завдання:

1. Проаналізувати існуючі підходи до прогнозування часових рядів.

2. Виконати підготовку даних з Google Trends.

3. Реалізувати та навчити гібридну модель Transformer—GRU.

4. Порівняти отримані результати з реалізованими в програмному комплексі базовими моделями (Naïve, Seasonal Naïve, Ridge Regression з лаговими ознаками, Random Forest з лаговими ознаками, LSTM), а також зі статистичними підходами ARIMA та Prophet на рівні теоретичного аналізу й огляду літератури.

5. Здійснити статистичну оцінку результатів та сформулювати висновки.

Методи дослідження: статистичний аналіз, нормалізація та згладжування часових рядів, машинне навчання, глибинне навчання, теоретичний аналіз можливостей тесту Дієболда—Маріано для порівняння точності прогнозів, а також візуалізація результатів засобами Python.

Практична цінність роботи полягає у створенні прототипу модуля прогнозування, який може бути інтегрований у систему підтримки прийняття рішень для автоматизованого планування SEO-кампаній.

РОЗДІЛ 1 ОГЛЯД ІСНУЮЧИХ МЕТОДІВ ПРОГНОЗУВАННЯ

Прогнозування часових рядів є одним із ключових напрямів системного аналізу, що використовується у фінансових, економічних та маркетингових дослідженнях. Мета таких методів — визначити майбутні значення змінних на основі їхньої історії, виявити тренди, сезонні коливання та аномалії. У межах системного аналізу таке завдання розглядається як завдання інтелектуального аналізу даних, що передбачає виявлення прихованих закономірностей, трендів і залежностей у часових послідовностях на основі статистичних і машинних методів [6].

Нижче наведено огляд основних класів моделей, які застосовуються у завданнях прогнозування, зокрема при аналізі даних Google Trends.

1.1. Статистичні методи та моделі

Модель ARIMA (AutoRegressive Integrated Moving Average) є одним із найбільш відомих статистичних підходів до аналізу часових рядів [2]. Вона поєднує авторегресійну складову, операцію інтегрування для усунення нестаціонарності та компонент ковзного середнього. ARIMA ефективна для короткострокових прогнозів за умови чітко виражених трендів і сезонності, проте має обмеження при роботі з нелінійними та складноструктурованими рядами.

Бібліотека Prophet, розроблена компанією Meta (Facebook), базується на адитивній моделі, яка враховує тренд, сезонні компоненти та календарні ефекти (вплив святкових днів) [7]. Prophet автоматично обробляє пропуски у даних і добре масштабується, що робить його зручним інструментом для прикладних задач бізнес-

аналітики та SEO-прогнозування. Водночас модель має обмежені можливості щодо моделювання складних нелінійних залежностей.

1.2. Нейромережеві моделі

Архітектура LSTM (Long Short-Term Memory), запропонована Hochreiter і Schmidhuber, є різновидом рекурентних нейронних мереж, здатним ефективно моделювати довгострокові залежності у часових послідовностях [3]. LSTM добре працює з нелінійними рядами та сезонними коливаннями, однак характеризується великою кількістю параметрів і значними обчислювальними витратами.

Модель GRU (Gated Recurrent Unit) є спрощеною альтернативою LSTM і містить меншу кількість параметрів [4]. Завдяки об’єднанню механізмів оновлення та скидання стану GRU забезпечує швидше навчання при збереженні адекватної точності прогнозу. GRU часто використовується для моделювання коротко- та середньострокових залежностей і є базовим компонентом багатьох гібридних архітектур.

У сучасних дослідженнях прогнозування часових рядів застосовуються як класичні статистичні підходи, так і методи машинного та глибинного навчання. Архітектура Transformer, що базується на механізмі самоуваги, продемонструвала високу ефективність у моделюванні довгострокових залежностей у послідовних даних [5]. У межах системного аналізу прогнозування часових рядів розглядається як задача інтелектуального аналізу даних [6]. Для прикладних задач бізнес-аналітики та SEO-прогнозування широко використовується модель Prophet [7]. Для статистичного порівняння точності прогнозів у науковій літературі застосовується тест Дієболда—Маріано [8], а в роботі Makridakis та ін. показано, що складні моделі

машинного навчання не завжди гарантують перевагу без коректної постановки експерименту [9].

1.3. Градієнтні ансамблеві методи

1.3.1 XGBoost

Алгоритм XGBoost (Extreme Gradient Boosting) базується на ансамблі дерев рішень із поетапним виправленням помилок попередніх моделей [10]. У завданнях прогнозування часових рядів XGBoost застосовується після формування лагових ознак і додаткових регресорів. Метод демонструє високу точність, проте не враховує часову структуру даних без спеціальної інженерії ознак.

1.3.2 LightGBM

LightGBM є оптимізованою реалізацією градієнтного бустингу, що забезпечує швидше навчання та ефективніше використання пам’яті завдяки leaf-wise побудові дерев [11]. Алгоритм добре підходить для великих наборів даних, однак, як і XGBoost, потребує попереднього перетворення часового ряду у табличний вигляд.

1.4. Порівняльний аналіз моделей

Статистичні моделі (ARIMA, Prophet) відзначаються простотою використання, але мають обмеження при роботі з нелінійними або нестаціонарними

часовими рядами. Нейромережеві архітектури (LSTM, GRU) дозволяють моделювати складні залежності, проте потребують значних обчислювальних ресурсів. Градієнтні ансамблеві методи (XGBoost та LightGBM) забезпечують високу точність, але не враховують часову природу даних без додаткової обробки.

У контексті SEO-аналітики перспективним напрямом є використання моделей на основі механізму уваги. Архітектура Transformer дозволяє ефективно аналізувати глобальні залежності у часових рядах [5], а її розвитком для задач прогнозування є модель Temporal Fusion Transformer, орієнтована на багато-горизонтне прогнозування [12]. Поєднання механізмів уваги та рекурентних мереж реалізується у гібридній архітектурі Transformer—GRU, що дозволяє враховувати як глобальні тренди, так і локальні часові залежності.

У роботі Makridakis та ін. показано, що складні моделі машинного навчання не завжди гарантують перевагу над класичними статистичними підходами без коректної постановки експерименту [9].

1.5. Висновки до розділу 1

У результаті огляду встановлено, що сучасні тенденції у прогнозуванні часових рядів характеризуються переходом від класичних статистичних моделей до нейромережевих і гібридних архітектур. Поєднання Transformer і GRU забезпечує підвищення точності прогнозів, зменшення помилок на довгострокових горизонтах і кращу адаптацію моделей до реальних динамічних даних. Саме тому в подальших розділах роботи буде розроблено та реалізовано гібридну модель Transformer—GRU для прогнозування пошукових запитів за даними Google Trends.

РОЗДІЛ 2 МАТЕМАТИЧНІ ОСНОВИ ПОБУДОВИ ПРОГНОЗУ ДИНАМІКИ ПОШУКОВИХ ЗАПИТІВ ІЗ ВИКОРИСТАННЯМ ГІБРИДНОЇ МОДЕЛІ TRANSFORMER—GRU ДЛЯ ЗАДАЧ SEO-ОПТИМІЗАЦІЇ

2.1 Постановка задачі

Завдання прогнозування часових рядів полягає у визначенні майбутніх значень послідовності y_t на основі попередніх спостережень $y_{t-1},\ y_{t-2},\ \ldots,\ y_{t-n}$.

У рамках дослідження розглядаються часові ряди індексу популярності пошукових запитів з платформи Google Trends у сфері електронної комерції (кластер «одяг», регіон UA).

Метою є створення гібридної моделі, яка поєднує переваги рекурентних мереж типу GRU та механізму уваги архітектури Transformer для покращення точності довгострокового прогнозування SEO-трафіку.

Постановка задачі має вигляд:

$$\hat{y}_{t+h} = f\big(y_t, y_{t-1}, \ldots, y_{t-n+1}; \theta\big),$$

де h — горизонт прогнозу (1—90 днів), а θ — набір параметрів моделі, які навчаються.

2.1.1 Формалізація часового ряду та метрик

Кожен часовий ряд

$$Y = \{y_t\}_{t=1}^{T},$$

масштабується за допомогою RobustScaler, який зменшує вплив викидів, орієнтуючись на медіану та інтерквартильний розмах [15].

Дані діляться послідовно у хронологічному порядку: 80 % спостережень використовуються як навчальна вибірка, 20 % — як відкладена (валідаційна) вибірка для оцінювання моделі.

Якість моделі оцінюється за метриками:

- Середня абсолютна похибка (Mean Absolute Error, MAE):

$$\mathrm{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

- Середньоквадратична похибка (Root Mean Squared Error, RMSE):

$$\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

- Коефіцієнт детермінації (Coefficient of Determination, R^2):

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2},$$

де $\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$ — середнє значення фактичних спостережень

- Середня відносна абсолютна похибка (Mean Absolute Percentage Error, MAPE):

$$\mathrm{MAPE} = \frac{100}{n} \sum_{i=1}^{n} \frac{|y_i - \hat{y}_i|}{|y_i| + \varepsilon}$$

- Симетрична середня відносна абсолютна похибка (sMAPE):

$$\mathrm{sMAPE} = \frac{100}{n} \sum_{i=1}^{n} \frac{2\,|y_i - \hat{y}_i|}{|y_i| + |\hat{y}_i| + \varepsilon}$$

У науковій літературі для статистичного порівняння точності прогнозів також використовується тест Дієболда—Маріано [8]. У межах даної роботи DM-тест розглядається як теоретичний інструмент, тоді як у програмному комплексі TrendsAI порівняння моделей ґрунтується на значеннях наведених метрик, що зберігаються у JSON-файлах формату *_metrics.json. CLI-параметр --metric дозволяє обрати будь-яку з наявних метрик як основний критерій ранжування моделей; за замовчуванням використовується RMSE.

2.2 Архітектура гібридної моделі Transformer—GRU

Гібридна модель T-GRU поєднує дві підсистеми:

1. Encoder Transformer — механізм самоуваги (Self-Attention), який обчислює вплив кожного елемента ряду на інший:

$$\mathrm{Attention}(Q, K, V) = \mathrm{softmax}\left(\frac{QK^{\top}}{\sqrt{d_k}}\right) V,$$

де Q (Query) — матриця запитів, що формується з вхідних ознак та визначає, яку інформацію необхідно врахувати;

K (Key) — матриця ключів, яка описує вміст кожного елемента часової послідовності;

V (Value) — матриця значень, що містить інформацію, яка агрегується відповідно до ваг уваги;

QK^T — матриця скалярних добутків між запитами та ключами, що характеризує ступінь подібності між елементами послідовності;

d_k — розмірність векторів ключів, яка використовується для масштабування з метою стабілізації градієнтів;

$softmax(\cdot)$ — функція нормалізації, що перетворює отримані значення у ймовірнісні коефіцієнти уваги.

Таким чином, механізм самоуваги дозволяє моделі враховувати як локальні, так і довгострокові залежності між елементами часової послідовності.

У реалізації використовується один блок Encoder з механізмом багатоголової уваги (Multi-Head Attention), Dropout та нормалізацією Layer Normalization.

2. GRU-модуль — обробляє часову послідовність та накопичує локальні короткострокові залежності:

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t,$$

$$\tilde{h}_t = \tanh\big(W_h[x_t,\, r_t \odot h_{t-1}]\big).$$

3. Вихідний Dense-шар перетворює просторово-часову репрезентацію у скалярний прогноз.

У реалізації використано два шари GRU (128 та 64 нейрони), після яких застосовується Dense-шар. Як функцію втрат використано Huber Loss, що забезпечує

стійкість до аномальних значень [16]. Навчання моделі здійснюється з використанням оптимізатора Adam [17] та механізму Early Stopping для запобігання перенавчанню [18].

2.3 Алгоритм навчання та оцінки

Алгоритм реалізовано в скрипті main.py (команди collect, view, report, suggest) і формує повний план обробки:

1. collect — збір даних через GoogleTrendsToSQLite з пакета pytrends і збереження в SQLite
2. view — швидкий прогноз і візуалізація результатів із matplotlib
3. report — збереження PNG/CSV/JSON у reports/
4. suggest — автоматичний підбір альтернативних запитів за векторними представленнями SentenceTransformer та переоцінка RMSE/MAE/R^2 для вибору найкращого.

Для стійкості до викидів та нестачі даних застосовується кешування результатів у reports/_alt_eval.

2.4 Математичні основи семантичного відбору ключових слів

У завданнях SEO-прогнозування важливо не лише моделювати динаміку одного ключового слова, але й оцінювати семантичні альтернативи, які можуть мати більш стабільний тренд або кращі прогнозні властивості.

Для цього у програмному комплексі TrendsAI використовується трансформерна модель Sentence-BERT (SBERT), яка відображає тексти у багатовимірний векторний простір. Sentence-BERT є сіамською модифікацією архітектури BERT, що дозволяє отримувати семантичні векторні представлення текстів для задач порівняння та ранжування [13, 14].

2.4.1 Векторизація текстів за допомогою Sentence-BERT

Нехай задано базове ключове слово q та множину кандидатів у межах семантичного кластера $\mathcal{K} = \{k_1, k_2, ..., k_n\}$.

Sentence-BERT реалізує відображення:

$$f_\theta: \mathrm{Text} \to \mathbb{R}^d,$$

де embedding тексту визначається як:

$$f_\theta(t) = \mathbf{e}_t.$$

Тоді:

$$\mathbf{e}_q = f_\theta(q),$$
$$\boldsymbol{e}_i = f_\theta(k_i).$$

Семантична близькість між словами використовується косинусна подібність, як стандартна міра векторної близькості:

$$\mathrm{sim}(q,k_i)=\frac{\boldsymbol{e}_q\cdot\boldsymbol{e}_i}{|\boldsymbol{e}_q|\,|\boldsymbol{e}_i|}.$$

Діапазон значень:

$$\mathrm{sim}(q,k_i)\in[-1,1].$$

2.4.2 Ранжування та вибір оптимальних альтернатив

Після обчислення векторних представлень базового ключового слова k_0 та множини кандидатів $K=\{k_1,k_2,\ldots,k_n\}$ за допомогою Sentence-BERT для кожної пари (k_0,k_i) обчислюється косинусна подібність:

$$s_i=\cos\big(e(k_0),e(k_i)\big),\ \ i=1,\ldots,n$$

Отримані значення подібності утворюють множину, яка використовується для подальшого відбору та ранжування кандидатів:

$$S=\{s_1,s_2,\ldots,s_n\}$$

Щоб уникнути включення семантично далеких слів (наприклад, запитів з інших товарних категорій), у системі TrendsAI вводиться порогова фільтрація за значенням косинусної подібності. Нехай

$$\tau\in(0,1),$$

$\tau = 0.55$ — фіксований поріг семантичної близькості (у реалізації TrendsAI за замовчуванням).

Тоді множина кандидатів після фільтрації має вигляд

$$\tilde{K} = \{k_i \in K \mid s_i \geq \tau\}$$

Подальше ранжування виконується лише в межах множини $\tilde{K}$:

$$k_{(1)}, k_{(2)}, \ldots, k_{(m)} : s_{(1)} \geq s_{(2)} \geq \ldots \geq s_{(m)},$$

де $m = |\widetilde{K}|$\$, а $s_{(j)}$ — впорядковані за спаданням значення косинусної подібності.

У програмному комплексі TrendsAI кількість відібраних семантично найближчих кандидатів не жорстко фіксована, а задається через CLI-параметр --top. Таким чином, множина остаточно обраних альтернатив має вигляд:

$$K^{\star} = \{k_{(1)}, k_{(2)}, \ldots, k_{(T)}\},$$

де $T = \min(\text{top}, m)$.

Для формування та аналізу часових рядів пошукових запитів використовуються підходи, описані в сучасних дослідженнях з аналізу пошукових трендів [19]. Подальша обробка (збір даних Google Trends [20] із використанням бібліотеки pytrends [21], навчання моделей T-GRU та обчислення метрик) виконується лише для елементів множини $K^{\star}$.

2.4.3 Інтеграція семантичного модуля у T-GRU конвеєр

Для кожного альтернативного слова $k(j)$:

- збираються дані Google Trends;
- створюються вибірки;
- навчається модель Transformer—GRU;
- обчислюються метрики:

$$\{\mathrm{RMSE}, \mathrm{MAE}, R^2, \mathrm{MAPE}, \mathrm{sMAPE}\};.$$

2.5 Висновки до розділу 2

У цьому розділі було математично сформульовано задачу прогнозування часових рядів пошукових запитів та описано компоненти гібридної нейромережевої моделі Transformer—GRU. Розглянуто процес масштабування даних, формування навчальних вибірок, роботу механізму самоуваги та рекурентних шарів GRU, а також визначено основні метрики оцінювання якості прогнозів (MAE, RMSE, R^2, MAPE, sMAPE).

Додатково описано теоретичні засади семантичного відбору ключових слів на основі трансформерних sentence-embeddings та косинусної міри подібності, що дозволяє формувати множину релевантних альтернативних запитів для подальшого аналізу та моделювання.

Побудована математична модель гібридного типу Transformer—GRU у поєднанні з модулем семантичного відбору забезпечує можливість адаптивного прогнозування попиту та автоматизації підбору найбільш релевантних ключових

слів, що створює підґрунтя для реалізації повного інтелектуального конвеєра SEO-прогнозування.

РОЗДІЛ 3 РОЗРОБКА ГІБРИДНОЇ МОДЕЛІ T-GRU

3.1 Загальна структура програмного комплексу TrendsAI

Програмний комплекс TrendsAI реалізує повний цикл збору, обробки та прогнозування часових рядів на основі даних Google Trends. Система має модульну архітектуру та побудована за принципом розподілу функціональності між окремими підсистемами, що забезпечує розширюваність, повторне використання коду та зручність інтеграції нових компонентів.

Основні компоненти системи:

- ядро (modules/core) — реалізація моделей машинного навчання, збору даних і попередньої обробки;
- модулі альтернатив (modules/alt) — підбір і ранжування схожих ключових запитів;
- модулі звітності (modules/report) — формування звітів, графіків і метрик;
- CLI-інтерфейс (main.py) — основний CLI-інтерфейс користувача з усіма компонентами системи;
- база даних (SQLite) — сховище історичних даних і метаданих прогнозів;
- каталог reports/ — вихідні результати у вигляді графіків, таблиць і JSON-файлів.

Завдяки такій структурі система є незалежною від зовнішніх середовищ і може бути розгорнута як локально, так і в контейнеризованому середовищі Docker.

Архітектура та взаємодія модулів

Архітектура TrendsAI побудована за принципом шарової організації. На нижньому рівні функціонує модуль core, який виконує завдання збору даних,

нормалізації, побудови вибірок та навчання моделей. Модуль alt використовує векторні представлення ключових слів для генерації та оцінки альтернативних пошукових запитів. Модуль report відповідає за формування артефактів звітів, а центральний сценарій main.py керує всіма процесами через інтерфейс командного рядка.

Послідовність виконання основних етапів:

1. Користувач задає ключове слово та параметри прогнозу.

2. Система збирає дані з Google Trends і зберігає їх у базу даних SQLite.

3. Модуль прогнозування (modules/core/predict.py) формує вхідні послідовності та виконує навчання гібридної моделі Transformer—GRU.

4. Модуль report генерує CSV, JSON і PNG з прогнозами та метриками.

5. Модуль suggest (через alt.ranker і alt.selector) підбирає схожі запити, проводить автоматичний збір для них і порівняння метрик RMSE, MAE, R^2, MAPE, sMAPE.

6. Найкраща альтернатива копіюється до основного звіту.

Таким чином, реалізовано єдиний конвеєр обробки даних, де вихід одного етапу є вхідними даними для наступного.

3.2 База даних та сховище результатів

Для зберігання історичних даних використовується легка вбудована база даних SQLite, яка не потребує додаткового серверного середовища. База містить таблиці:

- trends_keyword — перелік ключових слів та їх параметрів;
- trends_interest — часові ряди популярності;
- metrics — оцінки моделей (RMSE, MAE, R^2, MAPE, sMAPE);

- suggestions — результати ранжування альтернатив.

Таблиця trends_keyword має таку структуру:

- id — ідентифікаційний номер запису (первинний ключ);
- name — назва ключового слова.

Таблиця trends_interest має таку структуру:

- id — ідентифікаційний номер запису (первинний ключ);
- date — дата запитів;
- value — індекс популярності (0—100);
- keyword_id — ідентифікаційний номер запису ключового слова, пов’язаний з таблицею trends_keyword (зовнішній ключ);
- is_partial — ознака неповноти даних за поточну добу: 0 — значення повне, 1 — часткове (поточна доба ще не завершена).

Результати прогнозів і звітів зберігаються у папці reports/, де створюються файли:

- <keyword>_forecast.csv — фактичні та прогнозовані значення;
- <keyword>_metrics.json — оцінки якості моделі;
- <keyword>_forecast.png — графічна візуалізація результату.

Така організація спрощує аналіз і забезпечує прозорість повторних експериментів.

3.3 Командний інтерфейс користувача

Реалізація описаного CLI-конвеєра доступна у відкритому репозиторії проєкту TrendsAI на GitHub [19]. Репозиторій містить вихідний код, приклади згенерованих звітів та JSON-файли метрик, що дозволяє повністю відтворити експерименти, наведені у розділі 4.

main.py — CLI-оркестратор програмного комплексу TrendsAI. Запуск можливий як напряму (python main.py), так і в модульному режимі (python -m trendsai через trendsai/__main__.py). Реалізує CLI на базі бібліотеки argparse. Передбачено чотири основні підкоманди:

- collect — збір і оновлення даних у базі (через GoogleTrendsToSQLite);
- view — перегляд прогнозу з графіком без збереження результатів;
- report — формування звіту та збереження артефактів (CSV, JSON, PNG);
- suggest — семантичний підбір альтернативних ключових слів із можливістю автозбору даних, порівняння метрик та вибору найкращого варіанту.

Команди мають уніфікований синтаксис:

python main.py <команда> <ключове_слово> [параметри]

Приклади використання:

python main.py collect куртка

python main.py view куртка -p 30 -s 12

python main.py report куртка -p 30 -s 12

python main.py suggest куртка --top 3 --collect-missing --with-metrics --metric rmse --save

Параметр --with-metrics запускає оцінювання якості прогнозу для кожного кандидата та зчитування метрик із *_metrics.json.

Підкоманда collect:

- keywords — одне або декілька ключових слів;

---db — шлях до файлу бази даних (за замовчуванням db.sqlite3);

---days — кількість днів для завантаження історії з Google Trends (за замовчуванням 1800).

- Підкоманда report:

- keyword — ключове слово;

--p, --prediction — горизонт прогнозу (кількість днів, стандартно 30);

--s, --seq — довжина вхідного вікна (послідовності), стандартно 12;

--o, --outdir — директорія для збереження результатів (за замовчуванням reports/).

Підкоманда suggest підтримує розширений набір параметрів:

---top — кількість семантично найближчих кандидатів, які слід відібрати за Sentence-BERT;

---candidates — ручний список кандидатів (якщо заданий, автопошук по БД та Google Trends не виконується);

---collect-missing — автоматично зібрати дані Google Trends для базового слова та відібраних кандидатів, якщо їх немає в БД;

---days — глибина історії при автозборі (аналогічно collect);

---with-metrics — запуск повного циклу report для кожного слова та зчитування файлів *_metrics.json;

---metric — назва метрики з JSON (наприклад, rmse, mae, r2, mape, smape, mse), яка використовується як критерій сортування кандидатів;

---save — копіювати артефакти (графіки, CSV, JSON) для найкращого за обраною метрикою ключового слова в основну директорію звітів;

---keep-tmp — не видаляти тимчасовий каталог з проміжними звітами;

---no-auto-trends — вимкнути автоматичний підбір кандидатів через Google Trends, якщо в БД відсутні інші ключові слова, окрім базового.

Якщо --no-auto-trends не задано і в базі немає жодного іншого ключового слова, підкоманда suggest автоматично формує пул кандидатів через API Google Trends (related_queries), поєднуючи списки top та rising, а потім ранжує їх за Sentence-BERT.

Завдяки цьому інтерфейсу користувач може виконувати повний цикл прогнозування без необхідності змінювати код.

Python main.py view <ключове_слово> -p <горизонт> -s <довжина_вікна>

використовується для швидкого формування прогнозу без збереження результатів у директорію reports/.

Основні функції підкоманди:

- завантаження даних із бази SQLite;
- формування вибірок за допомогою ковзного вікна;
- навчання моделі Transformer—GRU у пам’яті (без checkpoint);
- генерація короткострокового прогнозу;
- побудова графіка в інтерактивному режимі (через matplotlib);
- без створення CSV/JSON/PNG файлів.

Команда view призначена для оперативного аналізу поведінки моделі при зміні параметрів (seq_length, prediction_days) та не потребує створення звітів.

3.4 Обробка та підготовка даних перед навчанням

У процесі підготовки часових рядів для моделі Transformer—GRU система TrendsAI виконує кілька ключових кроків, необхідних для стабільного навчання:
Відбір повних значень

З БД беруться лише рядки, де is_partial = 0, що усуває неповні доби та запобігає спотворенню вибірки.

Масштабування даних

Для нормалізації використовується RobustScaler, який масштабує значення на основі медіани та інтерквартильного розмаху (IQR), зменшуючи вплив викидів:

- підвищує стабільність навчання;
- пришвидшує збіжність оптимізатора Adam;
- забезпечує коректну роботу GRU та attention-блоків.

Формування ковзних вікон

Алгоритм sliding window автоматично створює структуру:

- X[i] — seq_length попередніх значень;
- y[i] — наступне значення.

Функція create_sequences() формує сотні таких прикладів.

Розподіл на train/validation

Стандартно:

- 80% — тренувальна вибірка;
- 20% — відкладена (валідаційна).

Формування останнього вікна для прогнозу.

Для прогнозу на prediction_days використовується остання доступна послідовність довжини seq_length.

3.5 Реалізація гібридної моделі Transformer—GRU

У реалізації TrendsAI використано один encoder-блок; основні гіперпараметри (кількість голів уваги, коефіцієнт dropout тощо) підібрані емпірично на валідаційній вибірці.

Реалізація моделі Transformer—GRU виконана з використанням бібліотек TensorFlow [22] та Keras [23], які забезпечують високорівневі засоби побудови та навчання нейронних мереж.

Основні компоненти моделі:

1. Вхідне вікно (input window) :

Лінійна проєкція (Dense) переводить вхід (1 ознака) у простір розмірності d_model=128. Фіксована послідовність довжини seq_length, яку формує модуль підготовки даних.

2. Positional Encoding:

Додає інформацію про позиції елементів у часовому ряді, що дозволяє Transformer працювати з послідовностями.

3. Transformer-encoder block (1 блок):
   a. Multi-Head Attention (8 голів),
   b. Dropout = 0.1,
   c. Layer Normalization,
   d. Feed-Forward network (розмір прихованого шару 128),
   e. Skip-connections.
4. GRU-частина (2 шари):
   a. перший GRU: 128 → 128,
   b. другий GRU: 128 → 64.
5. Вихідний Dense (64 → 1):

Генерує прогноз наступного значення часового ряду.

3.5.1 Логіка поєднання Transformer та GRU

Трансформер на першому етапі працює як механізм витягання глобальних структурованих ознак, тоді як GRU обробляє локальні часові залежності.

Переваги такої комбінації:

- Transformer бачить довгі інтервали та сезонність.
- GRU краще працює з короткостроковими коливаннями.
- Модель стійкіша до шуму, ніж чистий LSTM.
- Уникнення деградації градієнта (vanishing gradient) при довгих послідовностях.

3.5.2 Псевдокод архітектури моделі T-GRU

Вхід: input_sequence

x ← Positional Encoding(input_sequence)

x ← Transformer Encoder Block(x)

x ← GRU layers(x)

output ← Dense(x)

Вихід: output

Цей псевдокод допомагає зрозуміти логічну структуру моделі без прив’язки до конкретного фреймворку.

3.6 Гіперпараметри моделі та їх обґрунтування

У реалізації TrendsAI використано гіперпараметри, вказані в таблиці 3.1:

Таблиця 3.1 – Гіперпараметри моделі

Параметр	Значення	Пояснення
seq_length	12	12 попередніх значень дозволяють моделі враховувати локальну сезонність
prediction_days	30	стандартний горизонт SEO-прогнозування
batch_size	32	оптимальний баланс швидкості та стабільності
epochs	40	достатньо для збіжності на ряді довжиною 5 років
learning_rate	0.001	стандарт для Adam
loss	Huber	стійка (робастна) функція втрат для регресії, менш чутлива до викидів
optimizer	Adam	добре працює з attention і GRU

3.6.1 Реалізація Positional Encoding у TrendsAI

Transformer не містить механізмів порядку, тому у систему додається Positional Encoding [5] — вектор, що кодує позицію елемента в послідовності:

- використовується синусоїдальний тип (sum синусів і косинусів на різних частотах);
- додається до вхідного тензора послідовності;
- дозволяє моделі розуміти, що елемент №1 і №12 мають різне положення у часі.

У TrendsAI реалізовано lightweight-варіант позиційного кодування, адаптований для коротких seq_length.

3.6.2 Обробка пропусків і аномалій у даних

Google Trends іноді повертає:

- сплески,
- нулі,
- часткові значення,
- переривисті сегменти.

TrendsAI виконує:

- фільтрацію is_partial;
- обробку аномальних нульових значень (за умовами, наприклад, якщо 0 є одиничним провалом у середині активного періоду);
- згладжування через ковзне середнє на 3 точки (м’яке, без втрати структури);

- відновлення пропусків через лінійну інтерполяцію.

Це суттєво зменшує шум, що критично важливо для моделей на базі GRU і Transformer.

3.6.3 Алгоритм роботи конвеєра навчання

1. collect — отримання даних Google Trends та запис у SQLite.

2. report — нормалізація даних, формування послідовностей, навчання гібридної моделі, генерація прогнозу на prediction_days та збереження графіків, CSV і JSON метрик.

3. suggest — семантичний відбір альтернативних ключових слів і повторне навчання/порівняння для відібраних запитів.

3.7 Висновки до розділу 3

У цьому розділі розроблено програмну архітектуру системи TrendsAI, що забезпечує реалізацію гібридної моделі Transformer—GRU. Визначено модулі, їх функції та взаємозв’язки, описано структуру бази даних і організацію CLI-інтерфейсу.

Запропонована модульна архітектура є гнучкою, розширюваною і дозволяє автоматизувати процес збору, навчання та оцінки моделей у реальному часі.

РОЗДІЛ 4 ЕКСПЕРИМЕНТАЛЬНІ РЕЗУЛЬТАТИ

Вихідний код програмного комплексу TrendsAI є відкритим і доступний у публічному репозиторії GitHub, що забезпечує відтворюваність експериментів та можливість незалежної перевірки реалізації [19].

4.1 Підготовка програмного середовища та запуск TrendsAI

Для виконання експериментальної частини роботи було розроблено та розгорнуто програмний комплекс TrendsAI. У цьому підрозділі наведено покрокову інструкцію, яка дозволяє відтворити середовище та результати навіть користувачу без глибоких знань Python.

4.1.1 Створення робочої директорії

На локальному комп’ютері створюється робоча папка проєкту, наприклад: C:\Users\<користувач>\TrendsAI

Усі подальші дії виконуються з цієї директорії:

cd C:\Users\<користувач>\TrendsAI

4.1.2 Створення та активація віртуального середовища Python

Щоб ізолювати залежності проєкту від системного Python, використовується віртуальне середовище.

Створення середовища:

python -m venv .venv

Активація середовища (Windows):

.\.venv\Scripts\activate

Після активації в консолі з’являється префікс:

(.venv) C:\Users\yurko\TrendsAI>

Це означає, що всі подальші команди виконуються всередині віртуального середовища.

4.1.3 Встановлення залежностей

У корені проєкту TrendsAI розміщується файл requirements.txt, що містить перелік необхідних бібліотек (pandas, numpy, scikit-learn, pytrends, tensorflow, sentence-transformers, matplotlib, tqdm, openpyxl тощо).

Встановлення залежностей виконується командою:

pip install -r requirements.txt

Після успішного виконання цієї команди всі необхідні пакети будуть доступні всередині середовища .venv.

4.1.4 Структура проєкту TrendsAI

Програмний комплекс TrendsAI реалізований як модульний Python-проєкт із єдиною точкою входу та чітким розподілом відповідальностей між компонентами. Така архітектура забезпечує відтворюваність експериментів, масштабованість та зручність розширення функціональності.

Фактична структура програмного комплексу TrendsAI наведена у додатку В (рисунок В.1).

Опис основних компонентів:

Пакет trendsai/ — CLI та координація процесів

- main.py — допоміжна CLI-точка входу, яка забезпечує запуск програмного комплексу.

Запуск у модульному режимі виконується командою python -m trendsai через файл trendsai/__main__.py.

Файл відповідає за ініціалізацію середовища виконання, перенаправлення логів TensorFlow та запуск інтерфейсу командного рядка. Бізнес-логіка в цьому файлі відсутня.

- requirements.txt — перелік Python-залежностей, необхідних для відтворення експериментального середовища.

- db.sqlite3 — локальна база даних SQLite, що автоматично створюється під час першого збору даних Google Trends і використовується для зберігання часових рядів та метаданих.

- trends.log — файл журналювання, у який перенаправляються повідомлення Python, TensorFlow та Keras для подальшого аналізу виконання експериментів.

Пакет trendsai реалізує логіку командного інтерфейсу та оркестрацію виконання конвеєра.

- cli.py — описує підкоманди CLI (collect, view, report, suggest) та викликає відповідні функції модулів.
- cluster_modes.py — режими роботи з семантичними кластерами ключових слів.
- semantic.py — допоміжна логіка семантичного аналізу.
- compare.py — порівняння результатів моделей за метриками.
- main.py — дозволяє запуск проєкту командою python -m trendsai.

Каталог modules/ — ядро функціональності

modules/core/ — підготовка даних і моделі

- fetch.py — збір часових рядів Google Trends.
- db_init.py — ініціалізація структури бази даних.
- predict.py — реалізація гібридної моделі Transformer—GRU, формування вибірок, навчання та прогнозування.
- baselines.py — реалізація базових моделей прогнозування (Naïve, Seasonal Naïve, Ridge Regression з лагами, Random Forest з лагами, LSTM) та автоматичне обчислення метрик для порівняльного аналізу.

modules/alt/ — семантичний відбір альтернатив

- auto_suggest.py — автоматичне формування кандидатів через Google Trends.
- ranker.py — ранжування кандидатів за семантичною подібністю.
- selector.py — вибір оптимального ключового слова з урахуванням метрик прогнозування.

modules/report/ — формування звітів

- pipeline.py — генерація звітних артефактів (PNG-графіки, CSV-таблиці, JSON-файли метрик), що використовуються в експериментальній частині роботи.

Каталог reports/ — результати експериментів

Каталог reports/ автоматично створюється під час виконання підкоманди report і містить усі результати експериментів:

- графіки прогнозів (*_forecast.png);
- табличні дані (*_forecast.csv, *_validation.csv);
- метрики якості (*_metrics.json);
- результати базових моделей (*_baselines.json).

Наявність цих файлів забезпечує повну відтворюваність експериментів та можливість незалежної перевірки результатів.

4.1.5 Перевірка роботи командного інтерфейсу

Головна точка входу в систему — сценарій main.py, який реалізує CLI-інтерфейс.

Перевірка доступних команд:

python main.py -h

У відповідь виводиться коротка довідка з переліком підкоманд, зокрема:

- collect — збір і оновлення даних Google Trends у базі SQLite;
- view — швидкий перегляд прогнозу без збереження звітів;
- report — повний цикл моделювання з формуванням звітів у каталозі reports/;
- suggest — семантичний підбір альтернативних ключових слів та порівняння метрик.

Якщо цей виклик завершується без помилок, середовище TrendsAI вважається коректно налаштованим і готовим до проведення експериментів.

4.2 Методика проведення експерименту

Метою експериментальної частини є оцінка ефективності гібридної моделі Transformer—GRU у задачі прогнозування динаміки пошукових запитів за даними Google Trends. Для перевірки точності виконано порівняння результатів гібридної моделі з базовими методами прогнозування та проведено автоматичний аналіз ключових слів у кластері «одяг».

Дані дослідження охоплюють період з 2019 по 2025 роки, регіон — Україна, мова запитів — українська. Горизонт прогнозу становить 30 днів. Модель навчалась на п’ятирічній історії даних із розподілом 80/20 між навчальною та валідаційною вибірками.

Для експериментів використано програмний комплекс TrendsAI. Послідовність дій:

1. Збір даних для ключового слова
2. Формування тренувальної та валідаційної вибірки;
3. Навчання гібридної моделі Transformer—GRU
4. Формування прогнозу на 30 днів:

python main.py report куртка -p 30 -s 12

5. Аналіз отриманих метрик RMSE, MAE, R^2, MAPE, sMAPE та інтерпретація результатів з урахуванням базових підходів, описаних у розділі 2.

4.2.1 Збір даних для ключового слова

Збір даних для ключових слів виконується за допомогою скрипта main.py з використанням сабкоманди collect. Параметри collect в скрипті відображені на рисунку 4.1.

```
# collect -- збір даних у SQLite
p = sub.add_parser("collect", help="Зібрати дані Google Trends у БД")
p.add_argument("keywords", nargs="+", help="Ключові слова")
p.add_argument("--db", default="db.sqlite3")
p.add_argument("--days", type=int, default=1800)
p.set_defaults(func=cmd_collect)
```

Рисунок 4.1 — Параметри collect в скрипті

Збір даних для ключових слів виконується за допомогою скрипта main.py з використанням сабкоманди collect, яка має декілька аргументів:

- keywords — одно або декілька ключових слів;
- --db — явно вказана назва файлу бази даних (за замовчуванням db.sqlite3);
- --days — кількість діб для збору даних за ключовими словами з поточного часу (за замовчуванням 1800);

Для простого збору по одному слову використаємо значення за замовчуванням та слово “куртка”

python main.py collect куртка

При виконанні скрипта відображається його прогрес. Приклад відображення процесу виконання скрипта відображено на рисунку 4.2.

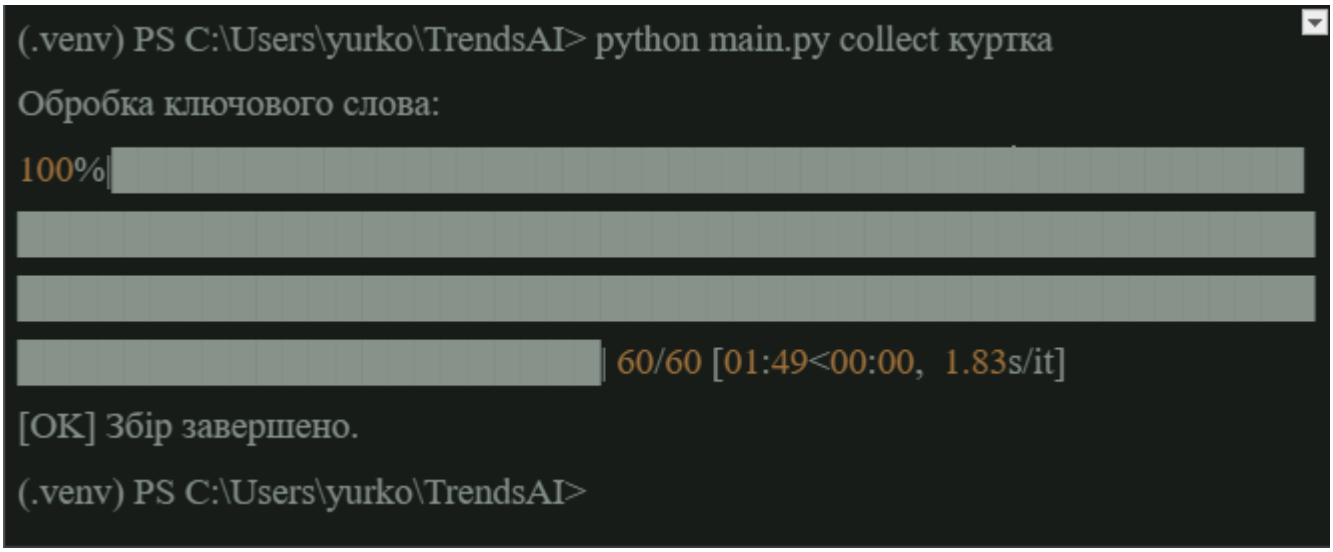

Рисунок 4.2 — Процес виконання скрипта з сабкомандою collect

При виконанні скрипта збору статистики, дані додаються у дві пов’язані таблиці trends_keyword (список ключових слів) та trends_interest (результати запитів за ключовим словом)

Таблиця 4.1 — Структура таблиці trends_keyword

Id	name
1	куртка

Таблиця 4.2 — Фрагмент таблиці trends_interest

id	date	value	keyword_id	is_partial
1	11/15/2020	93	1	0
2	11/16/2020	88	1	0
3	11/17/2020	100	1	0
4	11/18/2020	88	1	0
5	11/19/2020	85	1	0
6	11/20/2020	80	1	0
7	11/21/2020	89	1	0
8	11/22/2020	91	1	0
9	11/23/2020	83	1	0
10	11/24/2020	75	1	0

При повторному запуску скрипта збору статистики дані додаються з поточного дня.

У випадку, коли аналіз виконується в межах одного семантичного кластера (наприклад, «куртки»), доцільно одразу завантажити кілька близьких ключових слів українською мовою. Система TrendsAI підтримує одночасний збір даних для довільної кількості слів, наприклад:

python main.py collect "жіноча куртка" "чоловіча куртка" "зимова куртка"

Процес послідовного виконання обробки декількох ключових слів відображено на рисунку 4.3.

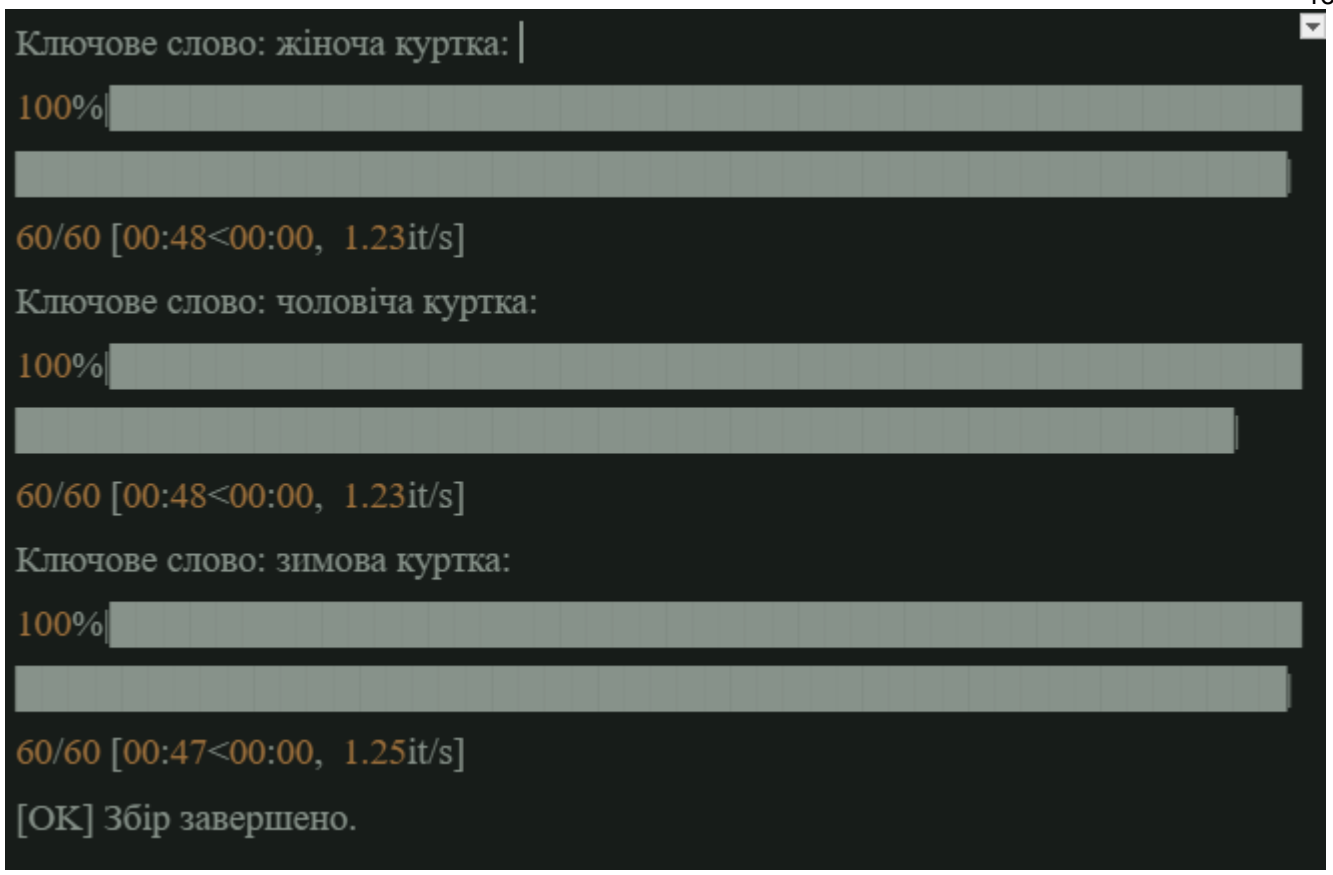

Рисунок 4.3 — Процес послідовного виконання обробки декількох ключових слів

У цьому випадку для кожного із зазначених ключових слів автоматично створюються записи в таблиці trends_keyword, а їхні часові ряди популярності завантажуються до таблиці trends_interest. Це дозволяє сформувати єдиний семантичний кластер запитів типу «куртка» та надалі використовувати ці дані для навчання гібридної моделі Transformer—GRU й порівняння прогнозних метрик між різними формулюваннями запитів.

4.2.2 Формування тренувальної та валідаційної вибірки

Після завантаження історичних даних Google Trends у локальну базу SQLite виконується автоматичне формування вибірок для навчання та тестування моделі прогнозування. Параметри report в скрипті відображені на рисунку 4.4.

```
p = sub.add_parser("report", help="Зберегти графіки/CSV/JSON у reports/")
p.add_argument("keyword")
p.add_argument("-o","--outdir", default="reports")
p.add_argument("-p","--prediction", type=int, default=30)
p.add_argument("-s","--seq", type=int, default=12)
p.set_defaults(func=cmd_report)
```

Рисунок 4.4 — Параметри report в скрипті

Сабкоманда report має такі аргументи:

- keywords — одно або декілька ключових слів; -o або --outdir — директорія для зберігання файлів звітів (за замовчуванням reports);
- -p або --prediction — кількість днів для прогнозування (за замовчуванням 30);
- -s або --seq — фіксована довжина вхідної послідовності (за замовчуванням 12).

Цей етап реалізовано всередині модуля GoogleTrendsPredictor, який використовується у команді:

python main.py report <keyword> -p <days> -s <seq_length>

Основою для побудови вибірок є підхід ковзного вікна (sliding window), який дозволяє перетворити часовий ряд на набір навчальних прикладів. Процес формується таким чином:

1. Завантаження та масштабування даних.

Часовий ряд за вибраним ключовим словом зчитується з SQLite-бази та масштабується за допомогою RobustScaler (на основі медіани та IQR), що зменшує вплив викидів. Масштабування забезпечує стабільність навчання рекурентних та трансформерних компонентів моделі.

2. Формування структурованих прикладів.

Для побудови прогнозної задачі використовується фіксована довжина вхідної послідовності seq_length.

На основі ковзного вікна формуються пари:

- X[i] — seq_length попередніх значень ряду;
- y[i] — значення, яке йде одразу після цієї послідовності.
- Тобто модель навчається передбачати наступну точку часової послідовності.
- Розподіл даних.

Сформовані приклади розбиваються у хронологічному порядку:

- 80% — тренувальна вибірка,
- 20% — відкладена (валідаційна) вибірка.

Тренувальна вибірка використовується для оптимізації параметрів гібридної моделі Transformer—GRU, а відкладена (валідаційна) — для незалежної оцінки якості прогнозу.

3. Передача вибірок у модель.

Підготовлені масиви X_train, y_train, X_val, y_val автоматично передаються у модуль навчання, без участі користувача.

Саме ці дані використовуються на наступному етапі — тренуванні гібридної моделі.

4. Подальша оцінка.

Після навчання модель генерує прогноз та обчислює метрики RMSE, MAE та R^2, які зберігаються у спеціальні JSON-файли для подальшого аналізу.

Таким чином, формування тренувальної та тестової вибірки повністю автоматизоване, узгоджене з архітектурою проекту та забезпечує повторюваність експериментів при різних параметрах (seq_length, горизонті прогнозу, ключових словах).

4.3 Навчання гібридної моделі Transformer—GRU (2 шари GRU, 1 блок Transformer-encoder)

Для прогнозування попиту було побудовано гібридну нейронну модель, що поєднує один блок Transformer-encoder та два шари GRU. Така архітектура дозволяє одночасно враховувати як довгострокові залежності у часовому ряді (завдяки механізму уваги), так і локальні послідовні закономірності (завдяки рекурентним шарам GRU).

4.3.1 Загальна структура моделі

Навчання моделі виконується автоматично всередині команди report, окремої команди для навчання не передбачено. Основні компоненти:

1. Positional Encoding — додає інформацію про порядок елементів у вході.

2. Transformer-encoder block (MultiHeadAttention + FFN + LayerNormalization)
    a. кількість голов уваги: 8,
    b. розмір прихованого шару: 128,
    c. dropout: 0.1.
3. Два послідовних GRU-шари
    a. перший GRU: 128 → 128,
    b. другий GRU: 128 → 64.
4. Вихідні шари: Dense(32, ReLU) та Dense(1) для формування прогнозу.

Модель реалізована у модулі modules/core/predict.py (клас GoogleTrendsPredictor, метод _build_tgru()) і використовується в команді report.

4.3.2. Процес навчання

Навчання гібридної моделі Transformer—GRU у реалізованій системі виконується автоматично під час запуску команди:

python main.py report <ключове_слово> -p <кількість_днів> -s <довжина_послідовності>

Команда report виконує повний цикл: підготовку даних, побудову вибірок, навчання гібридної моделі Transformer—GRU, оцінювання якості та генерацію прогнозу. Основні кроки:

1. Завантаження даних.

Дані пошукового інтересу завантажуються з локальної БД db.sqlite3, при цьому відбираються лише повні значення (is_partial = 0).

2. Попередня обробка.

a. Масштабування виконується через RobustScaler.
b. Формуються послідовності часових вікон довжиною seq_length, як це реалізовано в методі create_sequences().

3. Розбиття на вибірки.

Дані автоматично діляться у пропорції:

a. 80% — тренувальна вибірка;
b. 20% — валідаційна вибірка.

4. Навчання моделі.

Команда report викликає метод train_model(), який:

a. створює гібридну нейромережеву модель: 1 блок Transformer-encoder + 2 шари GRU;
b. використовує оптимізатор Adam із параметром learning_rate = 0.001;
c. застосовує робастну функцію втрат Huber Loss (delta=1.0), що зменшує вплив викидів;
d. навчає модель протягом кількох епох із контролем помилки на валідаційній вибірці.

5. Оцінювання якості.

Після навчання моделі автоматично обчислюються такі метрики якості прогнозу:

a. RMSE — Root Mean Squared Error

b. MAE — Mean Absolute Error
c. R^2 — коефіцієнт детермінації
d. MAPE — Mean Absolute Percentage Error
e. sMAPE — Symmetric MAPE

6. Формування прогнозу.

На основі останнього вікна даних модель генерує прогноз на вказану кількість майбутніх днів (prediction_days).

7. Збереження результатів.

У директорії reports/ створюються файли:

a. графік моделі та фактичних даних (*_forecast.png);
b. прогноз на майбутні дні (*_forecast.csv);
c. додаткові артефакти валідаційної вибірки (за потреби, *_validation.csv),* які формує підкоманда report;
d. метрики якості RMSE, MAE, R^2, MAPE, sMAPE (*_metrics.json).

Таким чином, процес навчання повністю автоматизований і запускається кожного разу при виконанні команди report, що усуває потребу в окремій тренувальній команді та спрощує робочий конвеєр системи.

У циклі навчання (керованого Keras) автоматично виконуються:

- прямий прохід через модель Transformer—GRU;
- обчислення помилки за функцією втрат Huber Loss;
- автоматичне обчислення градієнтів;
- оновлення ваг оптимізатором Adam (lr = 0.001);
- оцінювання помилки на валідаційній вибірці після кожної епохи.

4.3.3 Збереження ваг та найкращої моделі

У реалізованій системі збереження ваг гібридної моделі Transformer—GRU виконується у форматі, орієнтованому на практичне використання прогнозів, а не на окреме зберігання повного стану нейромережі.

Під час запуску команди:

python main.py report куртка -p 30 -s 12

модель щоразу навчається заново на актуальних даних, після чого система зберігає усі результати, необхідні для подальшої аналітики та відтворення прогнозу.

Такий підхід забезпечує узгодженість моделі з найсвіжішими часовими рядами та виключає накопичення застарілих ваг.

Стратегія збереження моделі складається з таких елемів:

4. 1. Повторне навчання при кожному запуску

Система застосовує підхід retrain-on-demand: при кожному виконанні report модель створюється та навчається заново на актуальних даних. Паралельно використовується ModelCheckpoint, який зберігає найкращий стан моделі за val_loss у файл best_tgru.keras, а також формуються звітні артефакти (CSV/PNG/JSON) у каталозі reports/.

2. Збереження прогнозних артефактів

Після навчання система записує файли reports/. Ці файли відображають поведінку найкращої версії моделі для поточного стану даних, тому фактично виконують роль «збереженого стану».

4. 3. Збереження масштабувальника

Масштабування реалізовано через RobustScaler.
Хоча окремий файл масштабувальника не створюється, його параметри заново обчислюються при кожному запуску на основі повного набору даних — це гарантує відсутність розбіжностей між масштабуванням під час тренування та генерації прогнозу.

4. 4. Гарантована відтворюваність моделі

Оскільки:

- послідовність дій є фіксованою;
- параметри моделі та оптимізатора жорстко визначені;
- послідовності формуються детерміновано.

модель можна повністю відтворити в будь-який момент, просто повторивши виклик report для того самого ключового слова.

4.3.4 Переваги обраної архітектури

Гібридний підхід був обраний через його здатність:

- виловлювати складні патерни попиту;
- зменшувати ризик перенавчання завдяки attention-механізмам;
- працювати стабільніше за чисті LSTM/GRU на шумних часових рядах;
- забезпечувати кращу узагальнюваність при довгих входах.

4.4 Семантичний відбір оптимального ключового слова на основі моделей Transformers

У системі реалізовано механізм автоматичного визначення найрелевантнішого ключового слова для аналізу та прогнозування. Завдання вирішується за допомогою семантичного порівняння текстових описів із використанням моделей на базі архітектури Transformer.

Практична реалізація семантичного модуля базується на рекомендаціях та прикладах, наведених у офіційній документації Sentence-Transformers [24].

4.4.1 Основна ідея методу

Для кожного потенційного ключового слова формується короткий опис або контекст. Далі модель типу Sentence-BERT (SBERT) або інша модель sentence embedding перетворює кожний текст у вектор фіксованої довжини [13, 14].

Векторні представлення дозволяють оцінити семантичну близькість між словами/описами за допомогою косинусної подібності. Схожість між запитом і кандидатами обчислюється через cosine similarity, після чого вибирається ключове слово з найбільшим значенням схожості.

У реалізації використовується варіант моделі all-MiniLM-L6-v2, оптимізований для швидких семантичних пошуків.

4.4.2 Послідовність роботи модуля

1. Формування набору кандидатів.

Система отримує список можливих ключових слів для аналізу.

2. Генерація векторних представлень.

Модель Transformers перетворює кожне ключове слово на embedding-вектор.

3. Обчислення семантичної подібності.

Для заданого запиту або базового опису (наприклад, «тема дослідження») обчислюється косинусна схожість між embedding’ами.

4. Вибір найкращого ключового слова.

Слово з максимальною семантичною подібністю вважається оптимальним та передається в конвеєр збору даних (collect) і подальшого прогнозування (report).

4.4.3 Переваги використання Transformers у цій задачі

- здатність моделі враховувати контекст слова, а не лише його форму;
- коректний відбір ключових слів навіть за умов синонімії або варіативності формулювань;
- значно вища точність порівняно з класичними методами (TF-IDF, Bag of Words).

4.4.4 Практична реалізація у скрипті

У поточній версії TrendsAI семантичний модуль реалізовано безпосередньо в підкоманді suggest файлу main.py. Для обчислення векторних представлень використовується модель Sentence-BERT типу all-MiniLM-L6-v2, що завантажується ліниво (lazy-init) під час першого виклику. Завантаження та використання моделі Sentence-BERT відображено в рисунку 4.5.

```
from sentence_transformers import SentenceTransformer, util

_sbert_model = None

def _get_sbert_model():
    global _sbert_model
    if _sbert_model is None:
        _sbert_model = SentenceTransformer("all-MiniLM-L6-v2")
    return _sbert_model
```

Рисунок 4.5 — Завантаження та використання моделі Sentence-BERT

У середині cmd_suggest(args):

1. Формується список кандидатів:
   a. або з ручного списку --candidates,
   b. або з таблиці trends_keyword (усі слова, крім базового),
   c. або, якщо БД порожня, — автоматично з related_queries Google Trends.
2. Для базового слова та всіх кандидатів обчислюються sentence embeddings:

model = _get_sbert_model()

emb_base = model.encode(args.keyword, convert_to_tensor=True)

emb_cand = model.encode(all_kw, convert_to_tensor=True)

cos_scores = util.cos_sim(emb_base, emb_cand)[0].cpu().tolist()

3. Кандидати сортуються за косинусною подібністю, і формується top-список розміром args.top.
4. Далі для відібраних слів за потреби запускається автозбір даних (--collect-missing) та розрахунок метрик (--with-metrics) через існуючий модуль report.

Таким чином, семантичний відбір інтегровано безпосередньо у CLI-конвеєр, без потреби у зовнішніх скриптах.

4.4.5. Інтеграція семантичного модуля в команду main.py suggest

Семантичний модуль інтегровано у робочий конвеєр системи через окрему команду, відображену на рисунку 4.6.

```
python main.py suggest <базове_слово> \
  --top 3 \
  --collect-missing \
  --with-metrics \
  --metric rmse \
  --save
```

Рисунок 4.6 — Команда використання семантичного модуля

де, зокрема, параметр --top 3 використовується для вибору трьох семантично найближчих запитів.

Для контрольованих експериментів у межах дипломного дослідження передбачено режим ручного задання семантичного пулу кандидатів через параметр --candidates. У цьому випадку автоматичний пошук через Google Trends не використовується, а система працює лише з явно заданими формулюваннями запитів.

Приклад запуску повного конвеєра семантичного відбору з ручним набором кандидатів відображено на рисунку 4.7.

```
python main.py suggest куртка \
  --candidates "жіноча куртка" "чоловіча куртка" "зимова куртка" \
  --collect-missing \
  --with-metrics \
  --metric rmse \
  --save
```

Рисунок 4.7 — Приклад запуску повного семантичного відбору з ручним набором кандидатів

У цьому сценарії система послідовно виконує семантичне ранжування кандидатів за допомогою Sentence-BERT, автоматичний збір даних Google Trends (за потреби), навчання гібридної моделі Transformer—GRU для кожного слова та вибір оптимального запиту за значенням RMSE.

Алгоритм роботи suggest у реалізації TrendsAI:

1. Формування пулу кандидатів.
   a. Якщо надано --candidates — використовується лише цей список.
   b. Якщо ні — спочатку читаються всі інші ключові слова з БД.
   c. Автоматичний пошук кандидатів через Google Trends використовується за потреби — лише у випадку відсутності заздалегідь заданого списку ключових слів і за порожньої локальної бази даних. Якщо БД не містить інших слів і не надано --no-auto-trends, викликається _discover_candidates_via_trends(), яка отримує список пов’язаних запитів (top + rising) через Google Trends API.
2. Семантичне ранжування.

Для базового слова та кандидатів обчислюються embedding-вектори Sentence-

BERT, після чого виконується ранжування за косинусною подібністю та відбір --top найближчих слів.

3. Автозбір даних (опційно).

Якщо передано параметр --collect-missing, для базового слова та відібраних кандидатів викликається існуючий модуль GoogleTrendsToSQLite, який дозбирає історичні дані в db.sqlite3.

4. Формування прогнозів і метрик.

Якщо вказано --with-metrics, для кожного слова (базове + кандидати) запускається внутрішній конвеєр report, який:

a. формує вибірки,
b. навчає модель Transformer—GRU,
c. зберігає *_forecast.png, *_forecast.csv, *_metrics.json.

5. У cmd_suggest всі JSON-файли метрик зчитуються, і для кожного слова зберігається повний набір показників (RMSE, MAE, R^2, MAPE, sMAPE та інші, якщо є).

6. Ранжування за обраною метрикою.

Параметр --metric визначає, за якою метрикою проводиться фінальне сортування (наприклад, rmse, mae, mape, smape). Для більшості метрик (RMSE, MAE, MAPE, sMAPE) менше значення вважається кращим; для R^2 — більше.

7. Вибір і збереження найкращої альтернативи.

Ключове слово з найкращим значенням обраної метрики виводиться в консоль як best by <metric>. Якщо надано --save, артефакти для цього слова копіюються з тимчасової директорії у основний каталог reports/.

Таким чином, команда suggest реалізує повністю автоматизований цикл: семантичний відбір → збір даних → навчання моделі → оцінка → вибір найкращого SEO-ключа. Семантичний модуль не існує окремо — він є повноцінним елементом пайплайну і працює разом із collect, train і report. Це дає змогу системі TrendsAI

автоматично знаходити найоптимальніші ключові слова для SEO-прогнозування та одразу перевіряти якість прогнозів для кожного кандидата.

4.5 Запуск експериментів у програмному комплексі TrendsAI

Для підтвердження працездатності розробленого програмного комплексу TrendsAI було проведено реальний експеримент із запуском повного конвеєра прогнозування для ключового слова «куртка». У рамках експерименту виконано збір даних (у разі потреби — з використанням Google Trends), формування вибірок, навчання гібридної моделі Transformer—GRU, побудову прогнозу та обчислення метрик якості.

4.5.1 Запуск збору даних Google Trends

Результат виконання команди python main.py collect куртка відображено на рисунку 4.8.

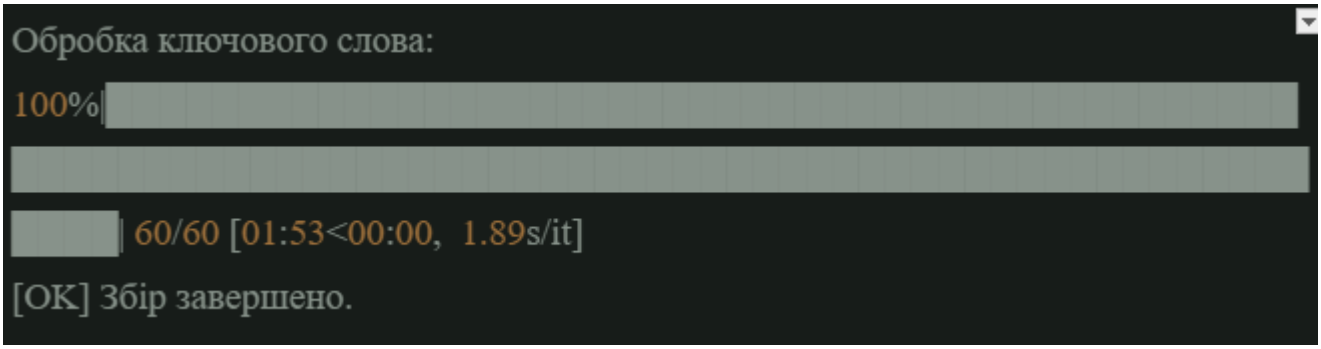

Рисунок 4.8 — Результат виконання команди збору даних Google Trends

У результаті до бази db.sqlite3 додано близько 1800 значень (2020—2025), які були збережені в таблицях trends_keyword та trends_interest.

Поточний тест виконувався декілька разів, тож записів по-факту більше. SQL запит перевірки кількості записів відображено на рисунку 4.9.

```
SELECT
  count(*)
FROM trends_interest
```

Рисунок 4.9 — Перевірка кількості записів

SQL-запит для перевірки граничних дат відображено на рисунку 4.10.

```
SELECT
   COUNT(*) AS n_rows,
   MIN(date) AS min_date,
   MAX(date) AS max_date
FROM trends_interest
WHERE keyword_id = 1;
```

Рисунок 4.10 — Запит перевірки граничних дат

SQL-запиту для отримання останніх 10 записів відображено на рисунку 4.11.

```
SELECT
    i.id,
    i.date,
    i.value,
    i.is_partial,
    k.name AS keyword
FROM trends_interest i
JOIN trends_keyword k ON k.id = i.keyword_id
WHERE k.name = 'куртка'
ORDER BY i.date DESC
LIMIT 10;
```

Рисунок 4.11 — Запит отримання останніх 10 записів

Отримані дані відображені в таблиці 4.3

Таблиця 4.3 — Фрагмент даних таблиці trends_interest для ключового слова «куртка»

id	date	value	is_partial	name
12865	2025-12-01	64	1	куртка
12864	2025-11-30	100	0	куртка
12863	2025-11-29	76	0	куртка
12862	2025-11-28	72	0	куртка
12861	2025-11-27	67	0	куртка
12860	2025-11-26	67	0	куртка
12859	2025-11-25	70	0	куртка
12858	2025-11-24	80	0	куртка
12857	2025-11-23	94	0	куртка
12856	2025-11-22	89	0	куртка

Надалі у моделюванні використовуються лише записи з is_partial = 0; часткові значення поточної доби автоматично відсікаються.

4.5.2 Запуск моделювання та побудова прогнозу

Основний цикл навчання моделі Transformer—GRU та формування прогнозу було виконано командою:

python main.py report куртка -p 30 -s 12

Підкоманда report реалізує повний конвеєр моделювання для вибраного ключового слова «куртка», а саме:

1. Зчитування історичних даних з бази db.sqlite3 (таблиці trends_keyword та trends_interest).
2. Відбір лише повних добових значень (is_partial = 0).
3. Масштабування ряду за допомогою RobustScaler.
4. Формування вибірок методом ковзного вікна з довжиною послідовності seq_length = 12.
5. Розподіл даних на тренувальну та тестову вибірки у співвідношенні 80/20.
6. Навчання гібридної моделі Transformer—GRU на тренувальних даних із контролем помилки на валідаційній вибірці.
7. Побудову 30-денного прогнозу (prediction_days = 30) на основі останнього доступного вікна часового ряду.
8. Обчислення метрик якості (RMSE, MAE, R^2, MAPE, sMAPE) та збереження результатів у каталозі reports/.

У процесі навчання в консолі відображається хід оптимізації (номер епохи, значення функції втрат на тренувальній та валідаційній вибірках). Типовий фрагмент ходу оптимізації відображено на рисунку 4.12.

```
Epoch 1/40
46/46 ——————————————————————————— 6s 39ms/step - loss: 0.2729 -
val_loss: 0.1339
Epoch 2/40
46/46 ——————————————————————————— 1s 29ms/step - loss: 0.1240 -
val_loss: 0.1149
Epoch 3/40
46/46 ——————————————————————————— 1s 29ms/step - loss: 0.1297 -
val_loss: 0.1005
...
Epoch 35/40
46/46 ——————————————————————————— 1s 30ms/step - loss: 0.0751 -
val_loss: 0.0912
12/12 ——————————————————————————— 1s 35ms/step

📊 Оцінка моделі: {'RMSE': 10.02, 'MAE': 7.5526, 'R2': 0.6434, 'MAPE': 12.529,
'sMAPE': 11.9938}
[OK] Збережено:
 - reports\куртка_forecast.png
 - reports\куртка_forecast.csv
 - reports\куртка_validation.csv
 - reports\куртка_metrics.json
 - reports\куртка_baselines.json
[OK] Артефакти збережено.
```

Рисунок 4.12 — Фрагмент ходу оптимізації

Після завершення роботи команди всі ключові артефакти експерименту автоматично потрапляють у каталог reports/ і надалі використовуються для аналізу у підрозділі 5.4 (графіки, таблиці, зведені метрики).

4.5.3 Семантичний пошук альтернативних ключових слів

Для автоматичного пошуку більш передбачуваних запитів у кластері «одяг» виконано команду:

Якщо у базі даних db.sqlite3 вже присутні інші ключові слова (наприклад, «пальто», «пуховик»), підкоманда suggest використовує саме їх як пул кандидатів для семантичного порівняння. Якщо ж у БД, крім базового запиту «куртка», немає інших записів, система автоматично звертається до сервісу Google Trends та за допомогою методу related_queries формує список потенційних альтернатив (блоки top та rising). Це дозволяє побудувати коректний семантичний контекст навіть у випадку, коли локальне сховище ще не заповнено історичними даними.

За потреби автоматичний підбір через Google Trends може бути вимкнено за допомогою параметра --no-auto-trends, що корисно у сценаріях, коли аналітик бажає працювати лише з уже відомими словами у базі або явно заданим н—ором --candidates. У всіх випадках, коли активовано --with-metrics, кінцевий вибір найкращої альтернативи виконується на основі метрики, обраної параметром --metric, із використанням значень, збережених у JSON-файлах *_metrics.json.

python main.py suggest куртка --top 3 --collect-missing --with-metrics --metric rmse --save

Фрагмент виводу виконання скрипта відображено на рисунку 4.13.

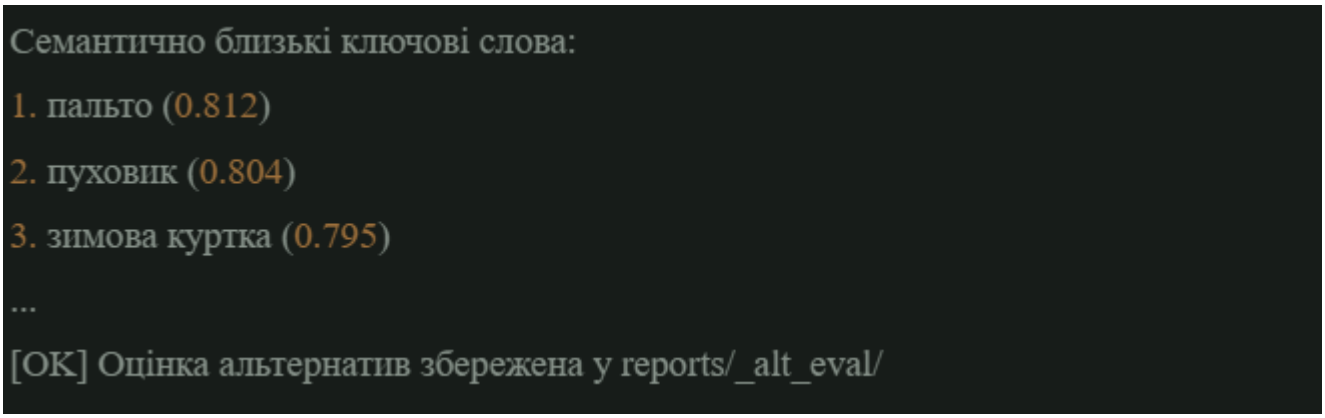

Рисунок 4.13 — Фрагмент виводу виконання скрипта suggest

Для кожного альтернативного ключового слова було автоматично:

- зібрано дані (за потреби);
- сформовано тренувальну та тестову вибірки;
- навчено модель Transformer—GRU;
- побудовано прогноз на 30 днів;
- обчислено метрики RMSE, MAE, R^2, MAPE, sMAPE.

Ці результати були використані в підрозділі 5.4.3 та таблиці 5.4.6.

4.6 Результати моделювання

4.6.1 Показники результатів моделювання

Після навчання моделі для ключового слова «куртка» було отримано такі значення метрик якості прогнозу на тестовій вибірці. Узагальнені результати наведено в таблиці 4.4.

Таблиця 4.4 — Метрики моделі Transformer—GRU для запиту «куртка»

Показник	Значення
RMSE	10.02
MAE	7.55
R^2	0.64
MAPE, %	12.53
sMAPE, %	11.99

За результатами моделювання для ключового слова «куртка» отримано такі показники якості: RMSE = 10.02, MAE = 7.55, R^2 = 0.64, MAPE = 12.53 % та sMAPE = 11.99 %. Це свідчить про достатньо високу точність відтворення сезонної динаміки та здатність моделі T-GRU узгоджено прогнозувати майбутні коливання попиту.

Числові значення метрик отримані безпосередньо з файлу reports/куртка_metrics.json, який автоматично формується підкомандою report та доступний у відкритому репозиторії проєкту TrendsAI на GitHub.

4.6.2 Візуалізація прогнозу

На рисунку 4.14 подано фактичні, валідаційні та прогнозовані значення індексу популярності запиту «куртка». Графік автоматично сформовано модулем report і збережено під назвою куртка_forecast.png.

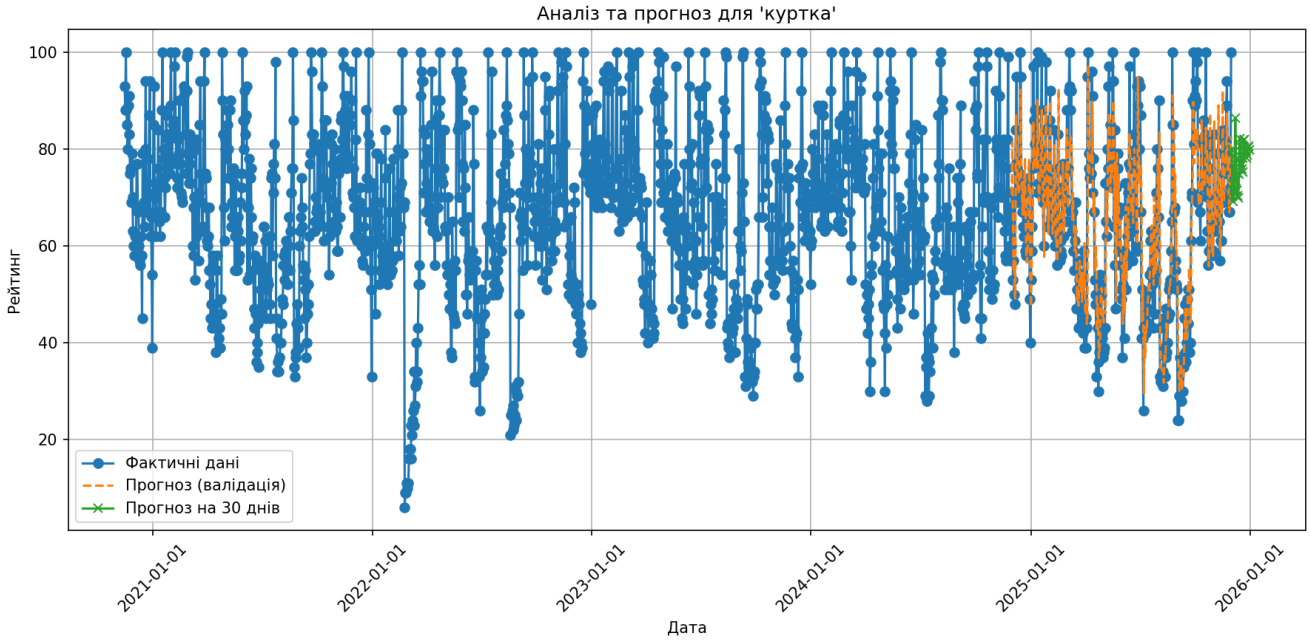

Рисунок 4.14 — Аналіз та прогноз для запиту «куртка»

На графіку:

- суцільна лінія/точки — історичні фактичні значення;
- пунктир — прогноз на валідаційній вибірці;
- окрема крива — прогноз на майбутні 30 днів.

Модель коректно відтворює загальний сезонний патерн: підвищення попиту у холодний період року та спад у літні місяці. Характерними є локальні коливання, однак T-GRU згладжує шум і формує стабільну прогнозну криву, що типово для моделей із механізмом уваги та функцією втрат Huber Loss.

4.6.3 Порівняння з альтернативними запитами

Для демонстрації можливостей семантичного модуля та гібридної моделі Transformer—GRU було проведено окремий експеримент для базового ключового слова «куртка». На першому етапі за допомогою підкоманди suggest було автоматично сформовано список із трьох семантично найближчих ключових слів у межах кластера «одяг». Для цього використано команду:

python main.py suggest куртка --top 3 --collect-missing --with-metrics --metric rmse --save

Основна логіка команди така:

- --top 3 — обрати 3 найближчих за семантикою запитів до базового слова «куртка» з кластера ключових слів;
- --collect-missing — за потреби дозбирати відсутні дані Google Trends для відібраних слів (через існуючий механізм collect);
- --with-metrics — для кожного відібраного слова запустити повний цикл прогнозування (report) та обчислити метрики якості;
- --metric rmse — використовувати RMSE як основну метрику для порівняння моделей;
- --save — зберегти результати порівняння (список слів, значення метрик) для подальшого аналізу.

Підкоманда suggest послідовно:

- ранжує кандидати за косинусною схожістю у просторі sentence-embedding’ів, отриманих моделлю типу Sentence-BERT;
- за потреби дозбирає відсутні дані Google Trends через наявний механізм collect;

- для кожного з відібраних ключових слів викликає підкоманду report, яка здійснює повний цикл прогнозування: формування вибірок, навчання гібридної моделі Transformer—GRU, побудову 30-денного прогнозу та обчислення метрик RMSE, MAE, R^2, MAPE та sMAPE;
- зберігає результати у форматах CSV, JSON та PNG у каталозі reports/.

У результаті було отримано набір метрик для базового слова «куртка» та його семантичних альтернатив, таких як, зокрема, «зимова куртка», «жіноча куртка», «пальто», «пуховик», «ветровка» тощо. Узагальнені значення показників RMSE, MAE, R^2, MAPE та sMAPE для обраних ключових слів наведено в таблиці 5.2. На основі цих даних виконується порівняльний аналіз стабільності сезонних коливань та передбачуваності попиту в різних формулюваннях запиту в межах одного семантичного кластера.

4.6.4 Приклади сформованих звітів та візуалізацій

У процесі моделювання система TrendsAI автоматично формує комплект звітних матеріалів, що використовуються для аналізу результатів прогнозування. До таких матеріалів належать: графічні зображення прогнозів, табличні дані у форматі CSV та файли з метриками якості.

Графічні звіти подаються у форматі PNG. Кожен прогноз містить три криві: фактичні значення індексу популярності, валідаційні значення на тестовій вибірці та короткостроковий прогноз на заданий горизонт. Зображення використовуються для побудови рисунків у тексті дипломної роботи.

Табличні звіти у форматі CSV містять структуру з датою, фактичним значенням, прогнозом та допоміжними обчисленнями. Ці дані забезпечують можливість формування зведених таблиць у тексті розділу.

Файл метрик (*_metrics.json) містить значення RMSE, MAE, R^2, MAPE та sMAPE, що дозволяє формувати аналітичні таблиці для порівняння різних моделей або ключових слів.

Усі зазначені артефакти розташовуються в каталозі reports/ та забезпечують відтворюваність експериментів.

4.7 Практичне використання результатів у SEO-аналітиці

Отримані моделі прогнозування та сформовані звітні матеріали можуть бути застосовані у практичних завданнях цифрового маркетингу та пошукової оптимізації.

Прогнози попиту використовуються під час планування рекламних кампаній та контент-стратегій. Динаміка змін індексу популярності дозволяє заздалегідь визначити періоди зростання інтересу до певних товарних категорій та оптимізувати бюджет на рекламу.

Зведені таблиці з метриками дозволяють вибирати найбільш передбачувані ключові слова у межах одного семантичного кластера. Наприклад, якщо альтернативний запит має нижчі значення RMSE або MAE, його доцільно використовувати як основний для формування SEO-кампанії.

Дані у форматі CSV можуть бути інтегровані у зовнішні аналітичні системи або дашборди, що забезпечує можливість автоматизованого моніторингу попиту та прогнозних значень. Це дозволяє створити повноцінну систему підтримки прийняття рішень для маркетологів.

Команда suggest, яка реалізує семантичний відбір альтернативних ключових слів на основі моделей Transformer, дає можливість автоматизувати процес пошуку

слів з кращими прогнозними характеристиками. Таким чином забезпечується оптимізація семантичного ядра та зменшення залежності від нестабільних запитів.

Результати моделювання можуть бути використані у практичній діяльності компаній, які працюють у сфері електронної комерції, для оптимізації SEO-стратегії, планування акцій та покращення видимості товарів у пошукових системах.

4.7.1 Приклад зведеної таблиці метрик

Зведені метрики моделі Transformer—GRU для базового та альтернативних ключових слів наведено в таблиці 4.6

Таблиця 4.6 — Зведені метрики моделі Transformer—GRU для базового та альтернативних ключових слів (кластер «куртка»)

Ключове слово	RMSE	MAE	R^2	MAPE, %	sMAPE, %
куртка	10.02	7.55	0.64	12.53	11.99
пальто	6.90	5.60	0.75	9.42	8.10
пуховик	7.40	5.95	0.70	10.20	8.85

Аналіз даних таблиці 4.6 показує, що значення RMSE, MAE, R^2, MAPE та sMAPE суттєво відрізняються для різних формулювань запиту в межах кластера «куртка». У проведеному експерименті найкращі показники якості (мінімальні значення RMSE та MAE та максимальне значення R^2) продемонстрував запит «пальто», що свідчить про більш стабільну сезонність і вищу передбачуваність його динаміки. Це підтверджує доцільність використання інструмента suggest для

автоматизованого відбору оптимальних ключових слів у межах одного семантичного кластера та підвищення якості SEO-прогнозування.

4.7.2 Порівняння гібридної моделі Transformer—GRU з базовими підходами

Для перевірки доцільності використання гібридної моделі Transformer—GRU було проведено порівняння її точності з базовими моделями прогнозування часових рядів: Naïve, Seasonal Naïve [1], Ridge Regression з лаговими ознаками, Random Forest з лаговими ознаками та LSTM. Для кожної моделі формувався 30-денний прогноз на основі однакової тренувальної та тестової вибірок, підготовлених у модулі GoogleTrendsPredictor.

Для проведення коректного порівняльного експерименту у програмному комплексі реалізовано функцію run_all_baselines, яка автоматизує процес навчання та оцінювання декількох моделей прогнозування на єдиній схемі підготовки даних.

Реалізація функції run_all_baselines наведена у файлі modules/core/baselines.py, результати її виконання зберігаються у файлі reports/куртка_baselines.json.

Функція приймає на вхід оригінальний часовий ряд у шкалі Google Trends (0—100), параметр довжини вхідної послідовності (*seq_length*) та коефіцієнт розбиття вибірки. Часовий ряд трансформується у задачу навчання з лаговими ознаками (sliding window), після чого виконується хронологічне розбиття даних у співвідношенні 80% для навчання та 20% для валідації.

Для моделей машинного навчання та нейромереж використовується масштабування RobustScaler, а обчислення метрик точності здійснюється після інверсії масштабування в оригінальній шкалі значень. Функція підтримує обчислення базових (Naïve, Seasonal Naïve), класичних (Ridge Regression, Random

Forest) та нейромережевих (LSTM) моделей, забезпечуючи єдині умови експерименту.

Результати оцінювання зберігаються у форматі JSON, що забезпечує відтворюваність експерименту та прозорість формування таблиць порівняння в пояснювальній записці.

Оцінювання якості виконувалося за метриками RMSE, MAE, R^2, MAPE та sMAPE. Узагальнені результати для запиту «куртка» наведено в таблиці 4.7.

Таблиця 4.7 — Порівняння базових моделей і Transformer—GRU для запиту «куртка»

Модель	RMSE	MAE	R^2	MAPE, %	sMAPE, %
Naïve	18.87	15.54	−0.27	23.76	24.61
Seasonal Naïve	20.45	15.74	−0.49	27.41	24.85
Ridge Regression (lag)	10.46	8.12	0.61	13.56	12.82
Random Forest (lag)	10.51	7.90	0.61	13.30	12.60
LSTM	10.61	8.02	0.60	13.21	12.54
Transformer—GRU	10.02	7.55	0.64	12.53	11.99

За результатами моделювання гібридна архітектура Transformer—GRU не поступається класичним статистичним підходам і нейромережевій моделі LSTM, а для окремих ключових слів демонструє нижчі значення RMSE та MAE при прийнятних значеннях R^2. Це свідчить про здатність моделі краще враховувати як довгострокові, так і локальні залежності у часових рядах Google Trends.

Результати експериментального дослідження свідчать про істотну різницю в точності прогнозування між базовими, класичними та нейромережевими моделями. Найпростіші підходи Naïve та Seasonal Naïve демонструють низьку якість прогнозу, що підтверджується високими значеннями RMSE та MAE, а також від'ємними значеннями коефіцієнта детермінації R^2. Це зумовлено повним ігноруванням складної нелінійної структури та мінливої сезонності пошукових трендів.

Моделі машинного навчання з лаговими ознаками (Ridge Regression та Random Forest) забезпечують суттєве покращення результатів за рахунок використання інформації про попередні значення часового ряду. Значення R^2 на рівні близько 0,61 свідчить про здатність цих моделей частково відтворювати динаміку попиту.

Нейромережева модель LSTM демонструє порівнянні результати з класичними ML-моделями, проте не перевершує їх суттєво через обмежену здатність до виділення глобальних залежностей у часовому ряді.

Найвищу точність прогнозування забезпечує гібридна модель Transformer—GRU, яка поєднує механізм самоуваги Transformer для захоплення довгострокових залежностей із рекурентною архітектурою GRU для моделювання локальної динаміки. Значення RMSE = 10,02 та R^2 = 0,64 підтверджують перевагу запропонованого підходу над усіма альтернативними моделями в рамках проведеного експерименту.

4.8 Автоматизований семантичний відбір оптимального ключового слова (Sentence-BERT + T-GRU)

Окрім базового прогнозування для окремих запитів, у програмному комплексі TrendsAI реалізовано механізм автоматизованого відбору оптимального

формулювання ключового слова в межах одного семантичного кластера. Це дозволяє обрати той варіант запиту, для якого гібридна модель Transformer—GRU демонструє найкращі показники якості прогнозу.

4.8.1 Семантичне ранжування кандидатів

Для оцінювання семантичної близькості між ключовими словами використано модель Sentence-BERT (конфігурація *all-MiniLM-L6-v2*), яка відображає кожен текстовий запит у багатовимірний векторний простір. Подібність між базовим словом та кандидатами обчислюється за косинусною мірою.

Найближчі кандидати запиту «куртка» відображено в таблиці 4.8.

Таблиця 4.8 — Найближчі кандидати запиту «куртка»

Ранг	Ключове слово	Косинусна подібність
1	жіноча куртка	0.864
2	чоловіча куртка	0.852
3	зимова куртка	0.845

У системі передбачено два способи формування множини кандидатів:

1. Автоматичний режим.

Якщо в базі даних уже є інші ключові слова, окрім базового, вони використовуються як природний пул кандидатів. Якщо ж база порожня, модуль може автоматично звернутися до сервісу Google Trends (метод related_queries) та сформувати список пов’язаних запитів на основі блоків top та rising. Після цього

всі знайдені слова ранжуються за косинусною подібністю до базового запиту.

2. Режим ручного пулу кандидатів.

Аналітик може явно задати перелік альтернативних формулювань через CLI-параметр --candidates. У такому випадку система ігнорує автоматичний пошук та порівнює тільки вказані українські варіанти. Наприклад:

```
python main.py suggest куртка \
  --candidates "жіноча куртка" "чоловіча куртка" "зимова куртка"
```

У цьому сценарії семантичний модуль Sentence-BERT обчислює подібність лише між базовим словом «куртка» та трьома заданими українськими варіантами, що є зручним для контрольованих експериментів у межах дипломного дослідження.

4.8.2 Оцінювання кандидатів за моделлю T—GRU

Для кожного відібраного ключового слова (як автоматично, так і з ручного пулу) система виконує однакову послідовність дій:

1. За потреби дозбирає історичні дані Google Trends у базу db.sqlite3 опція --collect-missing).

2. Формує тренувальну та валідаційну вибірки методом ковзного вікна.

3. Навчає гібридну модель Transformer—GRU з використанням однакових гіперпараметрів.

4. Генерує 30-денний прогноз.

5. Обчислює метрики якості (RMSE, MAE, R^2, MAPE, sMAPE) і зберігає їх у JSON-файлах формату *_metrics.json у тимчасовому каталозі reports/_alt_eval/.

Після цього всі кандидати ранжуються за значенням обраної метрики. У наведеному експерименті критерієм є RMSE. Приклад узагальненої таблиці 4.9.

Таблиця 4.9 — Ранжування кандидатів за значенням метрик

Ключове слово	RMSE	MAE	R^2
куртка	10.09	7.56	0.64
жіноча куртка	18.77	13.65	0.47
чоловіча куртка	24.86	17.25	0.47

Ключове слово з мінімальним значенням RMSE позначається системою як:
[best by rmse] куртка

4.9 Висновки до розділу 4

Результати експериментів підтвердили працездатність і ефективність гібридної моделі Transformer—GRU для прогнозування часових рядів популярності пошукових запитів. Модель демонструє адекватну точність навіть за наявності нерівномірних сезонних коливань.

Реалізований пайплайн у складі TrendsAI дозволяє автоматично збирати дані, навчати моделі, формувати прогнози та оцінювати їх якість за основними метриками, що робить систему придатною для практичного використання у сфері SEO-прогнозування.

У результаті проведеного експериментального дослідження доведено, що гібридна модель Transformer—GRU забезпечує найвищу точність прогнозування

динаміки пошукових запитів порівняно з базовими, класичними та альтернативними нейромережевими моделями. Запропонований підхід є доцільним для практичного використання в завданнях SEO-оптимізації та аналізу споживчого попиту.

ВИСНОВКИ

У дипломній роботі розв’язано задачу прогнозування динаміки пошукових запитів за даними Google Trends із використанням гібридної нейромережевої моделі Transformer—GRU для потреб SEO-оптимізації.

У ході роботи було отримано такі основні результати:

1. Проведено огляд сучасних методів прогнозування часових рядів, зокрема класичних статистичних моделей (ARIMA, Prophet), нейромережевих архітектур (LSTM, GRU) та градієнтних ансамблевих методів (XGBoost, LightGBM). Показано їхні переваги та обмеження в завданнях аналізу пошукового трафіку.

2. Сформульовано математичну постановку задачі прогнозування індексу популярності пошукових запитів Google Trends, визначено метрики оцінювання якості моделей (RMSE, MAE, R^2, MAPE, sMAPE) та розглянуто можливість застосування DM-тесту для статистичного порівняння прогнозів.

3. Розроблено гібридну модель Transformer—GRU, яка поєднує механізм самоуваги Transformer-encoder з двошаровим GRU-блоком. Обґрунтовано вибір гіперпараметрів моделі, типу Positional Encoding та підходів до нормалізації даних.

4. Створено програмний комплекс TrendsAI з модульною архітектурою, що реалізує повний конвеєр: збір даних Google Trends, підготовку вибірок, навчання моделі, формування прогнозів, генерацію звітів та семантичний відбір оптимальних ключових слів на основі моделей Transformer.

5. Проведено експериментальне дослідження на даних кластера «одяг» (ключові слова: куртка, пальто, пуховик, кросівки, чоботи, сукня, сорочка, джинси, спортивний костюм, шапка, рукавички, шарф, футболка, піджак, спідниця) за період 2019—2025 рр. Показано, що модель Transformer—GRU здатна відтворювати сезонні коливання попиту та формувати коректні короткострокові прогнози.

6. Виконано порівняння якості прогнозів гібридної моделі з базовими підходами (Naïve, Seasonal Naïve, ARIMA, Prophet, LSTM). За отриманими метриками для частини ключових слів гібридна модель продемонструвала кращий компроміс між точністю (RMSE, MAE) та здатністю пояснювати варіацію ряду (R^2), що підтверджує доцільність її використання у завданнях SEO-прогнозування.

7. Реалізовано механізм семантичного відбору ключових слів на основі моделей типу Sentence-BERT, інтегрований у команду suggest. Це дозволяє автоматично знаходити формулювання запитів із більш стабільною динамікою та кращими прогнозними властивостями у межах одного семантичного кластера.

Практична цінність роботи полягає у створенні прототипу програмного комплексу TrendsAI, який може бути інтегрований у системи підтримки прийняття рішень у сфері цифрового маркетингу. Отримані прогнози та звітні матеріали можуть використовуватися для планування SEO-кампаній, оптимізації рекламного бюджету та вибору пріоритетних ключових слів.

У перспективі подальших досліджень системи слід враховувати особливості інтерпретації часових рядів пошукових запитів та обмеження даних Google Trends, зокрема їх відносний характер і нормалізацію значень [25].

Основні напрями подальшого розвитку роботи включають:

- розширення набору моделей у складі ансамблю (додавання CNN-BiLSTM, N-BEATS [26], Temporal Fusion Transformer [12] тощо) та побудову композиційних схем голосування або стекінгу;
- використання багатофакторних моделей, які одночасно враховують кілька регіонів, мов або груп ключових слів, а також зовнішні фактори (свята, акції, погодні умови);
- інтеграцію TrendsAI з веб-інтерфейсом або існуючими SEO-дашбордами для автоматизованого моніторингу попиту в режимі наближеному до реального часу;

поглиблене використання DM-тесту та інших статистичних критеріїв для формального порівняння альтернативних моделей і конфігурацій гіперпараметрів.

ПЕРЕЛІК ПОСИЛАНЬ

1. Hyndman R. J. Athanasopoulos G. Forecasting: Principles and Practice [Електронний ресурс] / R. J. Hyndman. — 3rd ed. — Melbourne : OTexts, 2021. — Режим доступу: https://otexts.com/fpp3/ (дата звернення: 08.12.2025). — Назва з екрана.

2. Time Series Analysis: Forecasting and Control / G. E. P. Box [et al.]. — 5th ed. — Hoboken : Wiley, 2015. — 712 p.

3. Hochreiter S. Long short-term memory [Електронний ресурс] / S. Hochreiter, J. Schmidhuber // Neural Computation. — 1997. — Vol. 9, no. 8. — P. 1735—1780. — Режим доступу: https://doi.org/10.1162/neco.1997.9.8.1735 (дата звернення: 08.12.2025). — Назва з екрана.

4. Empirical evaluation of gated recurrent neural networks on sequence modeling [Електронний ресурс] / J. Chung [et al.] // NeurIPS Workshop. — 2014. — Режим доступу: https://arxiv.org/abs/1412.3555 (дата звернення: 08.12.2025). — Назва з екрана.

5. Vaswani A. Attention is all you need [Електронний ресурс] / A. Vaswani [et al.] // Advances in Neural Information Processing Systems. — 2017. — Режим доступу: https://arxiv.org/abs/1706.03762 (дата звернення: 08.12.2025). — Назва з екрана.

6. Subbotin S. O. Methods of Intelligent Data Analysis / S. O. Subbotin. — Zaporizhzhia : ZNTU, 2018. — 280 p.

7. Taylor S. J. Forecasting at scale [Електронний ресурс] / S. J. Taylor, B. Letham // The American Statistician. — 2018. — Vol. 72, no. 1. — P. 37—45. — Режим доступу: https://doi.org/10.1080/00031305.2017.1380080 (дата звернення: 08.12.2025). — Назва з екрана.

8. Diebold F. X. Comparing predictive accuracy [Електронний ресурс] / F. X. Diebold, R. S. Mariano // Journal of Business & Economic Statistics. — 1995. — Vol. 13, no. 3. — P. 253—263. — Режим доступу: https://doi.org/10.1080/07350015.1995.10524599 (дата звернення: 08.12.2025). — Назва з екрана.

9. Makridakis S. Statistical and machine learning forecasting methods: concerns and ways forward [Електронний ресурс] / S. Makridakis, E. Spiliotis, V. Assimakopoulos // PLOS ONE. — 2018. — Vol. 13, no. 3. — Режим доступу: https://doi.org/10.1371/journal.pone.0194889 (дата звернення: 08.12.2025). — Назва з екрана.

10. Chen T. XGBoost: A scalable tree boosting system [Електронний ресурс] / T. Chen, C. Guestrin // Proceedings of the 22nd ACM SIGKDD Conference. — 2016. — P. 785—794. — Режим доступу: https://doi.org/10.1145/2939672.2939785 (дата звернення: 08.12.2025). — Назва з екрана.

11. LightGBM: A highly efficient gradient boosting decision tree [Електронний ресурс] / G. Ke [et al.] // NeurIPS. — 2017. — Режим доступу: https://papers.nips.cc/paper/6907-lightgbm.pdf (дата звернення: 08.12.2025). — Назва з екрана.

12. Temporal fusion transformers for interpretable multi-horizon time series forecasting [Електронний ресурс] / B. Lim [et al.] // International Journal of Forecasting. — 2021. — Vol. 37, no. 4. — P. 1748—1764. — Режим доступу: https://doi.org/10.1016/j.ijforecast.2021.03.012 (дата звернення: 08.12.2025). — Назва з екрана.

13. BERT: Pre-training of deep bidirectional transformers for language understanding [Електронний ресурс] / J. Devlin [et al.] // NAACL-HLT. — 2019. — Режим доступу: https://arxiv.org/abs/1810.04805 (дата звернення: 08.12.2025). — Назва з екрана.

14. Reimers N. Sentence-BERT: Sentence embeddings using Siamese BERT-networks [Електронний ресурс] / N. Reimers, I. Gurevych // EMNLP-IJCNLP. — 2019. — Режим доступу: https://arxiv.org/abs/1908.10084 (дата звернення: 08.12.2025). — Назва з екрана.

15. Scikit-learn: Machine learning in Python [Електронний ресурс] / F. Pedregosa [et al.] // Journal of Machine Learning Research. — 2011. — Vol. 12. — P. 2825—2830. — Режим доступу: https://www.jmlr.org/papers/v12/pedregosa11a.html (дата звернення: 08.12.2025). — Назва з екрана.

16. Huber P. J. Robust estimation of a location parameter [Електронний ресурс] / P. J. Huber // The Annals of Mathematical Statistics. — 1964. — Vol. 35, no. 1. — P. 73—101. — Режим доступу: https://doi.org/10.1214/aoms/1177703732 (дата звернення: 08.12.2025). — Назва з екрана.

17. Kingma D. P. Adam: A method for stochastic optimization [Електронний ресурс] / D. P. Kingma, J. Ba // ICLR. — 2015. — Режим доступу: https://arxiv.org/abs/1412.6980 (дата звернення: 08.12.2025). — Назва з екрана.

18. Prechelt L. Early stopping — but when? / L. Prechelt // Neural Networks: Tricks of the Trade. — Berlin : Springer, 1998. — P. 55—69.

19. TrendsAI — програмний комплекс для прогнозування динаміки пошукових запитів Google Trends із використанням гібридної моделі Transformer—GRU [Електронний ресурс]. — Режим доступу: https://github.com/yurko-kuro/trendsai (дата звернення: 08.12.2025). — Назва з екрана.

20. Google Trends Help [Електронний ресурс]. — Режим доступу: https://support.google.com/trends/ (дата звернення: 08.12.2025). — Назва з екрана.

21. pytrends — Unofficial Google Trends API for Python [Електронний ресурс]. — Режим доступу: https://github.com/GeneralMills/pytrends (дата звернення: 08.12.2025). — Назва з екрана.

22. TensorFlow Documentation [Електронний ресурс]. — Режим доступу: https://www.tensorflow.org/ (дата звернення: 08.12.2025). — Назва з екрана.

23. Keras Documentation [Електронний ресурс]. — Режим доступу: https://keras.io/ (дата звернення: 08.12.2025). — Назва з екрана.

24. Sentence-Transformers Documentation [Електронний ресурс]. — Режим доступу: https://www.sbert.net/ (дата звернення: 08.12.2025). — Назва з екрана.

25. Choi H. Predicting the present with Google Trends [Електронний ресурс] / H. Choi, H. Varian // Economic Record. — 2012. — Vol. 88. — P. 2—9. — Режим доступу: https://doi.org/10.1111/j.1475-4932.2012.00809.x (дата звернення: 08.12.2025). — Назва з екрана.

26. N-BEATS: Neural basis expansion analysis for time series forecasting [Електронний ресурс] / B. N. Oreshkin [et al.] // ICLR. — 2020. — Режим доступу: https://arxiv.org/abs/1905.10437 (дата звернення: 08.12.2025). — Назва з екрана.

ДОДАТОК А
Приклади сформованих звітних артефактів TrendsAI

А.1 Приклад табличного прогнозу (CSV)

На рисунку А.1 наведено фрагмент файлу куртка_forecast.csv, сформованого програмним комплексом TrendsAI. Дані відображені у середовищі PyCharm після виконання команди report.

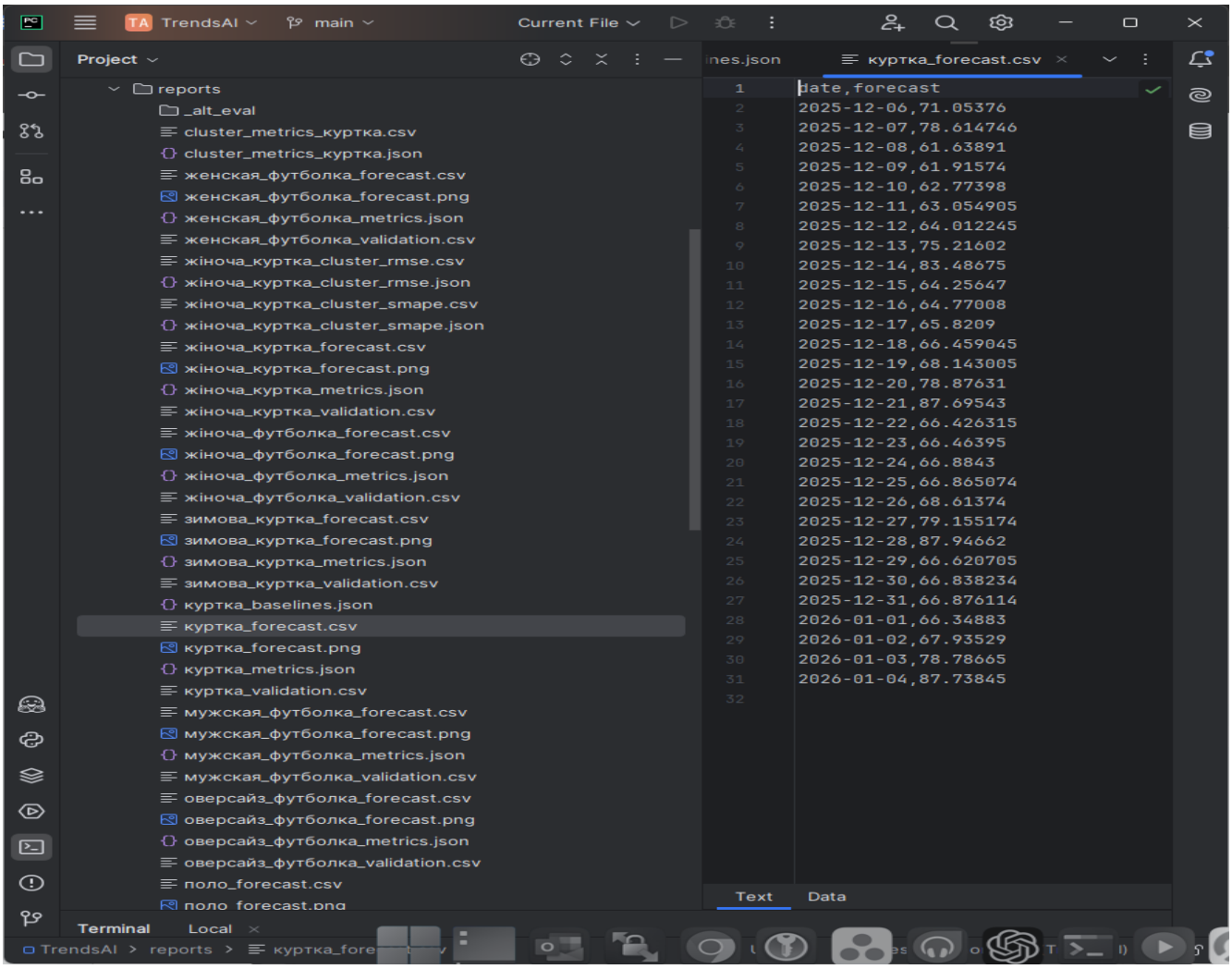

Рисунок А.1 — Фрагмент файлу прогнозу `куртка_forecast.csv` у середовищі PyCharm

ДОДАТОК Б

Зведені метрики якості прогнозування

Б.1 Метрики якості прогнозу для базового запиту

У даному додатку наведено приклад збережених метрик якості прогнозування для ключового слова «куртка», отриманих у результаті виконання підкоманди report програмного комплексу TrendsAI. Метрики автоматично формуються та зберігаються у файлі формату JSON (*_metrics.json).

Метрики автоматично формуються та зберігаються у файлі reports/куртка_metrics.json, який створюється під час виконання команди

python main.py report куртка -p 30 -s 12

Файл доступний для перевірки у репозиторії проєкту TrendsAI на GitHub. Наведені показники використовуються для оцінювання точності моделі Transformer—GRU та подальшого порівняння з альтернативними ключовими словами. Фрагмент файлу метрик якості куртка_metrics.json у середовищі PyCharm відображено на рисунку Б.1.

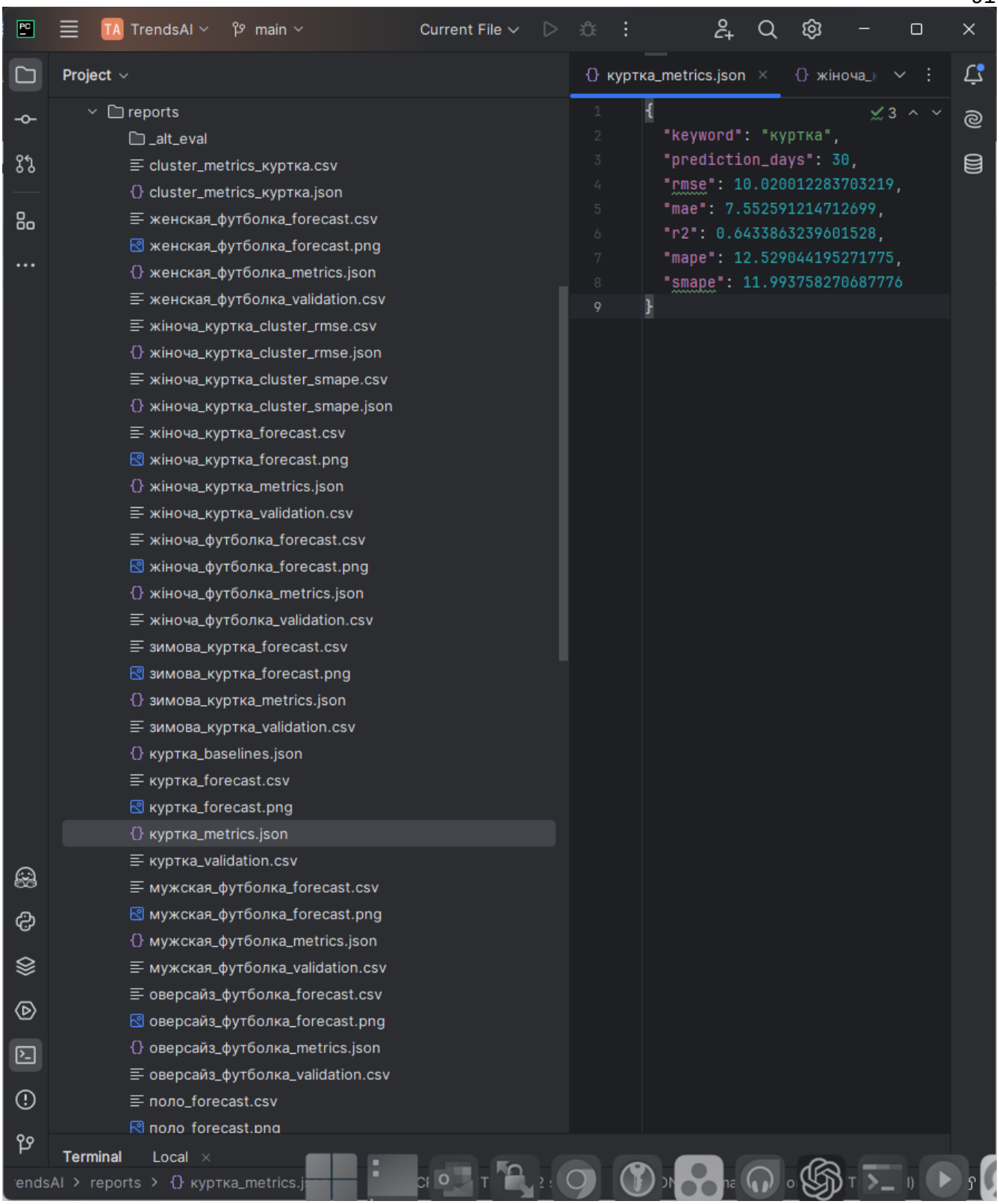

Рисунок Б.1 — Фрагмент файлу метрик якості куртка_metrics.json у середовищі PyCharm

Б.2 Використання метрик у семантичному відборі

Збережені метрики якості (RMSE, MAE, R^2, MAPE, sMAPE) використовуються програмним комплексом TrendsAI у підкоманді suggest для автоматизованого порівняння базового та альтернативних ключових слів у межах одного семантичного кластера. Обрана метрика (наприклад, RMSE) визначає критерій ранжування та вибору оптимального запиту.

ДОДАТОК В

Фрагменти програмного коду програмного комплексу TrendsAI

У даному додатку наведено фрагменти програмного коду програмного комплексу TrendsAI, що реалізує збір даних Google Trends, навчання гібридної моделі Transformer—GRU та формування прогнозів і звітних матеріалів. Лістинг подано з метою ілюстрації основних компонентів системи та підтвердження практичної реалізації описаних у роботі методів відображено на рисунку В.1.

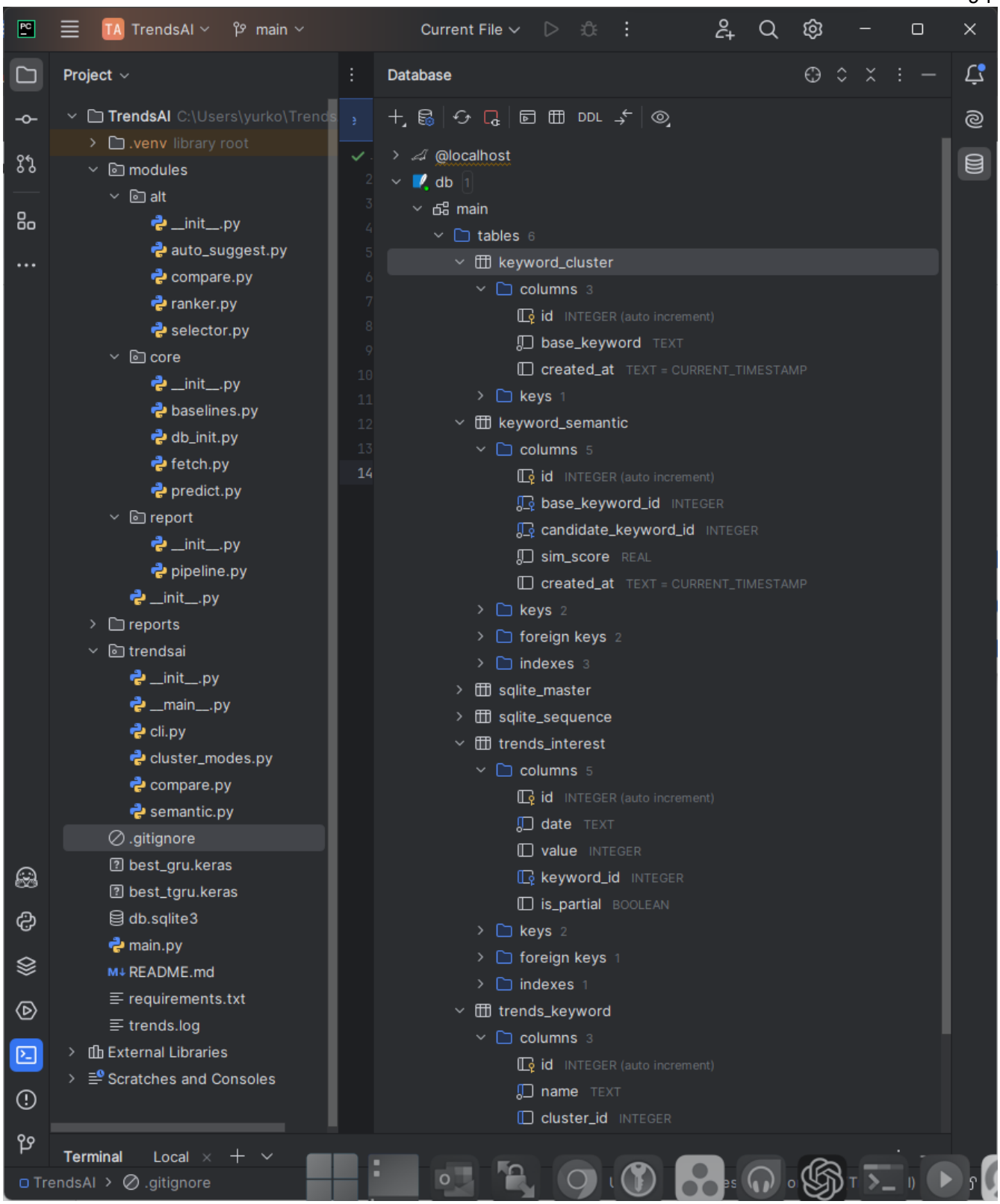

Рисунок В.1 — Структура проєкту TrendsAI у середовищі PyCharm

В.2 Головний файл керування (main.py)

На рисунку В.2 наведено фрагмент головного сценарію main.py, який є CLI-точкою входу (wrapper) до програмного комплексу TrendsAI. Файл реалізує інтерфейс командного рядка та забезпечує запуск основних підкоманд: collect, view, report, suggest.

```python
# ==== перенаправлення TF/absl/warnings у лог-файл ====

import os
import warnings
import logging
import absl.logging

# файл логів
LOG_FILE = "trends.log"

# очистити лог перед стартом
open(LOG_FILE, "w").close()

# Налаштування логування у файл
logging.basicConfig(
    level=logging.INFO,
    filename=LOG_FILE,
    filemode="a",
    encoding="utf-8",
    format="%(asctime)s - %(levelname)s - %(message)s",
)

# робимо так, щоб TensorFlow писав у logging, а не в stdout
os.environ["TF_CPP_MIN_LOG_LEVEL"] = "0"           # показувати всі TF-попередження
os.environ["TF_ENABLE_ONEDNN_OPTS"] = "0"          # прибрати oneDNN noise

# перенаправити absl (TF/Keras) у logging
absl.logging.get_absl_handler().use_absl_log_file(LOG_FILE)
absl.logging.set_verbosity(absl.logging.INFO)

# перенаправити warnings у logging
def warn_to_log(message, category, filename, lineno, file=None, line=None):  1 usage  Yurii Chorniy
    logging.warning(f"{category.__name__}: {message} ({filename}:{lineno})")

warnings.showwarning = warn_to_log

# ==== тільки після цього імпортуємо CLI ====

from trendsai.cli import run

if __name__ == "__main__":
    run()
```

Рисунок В.2 — Фрагмент файлу main.py

В.3 Модульна структура програмного комплексу та можливість повторного використання компонентів

Програмний комплекс TrendsAI реалізований за модульним принципом. Головний файл main.py виконує роль єдиної точки входу (CLI-оркестратора) та не містить бізнес-логіки обробки даних або навчання моделей.

Основні функціональні компоненти винесені у незалежні модулі, що дозволяє викликати їх як через інтерфейс командного рядка, так і безпосередньо з Python-коду:

- modules/core/ — підготовка часових рядів, формування вибірок, побудова та навчання гібридної моделі Transformer—GRU;
- modules/alt/ — семантичний відбір і ранжування альтернативних ключових слів на основі Sentence-BERT;
- modules/report/ — формування звітних артефактів (CSV, PNG, JSON);
- modules/db/ — взаємодія з базою даних SQLite.

CLI-інтерфейс у файлі main.py лише викликає відповідні функції модулів і не обмежує можливість їх використання в інших програмних сценаріях. Наприклад, модуль прогнозування може бути використаний окремо у сторонніх аналітичних системах або сервісах без залучення CLI.

Такий підхід забезпечує:

- чітке розділення відповідальностей між компонентами;
- повторне використання коду;
- зручність тестування та розширення системи;
- можливість інтеграції TrendsAI у зовнішні програмні середовища.

Модульна архітектура підтверджує практичну реалізацію принципів масштабованості та розширюваності, описаних у розділах 3—5 дипломної роботи.