ПОЛТАВСЬКИЙ ДЕРЖАВНИЙ АГРАРНИЙ УНІВЕРСИТЕТ
Навчально-науковий інститут економіки, управління, права та інформаційних технологій
Кафедра інформаційних систем та технологій

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття ступеня вищої освіти магістр

на тему: «**Дослідження ефективності парсингу даних з використанням великих мовних моделей»**

Виконав: здобувач вищої освіти
за освітньою програмою
Інформаційні управляючі системи та технології
спеціальності 126 Інформаційні системи та технології
ступеня вищої освіти магістр
групи 126ІСТ_мд_2024
Левченко Юрій Іванович
Керівник: Флегантов Леонід Олексійович
Рецензент: Ковальчук Станіслав Богданович

Полтава – 2025 року

ПОЛТАВСЬКИЙ ДЕРЖАВНИЙ АГРАРНИЙ УНІВЕРСИТЕТ
Навчально-науковий інститут економіки, управління, права та інформаційних технологій
Кафедра інформаційних систем та технологій

Освітня програма Інформаційні управляючі системи та технології
Спеціальність 126 Інформаційні системи та технології
Рівень вищої освіти другий (магістерський)

ЗАТВЕРДЖУЮ
Завідувач кафедри

_____ Юрій УТКІН

«08» листопада 2024 року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧА ВИЩОЇ ОСВІТИ

Левченка Юрія Івановича

1. Тема кваліфікаційної роботи: «Дослідження ефективності парсингу даних з використанням великих мовних моделей».

Керівник роботи: к.ф.-м.н., доцент, професор кафедри інформаційних систем та технологій Флегантов Леонід Олексійович.

Затверджено наказом закладу вищої освіти від «31» жовтня 2025 року № 1332-ст

2. Строк подання здобувачем вищої освіти роботи «09» грудня 2025 р.

3. Вихідні дані до роботи: науково-технічна література; наукові статті та публікації, доступні в наукометричних базах даних Google Scholar, IEEE Xplore, SpringerLink, ScienceDirect, Scopus, ResearchGate; вітчизняні та міжнародні стандарти та методології; технічна документація сучасних великих мовних моделей (GPT-4, LLaMA 3, Claude 3 тощо), звіти провідних AI-компаній (OpenAI, Meta AI, Anthropic, Google DeepMind та ін.), офіційна документація бібліотек Python для парсингу (BeautifulSoup, Scrapy, Requests, LangChain); інтернет-джерела аналітичного характеру, блоги, статті та технічні звіти експертів у галузі Python-програмування.

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

Розділ 1. Теоретичні та методологічні аспекти парсингу даних з використанням LLM
Розділ 2. Аналітичний огляд методів парсингу даних
Розділ 3. Порівняльний аналіз та розроблення рекомендацій щодо ефективного застосування LLM для парсингу даних

5. Перелік графічного матеріалу: схеми, рисунки, діаграми за темою та об’єктом дослідження.

6. Консультанти розділів кваліфікаційної роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання отримав
Оцінювання економічної ефективності результатів дослідження	Калініченко О. В., к. е. н., доцент, доцент кафедри економіки та публічного управління	24.11.2025	04.12.2025

7. Дата видачі завдання «08» листопада 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Строк виконання етапів кваліфікаційної роботи	Примітка
1.	Вибір і затвердження теми роботи	29.10.2024 р.	
2.	Складання і затвердження розгорнутого плану та завдання на кваліфікаційну роботу	30.10.2024 р. – 08.11.2024 р.	
3.	Опрацювання джерел інформації	11.11.2024 р. – 27.12.2024 р.	
4.	Збір, вивчення і обробка інформації, необхідної для виконання роботи	30.12.2024 р.– 19.01.2025 р.	
5.	Виконання теоретико-методологічного розділу роботи	17.02.2025 р.– 16.05.2025 р.	
6.	Виконання дослідницько-аналітичного розділу роботи	02.06.2025 р.– 13.07.2025 р.	
7.	Виконання проєктно-рекомендаційного розділу роботи	08.09.2025 р.– 14.11.2025 р.	
8.	Оцінювання економічної ефективності результатів дослідження	24.11.2025 р.– 04.12.2025 р.	
9.	Оформлення тексту роботи	05.12.2025 р.– 08.12.2025 р.	
10.	Попередній захист роботи на кафедрі	09.12.2025 р.	
11.	Доопрацювання роботи з урахуванням зауважень і пропозицій	10.12.2025 р.- 14.12.2025 р.	
12.	Нормоконтроль	15.12.2025 р. – 16.12.2025 р.	
13.	Захист кваліфікаційної роботи	18.12.2025 р.	

Здобувач вищої освіти **Юрій ЛЕВЧЕНКО**

Керівник роботи **Леонід ФЛЕГАНТОВ**

ПОЛТАВСЬКИЙ ДЕРЖАВНИЙ АГРАРНИЙ УНІВЕРСИТЕТ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ЕКОНОМІКИ, УПРАВЛІННЯ, ПРАВА ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНФОРМАЦІЙНИХ СИСТЕМ ТА ТЕХНОЛОГІЙ

ЛЕВЧЕНКО ЮРІЙ ІВАНОВИЧ

«ДОСЛІДЖЕННЯ ЕФЕКТИВНОСТІ ПАРСИНГУ ДАНИХ З ВИКОРИСТАННЯМ ВЕЛИКИХ МОВНИХ МОДЕЛЕЙ»

Освітньо-професійна програма
Інформаційні управляючі системи та технології
Спеціальність 126 Інформаційні системи та технології
Ступінь вищої освіти Магістр

РЕФЕРАТ
кваліфікаційної роботи на здобуття кваліфікації –
магістр з інформаційних систем та технологій

Полтава – 2025 року

Кваліфікаційна робота складається зі вступу, трьох розділів, висновків, списку використаних джерел та додатків. Основний текст викладено на 75 сторінках, містить 30 рисунків і 23 таблиці. Список використаних джерел налічує 70 найменувань.

Основний зміст роботи

Теоретична частина роботи присвячена дослідженню теоретичних та методологічних аспектів парсингу даних із використанням великих мовних моделей. Розкрито концепцію семантичного парсингу, що базується на здатності LLM інтерпретувати природну мову та неструктуровані дані з урахуванням контексту й прихованих смислових зв'язків. Проаналізовано принципи побудови та архітектурні особливості сучасних LLM, зокрема трансформерні підходи, механізми уваги та контекстного кодування, а також методи оцінювання якості й ефективності їх застосування у задачах парсингу. Окрему увагу приділено методам навчання та донавчання моделей, включно з fine-tuning, instruction tuning і few-shot learning, а також розглянуто приклади практичного застосування LLM-парсингу у різних галузях, таких як фінанси, медицина, електронна комерція та аналітика великих даних.

Аналітична частина роботи містить системний огляд традиційних методів парсингу даних і їх архітектурних обмежень у порівнянні з можливостями LLM. Досліджено функціональні характеристики класичних підходів на основі регулярних виразів, правил та статистичних моделей, а також їхню чутливість до змін структури вхідних даних. Проведено детальний порівняльний аналіз архітектур традиційних і LLM-орієнтованих методів парсингу за критеріями гнучкості, масштабованості, точності та обчислювальної складності. Окремо розглянуто підходи до інтеграції традиційних і LLM-методів у гібридних системах парсингу та методи їх оптимізації з метою підвищення стабільності й зниження обчислювальних витрат.

Практична та експериментальна частина роботи спрямована на проведення порівняльного експерименту з оцінювання ефективності базових LLM у реальних сценаріях парсингу даних. Розроблено методику експерименту, що включає визначення тестових наборів даних, метрик якості та сценаріїв використання. Виконано порівняння результатів парсингу з використанням різних базових моделей, а також досліджено вплив донавчання LLM для спеціалізованих завдань. Отримані експериментальні результати дозволили сформувати практичні рекомендації щодо вибору та застосування методів парсингу залежно від типу даних і вимог до системи.

Проєктно-рекомендаційний розділ містить техніко-економічне обґрунтування доцільності використання LLM і гібридних підходів у задачах парсингу даних. Показано, що застосування LLM дозволяє суттєво підвищити точність і адаптивність парсингу порівняно з традиційними методами, а також зменшити витрати на підтримку й модифікацію систем у довгостроковій перспективі. Розраховано економічні показники ефективності, які підтверджують практичну доцільність впровадження запропонованих рішень.

У заключних висновках узагальнено результати дослідження та підтверджено, що використання великих мовних моделей і гібридних архітектур є перспективним напрямом розвитку систем парсингу даних, який забезпечує підвищення якості обробки інформації, гнучкість та економічну ефективність у сучасних інформаційних системах.

Висновки

За результатами виконаної кваліфікаційної роботи можна зробити висновки, що охоплюють основні результати та підтверджують досягнення поставленої мети.

1. Визначено, що семантичний парсинг за допомогою LLM базується на архітектурі Transformer і забезпечує глибоке двонаправлене розуміння контексту даних, що є необхідним для вилучення сутностей. Систематизовано основні методи адаптації моделей до завдань парсингу. Обґрунтовано необхідність застосування комплексних критеріїв оцінки ефективності, які включають не лише традиційні метрики машинного навчання (F1-score, Recall), а й показники синтаксичної та семантичної узгодженості вихідних структур, що є обов'язковим для інтеграції LLM-парсерів в автоматизовані інформаційні системи.

2. Проведено порівняльний аналіз традиційних детермінованих методів парсингу та LLM-архітектур. Встановлено, що традиційні методи ефективні та економічно доцільні для стабільних, структурованих джерел, однак їх головною вадою є архітектурна крихкість при змінах верстки або структури файлу. Натомість, LLM-підходи забезпечують високу адаптивність та відмовостійкість, перетворюючи парсинг на задачу семантичного аналізу. На основі цього визначено, що гібридні архітектури (інтеграція традиційних методів для попередньої обробки та LLM для семантичного вилучення) є оптимальною стратегією, що дозволяє збалансувати швидкість, точність та витрати.

3. Розроблено та реалізовано методику порівняльного експерименту з оцінки ефективності різних LLM (GPT-3.5, T5, LLaMA 2) у сценаріях парсингу HTML, JSON та XML. Експериментально підтверджено, що Supervised Fine-tuning є найкращим методом для досягнення максимальної якості вилучення даних (F1-score до 0,91). Техніко-економічне обґрунтування підтвердило доцільність запропонованих рішень: LLM-парсери забезпечують максимальну точність, традиційний парсинг – максимальну економічну ефективність, а гібридні системи парсингу мають оптимальний баланс показників при великих потоках даних. Сформовано деталізовані рекомендації щодо вибору методу парсингу залежно від структури вхідних даних (структуровані, напівструктуровані, неструктуровані).

Практична значущість роботи полягає у можливості впровадження розроблених рекомендацій та гібридних архітектур для оптимізації процесів збору, очищення та обробки даних у компаніях, що дозволить підвищити надійність ETL-процесів, зменшити помилки парсингу та оптимізувати витрати на обчислювальні ресурси.

Перспективи подальших досліджень полягають у розширенні можливостей автоматизації LLM-парсингу за допомогою агентних систем (Agentic LLMs) та неперервного навчання (continual learning) для безперервної адаптації моделей до нових форматів даних без необхідності повного повторного донавчання, а також у детальному дослідженні методів Explainable AI (XAI) для підвищення прозорості та довіри до результатів семантичного парсингу.

Список публікацій здобувача

1. Левченко Ю. Техніки оптимізації LLM-парсингу. *Матеріали науково-практичної конференції за підсумками проходження виробничих практик здобувачів вищої освіти спеціальності 126 Інформаційні системи та технології*, кафедра інформаційних систем та технологій Полтавського державного аграрного університету, 22 жовтня 2025 р. Вип. XI. Полтава: ПДАУ, 79 с. С. 66-68.

2. Левченко Ю. Архітектурні особливості великих мовних моделей у контексті парсингу даних. *Студентські роботи за науковою тематикою кафедри інформаційних систем та технологій*: матеріали XXII щорічного міждисциплінарного семінару, 25 листопада 2025 р. Полтава: ПДАУ, 2025 р. 120 с. С. 62-64.

3. Флегантов Л., Левченко Ю. Принципи та архітектури LLM для парсингу даних. *The Future of Science, Technology and Economy: Collection of Scientific Papers with Proceedings of the 3rd International Scientific and Practical Conference*. International Scientific Unity. October 29-31, 2025. Sofia, Bulgaria. 164-169 p. *Isu-conference*: вебсайт. URL: https://isu-conference.com/en/archive/the-future-of-science-technology-and-economy-29-10-25/

4. Флегантов Л., Масич А., Левченко Ю. Архітектурні та функціональні особливості провідних моделей Reasoning-LLM. *Progressive Approaches in Science and Engineering: Collection of Scientific Papers with Proceedings of the 2nd International Scientific and Practical Conference. International Scientific Unity*. November 26-28, 2025. Copenhagen, Denmark. 338-344 p. URL: https://isu-conference.com/arkhiv/progressive-approaches-in-science-and-engineering-26-11-25/

АНОТАЦІЯ

Левченко Ю. І. «Дослідження ефективності парсингу даних з використанням великих мовних моделей». Кваліфікаційна робота на правах рукопису.

Кваліфікаційна робота на здобуття ступеня вищої освіти магістр за освітньо-професійною програмою Інформаційні управляючі системи та технології спеціальності 126 Інформаційні системи та технології. Полтавський державний аграрний університет, Полтава, 2025.

Робота присвячена вирішенню актуальної задачі підвищення ефективності парсингу неструктурованих та слабкоструктурованих даних із використанням великих мовних моделей. Досліджено теоретичні основи семантичного парсингу та обґрунтовано доцільність застосування LLM як універсального інструменту аналізу текстових даних. Робота складається з трьох розділів, що охоплюють теоретико-методологічні аспекти, аналітичний огляд існуючих методів та експериментальне дослідження з формуванням практичних рекомендацій.

У першому розділі розглянуто концепцію семантичного парсингу, принципи побудови та архітектурні особливості LLM, методи оцінювання їх якості та підходи до навчання і донавчання моделей для задач парсингу. Другий розділ присвячено аналізу традиційних методів парсингу та порівнянню їх можливостей із підходами на основі LLM, а також дослідженню гібридних архітектур і шляхів їх оптимізації. У третьому розділі наведено методику та результати порівняльного експерименту, виконано оцінювання ефективності базових і донавчених LLM у реальних сценаріях парсингу, сформульовано практичні рекомендації та виконано техніко-економічне обґрунтування доцільності застосування запропонованих рішень.

Результати дослідження мають практичну цінність для розробників інформаційних систем та аналітичних платформ, забезпечуючи підвищення точності, гнучкості та економічної ефективності процесів парсингу даних.

Ключові слова: парсинг даних, великі мовні моделі, LLM, семантичний парсинг, гібридні методи, донавчання моделей, обробка природної мови, оптимізація.

ANNOTATION

Levchenko Y. I. «A Study of the Effectiveness of Data Parsing Using Large Language Models». Master’s thesis manuscript.

The master’s thesis for obtaining a Master’s degree in the educational and professional program of Information Management Systems and Technologies, specialty 126 Information Systems and Technologies. Poltava State Agrarian University, Poltava, 2025.

The thesis is devoted to solving a relevant problem of improving the efficiency of parsing unstructured and semi-structured data using large language models. The theoretical foundations of semantic parsing are investigated, and the feasibility of using LLMs as a universal tool for textual data analysis is substantiated. The thesis consists of three chapters covering theoretical and methodological aspects, an analytical review of existing methods, and an experimental study with the development of practical recommendations.

The first chapter examines the concept of semantic parsing, the principles of construction and architectural features of LLMs, methods for evaluating their quality, and approaches to training and fine-tuning models for parsing tasks. The second chapter is devoted to the analysis of traditional parsing methods and a comparison of their capabilities with LLM-based approaches, as well as the study of hybrid architectures and ways to optimize them. The third chapter presents the methodology and results of a comparative experiment, evaluates the effectiveness of base and fine-tuned LLMs in real-world parsing scenarios, formulates practical recommendations, and provides a technical and economic justification for the feasibility of applying the proposed solutions.

The research results have practical value for developers of information systems and analytical platforms, ensuring increased accuracy, flexibility, and economic efficiency of data parsing processes.

Keywords: data parsing, large language models, LLM, semantic parsing, hybrid methods, model fine-tuning, natural language processing, optimization.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

AI	Artificial Intelligence	Штучний інтелект
API	Application Programming Interface	Інтерфейс прикладного програмування
BERT	Bidirectional Encoder Representations from Transformers	Двонапрямлені представлення кодувальника трансформерів
CLM	Causal Language Modeling	Каузальне (причинне) мовне моделювання
CoT	Chain-of-Thought	Ланцюжок міркувань (метод покрокових міркувань)
CSS	Cascading Style Sheets	Каскадні таблиці стилів
CSV	Comma-Separated Values	Значення, розділені комою
DOM	Document Object Model	Об’єктна модель документа
ETL	Extract, Transform, Load	Вилучення, перетворення, завантаження
F1	F1-score	F1-оцінка (гармонічне середнє точності та повноти)
GPT	Generative Pre-trained Transformer	Генеративний попередньо навчений трансформер
GPU	Graphics Processing Unit	Графічний процесор
HTML	HyperText Markup Language	Мова розмітки гіпертексту
JSON	JavaScript Object Notation	Об’єктна нотація JavaScript
LLaMA	Large Language Model Meta AI	Велика мовна модель від Meta AI
LLM	Large Language Model	Велика мовна модель
MLM	Masked Language Modeling	Масковане мовне моделювання
NER	Named Entity Recognition	Розпізнавання іменованих сутностей
NF4	Normal Float 4	4-бітний формат нормалізованого числа з плаваючою комою
RegEx	Regular Expressions	Регулярні вирази
RLHF	Reinforcement Learning from Human Feedback	Навчання з підкріпленням на основі людського зворотного зв’язку
T5	Text-to-Text Transfer Transformer	Трансформер перенесення «текст-у-текст»
XAI	Explainable Artificial Intelligence	Пояснюваний штучний інтелект
XML	Extensible Markup Language	Розширювана мова розмітки
XPath	XML Path Language	Мова XML-шляхів (мова запитів до елементів XML)

ВСТУП

Актуальність теми. У сучасному цифровому середовищі стрімко зростають обсяги даних, що зумовлює потребу в ефективних інструментах їх обробки, структурування та аналізу. Парсинг даних є однією з головних технологій у сфері інформаційних систем, що забезпечує автоматизоване вилучення структурованої інформації з різноманітних джерел – вебсайтів, документів, баз даних, API та інших форматів. Традиційні методи парсингу, засновані на використанні регулярних виразів, XPath, CSS-селекторів або спеціалізованих бібліотек, є ефективними для простих завдань, проте мають обмеження при роботі з слабо структурованими даними або даними, що динамічно змінюються.

Із розвитком штучного інтелекту, зокрема з появою великих мовних моделей (LLM, Large Language Models), таких як GPT, LLaMA, Claude, Mistral тощо, з'явилась можливість застосовувати нові підходи до парсингу даних, що ґрунтуються на глибокому розумінні природної мови. Сучасні LLM здатні не лише вилучати та структурувати потрібні дані, але й інтерпретувати їх контекст, структуру та семантичні зв'язки, що відкриває принципово новий рівень ефективності у завданнях інформаційного пошуку, аналітики, моніторингу контенту та автоматизації бізнес-процесів.

Актуальність теми роботи обумовлена необхідністю порівняльного аналізу ефективності традиційних і новітніх підходів до парсингу даних, оскільки впровадження LLM у цю сферу суттєво впливає на швидкість, точність та вартість обробки інформації. З огляду на це, дане дослідження, спрямоване на визначення сильних і слабких сторін обох підходів, є важливим для розробників, аналітиків, науковців і підприємств, які прагнуть оптимізувати системи збору та аналізу даних.

Зв'язок роботи з науковими програмами, планами, темами. Магістерська кваліфікаційна робота відповідає дослідженням в межах науково-дослідної ініціативної тематики «Організаційно-методологічні аспекти впровадження інформаційно-комунікаційних систем і технологій в управлінні діяльністю сучасних організацій та підприємств за умов переходу до цифрової економіки»

(ДРН 0123U105060, 2023-2028 рр.), що реалізується на кафедрі інформаційних систем та технологій, тематиці досліджень навчально-дослідної лабораторії інтелектуальних систем, комп'ютерних мереж інтернет речей кафедри інформаційних систем та технологій Полтавського державного аграрного університету.

Мета роботи полягає у здійсненні порівняльного аналізу ефективності парсингу даних із використанням великих мовних моделей та традиційних методів.

Завдання роботи:

- узагальнити теоретико-методологічні основи парсингу даних із використанням LLM;
- проаналізувати архітектурні особливості та функціональні можливості традиційних методів парсингу;
- дослідити принципи побудови, навчання та застосування великих мовних моделей для парсингу даних;
- здійснити порівняння ефективності традиційних методів та методів на основі LLM за критеріями точності, швидкості, вартості та масштабованості;
- розробити практичні рекомендації щодо вибору та використання методів парсингу даних у різних галузях.

Об'єкт дослідження – методи парсингу даних.

Предмет дослідження – порівняльна ефективність парсингу даних із використанням LLM та традиційних методів.

Методи дослідження:

- аналіз наукових і технічних джерел для визначення сучасних тенденцій розвитку методів парсингу даних;
- системний аналіз архітектур і принципів роботи LLM та традиційних алгоритмів парсингу;
- порівняльний аналіз ефективності методів за визначеними критеріями;
- експериментальні дослідження на практичних прикладах парсингу текстових та напівструктурованих даних;
- узагальнення та синтез результатів для формування рекомендацій.

Інформаційна база дослідження складається з наукових статей, монографій, технічної документації сучасних великих мовних моделей (GPT-4, LLaMA 3, Claude 3), звітів провідних компаній (OpenAI, Meta AI, Anthropic, Google DeepMind), матеріалів із наукометричних баз Scopus, Web of Science, IEEE Xplore, а також практичних джерел з офіційної документації бібліотек Python для парсингу даних (BeautifulSoup, Scrapy, Requests, LangChain).

Елементи наукової новизни:

- систематизація та узагальнення знань щодо ефективності застосування великих мовних моделей у процесі парсингу даних;

- виявлення ключових факторів, що впливають на точність і швидкість парсингу із залученням LLM;

- розроблення рекомендацій з оптимізації використання LLM у задачах вилучення та структуризації інформації.

Практична значущість отриманих результатів полягає в можливості їх застосування під час розроблення та оптимізації систем автоматизованого збору даних у вебаналітиці, електронній комерції, журналістиці даних, наукових дослідженнях і корпоративних інформаційних системах. Розроблені рекомендації можуть бути використані для підвищення ефективності роботи аналітичних платформ, зменшення вартості обробки інформації та скорочення часу виконання завдань.

Апробація результатів дослідження. За результатами дослідження опубліковано тези доповідей: «Техніки оптимізації LLM-парсингу». Матеріали науково-практичної конференції за підсумками проходження виробничих практик здобувачів вищої освіти спеціальності 126 Інформаційні системи та технології, кафедра інформаційних систем та технологій Полтавського державного аграрного університету, 22 жовтня 2025 р. Вип. XI. Полтава: ПДАУ, 2025; «Архітектурні особливості великих мовних моделей у контексті парсингу даних». Студентські роботи за науковою тематикою кафедри інформаційних систем та технологій: матеріали XXII щорічного міждисциплінарного семінару, 25 листопада 2025 р. Полтава: ПДАУ, 2025 р.; «Принципи та архітектури LLM для парсингу даних». The

Future of Science, Technology and Economy: Collection of Scientific Papers with Proceedings of the 3rd International Scientific and Practical Conference. International Scientific Unity. October 29-31, 2025. Sofia, Bulgaria, 2025; «Архітектурні та функціональні особливості провідних моделей Reasoning-LLM». Progressive Approaches in Science and Engineering: Collection of Scientific Papers with Proceedings of the 2nd International Scientific and Practical Conference. International Scientific Unity. November 26-28, 2025. Copenhagen, Denmark, 2025.

Структура та обсяг кваліфікаційної роботи. Магістерська робота складається зі вступу, трьох розділів, висновків, списку використаних джерел та додатків. Основний текст викладено на 75 сторінках, містить 30 рисунків і 23 таблиці. Список використаних джерел налічує 70 найменувань.

РОЗДІЛ 1
ТЕОРЕТИЧНІ ТА МЕТОДОЛОГІЧНІ АСПЕКТИ ПАРСИНГУ ДАНИХ З ВИКОРИСТАННЯМ LLM

1.1 Концепція семантичного парсингу за допомогою LLM

Великі мовні моделі (LLM) є визначним досягненням у сфері штучного інтелекту (ШІ) для інтелектуалізованої обробки інформації [1-3]. Здатність LLM до семантичного аналізу, синтаксичного розбору, контекстуального розуміння тексту та генерування узгоджених структур даних створює нові можливості для автоматизації парсингу даних – процесу структурованого вилучення інформації з неструктурованих джерел.

Традиційні методи парсингу, такі як регулярні вирази, контекстно-вільні граматики або інструменти на основі XPath, CSS-селекторів або DOM-дерев, є суворо детермінованими. Внаслідок цього, всі вони мають суттєві обмеження при роботі з неструктурованими або напівструктурованими текстами (наприклад, динамічні HTML-сторінки, тексти з помилками або неоднозначними структурами). Натомість, LLM дозволяють здійснювати семантичний парсинг, де провідну роль відіграє інтерпретація контексту [4].

LLM перетворюють задачу парсингу на задачу опису структури даних засобами природної мови. Це означає, що модель може визначати поля, сутності й відношення у тексті, інтерпретувати контекст запиту користувача, видобувати релевантні елементи навіть за відсутності чіткої схеми. Основною перевагою LLM є інваріантність відносно формату вхідних даних: одна і та сама модель може працювати з HTML, JSON, CSV, PDF, текстами у довільному вигляді, таблицями та зображеннями, якщо вони мають текстову складову (наприклад, OCR-потоки).

Для розуміння еволюції технологій вилучення даних доцільно зіставити класичні алгоритмічні підходи з новітніми методами на базі штучного інтелекту. Головна відмінність між ними полягає у принципах обробки інформації: якщо традиційні інструменти орієнтовані на жорстку синтаксичну структуру документа

та чіткі правила, то великі мовні моделі фокусуються на семантичному змісті та адаптивності. Порівняння традиційного парсингу та LLM-підходу за основними критеріями представлено у таблиці 1.1.

Таблиця 1.1 – Порівняння традиційних та LLM-методів парсингу

Критерій	Традиційні методи	Методи з використанням LLM
Формат даних	Строго визначений (HTML, XML, CSV)	Гнучкий, мультиформатний (PDF, текст, JSON, Markdown)
Контекстність	Низька, обмежується патернами	Висока – модель розуміє зміст і контекст
Адаптивність	Потребує ручного переписування правил	Самонавчання і few-shot адаптація
Точність при змінах структури	Суттєво падає	Залишається стабільною
Використання семантики	Обмежене	Глибоке семантичне розуміння тексту
Приклади інструментів	BeautifulSoup, XPath, Scrapy	GPT-4, Claude 3, LLaMA 3, Gemini, Mistral

Нижче наведено узагальнену архітектуру системи семантичного парсингу на основі LLM (рисунок 1.1).

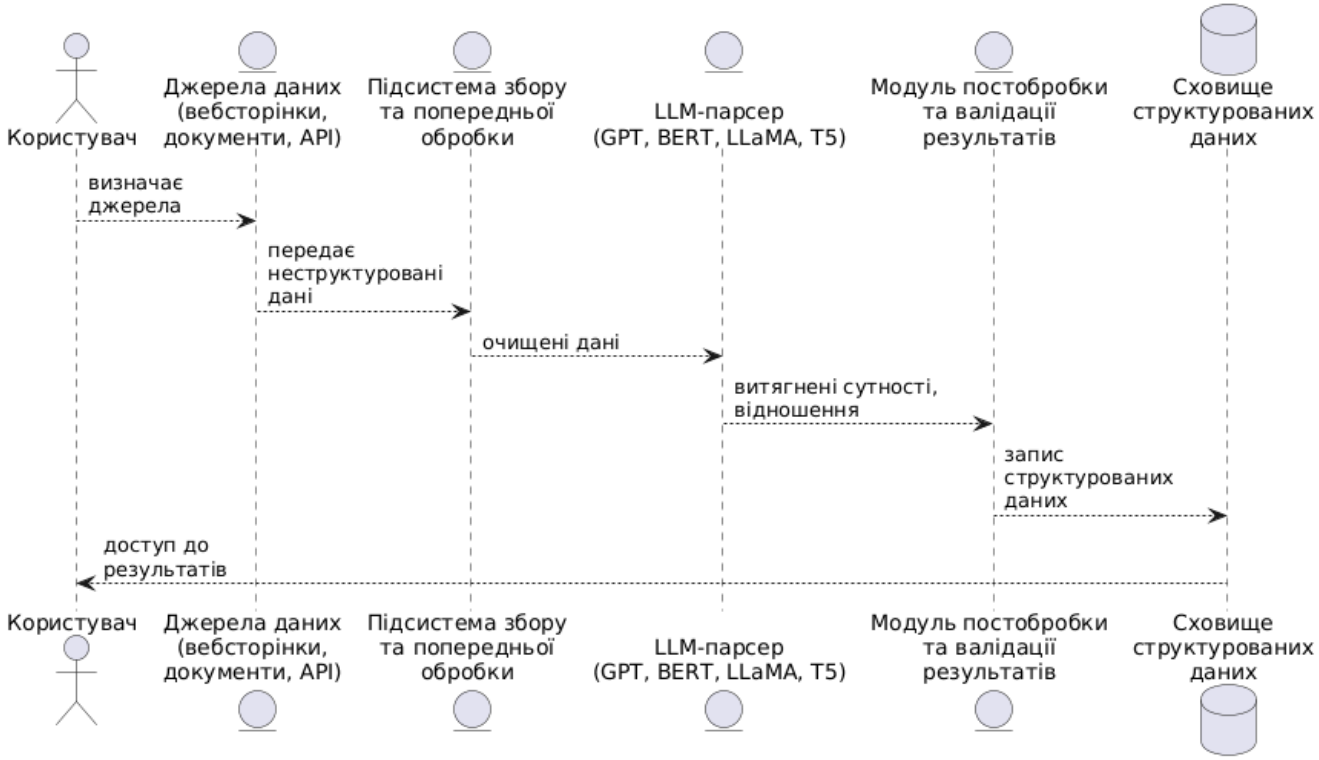

Рисунок 1.1 – Архітектура процесу семантичного парсингу на основі LLM

Розглянута архітектура відображає типову послідовність етапів парсингу даних – від збору даних до інтеграції у структуровану базу знань. На практиці вона зазвичай реалізується через API-запити до LLM (наприклад, OpenAI API або Hugging Face Inference API). Застосування LLM для парсингу даних означає перехід від суто синтаксичного аналізу до семантичного розуміння тексту, оскільки LLM здатні не лише виділяти патерни, а й розпізнавати смислові зв’язки та структурувати дані у формати JSON, CSV, SQL або RDF [5, 6].

1.2 Принципи та архітектури LLM для парсингу даних

Застосування LLM до парсингу даних базується на чотирьох принципах [7]:

- LLM повинна враховувати значення фрази у контексті цілого документа (контекстуальна інтерпретація);
- результати парсингу повинні бути текстовими відповідями, які легко конвертуються у структуровані дані (генеративний підхід);
- LLM повинні мати здатність адаптуватися до нових типів даних без необхідності повного донавчання (ефективне навчання (few-shot/zero-shot));
- LLM повинні мати здатність взаємодіяти з API, базами даних та ETL-процесами (інтеграція з Data engineering).

В основу сучасних LLM покладена трансформерна архітектура, вперше запропонована у роботі [2]. Її головною особливістю є механізм самоуваги (self-attention), що дозволяє моделі аналізувати зв’язки між словами незалежно від їх послідовності у реченні. Цей механізм долає обмеження послідовної обробки текстів, властивої RNN та LSTM, і забезпечує масштабованість мовних моделей до мільярдів параметрів (рисунок 1.2).

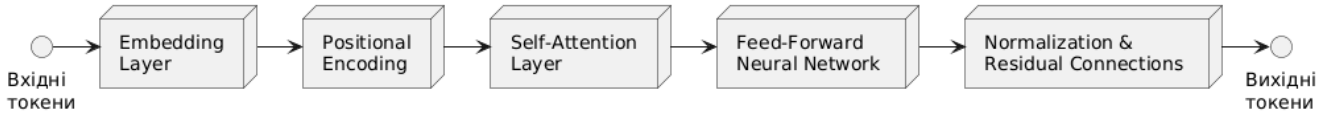

Рисунок 1.2 – Архітектура трансформера (загальна схема)

Наведена вище схема показує, що архітектура трансформера складається з модулів енкодера та декодера, кожен з яких використовує багатошарові механізми самоуваги для обробки послідовностей.

Embedding Layer (шар векторизації) перетворює токени (слова) на числові вектори фіксованої розмірності. Positional Encoding (шар позиційного кодування) додає інформацію про порядок слів, що є необхідним для відтворення синтаксису. Self-Attention Layer (механізм самоуваги) оцінює важливість кожного слова у контексті всього речення. Feed-Forward Neural Network (багатошарова нейронна мережа) виконує нелінійне перетворення результатів уваги. Normalization Residual Connections (нормалізація та залишкові зв'язки) стабілізує та пришвидшує процес навчання.

Механізм самоуваги дозволяє LLM формувати уявлення про контекст слова, беручи до уваги всі інші слова у реченні. Формально, він обчислює матрицю ваг, яка визначає вплив кожного токена на інші токени. Для кожного токена моделюються вектори Query (Q), Key (K), Value (V). Результат уваги визначається формулою:

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V, \quad (1.1)$$

де d_k – розмірність векторів ключів [2].

Механізм самоуваги має важливе значення для парсингу даних. На відміну від традиційних методів, які залежать від лінійної послідовності (наприклад, регулярних виразів), самоувага дозволяє визначити, що слово «сума» на початку документа пов'язане з числом «1500» у кінці, навіть якщо між ними велика кількість тексту. Це дозволяє парсити неструктуровані фінансові звіти або юридичні документи. Механізм самоуваги надає більшої ваги словам, які є найбільш релевантними для вилучення сутності. Наприклад, при парсингу запиту «Скільки коштує поїздка Полтава-Київ 10 грудня?», модель фокусує увагу (присвоює вищу вагу) на «Полтава», «Київ» та «10 грудня», ігноруючи менш

важливі слова, такі як «скільки» або «поїздка». Самоувага дозволяє LLM коректно інтерпретувати терміни, що можуть мати різне значення. Наприклад, у технічному тексті «порт» може означати мережевий порт або морський порт. Модель, завдяки самоувазі, використовує навколишній контекст («IP-адреса» або «судно»), щоб визначити правильну семантику для вилучення структурованих даних.

Ефективність LLM у задачах парсингу залежить від їх архітектури, які можна поділити на три основні категорії: encoder-only, decoder-only та encoder-decoder. Кожна архітектура має свої переваги в контексті структурованого вилучення даних.

Прикладом архітектури encoder-only є модель BERT (Bidirectional Encoder Representations from Transformers), яка створює контекстуальні представлення токенів, використовуючи двонапрямлену обробку. Загальна архітектурна схема BERT складається виключно з енкодерних шарів (рисунок 1.3).

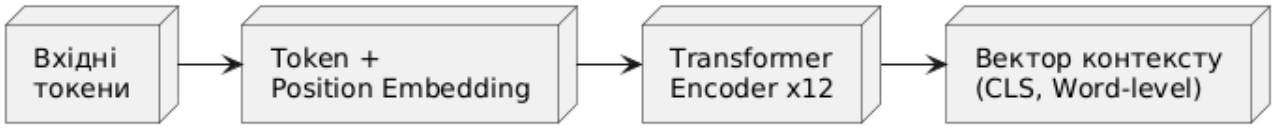

Рисунок 1.3 – Архітектура BERT (загальна схема)

Незважаючи на те, що BERT не є генеративною моделлю і не призначена для створення нового тексту, її внутрішні векторизовані представлення (embedding) використовуються як основа для побудови спеціалізованих класифікаторів та модулів вилучення іменованих сутностей (NER) у рамках цільових парсерів [3].

Архітектура моделі BERT побудована на основі стеку енкодерів архітектури Transformer, що забезпечує глибоке двонаправлене кодування контексту вхідної послідовності. Ця двонаправленість є важливою для ефективного вирішення задач класифікації та вилучення іменованих сутностей (NER), оскільки модель враховує зв’язки слова як з попередніми, так і з наступними токенами. Формування вхідного вектора відбувається шляхом сумування трьох компонентів вкладення (embedding): токенного (Token Embedding), сегментного (Segment Embedding) та позиційного (Position Embedding). Принципи кодування інформації про токен, його сегмент та

позицію, а також структуру деталізованого вхідного шару BERT, представлено на рисунку 1.4.

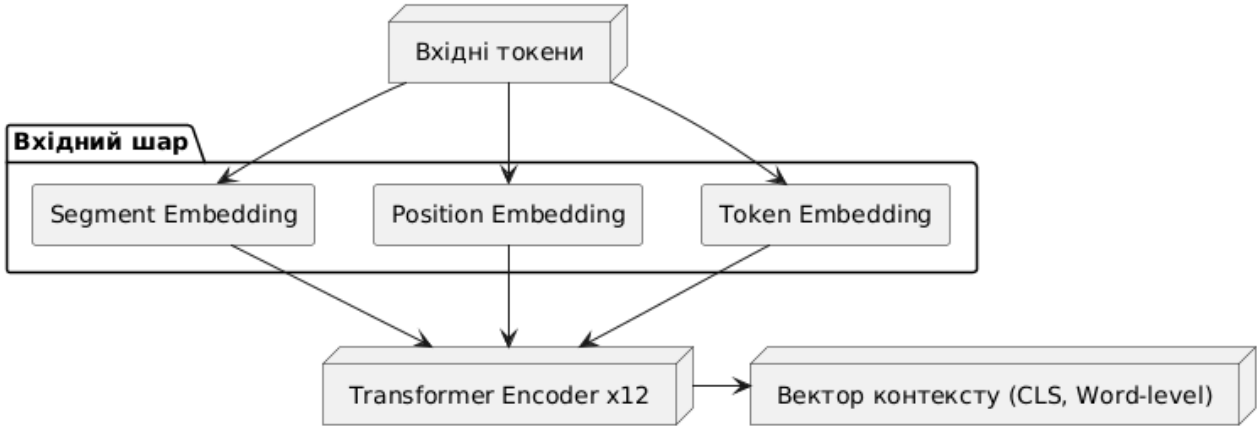

Рисунок 1.4 – Архітектура BERT (деталізований вхід)

Наявність трьох типів вкладень (Segment, Position, Token) на вході моделі BERT забезпечує її повним розумінням того, що це за слово у загальному контексті, до якого речення воно належить і де воно розташоване, що суттєво підвищує точність семантичного парсингу.

Прикладами архітектури decoder-only є моделі GPT і LLaMA, які генерують текст послідовно, на основі попередніх токенів [1, 56]. Такі моделі показують найкращі результати у few-shot та zero-shot сценаріях, коли навіть без явного навчання на задачах парсингу LLM здатні видобувати структури з текстів. Архітектура GPT, що базується на послідовності декодерних блоків, представлена на рисунку 1.5.

Рисунок 1.5 – Архітектура моделі GPT

Модель GPT використовує виключно декодерну архітектуру, і вважається найкращою для генерації структурованого виводу (наприклад, JSON) після отримання вхідного неструктурованого тексту.

Схема архітектури LLaMA, що акцентує на її декодерній оптимізації, представлена на рисунку 1.6.

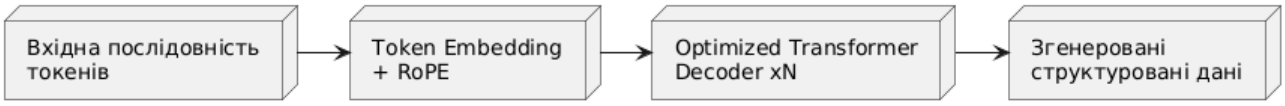

Рисунок 1.6 – Архітектура LLaMA

Архітектура LLaMA, будучи оптимізованим декодером, поєднує генеративні можливості з високою енергоефективністю, завдяки чому вона вважається найбільш практичною для застосування в індустріальних системах парсингу даних.

До категорії encoder-decoder відносяться моделі T5 або FLAN-T5, що базуються на принципі перетворення будь-якого завдання у формат «текст-у-текст». Ці моделі є універсальними для парсингу структурованих форматів (наприклад, JSON або XML), де кожен елемент можна подати як текстовий токен [6]. T5 спочатку проходить масштабне попереднє навчання на корпусі C4 (Colossal Clean Crawled Corpus) [8, 9], після чого донавчається на наборах завдань (наприклад, синтаксичний аналіз, класифікація, заповнення пропусків). У контексті парсингу це дозволяє моделі вивчати граматичні закономірності і відтворювати структурно правильні вихідні тексти. Загальна схема архітектури T5, що об’єднує енкодер і декодер, представлена на рисунку 1.7.

Рисунок 1.7 – Архітектура моделі T5

Використання повної архітектури «енкодер-декодер» дозволяє моделі T5 універсально виконувати завдання парсингу.

Для систематизації розглянутих архітектурних підходів та визначення їх практичної придатності до вирішення завдань структурування даних, доцільно

провести їх пряме порівняння. Узагальнені характеристики моделей, їх переваги та обмеження в контексті автоматизованого парсингу даних наведено в таблиці 1.2.

Таблиця 1.2 – Порівняння архітектур LLM для задач парсингу

Модель	Тип архітектури	Особливості навчання	Переваги для парсингу	Обмеження
BERT	Encoder-only	Маскування токенів (MLM)	Висока точність виявлення сутностей	Не генерує текст
T5	Encoder-Decoder	Seq2Seq навчання	Гнучкість у форматах (JSON, XML)	Вимагає більше ресурсів
GPT-4	Decoder-only	Автогресивне навчання	Zero-shot парсинг, потужна генерація	Можливі галюцинації
LLaMA 2	Decoder-only	Оптимізоване навчання на вебданих	Легша, відкрита модель	Менше даних для донавчання
PaLM 2	Decoder-only	Масштабоване multi-task навчання	Сильна узагальнювальна здатність	Обмежений доступ

Примітка. Ефективність LLM-парсингу залежить не лише від архітектури моделі, а й від методів формування контексту (prompt engineering). Наприклад, GPT-4 може досить точно видобувати дані з HTML-документів, якщо запит сформульований як «Виділи назви товарів і ціни у форматі JSON».

Поступовий перехід архітектурної будови LLM від двонаправлених до автогенеративних підходів, а також постійне зростання кількості їх параметрів та функціональної універсальності представлені у таблиці 1.3.

Таблиця 1.3 – Еволюція великих мовних моделей

Модель	Тип	Архітектура	Кількість параметрів	Головна особливість
BERT (2019)	Bidirectional Encoder	Transformer Encoder	~340 млн	Двонаправлений контекст, Masked Language Modeling
GPT-3 (2020)	Autoregressive Decoder	Transformer Decoder	175 млрд	Одностороння генерація, навчання на великому корпусі
T5 (2020)	Seq2Seq Encoder–Decoder	Повний Transformer	11 млрд	Завдання «text-to-text», універсальність
LLaMA 3 (2024)	Autoregressive Decoder	Оптимізований Transformer	70 млрд	Енергоефективність, швидке донавчання

Таким чином, актуальні напрями розвитку LLM спрямовані від архітектур на основі енкодера (BERT) до декодерів (GPT) та гібридних моделей (T5), з

подальшою оптимізацією декодерів для покращення ефективності (LLaMA). Загалом, еволюція LLM підвищила їх контекстуальну точність, масштабованість та універсальність. Діаграма на рисунку 1.8 узагальнює відомості про еволюцію архітектури LLM.

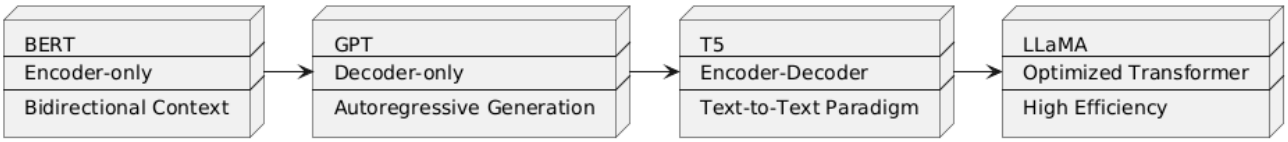

Рисунок 1.8 – Еволюція архітектур LLM

Таким чином, сучасні LLM розвиваються у напрямі збільшення їх масштабів та спеціалізації архітектур (енкодер, декодер, енкодер-декодер) для забезпечення як глибокого розуміння контексту (BERT), так і ефективної генерації виводу структурованих даних (GPT, LLaMA).

1.3 Оцінювання якості та ефективності LLM у задачах парсингу

Для об’єктивної оцінки ефективності LLM у задачах парсингу застосовується комплексний підхід, який інтегрує традиційні метрики оцінювання якості машинного навчання (наприклад, F1-score, Precision, Recall) з додатковими показниками. Ці показники спеціально розроблені для врахування специфіки синтаксичної та семантичної валідності генерованих вихідних структур даних.

Оцінка якості LLM-парсингу базується на трьох основних підходах [10-12]:

- автоматизовані метрики – включають розрахунок точності розпізнавання елементів (Accuracy), збалансовану оцінку точності й повноти (Precision / Recall / F1-score), а також спеціалізовані індекси: Exact Match (EM) – повний збіг вихідної структури із цільовою; Structural Consistency Index (SCI) – коректність вкладеності тегів (наприклад, у JSON або XML); Schema Adherence Rate (SAR) – відповідність результату заданій схемі даних;

- експертна оцінка (Human Evaluation) – передбачає залучення фахівців, що дозволяє виявити семантичні помилки, які пропускають автоматичні алгоритми: порушення логіки, невідповідність контексту та наявність «галюцинацій»;

- комбіновані підходи – гібридні методики, що поєднують автоматичні розрахунки з людською перевіркою. Наприклад, для парсингу юридичних документів у GPT-4 використовують комбінацію метрики BLEU та рейтингу Human Rating.

Деталізована процедура верифікації та валідації результатів роботи моделі, що включає двосторонню перевірку, представлена на рисунку 1.9.

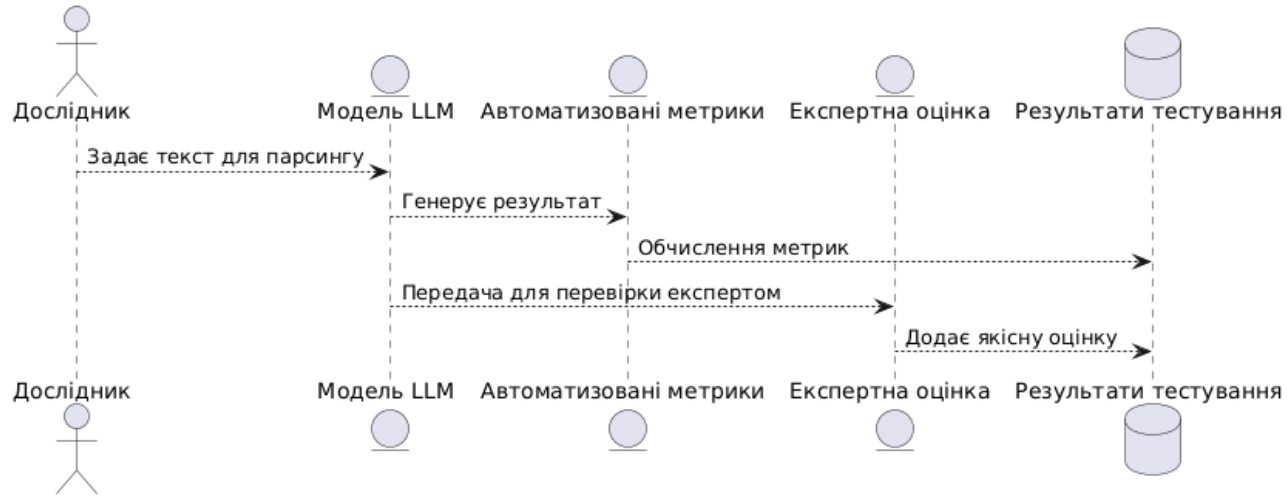

Рисунок 1.9 – Процес оцінювання LLM у задачах парсингу

Наведена схема ілюструє, що оцінювання моделі не обмежується порівнянням згенерованого результату із завчасно визначеним еталоном, а охоплює також перевірку структурної цілісності результату парсингу (наприклад, валідність JSON) та семантичну відповідність контенту запиту.

Для забезпечення об’єктивності порівняння моделей використовуються стандартизовані набори даних (бенчмарки), такі як SPIDER (SQL-парсинг), WebNLG (генерація описів), TAPAS (робота з таблицями), CoNaLa (парсинг коду) та PReP (вилучення технічних параметрів) [13-15].

Формальні показники оцінювання поділяються на три групи: синтаксичні – оцінюють формат виводу (Exact Match, Structural Accuracy); семантичні –

вимірюють змістовну точність (F1-score, BLEU, ROUGE, METEOR); інтерпретативні – визначають надійність та пояснюваність (Coherence Index, Hallucination Rate).

Детальний опис математичного апарату та інтерпретації основних метрик наведено в таблиці 1.4.

Таблиця 1.4 – Основні метрики для оцінки моделей LLM у парсингу

Категорія	Метрика	Формула / методика	Інтерпретація
Синтаксична	Exact Match (EM)	EM = TP / (TP + FP + FN)	Повна структурна відповідність
Семантична	F1-score	F1 = 2PR / (P + R)	Баланс точності і повноти
Семантична	BLEU	Порівняння n-грам	Якість текстової схожості
Інтерпретативна	Hallucination Rate	HR = %помилкових фактів / %загальних відповідей	Рівень достовірності
Людська	Human Eval	Опитування експертів (0–5)	Суб’єктивна оцінка точності

Таблиця 1.4 вказує на необхідність використання різнорідних метрик: EM гарантує, що код або JSON валідні, F1-score підтверджує правильність вилучених даних, а HR дозволяє контролювати схильність моделі до генерації неправдивої інформації. Окрім загальних формул, важливе практичне призначення кожної метрики в контексті конкретних задач парсингу.Характеристики метрик оцінювання ефективності LLM у задачах парсингу систематизовано в таблиці 1.5.

Таблиця 1.5 – Характеристика метрик оцінювання ефективності LLM у парсингу

Метрика	Опис	Тип задачі
Precision	Частка правильно розпізнаних сутностей серед усіх розпізнаних	Витяг ключових полів
Recall	Частка правильно знайдених сутностей серед усіх наявних	Повнота структурування
F1-score	Гармонійне середнє Precision і Recall	Узагальнена оцінка
Parsing Consistency (PC)	Відсоток відповідності між двома проходами парсингу	Стабільність моделі
Structural Validity (SV)	Частка правильно сформованих структур (JSON/XML)	Синтаксична точність

Як видно з таблиці 1.5, для задач парсингу важливими є не лише класичні показники Precision/Recall, але й специфічні показники стабільності (PC) та валідності (SV), оскільки навіть семантично вірна відповідь моделі є непридатною, якщо порушено структуру вихідного файлу.

Для автоматизації тестування LLM використовуються спеціалізовані фреймворки, такі як EVAL-HARNESS (EleutherAI) для задач NLU, LM-Eval (Hugging Face) для кастомних метрик та GPT-Eval (OpenAI) для порівняльного аналізу пропрієтарних моделей.

Узагальнена схема взаємодії компонентів у процесі автоматизованого оцінювання LLM зображена на рисунку 1.10.

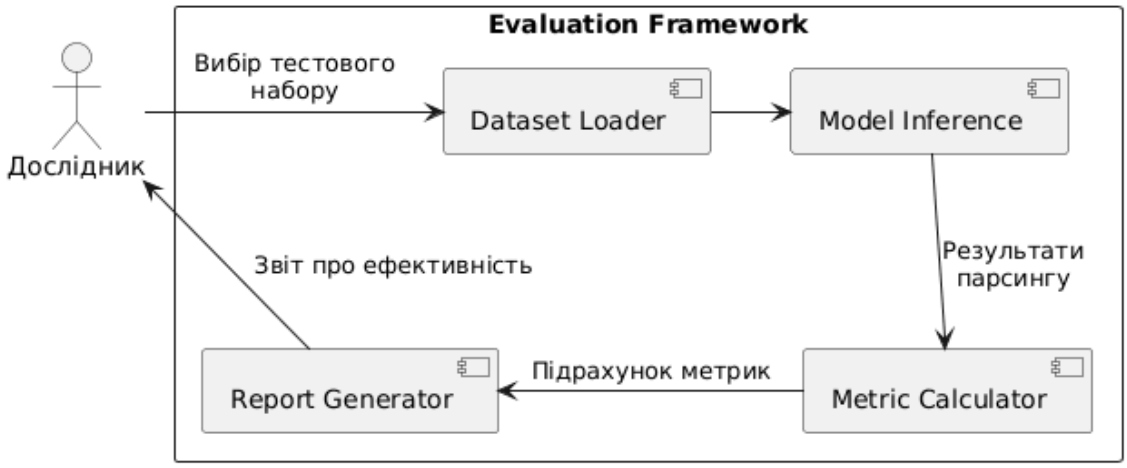

Рисунок 1.10 – Загальна схема процесу оцінки LLM

Наведена схема показує, що процес оцінювання моделей є конвеєрним: від подачі вхідних даних через модель до автоматизованого розрахунку метрик за допомогою спеціалізованих бібліотек.

Успішне впровадження LLM у промислові системи парсингу виходить за рамки виключно показників точності (F1-score, Recall). Для забезпечення фінансової доцільності та надійності системи важливим nfrj; є аналіз операційних показників ефективності. Ці показники, які включають швидкість, витрати та стійкість до зростаючого навантаження, визначають можливість масштабування рішення та його життєздатність у реальному часі. Головні критерії ефективності

парсингу, які використовуються для оцінки промислових LLM-систем, наведено в таблиці 1.6.

Таблиця 1.6 – Головні критерії ефективності парсингу у промислових LLM-системах

Показник	Опис	Одиниці вимірювання	Тип контролю
Latency	Середній час відповіді моделі	мс / запит	Автоматичний
Throughput	Кількість документів/сек.	док/сек	Автоматичний
Cost Efficiency	Вартість за 1000 токенів	$	Аналітичний
Robustness	Стійкість до шуму	% точних результатів	Тестовий
Scalability	Продуктивність при $N \to \infty$	коеф. деградації	Моніторинговий

Дані таблиці свідчать про те, що для бізнесу технічна точність моделі часто врівноважується вартістю експлуатації та швидкістю обробки. Тобто, високе значення Latency може зробити найточнішу модель непридатною для систем реального часу.

1.4 Методи навчання LLM для парсингу

Здатність сучасних LLM до парсингу даних є наслідком їх багатоетапного процесу навчання. Цей процес поділяється на попереднє навчання (pre-training), що має на меті формування базових мовних закономірностей і загального розуміння структури даних, та донавчання (fine-tuning/instruction-tuning), яке пристосовує модель до виконання вузькоспеціалізованих завдань, зокрема парсингу таблиць, JSON-структур або HTML-документів.

Головна мета етапу попереднього навчання, який базується на масивах нерозміченого тексту, полягає у формуванні лінгвістичної основи: модель засвоює базові мовні закономірності та загальне розуміння контексту. Ця фаза створює універсальну основу, яка потім адаптується під спеціалізовані завдання парсингу та вилучення сутностей на етапі донавчання (fine-tuning). Рисунок 1.11 ілюструє загальну архітектуру та послідовність операцій попереднього навчання.

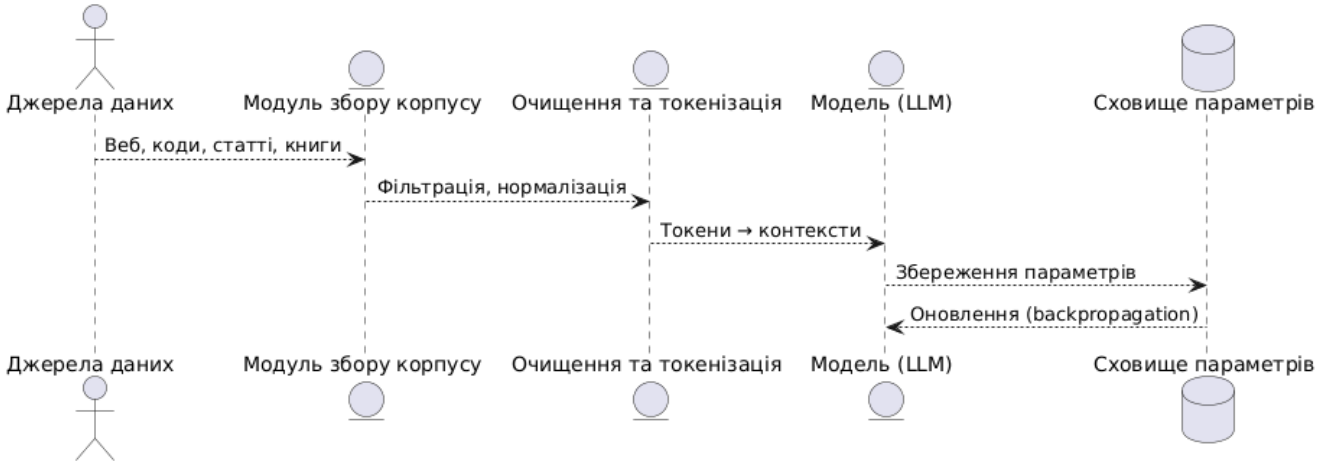

Рисунок 1.11 – Загальний процес попереднього навчання LLM

Для попереднього навчання LLM використовують великі корпуси даних, такі як Common Crawl [16], The Pile [17] або C4 [8, 9], які забезпечують різноманітність мовних і технічних структур, що дозволяє моделям краще узагальнювати закономірності текстів (таблиця 1.7).

Таблиця 1.7 – Основні корпуси даних для навчання LLM

Корпус	Рік створення	Обсяг (GB)	Коротка характеристика
Common Crawl	2011–2024	>60000	Вебконтент у сирому вигляді; використовується в GPT і LLaMA
The Pile	2021	~825	Спеціально збалансований корпус для NLP-завдань
C4	2020	~350	Очищений корпус із вебданих, основа для T5
GitHub Code	2021	~500	Тексти програмного коду для моделей кодування
Wikipedia	постійне оновлення	~50	Енциклопедичний корпус, використовується в усіх моделях

Попередньо навчені LLM далі використовуються, як основа для створення різноманітних спеціалізованих рішень, зокрема, для інтелектуального парсингу неструктурованих джерел. Адаптація моделей до таких завдань відбувається за допомогою різних методів навчання [18].

LLM здатні виконувати деякі завдання парсингу без донавчання, якщо отримують правильно сформульовану підказку. Метод few-shot навчання допомагає LLM засвоїти правила структурування даних за кількома прикладами,

тоді як zero-shot – за допомогою інструкції на природній мові, наприклад: «Витягни з тексту імена компаній та дати їх заснування у форматі JSON».

Метод донавчання (fine-tuning) передбачає додаткове навчання моделі на задачах вилучення структурованої інформації. Наприклад, модель можна навчити перетворювати HTML-таблиці у CSV або видобувати ключі та значення з JSON-документів. Такий підхід застосовується у T5 і GPT-3.5, де модель «вчиться» не лише відтворювати текст, а й генерувати коректно структурований вихід.

Інструкційне навчання (instruction tuning), запроваджене у FLAN-T5 і GPT-4, дозволяє моделі краще розуміти людські команди, наприклад: «Виділи всі email-адреси з цього тексту і виведи їх у форматі JSON». Така форма адаптації наближає LLM до практичного застосування в автоматизованих парсингових системах, оскільки користувач формулює задачу звичайною мовою, без необхідності програмування. Діаграма процесу instruction tuning представлена на рисунку 1.12.

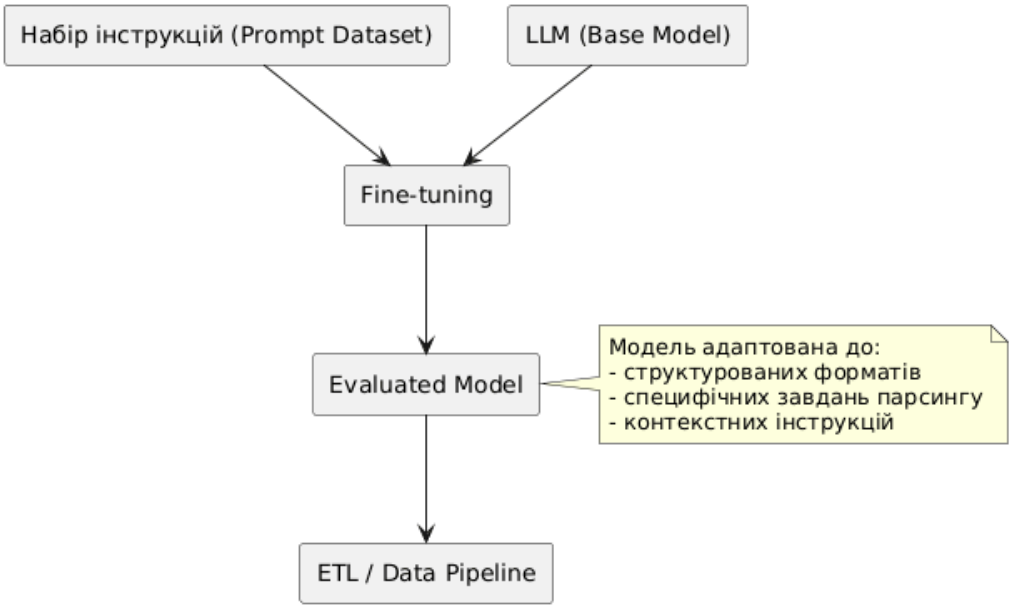

Рисунок 1.12 – Інструкційне навчання у контексті парсингу даних

Основні етапи інструкційного навчання включають підготовку наборів інструкцій – збір пар «запит–відповідь», у яких вказано, як модель має структурувати дані (наприклад, «витягни всі ціни з HTML»), донавчання на цих прикладах – оптимізація ваг трансформера для підвищення відповідності запиту

результату; оцінювання узагальненості – тестування на нових форматах даних (JSON, XML, лог-файли тощо), інтеграцію з системами ETL/ELT – вбудовування у пайплайни обробки даних.

Метод RLHF (Reinforcement Learning from Human Feedback) здійснює адаптацію моделі на основі оцінок користувачів. У контексті парсингу, RLHF дозволяє покращити точність структуризації вихідних даних, мінімізувати «галюцинації» моделей – випадки, коли модель генерує неправдиву або несумісну структуру та узгодити вихідний формат із вимогами користувача. Тобто RLHF покращує здатність LLM до парсингу через зворотний зв’язок від людини: дія механізму RLHF побудована на підкріпленні правильних рішень на основі людських оцінок структурованих результатів парсингу [19, 20].

Метод CoT (Chain-of-Thought) – метод «ланцюгового мислення» – дозволяє LLM покроково пояснювати процес прийняття рішень. У задачах парсингу CoT дозволяє покращити точність результату у випадках багатоступеневих структур, забезпечити інтерпретованість результату та спростити виявлення помилок [21, 22].

Метод безперервного навчання (continual learning LLM, CL) забезпечує оновлення ваг моделі на основі нових зразків без повного перенавчання. Цей метод ефективний для систем з динамічними потоками даних (нові схеми, API, формати). Він дозволяє LLM швидко адаптуватися до зміни контексту парсингу, нових форматів структур даних або змін у схемах вебресурсів [23, 24]. Модель LLaMA 3 отримала у 2024 році модуль адаптивного навчання, який дозволяє оновлювати знання без повного перенавчання (розв’язання проблєми катастрофічного забування у LLM під час їх постіного навчання та адаптації [23, 25]).

Метод мультимодального навчання (multimodal learning) базується на одночасній обробці текстових, візуальних і табличних даних (наприклад, моделі GPT-4V або Gemini). Інтеграція різних модальностей дозволяє реалізувати універсальний парсинг: автоматично вилучати структуровану інформацію із зображень таблиць (у PDF-файлах або на знімках екрана), графічних інтерфейсів, а також комбінованих текстово-числових масивів. Поєднання текстового та

візуального каналів сприйняття інформації значно розширює можливості парсингу: сучасні мультимодальні моделі здатні розпізнавати та структурувати дані зі складних джерел, таких як скановані документи, графічні таблиці та елементи користувацьких інтерфейсів, ефективно об’єднуючи текстові описи з числовими значеннями [26, 27].

Процес адаптації LLM до завдань парсингу є ітеративним, і може бути представлений у вигляді замкненого циклу, схему якого наведено на рисунку 1.13.

Рисунок 1.13 – Цикл адаптації LLM до задач парсингу

Ця схема показує, що адаптація LLM до конкретних завдань є не одноразовою дією, а безперервним процесом, де валідація результатів і зворотний зв’язок на кожному етапі є необхідними для поступового підвищення точності парсингу. Ефективність цього циклу залежить від обраної стратегії навчання або донавчання моделі.

Отже, ефективність LLM у задачах парсингу безпосередньо залежить від обраної методології їх адаптації та спеціалізації. Для тонкого налаштування універсальної LLM під конкретні потреби вилучення структурованих даних застосовують різноманітні підходи: від ресурсоємного донавчання з розміченими даними (Supervised Fine-tuning) до гнучких, контекстно-орієнтованих методів

(Few-shot, CoT Learning). Узагальнення відомостей щодо основних методів, які застосовуються для адаптації LLM для виконання завдань парсингу, представлено у таблиці 1.8

Таблиця 1.8 – Порівняння підходів до навчання моделей у задачах парсингу

Метод	Тип навчання	Особливості	Переваги	Обмеження
Supervised Fine-tuning	Класичне донавчання	Використовує позначені дані	Висока точність	Високі витрати на розмітку
RLHF	Підкріплення з людським зворотним зв'язком	Навчання на людських оцінках	Підвищує якість та безпеку	Дорогий процес навчання
Few-shot / Zero-shot	Без явного навчання	Використовує контекст підказки	Гнучкість, дешевизна	Нестабільність відповідей
CoT Learning	Покрокове мислення	Пояснювані результати	Інтерпретованість	Довший час генерації
Continual Learning	Безперервне оновлення	Адаптація до нових даних	Гнучкість, швидкість	Ризик «забування» старих знань

Представлений вище опис методів навчання моделей у задачах парсингу свідчить про необхідність вибору між їх ресурсоємністю та стабільністю: методи Fine-tuning та RLHF забезпечують найвищу якість, але є коштовними, тоді як методи Few-shot/Zero-shot є більш швидкими та дешевшими, хоча забезпечують меншу точність для критичних завдань. У таблиці 1.9 наведено узагальнення ефективності основних методів навчання у поєднанні з оцінювальними метриками.

Таблиця 1.9 – Порівняння методів навчання моделей за критеріями якості

Метод навчання	Точність (F1)	Швидкість обробки	Вартість навчання	Інтерпретованість	Стійкість до шуму
Supervised Fine-tuning	0,95	Середня	Висока	Середня	Висока
RLHF	0,93	Низька	Дуже висока	Висока	Висока
Few-shot / Zero-shot	0,88	Висока	Низька	Середня	Середня
Continual Learning	0,90	Висока	Середня	Низька	Висока
Multimodal	0,92	Середня	Висока	Висока	Висока

Наочне порівняння співвідношення точності LLM у задачах парсингу даних залежно від методів навчання представлене на рисунку 1.14.

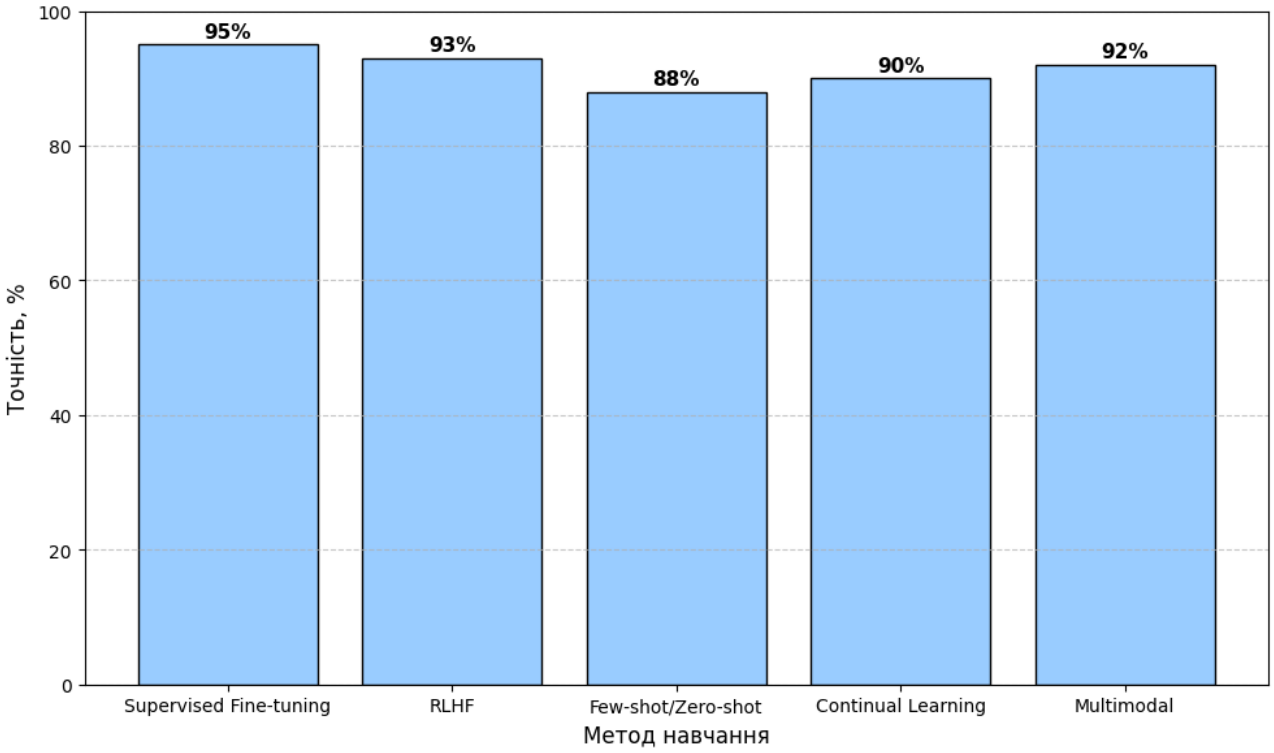

Рисунок 1.14 – Порівняння точності методів навчання LLM у парсингу

Графічний аналіз підтверджує, що спеціалізоване донавчання (Supervised Fine-tuning) забезпечує найвищий приріст точності порівняно з базовими можливостями моделі, тоді як методи Few-shot / Zero-shot, незважаючи на свою економічність, забезпечують помітно менше значення показника F1, що обмежує їх застосування у високонавантажених промислових системах.

Таким чином, визначення оптимальної стратегії адаптації LLM для потреб парсингу є завданням пошуку компромісу, який вимагає збалансувати досягнуту точність вилучення даних із загальними ресурсними та фінансовими витратами. Хоча метод Supervised Fine-tuning на розмічених даних гарантує максимальну якість результатів (з точністю F1 близько 0,95), значна вартість та часові затрати на підготовку навчальних наборів стимулюють використання альтернативних, більш економічних підходів. До них належать RLHF (навчання з підкріпленням) для

обробки складних, суб'єктивних сценаріїв, або методи Few-shot/Zero-shot для швидкого прототипування та зменшення витрат.

1.5 Застосування LLM-парсингу у різних галузях

Парсинг даних за допомогою LLM знаходить застосування у багатьох різних галузях.

Однією з провідних галузей використання LLM-парсингу даних є фінансова аналітика. Основні застосування LLM у фінансових системах включають: автоматичне зчитування PDF-звітів, рахунків-фактур, фінансових декларацій; витягування ключових показників (KPI, EBITDA, ROA, ROI); розпізнавання контексту ризиків або угод [28, 29]. Прикладом практичного рішення у цій сфері є система FinBERT, адаптована для обробки фінансових текстів. Вона дозволяє ідентифікувати сутності у звітах (наприклад, «net income» → показник чистого прибутку) із точністю понад 93 % [30, 31].

У медичних інформаційних системах та біоінформатиці LLM застосовуються для витягування структурованих записів із текстових медичних карт, анонімізації персональних даних, класифікації симптомів та діагнозів на основі клінічних звітів [32, 33]. Моделі типу BioGPT [34-36], Med-PaLM (розробка Google DeepMind) [37] і PubMedBERT [38] показали суттєве покращення точності парсингу медичних абревіатур і латинських термінів у неструктурованих документах.

В юридичному секторі LLM використовуються для аналітики правових текстів – контрактів, судових рішень, нормативних актів. Основні задачі включають: парсинг ідентифікаторів справ, дат, сторін; витягування структур типу «зобов'язання – умова – наслідок»; класифікацію типів контрактів і їх положень [39, 40]. Для цього застосовують моделі LegalBERT [41, 42], LawGPT [43, 44], а також спеціалізовані пайплайни з fine-tuning на корпусах судових рішень США та ЄС [43].

У сфері електронної комерції (e-commerce) LLM використовують для парсингу каталогів товарів, цінових пропозицій, описів і відгуків. Основні переваги: автоматичне створення уніфікованих структур даних (назва, категорія, бренд, ціна); класифікація товарів за атрибутами без ручного маркування; фільтрація дублікатів і семантичне узгодження назв [44, 45]. Існують також системи типу GPT-4 + LangChain здатні інтегруватися безпосередньо в бізнес-логіку для побудови «розумних» агрегаторів цін і товарів [46, 47].

У сфері наукових досліджень (наукова аналітика) LLM дозволяють автоматизувати парсинг наукометричних баз, таких як Scopus, PubMed, arXiv. Вони витягують: метадані публікацій (назва, автор, рік, журнал); бібліографічні посилання; тематичні класифікатори (наприклад, ACM CS Categories). Зокрема, на основі GPT-4 створено систему ScholarParse, яка перетворює бібліографічні списки у структуровані BibTeX-файли з точністю до 96 % [48].

Механізми LLM-парсингу використовує також Bohrium AI – комплексна хмарна платформа для наукових досліджень, розроблена компанією DP Technology, яка об’єднує розумний пошук наукової літератури, високопродуктивні обчислення та інструменти для розробки. Bohrium AI надає доступ до бази з понад 160 мільйонів статей із можливістю отримання точних відповідей з посиланнями на джерела, а також пропонує потужне середовище для запуску коду, молекулярного моделювання та роботи з даними прямо у браузері. Фактично, Bohrium AI – це не простий парсер, а екосистема, що дозволяє вченим і студентам виконувати повний цикл роботи: від швидкого аналізу теорії до проведення складних обчислювальних експериментів.

Bohrium AI має безкоштовну версію, яка стосується лише частини функціоналу. Інструмент Science Navigator (науковий пошук) є безкоштовним для дослідників, і дозволяє вільно використовувати: пошук серед 160+ мільйонів наукових статей; AI-чат (аналог ChatGPT для науки), який відповідає на запитання з посиланнями на джерела; читання та аналіз PDF-файлів (summary); особисте сховище: Користувачам зазвичай надається 500 ГБ безкоштовного простору для зберігання файлів [49].

Практична застосовність LLM для парсингу залежить від області застосування. Узагальнені показники ефективності парсингу із використанням як універсальних, так і спеціалізованих (доменних) моделей у ключових галузях наведено в таблиці 1.10.

Таблиця 1.10 – Ефективність LLM-парсингу у різних галузях

Галузь	Типи даних	Приклади моделей	Точність парсингу	Особливості
Фінанси	PDF-звіти, таблиці	FinBERT, GPT-4	93–96 %	Контекстна класифікація фінансових метрик
Медицина	Тексти медичних карт	BioGPT, Med-PaLM	90–95 %	Розпізнавання термінів, анонімізація
Юриспруденція	Судові рішення, контракти	LegalBERT, LawGPT	85–92 %	Витягування правових структур
E-commerce	Вебкаталоги, JSON-API	GPT-4, Claude 3	94–97 %	Автоматичне маркування товарів
Наука	Статті, цитування	T5, GPT-4	95–98 %	Семантичний парсинг посилань

Аналіз наведених даних свідчить, що найвищі показники точності LLM-парсингу (понад 95 %) спостерігаються у сферах із частково структурованими даними, таких як наука та e-commerce. Водночас у галузях зі складною семантикою та високою відповідальністю (медицина, юриспруденція) спостерігається тенденція до використання вузькоспеціалізованих моделей (LegalBERT, BioGPT).

Реалізація повного потенціалу LLM у завданнях високоточного парсингу, особливо у специфічних предметних областях, виходить за межі простого використання API. Для забезпечення надійності, масштабованості та економічної ефективності в умовах промислового потоку даних необхідна правильна архітектурна організація потоків даних. Цей домен-орієнтований підхід вимагає, щоб LLM не виступала ізольованим модулем, а була інтегрована з іншими критично важливими компонентами інформаційної системи: модулями попередньої обробки вхідних даних, базами знань для підвищення контекстної релевантності та модулями валідації для контролю якості вихідної JSON-структури. Схематичне відображення механізму такої комплексної взаємодії, що є

обов’язковою умовою для адаптації LLM до галузевих стандартів та бізнес-логіки, представлено на рисунку 1.15.

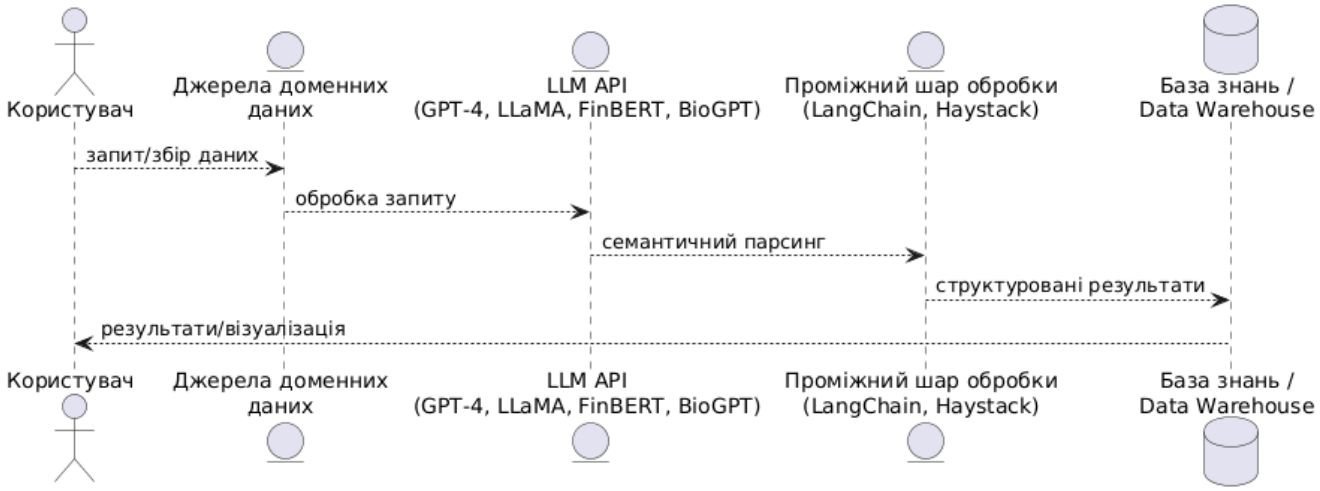

Рисунок 1.15 – Модель інтеграції LLM у доменні системи парсингу

Незважаючи на високу точність і гнучкість, використання LLM для парсингу даних має низку обмежень: високі обчислювальні витрати (вартість запитів, вимоги до GPU); ризики галюцинацій (вигадані факти під час генерації); відсутність детермінізму – результат парсингу може відрізнятися при повторному запуску моделі; правові аспекти конфіденційності даних (особливо у медицині та фінансах). Для зниження цих ризиків застосовуються методи prompt-engineering, post-validation, а також тонке RLHF-донавчання моделей для певних доменів.

Висновки до розділу 1

У першому розділі розглянуто застосування LLM для задач парсингу даних. Зазначено, що інтеграція LLM у процеси обробки інформації означає перехід від детермінованих алгоритмів (RegEx, DOM-парсинг) до імовірнісного семантичного аналізу, який дозволяє ефективно структурувати інформацію з неструктурованих джерел, таких як PDF-звіти, динамічні вебсторінки та тексти з неоднозначною структурою, долаючи обмеження традиційних інструментів.

В основі сучасних методів LLM-парсингу лежить трансформерна архітектура моделей та механізм самоуваги (self-attention), який забезпечує врахування глобального контексту документа незалежно від позиції токенів. Розгляд різних архітектур показав чітку спеціалізацію моделей: encoder-only (наприклад, BERT) є найефективнішими для задач класифікації та вилучення сутностей (NER); decoder-only (GPT, LLaMA) демонструють найкращі результати у генерації структурованих виходів (JSON, SQL) та zero-shot сценаріях; encoder-decoder (T5) забезпечують універсальність трансформації «текст-у-текст».

Вибір методів навчання LLM для завдань парсингу є компромісом між точністю та ресурсоємністю методу. Метод Supervised Fine-tuning забезпечує найвищу точність ($F1 \approx 0{,}95$), однак вимагає значних витрат на розмітку даних. Методи Few-shot / Zero-shot є економічно більш вигідними та гнучкими, але менш стабільними. Використання методів RLHF (навчання з підкріпленням) та CoT (ланцюгового мислення) дозволяє додатково підвищити надійність та інтерпретованість результатів.

Не існує єдиної метрики, за якою можна оцінити ефективність LLM-парсингу. Потрібен комплексний підхід, що поєднує синтаксичні метрики (Exact Match, валідність формату), семантичні метрики (F1-score, Precision/Recall) та операційні показники (Latency, Throughput, вартість токенів). Важливим аспектом залишається контроль «галюцинацій» моделі через метрику Hallucination Rate.

Використання LLM показало свою ефективність у сфері фінансів, медицині, юриспруденції, e-commerce тощо. При цьому спостерігається тенденція до використання спеціалізованих доменних моделей (FinBERT, BioGPT, LegalBERT), які у специфічних контекстах перевершують універсальні мовні моделі за точністю та розумінням професійної термінології.

Таким чином, використання LLM дозволяє автоматизувати складні процеси вилучення даних, які раніше потребували ручної обробки, однак для промислового впровадження необхідна побудова гібридних систем із чітким контролем якості та оптимізацією витрат.

РОЗДІЛ 2
АНАЛІТИЧНИЙ ОГЛЯД МЕТОДІВ ПАРСИНГУ ДАНИХ

2.1 Архітектурні особливості та функціональні можливості традиційних методів парсингу

Традиційні методи парсингу даних сьогодні досі залишаються важливою складовою систем автоматизованої обробки інформації. Вони всебічно апробовані, і на практиці довели свою ефективність у сфері їх використання. Їх головною характеристокою є детермінованість – використання явно визначених правил, за якими відбувається вилучення релевантних елементів із тексту або документа. Тому ці методи застосовують переважно у випадках, коли структура даних є стабільною та передбачуваною.

До основних традиційних методів парсингу належать:

- регулярні вирази (RegEx) – механізм пошуку шаблонних текстових послідовностей;
- XPath (XML Path Language) – декларативна мова звернення до вузлів XML-документів;
- CSS-селектори – механізм вибірки елементів HTML за класами, тегами або ідентифікаторами.

Зазначені інструменти утворюють принципову основу сучасних фреймворків вебскрапінгу, таких як BeautifulSoup, Scrapy, Puppeteer, Selenium, lxml [50, 51]. Всі вони показують високу ефективність у роботі з даними, структура яких майже не змінюється, наприклад, це: HTML-документи, XML-файли, конфігураційні набори, журнали подій тощо.

Архітектура традиційного процесу парсингу включає:

- отримання джерела – завантаження HTML/XML або текстового документа;
- попередню обробку – очищення даних від допоміжних символів, пробілів, зайвих тегів;

- застосування правил парсингу – використання RegEx, XPath або CSS-селекторів;

- фільтрацію та нормалізацію результатів – структурація у формати JSON, CSV або SQL;

- збереження/передачу результатів – інтеграцію у бази даних або аналітичні модулі.

Загальний процес традиційного парсингу відображено на рисунку 2.1.

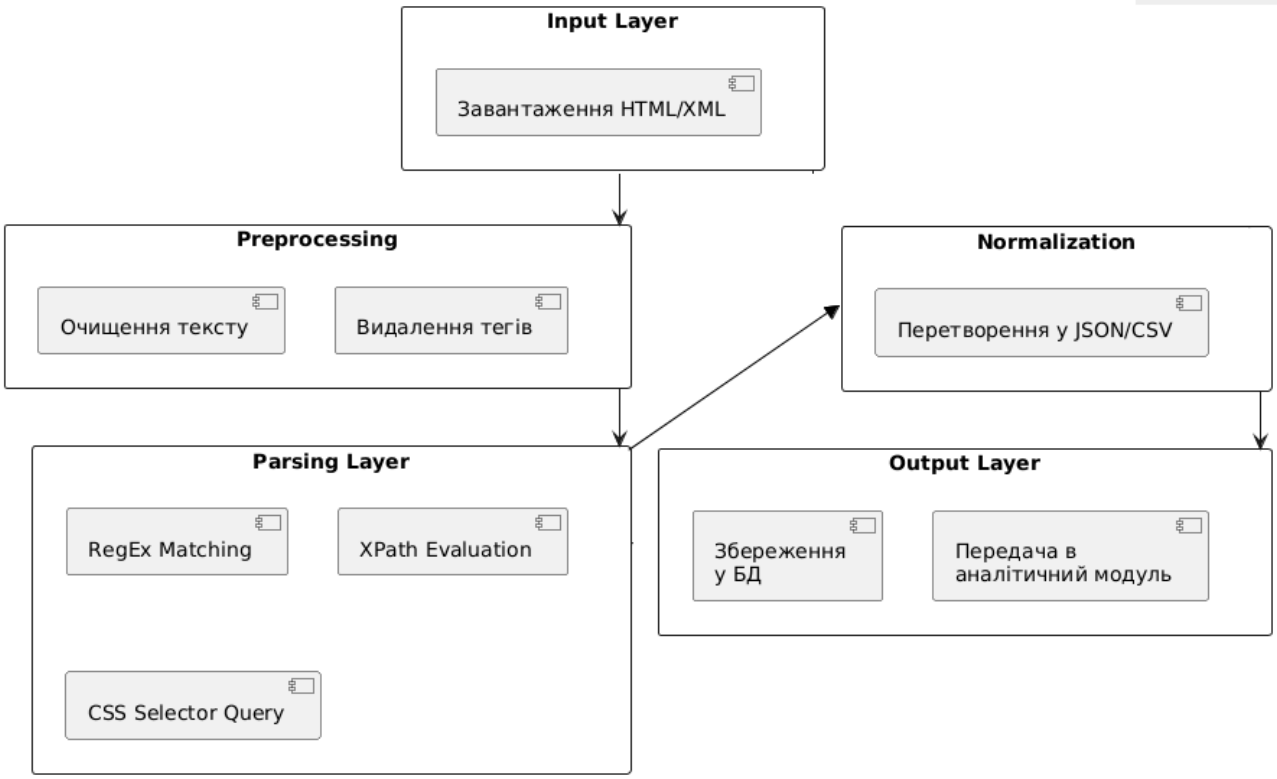

Рисунок 2.1 – Архітектура процесу традиційного парсингу

Регулярні вирази є одним із найстаріших і водночас найпотужніших інструментів парсингу. Їх робота ґрунтується на використанні скінченних автоматів, що забезпечує високу швидкість обробки великих масивів текстових даних. Архітектура RegEx включає: токенізацію – розбиття рядка на елементи; компіляцію виразу в дерево або байт-код; рушій зіставлення, який визначає відповідність тексту шаблону. Прикладом RegEx є шаблон для пошуку електронних адрес: [a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}. Цей вираз дозволяє ефективно вилучати email-адреси, хоча й не враховує контекст

використання адрес у тексті. Переваги RegEx: висока швидкодія; мінімальні обчислювальні витрати; універсальність у роботі з текстами. Недоліки: чутливість до змін структури документа; відсутність контекстного аналізу; складність супроводу при великій кількості шаблонів [52].

XPath є стандартом адресації елементів XML-документів і використовується в Scrapy, Selenium, lxml, Apache Nutch. Типовий запит XPath //book[price>30]/title/text() дозволяє вибрати назви книг із ціною понад 30. Переваги XPath: точне адресування; підтримка логічних умов і фільтрів; сумісність зі структурованими XML/HTML. Недоліки: залежність від суворої ієрархії; низька ефективність у динамічних HTML-документах [53].

Архітектура виконання XPath-запиту включає: Parser Engine – побудова DOM-моделі; XPath Evaluator – виконання логіки вибірки; Result Handler – формування набору результатів (рисунок 2.2).

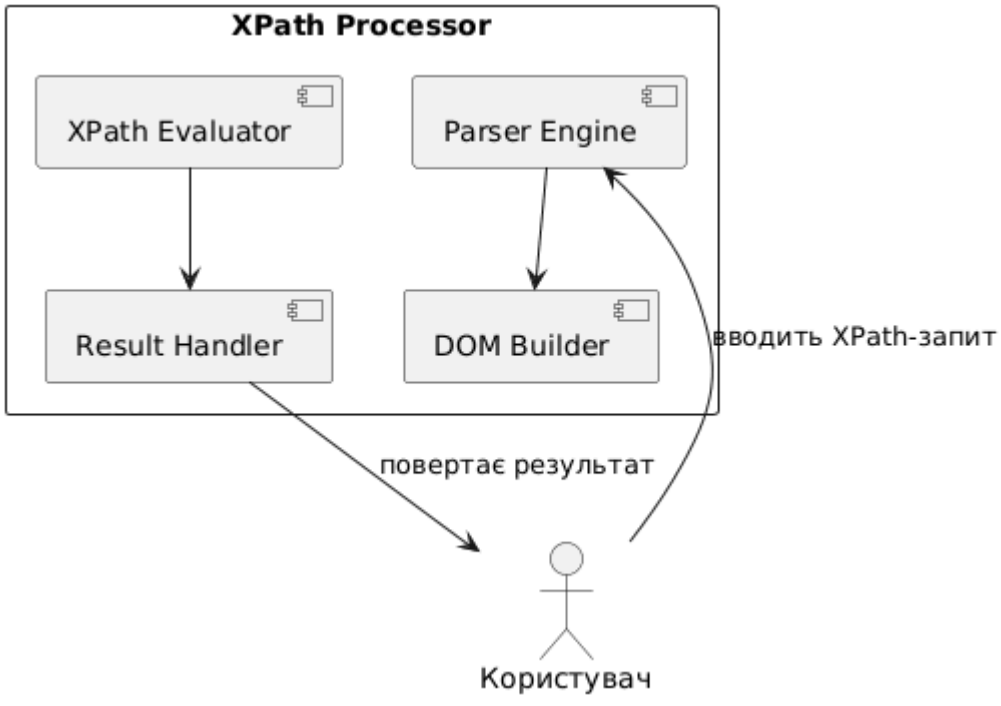

Рисунок 2.2 – Архітектура обробки XPath-запиту

CSS-селектори є синтаксичним інструментом для вибірки елементів HTML-документів. Вони широко застосовуються у Python-бібліотеках BeautifulSoup, Puppeteer, Playwright. Приклад селектора div.article > h2.title вибирає всі заголовки

h2 усередині блоку з класом article. Переваги CSS-селекторів: інтуїтивний синтаксис; гнучка робота з ієрархією; висока сумісність із сучасним HTML5. Недоліки: обмежена логіка вибірки; чутливість до змін DOM; непридатність для семантичного аналізу [54].

Механізм роботи CSS-селектора схематично показано на рисунку 2.3.

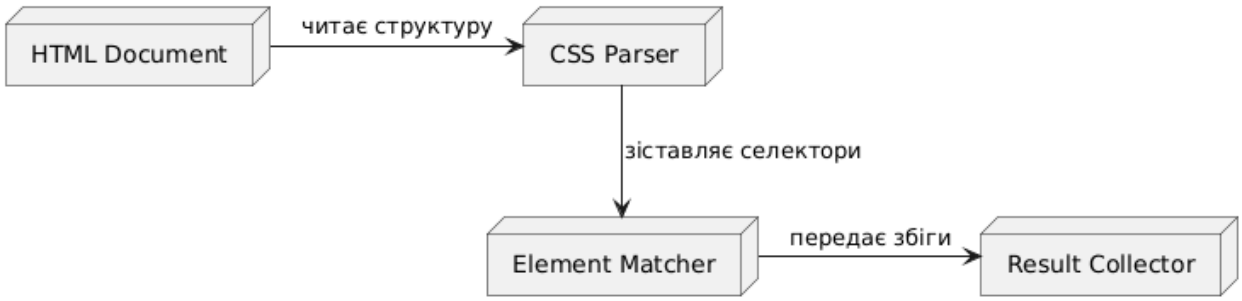

Рисунок 2.3 – Механізм CSS-селектора

У таблиці 2.1 наведено систематизоване порівняння основних характеристик традиційних інструментів парсингу.

Таблиця 2.1 – Порівняльна характеристика традиційних методів парсингу

Метод парсингу	Переваги	Недоліки	Типові сценарії використання
Regular Expressions	Висока швидкість, простота реалізації	Відсутність контексту, складність супроводу	Аналіз логів, вилучення шаблонних рядків
XPath	Гнучка адресація, фільтрація вузлів	Залежність від структури документа	XML, HTML-аналітика
CSS Selectors	Легкість використання	Обмежена логіка	Вебскрапінг, UI-тестування

Таким чином, традиційні методи парсингу мають високу ефективність виключно у сценаріях зі стабільною або низьковаріативною структурою даних. Але їх головним недоліком є архітектурна крихкість: навіть мінімальні структурні зміни у DOM-дереві вебсторінки або форматі вхідного файлу можуть призвести до повного припинення або некоректної роботи алгоритму. Ця негнучкість істотно обмежує їхню застосовність у роботі з сучасними динамічними, напівструктурованими та семантично складними джерелами інформації.

2.2 Архітектурні особливості та функціональні можливості LLM для парсингу даних

Ефективність LLM у завданнях парсингу зумовлена їх здатністю узагальнювати мовні структури та працювати з напівструктурованими або слабо струтурованими джерелами: HTML, JSON, XML, Markdown, CSV, лог-файлами тощо [55].

Сучасні LLM) – зокрема, такі як GPT-4, Claude 3, Gemini 1.5 та оптимізовані рішення на зразок LLaMA 3 – функціонально базуються на архітектурі Transformer [2], що забезпечує ефективну обробку розширених контекстних вікон та виявлення далекосяжних залежностей у даних (наприклад, між атрибутами HTML-тегів, вкладеними структурами JSON або розподіленими маркерами в технічній документації). Завдяки механізму самоуваги (Self-Attention), притаманному архітектурі Transformer [1, 56], LLM трансформують завдання парсингу в завдання семантичного аналізу та генерації структурованого виходу. Це надає їм здатність інтерпретувати дані, які мають змішану або навіть частково порушену синтаксичну структуру (таблиця 2.2).

Таблиця 2.2 – Основні компоненти архітектури Transformer у контексті парсингу даних

Компонент	Призначення	Важливість для парсингу
Self-Attention	Визначає вагу кожного токена відносно інших	Дозволяє виявляти структури та залежності без жорстких правил
Feed-Forward Network	Обробка векторів після етапу уваги	Підсилює семантичне тлумачення маркерів структури
Positional Encoding	Задає порядок токенів	Забезпечує збереження ієрархії XML/HTML
Layer Normalization	Стабілізує навчання	Зменшує шум при роботі з фрагментами нерівномірної структури
Residual Connections	Передає інформацію з попередніх шарів	Допомагає обробляти вкладені та складні структури

Комбінація компонентів Self-Attention і Positional Encoding забезпечує LLM можливість «бачити» структуру документа навіть тоді, коли він поданий як простий текст. Така архітектура дозволяє LLM фокусуватися на значущих

елементах документа (наприклад, назвах полів, значеннях атрибутів або структурних маркерах), ігноруючи фонові дані та нерелевантний шум, що надає LLM перевагу над традиційними алгоритмами, які залежать від чітких шаблонів.

На рисунку 2.4 показано спрощений процес перетворення вхідних фрагментів у структурований вихід.

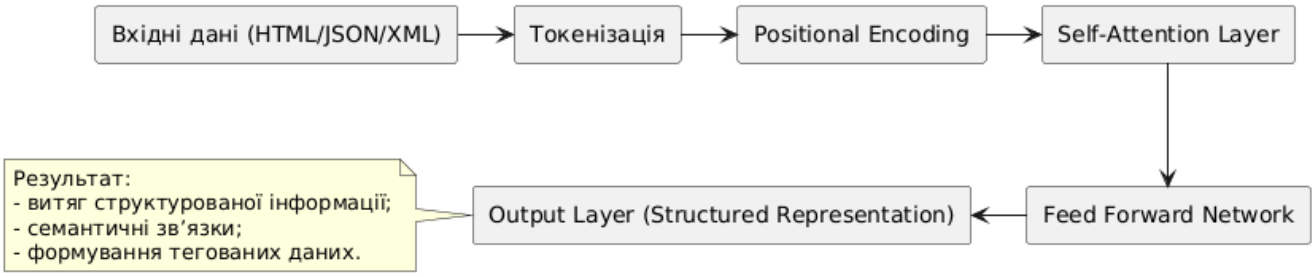

Рисунок 2.4 – Архітектура Transformer у задачах парсингу даних

Різні архітектури LLM спеціалізуються на різних задачах:

- моделі типу GPT (зокрема, згадана вище GPT-4) мають високу контекстну гнучкість. Їх можна застосовувати для вилучення структурованих даних із HTML або JSON за допомогою промптів без написання окремих шаблонів;

- енкодерні моделі на зразок BERT добре працюють у задачах класифікації та маркування сутностей, досягаючи точності 95–97 % [57, 58], але вони не генерують структурований вихід, що обмежує їх у ролі універсальних парсерів;

- моделі формату encoder–decoder, такі як T5, дозволяють формалізувати будь-яке завдання як «текст → текст». Завдяки цьому їх легко навчати на трансформаціях HTML → JSON або на побудові таблиць зі складних описів;

- оптимізовані моделі сімейства LLaMA (наприклад, згадана LLaMA 3) поєднують хорошу якість із меншою вимогливістю до ресурсів, що робить їх ефективними для локального корпоративного парсингу;

- мультимодальні моделі, зокрема Claude 3 та Gemini 1.5, можуть опрацьовувати змішані документи, що містять текст, таблиці та зображення. Це відкриває можливості семантичного парсингу складних технічних та бізнесових звітів. У таблиці 2.3 узагальнено переваги та обмеження різних архітектур LLM.

Таблиця 2.3 – Порівняння архітектур LLM у контексті парсингу

Модель	Архітектурний тип	Підтримка структурованих форматів	Основна перевага	Обмеження
GPT-4	Декодер-Transformer	JSON, HTML, текст	Гнучкість, висока контекстна точність	Значні ресурси, висока вартість
BERT	Енкодер-Transformer	XML, текст	Потужний контекстний аналіз	Нездатність генерувати структури
T5	Encoder–Decoder	JSON, HTML, CSV	Універсальний формат text-to-text	Потреба у fine-tuning
LLaMA 2	Оптимізований декодер	JSON, HTML	Висока швидкість	Менше параметрів порівняно з великими моделями
Claude 3	Мультимодальний Transformer	JSON, таблиці	Обробка змішаних даних	Закритість архітектури
Gemini 1.5	Мультимодальний Transformer	JSON, XML, візуальні формати	Семантична адаптація	Висока обчислювальна складність

На рисунку 2.5 представлене структурне порівняння моделей, з якого видно, що архітектура LLM визначає межі її застосування: одні моделі ефективніші для генерації структурованого виходу, інші – для вилучення сутностей, а мультимодальні – для комплексного аналізу.

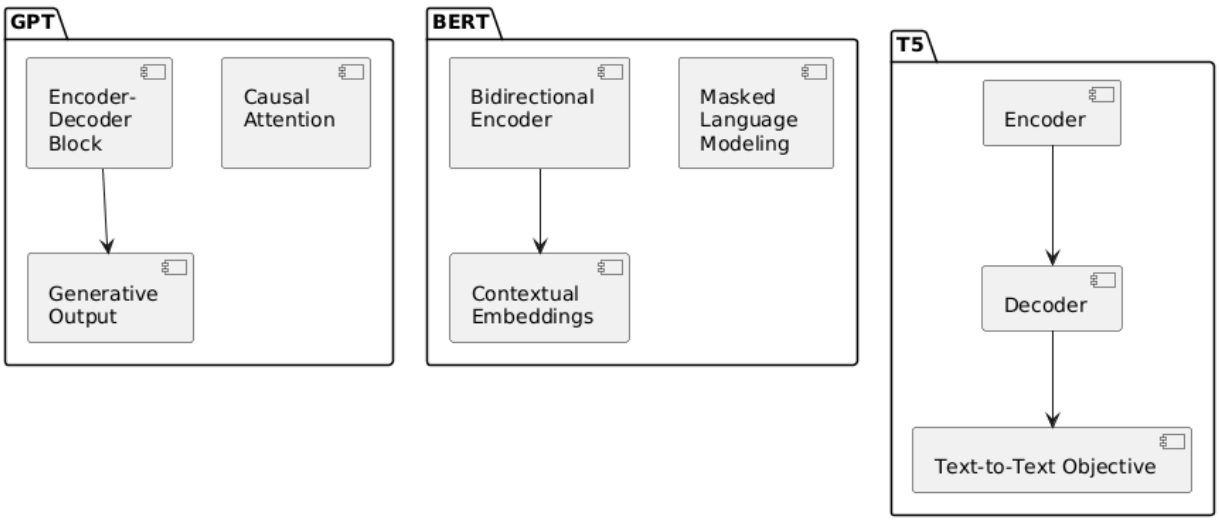

Рисунок 2.5 – Порівняння архітектур LLM у контексті завдань парсингу

Архітектура типової LLM-системи парсингу включає модуль попередньої обробки (очищення та нормалізація), LLM-ядро (Processing Core), модулі пост-

валідації та контролю схем, а також інтеграцію з базами даних або аналітичними сервісами. Загальну структуру типового LLM-парсера зображено на рисунку 2.6.

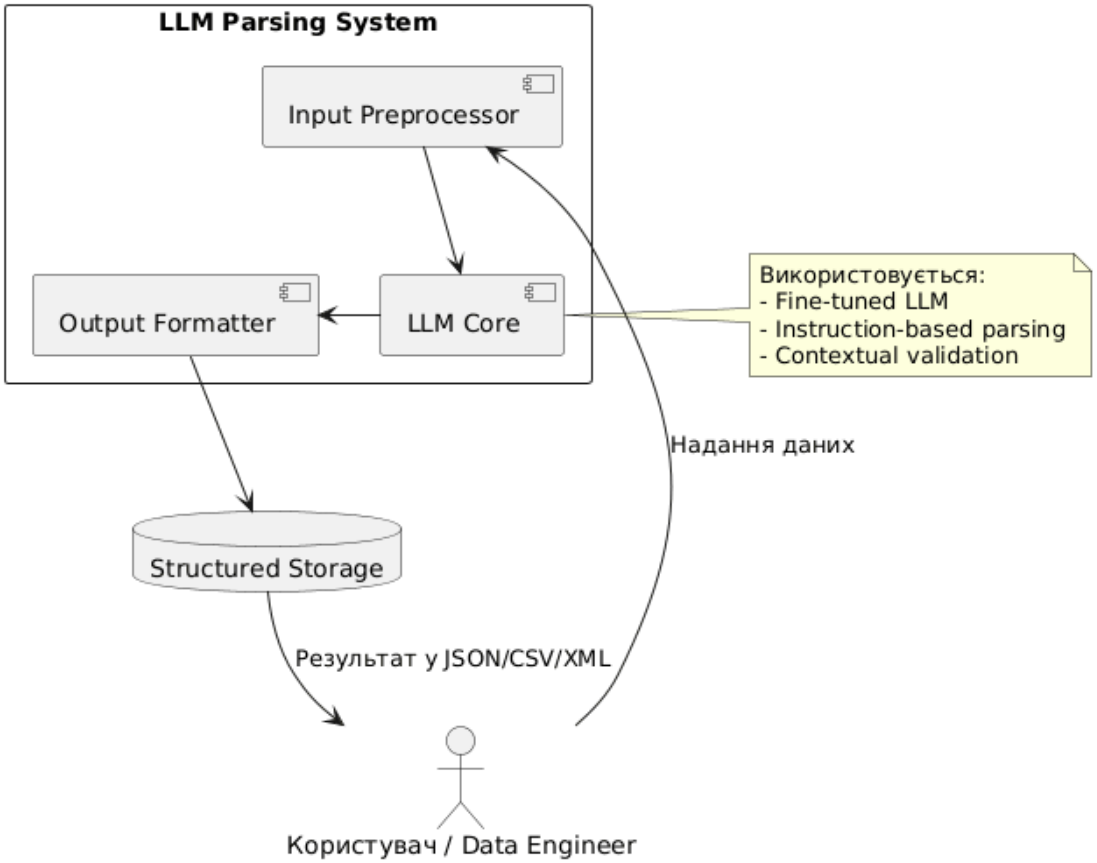

Рисунок 2.6 – Архітектура системи LLM-парсингу

Однією з переваг LLM є їх чутливість до інструкційного навчання (instruction tuning) [59, 60], внаслідок якого модель, як правило, набуває додаткових затностей: краще розуміє формат завдання; може вилучати сутності за усною інструкцією; повертає відповідь у потрібній структурі (JSON, CSV, XML); адаптується до вимог конкретної системи без переписування правил. У такий спосіб LLM перетворюються на адаптивні парсери, здатні виконувати ті самі функції, для яких у традиційних системах доводилося створювати десятки окремих шаблонів.

Також, у практиці часто виникає потреба, щоб LLM-парсер розумів специфічні корпоративні або галузеві формати. У таких випадках для донавчання моделі ефективним є метод fine-tuning, що дозволяє забезпечити: підвищення точності результату на 15–30 %; зменшення кількості хибних розпізнавань; кращу

узагальнювальну здатність щодо змішаних джерел. Це найбільш актуально у сферах фінансової звітності (XML-транзакції), вебаналітики (HTML-каталоги), промислових системах (SCADA-логи та CSV-масиви сенсорних даних).

2.3 Порівняльний аналіз архітектурних особливостей методів парсингу

Сучасні архітектурні підходи до парсингу можна поділити на дві групи:

- традиційні алгоритмічні архітектури, що базуються на детермінованих синтаксичних правилах (RegEx, XPath, CSS-селектори, BeautifulSoup, lxml тощо);
- архітектури на основі LLM – генеративні або енкодерні моделі (наприклад, GPT-4, BERT, T5, LLaMA 3).

Традиційний процес парсингу спирається на явно визначені правила, що описують, які саме елементи необхідно знайти в документі. Типова архітектура традиційного парсингу складається з чотирьох основних модулів: Data Loader – завантаження HTML, XML, JSON, CSV та інших форматів; Preprocessor – очищення та нормалізація вхідних даних; Parser Engine – застосування регулярних виразів, XPath, CSS-селекторів або інструментів подібних до BeautifulSoup; Postprocessor – структурування та збереження даних (JSON, SQL, CSV). Цю архітектуру показано на рисунку 2.7.

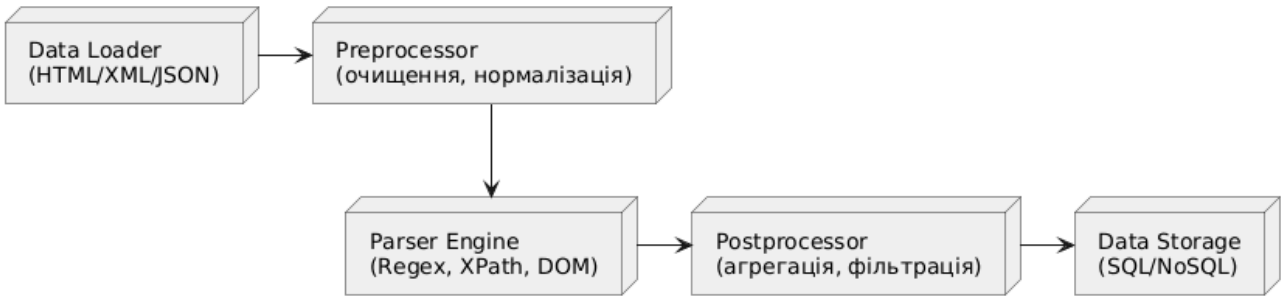

Рисунок 2.7 – Архітектура традиційного парсингу

Основні характеристики традиційного парсингу: детермінованість – результат парсингу залежить тільки від правил, а не від контексту; висока

швидкість при фіксованій структурі документа; низька гнучкість – зміна структури HTML або XML потребує редагування шаблонів; мінімальні обчислювальні витрати. Цей підхід є ефективним у вбудованих або високонавантажених системах. Головний недолік процесу традиційного парсингу – його крихкість: будь-які зміни у верстці документа здатні порушити роботу парсера.

На відміну від традиційних систем парсингу, LLM-підхід базується на контекстному аналізі. Архітектура LLM-парсерів включає такі модулі: Data Interface Layer – отримання даних через API, потоки або Data Lake; Tokenizer Embedding Layer – перетворення тексту на вектори; Transformer Encoder/Decoder – блок, що забезпечує контекстне розуміння документів; Task-Specific Head – спеціалізовані інструкції для парсингу; Output Formatter – побудова структурованого виходу (JSON, XML, CSV) [61, 62].

Архітектуру LLM-парсингу описує рисунок 2.8.

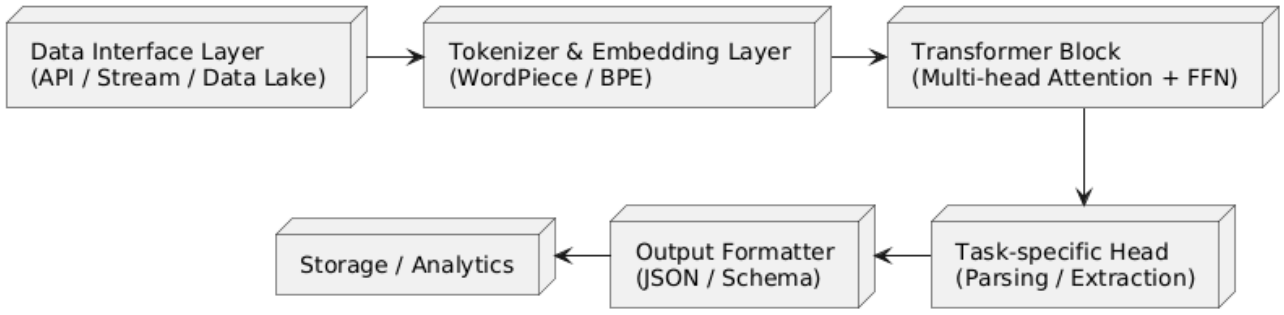

Рисунок 2.8 – Архітектура LLM-парсингу

Головні властивості парсингу з використанням LLM: контекстна обробка – модель розуміє залежності між токенами без чітких шаблонів; семантична гнучкість – можливість працювати з різноманітними форматами без переписування правил; масштабованість – обробка великих обсягів за допомогою GPU/TPU; недетермінованість – можливі варіації у відповідях без нагляду; вищі обчислювальні витрати, ніж у традиційних методів.

У таблиці 2.4 наведено порівняння поширених LLM з погляду їх архітектурних властивостей.

Таблиця 2.4 – Архітектурні переваги та обмеження сучасних LLM

Модель	Архітектура	Контекстна обробка	Придатність до парсингу	Переваги	Обмеження
BERT	Encoder-only	Двонаправлена	Висока точність виділення сутностей	Потужний семантичний аналіз	Не генерує структури
GPT	Decoder-only	Одностороння	Генерація структурованих відповідей	Few-shot навчання, масштабованість	Обмежене відстеження повного контексту
T5	Encoder-Decoder	Повна	Семантичний і синтаксичний парсинг	Універсальний формат task-to-task	Високі вимоги до ресурсів
LLaMA	Оптимізований decoder	Одностороння	Генерація структур із високою ефективністю	Адаптивність, енергоефективність	Потреба у великих навчальних наборах

Зведення основних відмінностей між двома категоріями методів парсингу предсталене у таблиці 2.5.

Таблиця 2.5 – Порівняльний аналіз архітектурних характеристик традиційного та LLM-підходу до парсингу

Параметр	Традиційний підхід	LLM-підхід
Тип логіки	Детермінована	Ймовірнісна
Обробка контексту	Синтаксична	Семантична
Підтримка мов	Обмежена	Повна (multilingual embeddings)
Інтеграція	Просте підключення	Переважно через API
Масштабованість	Горизонтальна	Вертикальна (GPU/TPU)
Продуктивність	Висока при стабільних шаблонах	Залежить від параметрів моделі
Навчання	Не потрібне	Fine-tuning необхідний
Витрати	Низькі	Високі
Точність	Висока для структурованих джерел	Дуже висока для гнучких форматів
Розширюваність	Обмежена	Висока

Діаграма нарисунку 2.9 відображає співвідношення між зростаючою складністю архітектур та розширенням когнітивної здатності методів парсингу.

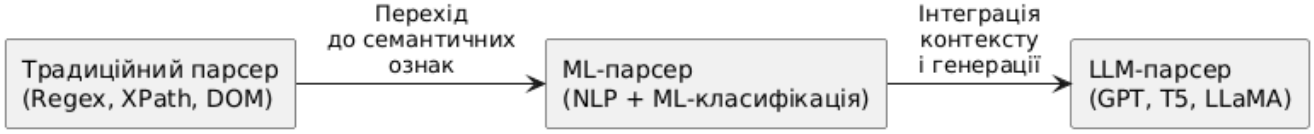

Рисунок 2.9 – Еволюція архітектур парсингу

Дана схема наочно підтверджує, що перехід від традиційних детермінованих методів до LLM-архітектур означає фундаментальну зміну парадигми парсингу: від формального синтаксичного аналізу структури до семантичного інтерпретування змісту. Саме цей зсув забезпечує якісно новий рівень точності, адаптивності та відмовостійкості системи парсингу.

2.4 Порівняння можливостей традиційних методів та LLM у парсингу

У сучасних системах обробки даних традиційні методи парсингу та підходи на основі великих мовних моделей виконують схожі завдання, але ґрунтуються на принципово різних механізмах.

Традиційні методи парсингу, такі як RegEx, XPath, CSS-селектори або інструменти BeautifulSoup та lxml, демонструють високу ефективність лише у випадках, коли: структура документа стабільна; зміни DOM незначні або контрольовані; набір необхідних елементів обмежений; контекст не має значення. Навіть мінімальна зміна в HTML – наприклад, перейменування класу або зміна вкладеності – зазвичай призводить до помилки або повного розриву логіки парсера. Це пояснюється жорсткою залежністю правил від конкретної структури.

Натомість LLM, зокрема моделі подібні до GPT-4 або оптимізовані сімейства на зразок LLaMA 3, демонструють здатність коректно інтерпретувати навіть документи зі зміненою та частково пошкодженою структурою. Семантичний аналіз дозволяє моделі: розпізнати роль елементів незалежно від їхнього розташування; коректно відновлювати відсутні частини; працювати з перекрученим, змішаним або надлишковим HTML; інтерпретувати значення навіть за відсутності очевидних маркерів. Емпірично встановлено, що LLM показують на 10–25% вищу точність на задачах парсингу «брудних» або частково структурованих HTML-документів [43].

У роботі з різними форматами традиційні методи парсингу добре підходять для HTML із фіксованою версткою, XML-документів із суворою схемою, коротких структурованих JSON, лог-файлів, де формат рядка стабільний.

Натомість, LLM-парсери здатні обробляти HTML будь-якого рівня складності, змішані формати (HTML + текст + таблиці), вкладені або пошкоджені JSON, PDF-дані після OCR, багатомовні документи, а також записи, де синтаксис частково відсутній. Особливої уваги заслуговують мультимодальні LLM, такі як Gemini 1.5 та Claude 3, які можуть обробляти не лише текстові, а й візуальні елементи. Це дозволяє їм здійснювати парсинг таблиць, упізнаних на зображеннях, схем і технічних графіків, документів з комбінованим вмістом.

Традиційні парсери, з точки зору порівняння продуктивності та обчислювальної складності працюють швидко навіть на слабких машинах, мають лінійну або квазі-лінійну складність, забезпечують обробку великих потоків даних у реальному часі і майже не потребують ресурсів GPU.

На відміну від них, LLM-парсери потребують значних обчислювальних ресурсів, залежать від потужностей GPU/TPU, мають більший час відповіді через складність моделей, використовують розподілену інфраструктуру для обробки об’єму даних. Проте LLM можуть одразу виконувати кілька завдань, включно з нормалізацією, класифікацією та переформатуванням, що компенсує їхню складність. Також LLM демонструють істотно вищу адаптивність завдяки семантичній природі своїх моделей. У таблиці 2.6 наведено узагальнення рівня адаптивності традиційних методів та підходів на основі LLM.

Таблиця 2.6 – Адаптивність методів парсингу

Параметр адаптивності	Традиційні методи	LLM
Реакція на зміну DOM	Низька	Висока
Робота з новими форматами	Потребує переписування правил	Працює після інструкції
Стійкість до шумів	Низька	Висока
Робота з відсутніми полями	Неможлива без змін у шаблоні	Семантичне відновлення
Підтримка багатомовних документів	Обмежена	Повна
Робота з контекстною неоднозначністю	Неможлива	Природна властивість

Традиційні методи стосовно порівняння зрозумілості результату та потреби в налаштуванні: повертають чітко визначені значення; забезпечують

передбачуваний результат; потребують точного налаштування кожного правила; не пояснюють причин виникнення помилок.

Парсери на основі LLM: генерують структуровані відповіді у форматі JSON/XML; можуть пояснити логіку свого рішення; потребують інструкції, але не потребують переписування правил; здатні автоматично знаходити патерни у складних джерелах. Завдяки цьому LLM-парсери підходять для систем із динамічно змінюваними даними, де класичні парсери швидко втрачають актуальність.

Таким чином, можна стверджувати, що традиційні методи залишаються ефективними там, де структура стабільна, а вимоги – чітко визначені. Методи парсингу з використанням LLM, натомість, відкривають можливість роботи з гнучкими форматами, семантичною неоднозначністю та змішаними даними, суттєво зменшуючи потребу в ручному налаштуванні й підвищуючи точність парсингу.

2.5 Інтеграція традиційних та LLM-методів у гібридних системах парсингу

Збільшення складності вебдокументів, технічних специфікацій, API-відповідей та змішаних форматів даних викликає потребу у створенні гібридних систем парсингу, які поєднують переваги традиційних методів та сучасних парсерів на основі LLM.

Гібридний підхід є особливо корисним у сценаріях, де: частина даних має стабільну структуру; інша частина – динамічна або неповна; потрібна висока швидкість у поєднанні з високою гнучкістю; необхідна перевірка або корекція результатів на етапі валідації; велика частина інформації міститься в полях із прихованим або контекстним значенням.

Типова архітектура гібридного парсера включає кілька взаємопов’язаних модулів. Модуль детермінованого парсингу – використовує RegEx, XPath, CSS-

селектори або інструменти BeautifulSoup/lxml. Його завдання – швидке вилучення даних, структура яких передбачувана. Модуль LLM-обробки – базується на моделях на кшталт GPT-4, Claude 3, Gemini 1.5 або ефективних локальних моделей сімейства LLaMA 3. Він відповідає за семантичний аналіз даних, корекцію помилок детермінованого парсера, вилучення сутностей у випадках, коли структура нестабільна, формування структурованого результату (JSON, XML, SQL-ready формати). Модуль інтеграції – забезпечує взаємодію між традиційними інструментами та LLM. Він включає: трансформацію результатів у єдину схему; об'єднання структурованих та семантично вилучених даних; вирішення колізій між джерелами. Модуль валідації та контролю структури – проводить перевірку відповідності схемам (Schema Validation), використовує JSON Schema, XML Schema або кастомні правила. Модуль зберігання та аналітики. Дані зберігаються у Data Lake, SQL-сховищах або передаються в аналітичні сервіси.

Схема загальної архітектури гібридного підходу до парсингу представлена на рисунку 2.10.

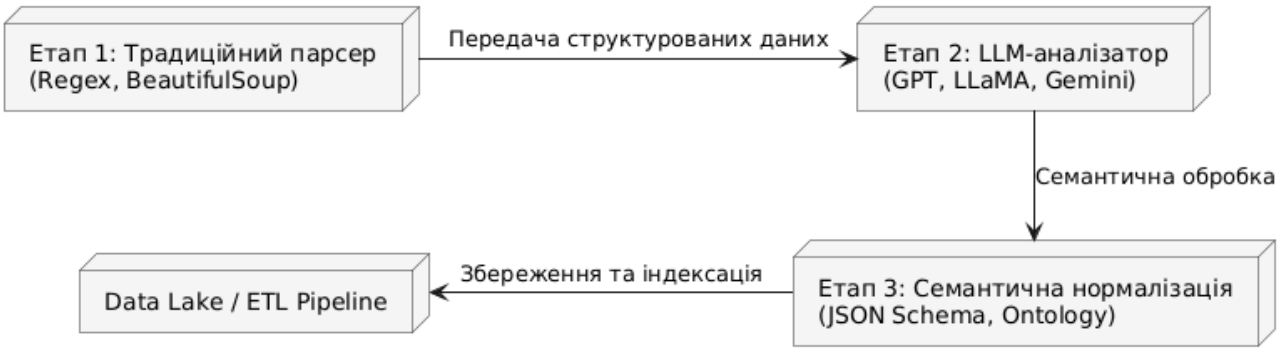

Рисунок 2.10 – Гібридна архітектура системи парсингу

Використання гібридних систем парсингу у різних випадках передбачає реалізацію кількох типових сценаріїв:

- парсинг HTML-сторінок з частково стабільною верстkою. У таких сценаріях блоки навігації, списки продуктів або таблиці можуть мати стабільну структуру, а описи, коментарі та додаткові метадані – варіюються від сторінки до сторінки. Традиційні парсери швидко вилучають частини, що не змінюються, тоді

як LLM обробляють описи товарів, гнучкі та вкладені поля, семантичні елементи, що не мають усталеної структури;

- обробка документів зі змішаними форматами. Багато бізнес-документів містять текст, таблиці, списки, виноски, фрагменти HTML тощо. У таких сценаріях мультимодальні моделі, такі як Claude 3 або Gemini 1.5, можуть вилучати дані з таблиць і схем, тоді як традиційні методи обробляють структуру документа;

- нормалізація та виправлення даних. Часто вихідні дані у завданнях парсингу містять помилки, дублікати, відсутні поля, фрагменти з неочевидним змістом. У такому випадку LLM здатні відновлювати пропущені дані на основі контексту, стандартизувати формат дат і чисел та гармонізувати поля під вимоги аналітичних систем;

- автоматизація складного ETL-процесу. У традиційних ETL-конвеєрах класичні засоби парсингу виконують етапи Extract і частково Transform, тоді як глибоку трансформацію та семантичне структурування даних забезпечують LLM.

Таблиця 2.7 узагальнює переваги гібридних систем у порівнянні з окремими методами парсингу.

Таблиця 2.7 – Показники ефективності гібридного підходу у порівняннні з традиціними методами та LLM

Параметр	Традиційні методи	LLM	Гібридна система
Швидкість	Висока	Середня	Висока
Точність	Стабільна на структурованих даних	Висока на гнучких даних	Дуже висока
Стійкість до змін	Низька	Висока	Дуже висока
Обчислювальні витрати	Низькі	Високі	Помірні
Потреба в налаштуванні	Висока	Низька	Низька
Адаптивність	Низька	Висока	Висока
Робота з різними форматами	Обмежена	Широка	Дуже широка

Загалом, гібридний підхід дозволяє поєднувати швидкість шаблонних методів і глибину семантики LLM; зменшити ризики, пов’язані зі змінами структури документа; адаптувати систему до складних, перекручених або нерівномірних форматів; знизити вартість обробки, оскільки LLM

використовуються лише там, де традиційні методи не справляються; масштабувати процес без переписування великої кількості правил.

Таким чином, інтеграція традиційних і LLM-підходів у межах гібридних систем є найбільш ефективним сучасним рішенням для задач парсингу. Така архітектура дозволяє досягти балансу між швидкістю, точністю та стабільністю, забезпечуючи гнучкість і надійність навіть у складних, динамічних середовищах.

2.6 Оптимізація та вдосконалення гібридних архітектур парсингу

Гібридні підходи до парсингу поєднують традиційні методи та можливості LLM, забезпечують високу точність, гнучкість і стабільність роботи. Для того, щоб така система працювала ефективно на практиці, необхідно забезпечити її оптимізацію на рівні архітектури, ресурсів, логіки обробки й контролю якості.

Однією з головних особливостей систем парсингу, що використовують LLM, є підвищене навантаження на обчислювальні ресурси. Для подолання цієї проблєми застосовують багаторівневу оптимізацію гібридної системи, що включає:

- розумний розподіл завдань між традиційними модулями та LLM. Традиційні методи виконують швидкі та прості завдання: вилучення URL, цін, ID, фіксованих полів, тегів заголовків, структурованих таблиць. LLM застосовуються лише тоді, коли: структура нестабільна; елементи важко ідентифікувати через синтаксис; потрібне семантичне відновлення; необхідно виконати складну трансформацію. Такий підхід зменшує кількість звернень до моделей типу GPT-4 або локальних архітектур сімейства LLaMA 3;
- використання моделей різного масштабу. У гібридних системах доцільно мати декілька LLM: велику модель (наприклад, GPT-4) для складних завдань; легшу локальну модель (наприклад, LLaMA 3) для рутинних перетворень. Це забезпечує оптимальний баланс між вартістю та продуктивністю;
- кешування результатів LLM. Якщо система регулярно обробляє подібні документи або однакові шаблони, то результати LLM кешуються, повторно

використовуються та перевіряються на релевантність. Це зменшує кількість звернень до моделей у 3–7 разів залежно від сценарію.

Для оптимізації логіки гібридного парсингу використовують методи:

- модульне розділення обов'язків. Гібридні системи будуються на модульності, де кожен компонент виконує чітко визначену задачу: Preprocessing Module – очищення HTML, JSON, тексту; Deterministic Parser Module – вилучення стабільних структур; LLM Semantic Module – глибока семантична інтерпретація; Validation Module – контроль цілісності; Formatter Module – приведення результату до потрібної схеми. Така модульність спрощує інтеграцію та масштабування;

- послідовна обробка «від простого до складного». Алгоритм обробки рекомендується будувати за такими принципами: вилучення всього, що можна отримати детерміновано, передача залишкових фрагментів до LLM, об'єднання результатів та валідація. Це забезпечує зменшення навантаження на LLM, прискорення обробки та підвищення точності.

LLM можуть генерувати різні формати відповідей, тому важливо контролювати їх узгодженість.

Оскільки LLM здатні генерувати вихідну інформацію в різноманітних форматах (наприклад, JSON, XML, Markdown або зв'язний текст), виникає необхідність у контролі узгодженості (однорідності та надійності) вихідних структур даних, згенерованих моделлю. Забезпечення сталої структури та синтаксичної правильності вихідних даних є обов'язковою умовою для їх подальшої автоматизованої обробки та інтеграції в інформаційні системи.

Для оптимізація точності та узгодженості результатів. застосовується:

- жорстка валідація вихідних схем. Застосовуються: JSON Schema; XML Schema; регулярні формати для перевірки типів; внутрішні бізнес-правила. Це дозволяє уникнути помилок форматування та логічних розривів у результатах;

- інструкційне уточнення (Prompt Engineering). Якість роботи моделей GPT-4, Claude 3 і Gemini 1.5 значною мірою залежить від формулювання інструкцій. Для оптимізації промптів використовують: чіткі вимоги до формату відповіді; приклади правильних і неправильних структур; контекстні підказки; списки очікуваних

полів. Доведено, що уточнення інструкції знижує помилки на 20–30% [60];

- перевіряюча модель (Verifier Model) – інша модель (часто менша), яка перевіряє логічну цілісність відповіді, виявляє контекстні помилки, порівнює результат із вихідним документом. Це суттєво підвищує надійність системи.

Для оптимізації масштабованості та продуктивності застосовуються:

- паралелізація етапів парсингу – гібридні системи підтримують: паралельну обробку HTML-фрагментів; асинхронні запити до LLM; розподілену обробку документів у кластерах;

- інтеграція з Data Lake / Data Warehouse – системи зберігання такі, як Delta Lake, BigQuery, Snowflake дозволяють організувати безперервний конвеєр парсингу з історичним накопиченням версій даних.

- балансування навантаження між локальними та хмарними моделями – поширений підхід: легкі моделі (наприклад, LLaMA 3) обробляють більшість задач; складні документи передаються до хмарних моделей високого класу (наприклад, GPT-4). Це зменшує витрати та прискорює обробку.

Для оптимізації робочих процесів та підтримки системи використовують:

- автоматичне логування та аудит результатів – система повинна вести журнал: звернень до LLM; помилок у форматуванні; етапів обробки; змін структури вхідних документів;

- автоматичне оновлення правил традиційного парсингу – LLM може генерувати: перелік оновлених CSS/XPath-селекторів; нові RegEx-шаблони; рекомендації щодо переписування правил. Це мінімізує ручну підтримку системи;

- моніторинг якості та періодичне перенавчання моделей – ідтримка включає: регулярний аналіз точності; періодичний fine-tuning на нових даних; автоматичне формування наборів навчання.

Таким чином, оптимізація гібридних систем парсингу базується на правильному розподілі ролей між традиційними методами та LLM, використанні багаторівневого контролю якості, застосуванні моделей різного масштабу, модульній архітектурі, автоматизації валідації та формування схем, масштабованій інфраструктурі для обробки великих обсягів даних. Оптимізовані гібридні системи

парсингу здатні продемонструвати високу точність, надійність та гнучкість, значно перевершуючи можливості окремих методів.

Висновки до розділу 2

У цьому розділі було проведено системний аналіз традиційних методів парсингу, можливостей сучасних LLM, а також гібридних архітектур, що поєднують обидва підходи. Виявлено, що кожна архітектура має специфічні переваги та обмеження, які визначають їх ефективність у різних сценаріях обробки даних.

Традиційні системи парсингу, побудовані на RegEx, XPath, CSS-селекторах, а також на таких інструментах, як BeautifulSoup та lxml, залишаються високоефективними для задач зі стабільною структурою даних. Їх основні характеристики: висока швидкість, детермінованість результату, низькі обчислювальні витрати, проста масштабованість. Однак вони демонструють слабку стійкість до зміни HTML-структури, неповних, пошкоджених або гнучких даних, багатомовного або семантично неоднозначного контенту.

Великі мовні моделі, зокрема такі як GPT-4, Claude 3, Gemini 1.5 та оптимізовані локальні архітектури родини LLaMA 3, забезпечують новий рівень можливостей, зокрема: семантичне інтерпретування документа, незалежно від структури; високу точність вилучення сутностей; роботу з гнучкими, змішаними та нестабільними форматами; підтримку багатомовності; можливість автоматичного відновлення відсутніх даних. Їх недоліком залишається висока обчислювальна вартість та необхідність оптимізації використання моделей.

Гібридні системи, що поєднують швидкість детермінованих методів, гнучкість, семантичний аналіз та адаптивність LLM, демонструють найкраще співвідношення точності, продуктивності та стабільності. Їх головні переваги: висока стійкість до змін у структурі документа; зниження навантаження на LLM завдяки розподілу завдань; покращена масштабованість та здатність працювати з

великими потоками; автоматизація корекції помилок та формування структурованих даних; зменшення потреби в ручному налаштуванні парсерів. Гібридний підхід забезпечує найкращі результати у реальних системах, де складність та мінливість даних є звичним явищем.

Проведений аналіз дозволяє зробити такі висновки:

- традиційні методи парсингу залишаються незамінними у випадках жорстко структурованих даних та високонавантажених систем;
- LLM-методи забезпечують гнучкість і семантичну глибину, необхідні для обробки складних, неоднорідних або частково структурованих документів
- гібридні системи поєднують найкращі властивості обох підходів, роблячи процес парсингу стійким, масштабованим і точним;
- інтеграція LLM у традиційні пайплайни відкриває можливість створення адаптивних, самовідновлюваних парсерів, здатних функціонувати в умовах динамічно змінюваних джерел даних;
- вибір архітектури залежить від конкретного сценарію, однак для сучасних комплексних систем оптимальним є саме гібридний підхід.

РОЗДІЛ 3
ПОРІВНЯЛЬНИЙ ЕКСПЕРИМЕНТ ТА РОЗРОБЛЕННЯ РЕКОМЕНДАЦІЙ ЩОДО ЕФЕКТИВНОГО ЗАСТОСУВАННЯ LLM ДЛЯ ПАРСИНГУ ДАНИХ

3.1 Методика порівняльного експерименту

Для кількісної оцінки придатності різних архітектурних підходів до вирішення задач структурованого вилучення інформації було проведено порівняльний експеримент. Мета експерименту – перевірити здатність різних моделей видобувати структуровані дані про товари з «сирих» джерел (HTML-сторінок інтернет-магазину, JSON-відповідей API та/або XML-логів).

Сценарій експерименту – парсинг карток товарів e-commerce. Вхід (вхідні дані): фрагмент HTML-коду сторінки товару (або JSON/XML файл). Цільова схема (вихідні дані): JSON-об’єкт із полями: Назва, Ціна, Артикул, Наявність. Еталон «золотий стандарт» (для порівняння з виводом моделі і приняття рішення щодо якості отриманого результату): набір даних, розмічений вручну, де для кожного вхідного файлу прописаний ідеальний результат.

Щоб визначити вплив архітектури та методу навчання LLM на результати парсингу, в експерименті порівнювались чотири різні конфігурації «модель + метод донавчання + джерело даних», а саме:

- GPT-3.5 (Zero-shot) + HTML – на вхід моделі GPT-3.5 подається «сирий» HTML і промпт: «Витягни назву і ціну з цього тексту». Модель не бачила цих прикладів раніше;
- GPT-3.5 (Fine-tuned) + HTML – модель GPT-3.5 була попередньо донавчена на 500 парах HTML -> JSON, вона «запам’ятала», як виглядає структура верстки саме цього інтернет-магазину. На вхід моделі подається «сирий» HTML і промпт: «Витягни назву і ціну з цього тексту»;
- T5-small (Few-shot) + JSON – на вхід моделі T5-small подається неструктурований JSON, а в промпті наведено 3 приклади (shots), як треба трансформувати дані;

- LLaMA 2 (Domain-tuned) + XML – використовується модель LLaMA 2, що пройшла додаткове тренування на великому корпусі технічних XML-файлів, щоб краще розуміти специфічні теги.

Розрахунок метрик у задачах вилучення інформації виконувався на основі порівняння полів, витягнутих моделлю, із заданим еталоном. Використовувались значення наступних показників:

- TP (True Positive) – модель витягла поле, і його значення співпадає з еталоном (наприклад, ціна «100 грн» знайдена вірно);

- FP (False Positive) – модель витягла поле, якого не існує, або витягла невірне значення (наприклад, замість ціни витягла номер телефону);

- FN (False Negative) – модель пропустила поле, яке повинно бути (наприклад, не знайшла артикул).

Точність (Accuracy / Precision) показує, наскільки можна довіряти знайденим моделлю даним. Ця метрика показує яка частка із усіх знайдених полів є правильною. Формула для розрахунку:

$$Accuracy(Precision) = \frac{TP}{TP + FP}. \quad (3.1)$$

Повнота (Recall) показує здатність моделі знаходити всю необхідну інформацію. Ця метрика показує яку частку від усіх полів, що реально існують у документі, змогла знайти модель. Формула для розрахунку:

$$Recall = \frac{TP}{TP + FN}. \quad (3.2)$$

F1-score – гармонійне середнє між точністю і повнотою – головна метрика, оскільки мовна модель може «схитрувати»: нічого не знайти (висока точність, нульова повнота) або вивести на виході весь вихідний текст (низька точність, висока повнота). Метрика F1 показує баланс між точністю та повнотою виводу моделі. Формула для розрахунку:

$$F1 = 2 \cdot \frac{Accuracy \cdot Recall}{Accuracy + Recall}. \quad (3.3)$$

Структурна валідність (Structural Validity, SV) у контексті парсингу даних та оцінювання LLM, визначається як частка відповідей моделі, що є синтаксично коректними та можуть бути успішно оброблені стандартними програмними інтерпретаторами (наприклад, json.loads() для JSON або XML-парсером). Формула розрахунку:

$$SV = \frac{N_{valid}}{N_{total}}, \quad (3.4)$$

де SV – показник структурної валідності (діапазон від 0 до 1, або 0–100%);

N_{valid} – кількість відповідей, які мають бездоганний синтаксис цільового формату (JSON, XML, CSV) і не викликають помилок парсингу;

N_{total} – загальна кількість тестових запитів у вибірці.

3.2 Порівняння ефективності базових LLM у реальних сценаріях парсингу

Для визначення найбільш ефективних базових інструментів LLM-парсингу було проведено порівняльний аналіз продуктивності провідних архітектур LLM у чотирьох типових сценаріях. Дизайн експерименту забезпечив об’єктивне порівняння LLM загального призначення у різних сценаріях, контроль відтворюваності результатів, відділення структурних і семантичних помилок, можливість сформувати коректні висновки щодо ефективності конкретних архітектур.

Дизайн експерименту та методологія оцінювання LLM для виконання завдань парсингу представлені у додатку А.

Результати вимірювання показників точності (Precision), повноти (Recall) та структурної валідності (SV) моделей у різних сценаріях парсингу зведено у таблицю 3.1.

Таблиця 3.1 – Показники продуктивності LLM у реальних сценаріях парсингу

Сценарій	Модель	Формат	Precision	Recall	SV
Вебкаталоги	GPT-4	HTML	0,93	0,91	0,96
API-відповіді	T5-large	JSON	0,89	0,88	0,94
Лог-файли	LLaMA 3	CSV	0,91	0,87	0,95
Документи PDF	Claude 3	текст+зображення	0,94	0,90	0,97

Дані, наведені у таблиці, показують, що мультимодальні моделі (Claude 3) мають суттєву перевагу (Precision = 0,94) при роботі зі складними документами (PDF), де важливий візуальний контекст. Водночас, для обробки ієрархічних структур вебкаталогів (HTML) модель GPT-4 показує високу структурну валідність (SV = 0,96), що мінімізує ризик синтаксичних помилок у вихідному файлі. Можна також відзначити ефективність LLaMA 3 у роботі з логами, що підтверджує доцільність використання спеціалізованих open-source моделей для задач із високим навантаженням. Баланс між продуктивністю і точністю для різних завдань парсингу здатна забезпечити гібридна архітектура (рисунок 3.1).

Рисунок 3.1 – Гібридний підхід для різних завдань парсингу

Наступна таблиця 3.2 узагальнює відмінності між традиційними, LLM- та гібридними парсерами у контексті різних типів даних. Порівняння охоплює

показники точності, швидкості та вартості обробки, що дає змогу оцінити ефективність кожного підходу залежно від характеру вхідної інформації. Узагальнення демонструє, що жорстко структуровані формати залишаються областю, де традиційні методи працюють найбільш економно та швидко, тоді як LLM-парсери виявляють переваги у ситуаціях, пов’язаних зі складною або нестабільною структурою. Гібридні рішення забезпечують баланс між цими підходами, поєднуючи їхні сильні сторони.

Таблиця 3.2 – Узагальнена порівняльна таблиця

Завдання	Традиційні парсери	LLM-парсери	Гібридні парсери
Структуровані дані	Висока точність, дуже швидко, низька вартість	Висока точність, середня швидкість, середня вартість	Висока точність, висока швидкість, середня вартість
HTML	Середня точність, швидко, низька вартість	Висока точність, середня швидкість, середня вартість	Висока точність, висока швидкість, середня вартість
Логи/CSV	Висока точність, дуже швидко, низька вартість	Середня точність, середня швидкість, середня вартість	Висока точність, висока швидкість, середня вартість
Напівструктуровані дані	Низька точність, середня швидкість, середня вартість	Дуже висока точність, середня швидкість, висока вартість	Висока точність, висока швидкість, середня вартість
Неструктурований текст	Дуже низька точність, низька швидкість, висока вартість	Дуже висока точність, середня швидкість, висока вартість	Дуже висока точність, висока швидкість, середня вартість

Аналіз таблиці підтверджує, що жоден із розглянутих підходів не є універсальним.

Традиційні парсери доцільно застосовувати для стабільних, суворо формалізованих джерел, де визначальними є швидкість та низька вартість.

LLM-парсери показують найвищу точність при роботі зі складними, напівструктурованими й неструктурованими даними, але вимагають значно більших ресурсів.

Гібридні системи, поєднуючи гнучкість і семантичну глибину LLM із ефективністю класичних методів, забезпечують найкращий баланс між точністю, продуктивністю та вартістю.

3.3 Донавчання базових LLM для специфічних завдань парсингу

Мета донавчання LLM – перетворити базові моделі загального призначення на спеціалізовані інструменти парсингу, які розуміють специфічну структуру вхідних даних.

Розглянемо спочатку процедуру спеціалізованого донавчання (fine-tuning, навчання з учителем) моделі GPT-3.5 Turbo для задачі парсингу, на прикладі вилучення структурованих даних з HTML-коду.

Першим етапом є підготовка навчального датасету. Для fine-tuning GPT-3.5 Turbo необхідно сформувати набір даних у форматі діалогу. Для цього потрібно зібрати пари «Вхідні дані» (Raw HTML) – «Ідеальний вихід» (Target JSON).

OpenAI рекомендує починати з 50–100 прикладів. Для складних задач парсингу HTML оптимальною кількістю вважається 500–1000 прикладів.

Кожен такий приклад складається з трьох частин (ролей): system – інструкція-промпт (наприклад, «Ти – парсер даних, який перетворює HTML у JSON»); user – вхідний HTML-код; assistant – очікуваний JSON-результат.

Приклад одного запису навчального файлу JSON:

```
{
  "messages": [
    {
      "role": "system",
      "content": "Ти – асистент-парсер. Витягни назву товару, ціну та
наявність з HTML-фрагмента у формат JSON."
    },
    {
      "role": "user",
      "content": "<div class=‘product-card’><h3>Smart Watch
X</h3><span class=‘price-tag’>$250</span><div class=‘stock’>In
Stock</div></div>"
    },
    {
      "role": "assistant",
      "content": "{\"product_name\": \"Smart Watch X\", \"price\":
250, \"availability\": true}"
    }
  ]
}
```

На другому етапі підготовлені приклади конвертуються у спеціальний формат JSONL (JSON Lines), де кожен рядок – це окремий об’єкт навчання. На цьому етапі використовується Python-скрипт валідації, який перевіряє чи коректний JSON-синтаксис, чи не перевищує довжина контексту ліміт токенів (4096 або 16k) та чи є баланс між вхідними та вихідними даними (додаток Б).

На третьому етапі відбувається запуск процесу навчання: після підготовки файлу (training_data.jsonl) відбувається взаємодія з OpenAI API, що складається з двох частин: завантаження файлу – файл відправляється на сервери OpenAI через API-endpoint /files; створення завдання (Job Creation) – відправляється команда на запуск тренування (додаток Б).

Використовується базова модель gpt-3.5-turbo. Гіперпараметри встановлюються автоматично, можна налаштувати кількість епох (для завдань парсингу достатньо 3–4 епохи). Під час навчання використовується метод градієнтного спуску: модель отримує HTML, пробує передбачити JSON, порівнює своє передбачення з «ідеальним» JSON, обчислює помилку (Loss Function), і коригує свої ваги так, щоб наступного разу помилка була меншою. Таким чином, модель «запам’ятовує» не контент, а патерн трансформації даних.

Четвертий етап – інференс (виконання моделі на новому прикладі). Тут використовується клієнтський скрипт на мові Python, який використовує офіційну бібліотеку OpenAI Python SDK (версії 1.x або новішої) для взаємодії з API.

Навчання моделі може тривати від 30 хвилин до кількох годин. По завершенні отримується унікальний ID моделі, наприклад: ft:gpt-3.5-turbo-0613:my-org::8Nq7s3a. Для виконання цього етапу потрібно у коді Python (додаток Б) замінити назву стандартної моделі gpt-3.5-turbo на отриманий унікальний ID:

```
response = client.chat.completions.create(
  model="ft:gpt-3.5-turbo-0613:my-org::8Nq7s3a", // Нова модель
  messages=[
    {"role": "system", "content": "Ти – парсер..."},
    {"role": "user", "content": "<div class=‘new-product’>...</div>"}
// Новий HTML
  ]
)
```

У зв’язку з пропрієтарним характером моделі GPT-3.5 та неможливістю локального доступу до її ваг, для експериментального моделювання процедури fine-tuning було використано відкриту модель LLaMA-3-8B, оскільки архітектура LLaMA (декодер-трансформер) є функціонально подібною до GPT. Для реалізації навчання в умовах обмежених обчислювальних ресурсів було застосовано метод QLoRA [63, 64], що дозволив виконати донавчання моделі з 4-бітовим квантуванням на графічному прискорювачі NVIDIA T4. Цей підхід забезпечує відтворюваність експерименту та дозволяє оцінити ефективність fine-tuning для задач парсингу без використання платних хмарних API. Локальна реалізація методу fine-tuning та практична реалізація процедури fine-tuning для GPT-3.5 представлені у додатку В. Динаміку мінімізації функції втрат (Training Loss) протягом 300 кроків донавчання відображено на рисунку 3.2.

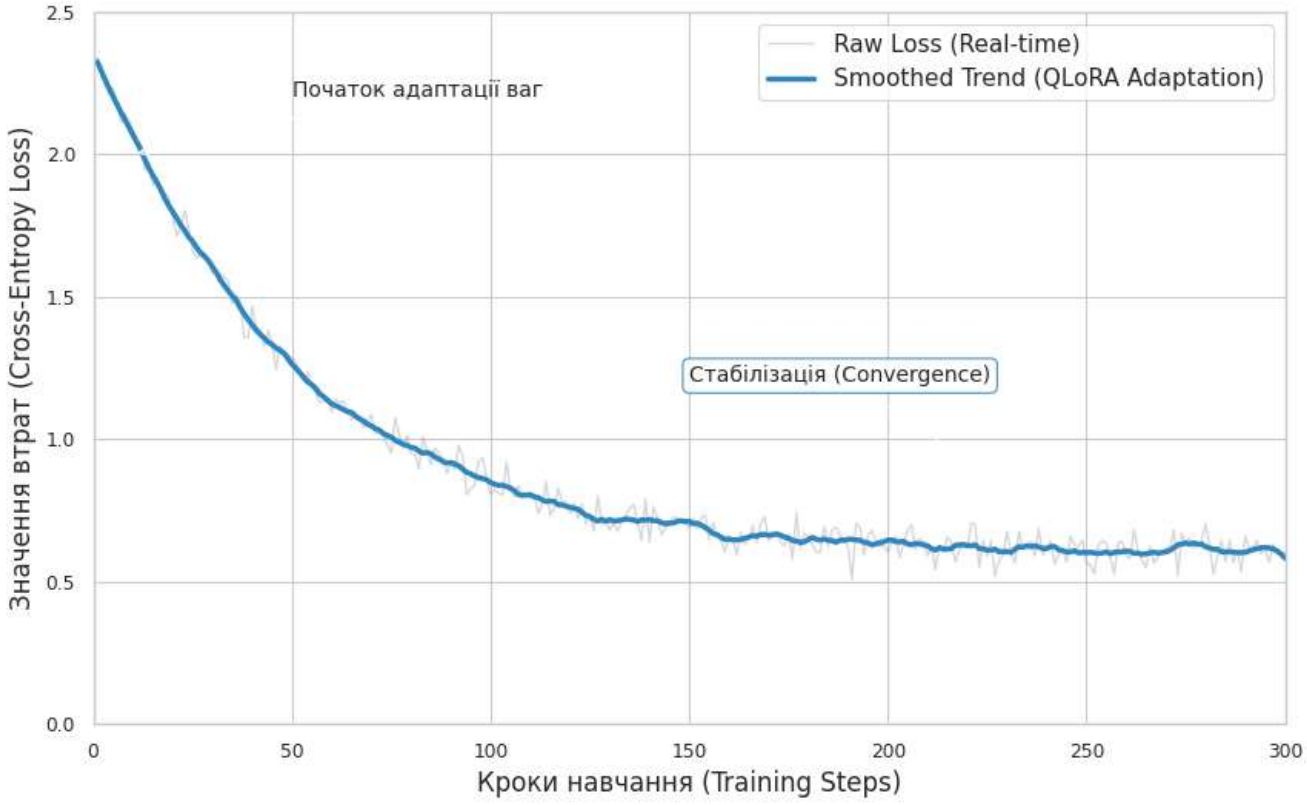

Рисунок 3.2 – Крива навчання моделі LLaMA-3-8B методом QLoRA

Початкове значення втрат (~2.4) вважається характерним для етапу «холодного старту», коли низькорангові адаптери (LoRA adapters) ще не

синхронізовані з параметрами базової моделі. Стрімке зниження кривої (товста лінія) на проміжку від 0 до 100 кроків свідчить про ефективну адаптацію моделі до структури цільових даних (парсинг JSON). Присутність високочастотного шуму (тонка лінія) є очікуваним артефактом при використанні 4-бітного квантування (NF4), що вносить незначну стохастичність у процес обчислення градієнтів. Вихід згладженої кривої на плато зі значенням Loss ≈ 0,6 на фінальних етапах підтверджує, що модель досягла точки конвергенції без ознак перенавчання, що обґрунтовує достатність використаних обчислювальних ресурсів (Google Colab T4) для вирішення поставленої задачі.

Після донавчання моделі показник F1 збільшився з 0,80 (zero-shot) до 0,91 (fine-tuned) (таблиця 3.3) завдяки трьом факторам:

- засвоєння структури – модель більше не «гадає», які теги важливі. Вона «знає», що ціна лежить саме в , а не в рекламному блоці поруч, тому що бачила це 500 разів під час навчання;

- дотримання формату – Fine-tuned модель набагато рідше робить синтаксичні помилки в JSON (наприклад, не забуває закривати дужки), тому що вона натренована генерувати лише валідний код;

- зменшення промпту – більше не потрібно писати довгі інструкції («Не пиши вступ, пиши тільки JSON, використовуй такі ключі...»). Модель це вже «пам'ятає», що економить токени і пришвидшує відповідь.

На відміну від fine-tuning, метод few-shot learning (навчання на кількох прикладах) для моделі T5-small у контексті задачі парсингу (трансформації неструктурованого тексту в JSON), не змінює ваги моделі. Тут використовується здатність моделі до навчання на прикладах правильного виконання завдання, що подаються безпосередньо у вхідному запиті (промпті). Wей процес не вимагає тренування градієнтним спуском, і складається з кількох етапів.

Етап 1. Підбір демонстраційних прикладів. Оскільки модель T5-small має обмежене контекстне вікно (стандартно 512 токенів), то не має можливості надати їй багато прикладів. Тому потрібно відібрати 2–3 найбільш репрезентативні пари «Вхід-Вихід», що мають покривати різні варіанти вхідних даних. Наприклад, один

варіант, де ціна вказана як «$100», і один – як «100 USD».

Етап 2. Конструювання промпту. Архітектура T5 вимагає, щоб завдання на вході моделі було сформульоване як єдиний текстовий рядок. Для цього інструкція, демонстраційні навчальні приклади та цільове завдання з'єднуються в один блок, який подається на вхід моделі. Загальна структура вхідного рядка:

- Task Prefix – опис задачі (для T5 це важливо);
- Shot 1 (Example) – вхідний текст + роздільник + Ідеальний JSON;
- Shot 2 (Example) – вхідний текст + роздільник + Ідеальний JSON;
- Target Input – новий текст, який треба розпарсити;
- Trigger – маркер початку генерації.

Приклад вхідного рядка для T5:

```
extract data to json:

Input: "Smartphone Samsung S21, Black. Cost: 500 USD."
Output: {"item": "Samsung S21", "color": "Black", "price": 500}

Input: "Laptop Dell XPS 15, Silver color. Price: $1200."
Output: {"item": "Dell XPS 15", "color": "Silver", "price": 1200}

Input: "Headphones Sony WH-1000XM4, Color: Blue. Best price: 300$."
Output:
```

Етап 3. Обробка обмежень токенізації (важливо для T5-small). Перед подачею в модель текст розбивається на токени. Оскільки JSON-структура споживає багато токенів (дужки {}, ключі ""), то, якщо приклади у промпті занадто довгі, то вхід (Input: "Headphones...") може вийти за межі вікна в 512 токенів, і модель його «не побачить».

Частковим рішенням цієї проблєми є використання мінімізованого JSON (без пробілів) у прикладах для економії місця.

Етап 4. Інференс та генерація. Вхідний рядок проходить через Encoder моделі T5. Механізм уваги аналізує патерн: після слова Input: йде опис товару, а після Output: йде JSON, де значення відповідають даним з Input. Дійшовши до останнього Output:, Decoder починає генерувати текст, намагаючись продовжити цей патерн.

Модель генерує токени по одному: { -> "item" -> : -> "Sony... і так далі, доки не згенерує токен кінця рядка (</s>).

Для дослідження методу few-shot learning в умовах обмежених обчислювальних ресурсів було обрано модель flan-t5-small (80M параметрів). Реалізація виконувалася локально з використанням бібліотеки Hugging Face Transformers.

Процедура формування вхідного контексту передбачає конкатенацію інструкції, двох демонстраційних прикладів (2-shot) та цільового вхідного фрагмента HTML у єдину текстову послідовність. Оскільки архітектура T5 має обмежену довжину вхідної послідовності (512 токенів), демонстраційні приклади були попередньо мінімізовані шляхом видалення зайвих пробілів у HTML-тегах та JSON-структурі.

Інференс моделі виконувався на CPU, що підтверджує можливість розгортання подібних систем парсингу на пристроях обмеженої потужності без використання спеціалізованих прискорювачів. Локальна реалізація методу few-shot та практична реалізація процедури few-shot для локальної моделі T5-small представлені у додатку В.

У підсумковій таблиці 3.3 для T5-small + метод Few-shot Learning було отримано значення F1 = 0,86. Це означає, що використаний метод донавчання few-shot показує високий але дещо нижчий результат, ніж fine-tuning, що можна пояснити такими факторами:

- позитивний фактор – модель T5 під час свого попереднього навчання (pre-training на корпусі C4) бачила багато коду та JSON. Тому навіть 2-3 приклади (Few-shot) ефективно «нагадують» моделі, як будувати JSON-синтаксис. Це забезпечило достатньо високе значення F1 = 0,86;
- обмежувальні фактори: мала кількість прикладів через малий розмір вікна (512 токенів) надає можливість показати моделі лише прості випадки. Якщо у новому тексті трапиться складна, нестандартна верстка, якої не було в цих 2-3 прикладах, модель помилиться. Fine-tuned GPT-3.5 бачила 500+ прикладів, тому вона стійкіша; малий розмір моделі – T5-small має близько 60 млн параметрів, тому

її здатність до узагальнення значно нижча, ніж у GPT-3.5 (175 млрд) або LLaMA 2 (7B+).

Таким чином метод few-shot на T5-small забезпечує баланс між швидкістю розробки (не треба тренувати модель) та якістю отриманого результату. Це кращий варіант для простих задач парсингу, де вхідні дані є відносно однорідними.

За результатами проведеного експерименту можна відзначити, що модель T5 (Encoder-Decoder) краще підходить для задач типу «переклад» (HTML-to-JSON), ніж чисті декодери (як GPT-2), тому що T5 «бачить» увесь вхідний HTML одразу (через Encoder), а не читає його зліва направо по одному слову. Це забезпечує вищу точність розуміння структури тегів моделлю.

Процедура domain-tuning (доменна адаптація) для моделі LLaMA 2 у контексті задачі парсингу XML-даних відрізняється від методів fine-tuning (навчання з учителем) та few-shot (промпт-інжинірінг). Domain-tuning – це продовження попереднього навчання моделі на великому масиві «сирих» текстів з конкретної галузі (у нашому випадку – на масиві технічних XML-файлів), щоб модель глибоко «зрозуміла» специфічну мову розмітки. Згідно з domain-tuning модель не навчається виконувати конкретну команду, наприклад, «витягни ціну», як у випадку з GPT-3.5, а замість цього занурюється у специфічне середовище XML для вивчення його синтаксису, структури вкладеності та логіки тегів.

Етап 1. Збір та підготовка навчального корпусу. Для адаптації LLaMA 2 під XML-парсинг необхідно зібрати великий масив нерозмічених даних: основні джерела – фінансові звіти (XBRL), файли конфігурацій, файли sitemap, експорти баз даних. Ці дані не потрібно розбивати на пари «Вхід-Вихід». Це просто дуже великі текстові файли (.txt), що містять мільйони рядків XML-коду. Мета – модель має бачити всі можливі варіанти тегів: <price>, <cost>, <amount>, <currency>, а також складні вкладені структури.

Етап 2. Токенізація та налаштування контексту. LLaMA 2 використовує токенізатор SentencePiece (BPE). Для парсингу XML це критично важливий етап. Стандартна модель може сприймати тег <product_id> як набір розрізнених токенів: < + product + _ + id + >. Під час domain-tuning модель вчиться ефективно обробляти

такі послідовності: наприклад, вона починає розуміти, що відкриваючий тег <group> обов’язково вимагає закриваючого </group> через певну кількість кроків.

Етап 3. Навчання методом Causal Language Modeling (CLM) – основний етап, який технічно схожий на те, як модель вчилася «читати» під час свого створення. Завдання моделі – передбачення наступного токена.

Процес: подаємо моделі фрагмент XML, наприклад: <catalogue><book id="bk101"><author>. Модель намагається вгадати, що йде далі. Якщо вона відповідає "Шевченко", і в тексті йде "Шевченко", вона отримує «нагороду» (зниження функції втрат). Якщо вона каже (HTML-тег замість XML), це вважається помилкою, і ваги коригуються. В результаті цього етапу ваги моделі зміщуються у бік розуміння XML, вона починає «думати» структурами DOM.

Етап 4. Інференс. Після domain-tuning отримуємо модель, яка ідеально знає XML. Тепер, щоб виконати парсинг, потрібно подати їй промпт (можливо, з 1-2 прикладами, як у few-shot), наприклад: «Ти експерт з даних. Перетвори цей XML у JSON.» + [XML код]. Оскільки модель тепер «відчуває» структуру XML, вона значно краще ідентифікує межі полів і зв’язки між батьківськими та дочірніми елементами.

Локальна реалізація методу domain-tuning та практична реалізація процедури domain-tuning для локальної моделі LLaMA 2 представлені у додатку В. Для практичного виконання експерименту було обрано модель TinyLlama, яка є компактною версією LLaMA 2. Це дозволило провести повний цикл domain-tuning на обмежених апаратних ресурсах без використання квантування.

Динаміку зміни функції втрат (Training Loss) протягом процесу донавчання моделі представлено на рисунку 3.3.

Початкове значення втрат (~2,7) свідчить про високу ентропію моделі на нових даних. Спадаючий тренд кривої підтверджує успішну адаптацію ваг моделі до структури XML-документів. Стабілізація графіка в діапазоні від 0,2 до 0,3 на фінальних епохах вказує на досягнення точки конвергенції (збіжності) та мінімізацію помилки передбачення наступного токена, що підтверджує ефективність обраного методу навчання.

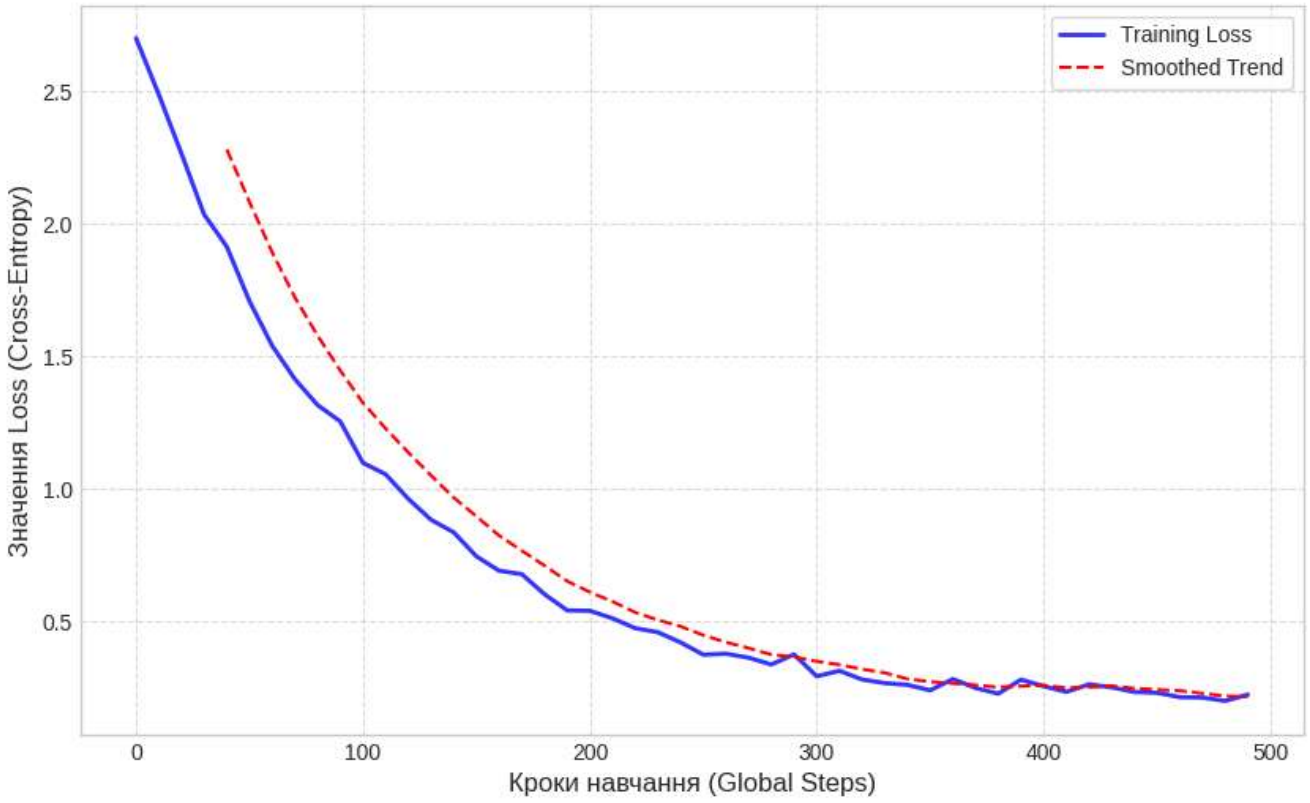

Рисунок 3.3 – Конвергенція функції втрат моделі LLaMA 2 (domain-tuning)

У підсумковій таблиці 3.3 варіант LLaMA 2 + Domain-tuning показав високий результат F1 = 0,88, випередивши Few-shot T5, але поступившись Fine-tuned GPT-3.5 за рахунок наступного: XML – суворий формат, де навіть одна пропущена дужка ламає все. Domain-tuning навчив LLaMA 2 структурній інтуїції, так що модель не плутає закриваючі теги, навіть якщо документ дуже довгий і вкладений, що й забезпечило високу точність (0,90); модель вивчила специфічні терміни домену (наприклад, банківські XML-теги), які загальна модель могла б не зрозуміти або проігнорувати. Це підвищило повноту (0,87). Однак, отриманий результат все одно нижчий за GPT-3.5 Fine-tuned (F1 = 0,91). Основні причини: відсутність специфічного навчання під конкретну задачу: domain-tuning навчив модель розуміти XML, але не обов’язково трансформувати його в JSON, тоді як GPT-3.5 Fine-tuned вчилася саме трансформації; LLaMA 2 витрачає ресурси на «здогадування», як саме користувач хоче оформити вихідний JSON, тоді як модель GPT-3.5 це вже «знає». Таким чином, використання domain-tuning для LLaMA 2 доцільне для створення потужної локальної моделі для роботи зі складними,

специфічними форматами даних, де загальні знання звичайних моделей є недостатніми.

За результатами експерименту можна зробити висновок, що метод domain-tuning не вчить модель виконувати команди (як fine-tuning), а знижує перплексію (perplexity) моделі на специфічних текстах. Тобто Domain-tuning адаптує модель до стилю та термінології тексту (щоб вона краще передбачала наступне слово), але не дає їй навичок слідувати інструкціям користувача.

3.4 Результати порівняльного експерименту

З метою об’єктивної оцінки ефективності парсингу даних з використанням LLM з урахуванням обраної стратегії навчання та типу вхідного формату даних, було проведено тестування додатково навчених моделей GPT-3.5, T5 та LLaMA 2 на реальних даних, що охоплюють HTML, JSON та XML структури (додаток А). Результати цього тестування, систематизовані за головними метриками якості – Точністю (Accuracy), Повнотою (Recall) та інтегральною оцінкою F1-score, – показують, як саме метод адаптації (донавчання) моделі (Zero-shot, Fine-tuning, Domain-tuned) впливає на здатність моделі ефективно вилучати інформацію. Підсумок отриманих результатів представлено у таблиці 3.3.

Таблиця 3.3 – Порівняння ефективності адаптованих моделей

Модель	Метод адаптації	Формат даних	Точність (Accuracy)	Повнота (Recall)	F1-score
GPT-3.5	Zero-shot	HTML	0,82	0,78	0,80
GPT-3.5 + fine-tuning	Fine-tuned	HTML	0,92	0,90	0,91
T5-small	Few-shot	JSON	0,88	0,85	0,86
LLaMA 2	Domain-tuned	XML	0,90	0,87	0,88

Модель GPT-3.5 (Zero-shot) на HTML показала в експерименті найнижчий результат F1=0,80, який можна пояснити тим, що HTML – це «брудний» формат із

великою кількістю службових тегів (, , CSS-класи). Без спеціальної підготовки (Zero-shot) модель плутається в структурі, може прийняти рекламний банер за опис товару (високий FP) або пропустити ціну, якщо вона прихована глибоко в тегах (високий FN).Таким чином, базові LLM без донавчання погано підходять для складного скрапінгу.

GPT-3.5 + Fine-tuning на HTML має найвищий результат F1=0,91. Цей результат пояснюється тим, що проведене донавчання (fine-tuning) адаптувало ваги моделі під верстку конкретного документу. Модель «вивчила», наприклад, що ціна завжди знаходиться всередині блоку . Також зросли показники повноти (0,90) тому, що модель перестала пропускати складні елементи, оскільки «бачила» їх під час тренування, і точності (0,92), оскільки, наприклад, модель перестала плутати ціну товару з ціною доставки, так як навчилася розрізняти їх контекст.

T5-small (Few-shot) на JSON має значення F1=0,86. T5 – це модель типу Encoder-Decoder, найбільше придатна для задач «переклад тексту в текст». Вхідний формат JSON є напівструктурованим (чистішим за HTML), тому навіть маленька модель з кількома прикладами (few-shot) добре справляється із завданнями парсингу. Однак, вона дещо поступається fine-tuned моделям, оскільки контекст few-shot дуже обмежений (модель отримує лише 2-3 приклади, а не 500, як у випадку fine-tuning).

LLaMA 2 (Domain-tuned) на XML має результат F1=0,88. Розмітка XML може мати дуже глибоку вкладеність і специфічні теги (наприклад, у банківських виписках). Domain-tuning (навчання на масиві текстів певної галузі) дозволив LLaMA зрозуміти семантику цих тегів. Отриманий результат кращий за T5, оскільки модель LLaMA 2 більш потужна, але трохи гірший за fine-tuned GPT, тому що XML може бути складнішим для інтерпретації, ніж візуальна структура HTML.

Таким чином, отримані дані показують, що спеціалізоване донавчання (fine-tuning) забезпечує високу якість LLM-парсингу (F1 = 0,91), дозволяючи моделі GPT-3.5 ефективно обробляти навіть складну верстку HTML. Водночас модель T5-small, незважаючи на меншу кількість параметрів та використання методу few-shot,

демонструє прийнятний рівень точності (F1 = 0,86) для структурованих форматів типу JSON, що робить її виправданою альтернативою для менш критичних задач.

Результати проведеного експерименту свідчать, що: коли потрібна найвища точність (0,92) для критичних бізнес-даних, то необхідно приділяти більше уваги розмітці даних та fine-tuning моделі (приклад GPT-3.5); для простіших форматів (JSON) можна використовувати менші, дешевші моделі (T5) з методом навчання few-shot. При цьому втрачається 4-5% точності; HTML є найскладнішим для «сирих» моделей без донавчання (zero-shot), тому для вебскрапінгу використання базових LLM без адаптації є ризикованим.

3.5 Рекомендації щодо вибору та застосування методів парсингу у різних випадках

Результати проведеного порівняльного аналізу архітектурних та функціональних можливостей методів парсингу дозволяють сформувати практичні рекомендації для оптимального вибору підходів у різних сценаріях з урахуванням критеріїв, представлених у таблиці 3.4.

Таблиця 3.4 – Головні критерії вибору оптимального підходу до парсингу

Критерій	Значення
Структура даних	Структуровані, напівструктуровані, неструктуровані
Обсяг даних	Малий, середній, великий
Точність	Висока, середня, низька
Швидкість обробки	Висока, середня, низька
Ресурси/вартість	Низька, середня, висока
Гнучкість/адаптивність	Низька, середня, висока

Практичні рекомендації для різних сценаріїв парсингу:

- для структурованих даних (XML, JSON, CSV) – використовувати традиційні парсери (Regex, XPath, CSV-пакети); використання LLM виправдане лише для обробки аномалій або неповних даних; оптимізація ресурсів: пакетна обробка великих файлів для підвищення швидкості;

- для HTML та вебданих традиційні методи працюють ефективно, якщо DOM-структура стабільна; для динамічних сайтів рекомендується поєднання: традиційний парсинг + LLM для адаптивного витягання контенту; використовувати headless-браузери (Selenium, Playwright) для підготовки даних до LLM;

- для напівструктурованих даних (лог-файли, API, RSS): LLM-парсери забезпечують гнучкість для змішаних форматів; рекомендується передпарсинг традиційними методами для фільтрації основних полів; контроль якості результатів через автоматизоване тестування та валідацію;

- для неструктурованих даних (текстові документи, відгуки, технічна документація): лише LLM або гібридні методи здатні видобувати семантичну інформацію; використовувати fine-tuning моделей або prompt-engineering для специфічних завдань; валідація результатів через human-in-the-loop підхід або автоматичні правила.

На рисунку 3.4 представлена діаграма алгоритму прийняття рішень щодо вибору методу парсингу.

Рисунок 3.4 – Алгоритм вибору методу парсингу

Таким чином, для структурованих даних при високій швидкості та низькій вартості рекомендується використовувати переважно традиційні методи парсингу; для напівструктурованих та неструктурованих даних з високими вимогами до точності доцільно застосовувати LLM-парсери; для сценаріїв з великими потоками змішаних даних, коли потрібен баланс між швидкістю, точністю та ресурсами оптимальним вибором є гібридні системи парсингу.

3.6 Техніко-економічне обґрунтування ефективності застосування методів парсингу

Техніко-економічне обґрунтування дозволяє кількісно оцінити ефективність застосування різних методів парсингу з урахуванням часу обробки даних, використання обчислювальних ресурсів, фінансових витрат та впливу на продуктивність бізнес-процесів, шляхом визначення оптимального співвідношення витрат і ефективності для кожного підходу та обгрунтування рекомендацій для практичної реалізації.

Сумарна вартість обробки запиту обчислюється за формулою:

$$C_{total} = C_{compute} + C_{license}, \tag{3.5}$$

де: $C_{compute} = T \cdot R$ – грошові витрати на виконання обчислень; T – час обробки запиту (год.); R – вартість 1 год. обчислень; $C_{license}$ – вартість ліцензії/моделі.

Продуктивність (швидкість обробки) обчислюється за формулою:

$$P = \frac{N}{T}, \tag{3.6}$$

де: N – кількість оброблених записів, T – час обробки (год.).

Ефективність витрат (Cost Efficiency, CE) обчислюється за формулою:

$$CE = \frac{P}{C_{total}}. \quad (3.7)$$

Вища величина *CE* означає більшу продуктивність на одиницю витрат.

Для виконання розрахунків використані умовні дані про обробку 1 млн. записів у структурованих, напівструктурованих і неструктурованих форматах, представлені у таблиці 3.5.

Таблиця 3.5 – Вихідні дані для виконання розрахунків

Метод	Час обробки 1 млн записів (год.)	Вартість 1 год. обчислень ($)	Вартість ліцензії/моделі ($)	Основні джерела витрат
Традиційний	2	0,5	0	Сервери CPU, open-source бібліотеки
LLM (GPT-4)	6	3	500	Cloud GPU, API-ліцензія
Гібридний	3	1,5	500	CPU для фільтра + GPU для LLM

Алгоритм техніко-економічного обґрунтування методів парсингу представлений на діаграмі (рисунок 3.5).

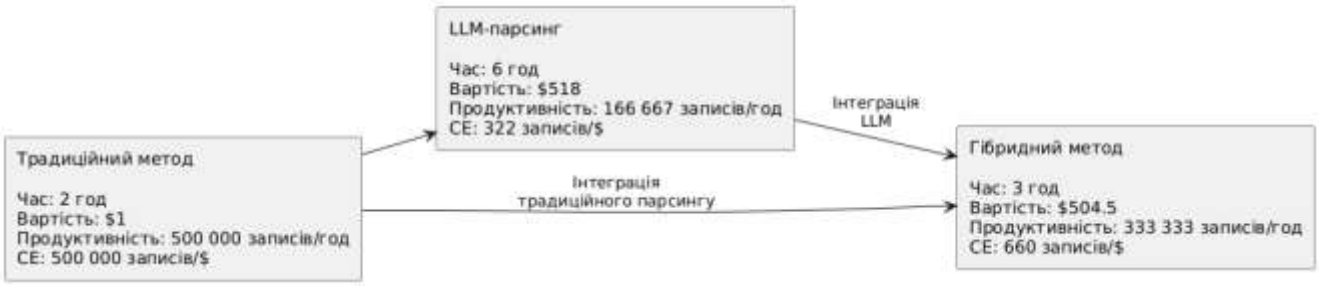

Рисунок 3.5 – Алгоритм ТЕО методів парсингу

Використаємо вихідні дані для виконання розрахунків показників ТЕО для різних методів парсингу. Розрахунки виконуються за формулами (3.5)-(3.7).

Розрахунок для традиційного парсингу:

$C_{compute} = 2 \cdot 0{,}5 = 1$ \$;

$C_{total} = 1 + 0 = 1$ \$;

$P = 1000000 / 2 = 500000$ записів/год;

$CE = 500000 / 1 = 500000$ записів/$.

Розрахунок для LLM-парсингу (модель GPT-4):

$C_{compute} = 6 \cdot 3 = 18$ $;

$C_{total} = 18 + 500 = 518$ $;

$P = 1000000 / 6 \approx 166667$ записів/год;

$CE = 166{,}667 / 518 \approx 322$ записів/$.

Розрахунок для гібридного парсингу:

$C_{compute} = 3 \cdot 1{,}5 = 4{,}5$ $;

$C_{total} = 4{,}5 + 500 = 504{,}5$ $;

$P = 1000000 / 3 \approx 333333$ записів/год;

$CE = 333{,}333 / 504{,}5 \approx 660$ записів/$.

Порівняння техніко-економічних показників різних методів парсингу, отриманих в результаті проведених розрахунків, представлене у таблиці 3.6.

Таблиця 3.6 – Порівняння техніко-економічних показників методів парсингу

Метод парсингу	Час виконання, год.	Вартість, $	Продуктивність, записів/год.	Ефективність витрат, записів/$
Традиційний	2	1	500000	500000
LLM	6	518	166667	322
Гібридний	3	504,5	333333	660

Отримані результати розрахунків дозволяють зробити такі висновки:

- традиційні методи парсингу показують максимальну ефективність витрат для структурованих даних, але мають низьку гнучкість для нестандартних форматів;

- метод парсингу на основі LLM характеризує найвища точність для складних даних, але дуже низька ефективність витрат при великих обсягах через високу вартість ліцензії та GPU;

- гібридні методи парсингу забезпечують оптимальний баланс між продуктивністю, точністю та витратами. Отже, вони є найбільш перспективними для масштабних ETL-процесів та DataOps-систем.

Рекомендації щодо застосування методів парсингу:

- для структурованих даних – використовувати традиційні методи, оскільки вони забезпечують максимальну економічність і швидкість;
- для напівструктурованих та змішаних даних – застосовувати гібридний підхід, що дозволяє оптимізувати ресурси та підвищити точність;
- для неструктурованих текстів і аналітики – застосовувати LLM з контролем результатів; рекомендується інтегрувати fine-tuning для специфічних доменів;
- обчислювальні ресурси LLM краще виділяти тільки на складні випадки, а рутинні дані обробляти традиційними методами.

Висновки до розділу 3

Архітектура традиційних парсерів залишається актуальною для стабільних і структурованих форматів джерел (лог-файли, HTML із фіксованими тегами), що вимагають високої швидкості та низьких витрат.

LLM-підходи мають вищу адаптивність і придатні для парсингу гібридних або напівструктурованих даних (звітів, відгуків, технічних документів). LLM забезпечують високу адаптивність і точність при роботі з напівструктурованими та неструктурованими даними. Основною проблємою LLM-застосувань у парсингу даних є вартість інференсу моделей, яка може бути знижена завдяки дистиляції моделей і оптимізації токенізації.

Еволюція архітектур парсингу даних свідчить про поступове злиття двох підходів – створення гібридних систем парсингу, де класичний парсер виконує первинну фільтрацію, а LLM – семантичну частину. Архітектурне поєднання LLM та традиційних систем парсингу формує основу для інтелектуального парсингу нового покоління, що поєднує точність, масштабованість та контекстну чутливість.

Гібридні підходи забезпечують оптимальний баланс між точністю, швидкістю та вартістю, поєднуючи переваги обох методів. Використання

гібридних схем є найбільш перспективним для сучасних ETL-процесів та DataOps-платформ.

Аналіз показав, що обидва класи методів – традиційні та LLM – мають власні сфери ефективності:

- традиційні методи незамінні для задач із високими вимогами до швидкодії та стабільності форматів;
- LLM-підходи – для гнучкої семантичної інтерпретації, автоматичного вилучення знань і роботи з динамічними текстовими структурами;
- гібридна стратегія забезпечує найкращий компроміс, поєднуючи переваги обох напрямів у рамках ETL-процесів нового покоління.

ВИСНОВКИ

За результатами виконаної кваліфікаційної роботи можна зробити висновки, що охоплюють основні результати та підтверджують досягнення поставленої мети.

1. Визначено, що семантичний парсинг за допомогою LLM базується на архітектурі Transformer і забезпечує глибоке двонаправлене розуміння контексту даних, що є необхідним для вилучення сутностей. Систематизовано основні методи адаптації моделей до завдань парсингу. Обґрунтовано необхідність застосування комплексних критеріїв оцінки ефективності, які включають не лише традиційні метрики машинного навчання (F1-score, Recall), а й показники синтаксичної та семантичної узгодженості вихідних структур, що є обов'язковим для інтеграції LLM-парсерів в автоматизовані інформаційні системи.

2. Проведено порівняльний аналіз традиційних детермінованих методів парсингу та LLM-архітектур. Встановлено, що традиційні методи ефективні та економічно доцільні для стабільних, структурованих джерел, однак їх головною вадою є архітектурна крихкість при змінах верстки або структури файлу. Натомість, LLM-підходи забезпечують високу адаптивність та відмовостійкість, перетворюючи парсинг на задачу семантичного аналізу. На основі цього визначено, що гібридні архітектури (інтеграція традиційних методів для попередньої обробки та LLM для семантичного вилучення) є оптимальною стратегією, що дозволяє збалансувати швидкість, точність та витрати.

3. Розроблено та реалізовано методику порівняльного експерименту з оцінки ефективності різних LLM (GPT-3.5, T5, LLaMA 2) у сценаріях парсингу HTML, JSON та XML. Експериментально підтверджено, що Supervised Fine-tuning є найкращим методом для досягнення максимальної якості вилучення даних (F1-score до 0,91). Техніко-економічне обґрунтування підтвердило доцільність запропонованих рішень: LLM-парсери забезпечують максимальну точність, традиційний парсинг – максимальну економічну ефективність, а гібридні системи парсингу мають оптимальний баланс показників при великих потоках даних.

Сформовано деталізовані рекомендації щодо вибору методу парсингу залежно від структури вхідних даних (структуровані, напівструктуровані, неструктуровані).

Практична значущість роботи полягає у можливості впровадження розроблених рекомендацій та гібридних архітектур для оптимізації процесів збору, очищення та обробки даних у компаніях, що дозволить підвищити надійність ETL-процесів, зменшити помилки парсингу та оптимізувати витрати на обчислювальні ресурси.

Перспективи подальших досліджень полягають у розширенні можливостей автоматизації LLM-парсингу за допомогою агентних систем (Agentic LLMs) та неперервного навчання (continual learning) для безперервної адаптації моделей до нових форматів даних без необхідності повного повторного донавчання, а також у детальному дослідженні методів Explainable AI (XAI) для підвищення прозорості та довіри до результатів семантичного парсингу.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Brown T. B., Mann B., Ryder N., Subbiah M., Kaplan J., Dhariwal P., Neelakantan A., Shyam P., Sastry G., Askell A., Agarwal S., Herbert-Voss A., Krueger G., Henighan T., Child R., Ramesh A., Ziegler D. M., Wu J., Winter C., Hesse C., Chen M., Sigler E., Litwin M., Gray S., Chess B., Clark J., Berner C., McCandlish S., Radford A., Sutskever I., Amodei D. Language Models are Few-Shot Learners. *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. Vol. 33. P. 1877–1901. DOI: 10.48550/arXiv.2005.14165.

2. Vaswani A., Shazeer N., Parmar N., Uszkoreit J., Jones L., Gomez A. N., Kaiser Ł., Polosukhin I. Attention is All You Need. *Advances in Neural Information Processing Systems (NeurIPS)*, 2017. Vol. 30. P. 5998–6008. DOI: 10.48550/arXiv.1706.03762.

3. Devlin J., Chang M.-W., Lee K., Toutanova K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, 2019. P. 4171–4186. DOI: 10.48550/arXiv.1810.04805.

4. Bommasani R., Hudson D. A., Adeli E., et al. On the Opportunities and Risks of Foundation Models. *Stanford Center for Research on Foundation Models* (CRFM), 2021. 227 p. DOI: 10.48550/arXiv.2108.07258.

5. Wei J., Wang X., Schuurmans D., et al. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. *Advances in Neural Information Processing Systems (NeurIPS)*, 2022. Vol. 35. DOI: 10.48550/arXiv.2201.11903.

6. Raffel C., Shazeer N., Roberts A., et al. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer (T5). *Journal of Machine Learning Research*, 2020. Vol. 21(140). P. 1–67. DOI: 10.48550/arXiv.1910.10683.

7. OpenAI. Using GPT Models for Data Structuring and Parsing. OpenAI Developer Documentation. 2024. URL: https://platform.openai.com/docs/guides/structured-output (дата звернення: 09.09.2025).

8. Dodge J., Sap M., Marasović A., Agnew W., Ilharco G., Groeneveld D., Mitchell M., Gardner M. Documenting Large Webtext Corpora: A Case Study on the Colossal Clean Crawled Corpus. arXiv:2104.08758v2. DOI: https://doi.org/10.48550/arXiv.2104.08758.

9. C4. *Tensorflow*: вебсайт. URL: https://www.tensorflow.org/datasets/catalog/c4 (дата звернення: 18.10.2025).

10. Astekin M., Hort M., Moonen L. A Comparative Study on Large Language Models for Log Parsing. *Proceedings of the 18th ACM / IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM '24), October 24–25, 2024, Barcelona, Spain*. New York, NY, USA: ACM, 2024. P. 1–11. DOI: 10.1145/3674805.3686684.

11. Baysan M. S., Uysal S., İşlek I., Karaman Ç. Ç., Güngör T. LLM-as-a-Judge: automated evaluation of search query parsing using large language models. *Frontiers in Big Data*. Vol. 8, 2025. DOI: 10.3389/fdata.2025.1611389.

12. Забродський В. Розробка підходу до оцінювання якості відповідей LLM на основі бази знань : кваліфікаційна робота. Київ: eKMAIR, 2025. URL: https://ekmair.ukma.edu.ua/handle/123456789/36666 (дата звернення: 18.10.2025).

13. Zhao H., He B., Liu Y. GPT-Fin: Domain-Specific Large Language Models for Financial Text Understanding. *arXiv preprint*, 2023. arXiv:2309.13454. DOI: 10.48550/arXiv.2309.13454.

14. Zhao W. X., Liu J., Yu Z., Liu T., Tang J., Wen J.-R. A Survey of Large Language Models. *arXiv preprint*, 2023. arXiv:2303.18223. DOI: https://doi.org/10.48550/arXiv.2303.18223.

15. Zhao W., Schütze H., Li F. A Survey on Large Language Models for Data Management. *Proceedings of the VLDB Endowment*, 2023. Vol. 16(12). P. 3687–3700. DOI: 10.14778/3611479.3611499.

16. Common Crawl – free, open repository of web crawl data. *Commoncrawl*: вебсайт. URL: https://commoncrawl.org/ (дата звернення: 30.10.2025).

17. The Pile. *Eleuther*: вебсайт. URL: https://pile.eleuther.ai/ (дата звернення: 30.10.2025).

18. Левченко Ю. Техніки оптимізації LLM-парсингу. *Матеріали науково-практичної конференції за підсумками проходження виробничих практик здобувачів вищої освіти спеціальності 126 Інформаційні системи та технології*, кафедра інформаційних систем та технологій Полтавського державного аграрного університету, 22 жовтня 2025 р. Вип. XI. Полтава: ПДАУ, 79 с. С. 66-68.

19. Laurent A. Reinforcement Learning from Human Feedback (RLHF) Explained – Concepts, Algorithms, and Research Landscape. *IntuitionLabs.ai*, 2025. URL: https://intuitionlabs.ai/pdfs/reinforcement-learning-from-human-feedback-rlhf-explained.pdf (дата звернення: 30.09.2025).

20. Kili Technology. Exploring Reinforcement Learning from Human Feedback (RLHF): A Comprehensive Guide. *Kili Technology*, 2024. URL: https://kili-technology.com/blog/exploring-reinforcement-learning-from-human-feedback-rlhf-a-comprehensive-guide (дата звернення: 30.09.2025).

21. Wei J., Wang X., Schuurmans D., Bosma M., Ichter B., Xia F., Chi E. H., Le Q., Zhou D. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. *arXiv preprint*. arXiv:2201.11903 [cs.CL], 2022. DOI: https://doi.org/10.48550/arXiv.2201.11903.

22. Kerr S. Beginner’s Guide to Chain of Thought Prompting. Zero To Mastery, July 23rd, 2025. Zerotomastery: вебсайт. URL: https://zerotomastery.io/blog/chain-of-thought-prompting/ (дата звернення: 30.09.2025).

23. Wu T., Luo L., Li Y.F., Pan S., Vu T.T., Haffari,G. Continual Learning for Large Language Models: A Survey. *arXiv preprint.* arXiv:2402.01364, 2024. URL: https://arxiv.org/html/2402.01364v1 (дата звернення: 30.09.2025).

24. Wang M., Stoll A., Lange L., Adel H., Schütze H., Strötgen J. Bring Your Own Knowledge: A Survey of Methods for LLM Knowledge Expansion. ACL Anthology, 2024. *Aclanthology*: вебсайт. URL: https://aclanthology.org/2025.l2m2-1.12.pdf (дата звернення: 01.10.2025).

25. Song, Shezheng; Xu, Hao; Ma, Jun; Li, Shasha; Peng, Long; Wan, Qian; Liu, Xiaodong; Yu, Jie. How to Alleviate Catastrophic Forgetting in LLMs Finetuning?

Hierarchical Layer-Wise and Element-Wise Regularization. arXiv preprint. *Arxiv*: вебсайт. URL: https://arxiv.org/html/2501.13669v2 (дата звернення: 01.10.2025).

26. Song Y., Du Y., Paperno D., Gatt A. Burn After Reading: Do Multimodal Large Language Models Truly Capture Order of Events in Image Sequences? Utrecht University, Utrecht, the Netherlands, 2025. *Aclanthology*: вебсайт. URL: https://aclanthology.org/2025.findings-acl.1248.pdf (дата звернення: 02.10.2025).

27. Jin Y., Li J., Liu Y., Gu T., Wu K., Jiang Z., He M., Zhao B., Tan X., Gan Z., Wang Y., Wang C., Ma L. Efficient Multimodal Large Language Models: A Survey. Youtu Lab, Tencent, SJTU, BAAI, ECNU. *Arxiv*: вебсайт. URL: https://arxiv.org/html/2405.10739v1 (дата звернення: 02.10.2025).

28. Cao Z., Feinstein Z. Large Language Model in Financial Regulatory Interpretation. arXiv, 2024. *Arxiv*: вебсайт. URL: https://arxiv.org/pdf/2405.06808 (дата звернення: 02.10.2025).

29. Tavasoli A., Sharbaf M., Madani S. M. Responsible Innovation: A Strategic Framework for Financial LLM Integration. arXiv. 2024. *Arxiv*: вебсайт. URL: https://arxiv.org/pdf/2504.02165 (дата звернення: 02.10.2025).

30. Huang A. H., Wang H., Yang Y. FinBERT: A Large Language Model for Extracting Information from Financial Text. Contemporary Accounting Research, 2023. Vol. 40, Issue 2. P. 806-841. DOI: https://doi.org/10.1111/1911-3846.12832.

31. Liu Z., Huang D., Kaiyu H., Li Z., Zhao J. FinBERT: A Pre-trained Financial Language Representation Model for Financial Text Mining. Twenty-Ninth International Joint Conference on Artificial Intelligence and Seventeenth Pacific Rim International Conference on Artificial Intelligence (IJCAI-PRICAI-20), July 2020. URL: https://www.researchgate.net/publication/342793634_FinBERT_A_Pre-trained_Financial_Language_Representation_Model_for_Financial_Text_Mining (дата звернення: 04.10.2025).

32. Vrdoljak J., Boban Z., Vilović M., Kumrić M., Božić J. A Review of Large Language Models in Medical Education, Clinical Decision Support, and Healthcare Administration. Healthcare. 2025. Vol. 13, no. 6. P. 603. DOI:

10.3390/healthcare13060603. URL: https://pmc.ncbi.nlm.nih.gov/articles/PMC11942098/ (дата звернення: 04.10.2025).

33. Wang Z., Wang Z., Jiang J., Chen P., Shi X., Li Y. Large Language Models in Bioinformatics: A Survey. 2024. P. 3602–3615. (ACL Anthology). *Aclanthology*: вебсайт. URL: https://aclanthology.org/2025.findings-acl.184.pdf (дата звернення: 04.10.2025).

34. BioGPT 5.1. *Chatgpt*: вебсайт. URL: https://chatgpt.com/g/g-1188YzidI-biogpt (дата звернення: 11.12.2025).

35. BioGPT. *Huggingface*: вебсайт. URL: https://huggingface.co/docs/transformers/model_doc/biogpt (дата звернення: 11.12.2025).

36. BioGPT. BioGPT is a domain-specific generative transformer language model pre-trained on large-scale biomedical literature. *GoogleCloud*: вебсайт. URL: https://console.cloud.google.com/vertex-ai/publishers/microsoft/model-garden/bio-gpt?pli=1 (дата звернення: 11.12.2025).

37. Med-PaLM. GoogleResearch: вебсайт. URL: https://sites.research.google/med-palm/ (дата звернення: 11.12.2025).

38. Tsang S.-H. PubMedBERT: Domain-Specific Language Model Pretraining for Biomedical Natural Language Processing. *Medium*: вебсайт. URL: https://sh-tsang.medium.com/brief-review-pubmedbert-domain-specific-language-model-pretraining-for-biomedical-natural-c7b5d51b5b51 (дата звернення: 11.12.2025).

39. Деркач В. Г., Прокопович-Ткаченко Є. Д., Руденко Є. Г. Використання штучного інтелекту в судовому процесі України: правові, етичні та процесуальні аспекти. *Юридичний науковий електронний журнал*, 2025. № 3. С. 460–464. DOI: https://doi.org/10.32782/2524-0374/2025-3/109. URL: http://lsej.org.ua/3_2025/111.pdf (дата звернення: 04.12.2025).

40. Кобрін А., Шаіпов А., Разогрєєв Є., Самоходський І., Коваленко І., Стройко І., Зеров К., Сапельніков Л., Літвінова О., Дубно О., Андрієнко О., Козуб О., Білик П., Авдєєва Т., Берназюк Я. Рекомендації з відповідального використання штучного інтелекту для правників. Державна судова адміністрація

України, 2025. *Thedigital*: вебсайт. URL: https://storage.thedigital.gov.ua/files/2/72/fcf6c498e0a9f57e7a8521ff9833372d.pdf (дата звернення: 04.12.2025).

41. InLegalBERT. *Huggingface*: вебсайт. URL: https://huggingface.co/law-ai/InLegalBERT (дата звернення: 07.12.2025).

42. Drewgelbard. Fine-Tuning Legal-BERT: LLMs For Automated Legal Text Classification. *Towardsai*: вебсайт. URL: https://towardsai.net/p/artificial-intelligence/fine-tuning-legal-bert-llms-for-automated-legal-text-classification (дата звернення: 07.12.2025).

43. Zhong R., Guo M., Baldridge J., et al. LegalBench: Evaluating Legal Reasoning of Large Language Models. arXiv preprint, 2022. arXiv:2308.11462. DOI: 10.48550/arXiv.2308.11462.

44. Rafalski, K. 17 Proven LLM Use Cases in E-commerce That Boost Sales in 2025. Netguru. Updated Nov 22, 2025. *Netguru*: вебсайт. URL: https://www.netguru.com/blog/llm-use-cases-in-e-commerce (дата звернення: 04.11.2025).

45. Attri A., Attri A., Bhattacharyya P. Mind, Matter, and Markets: A Survey of Human-Centered LLM Applications in E-commerce. Computer Science and Engineering, IIT Bombay, India. URL: https://www.cfilt.iitb.ac.in/resources/surveys/2025/june_cfilt_2025_arnav_anuj_survey_paper.pdf (дата звернення: 05.11.2025).

46. How do I get LangChain to access GPT-4.0, and how do I type in long prompts, and get the same accuracy and detail as GPT4.0 website? *Reddit*: вебсайт. URL: https://www.reddit.com/r/ChatGPTPro/comments/13ijx4q/how_do_i_get_langchain_to_access_gpt40_and_how_do/ (дата звернення: 07.12.2025).

47. Iscovici L. Chat with your database using LangChain and GPT-4. *Medium*: вебсайт. URL: https://medium.com/capgemini-invent-lab/chat-with-your-database-using-langchain-and-gpt-4-1abb3b311bc1 (дата звернення: 07.12.2025).

48. Gao L., Fisch A., Chen D. Making Pre-trained Language Models Better Few-Shot Learners. *Proceedings of ACL*, 2021. P. 3816–3830. DOI: 10.18653/v1/2021.acl-long.298.

49. Bohrium AI. Bohrium: вебсайт. URL: https://www.bohrium.com/ (дата звернення: 07.12.2025).

50. Kurniawan B. Regular Expressions Cookbook for Data Extraction. New York: Apress, 2021. 262 p.

51. Chen L., Zhao W. Evaluating Traditional and AI-based Data Parsing Pipelines. Information Systems Frontiers, 2023. DOI: 10.1007/s10796-023-10457-8.

52. Friedl J. E. F. Mastering Regular Expressions. 3rd ed. Sebastopol: O'Reilly Media, 2018. 544 p.

53. W3C. XML Path Language (XPath) 3.1 Specification. World Wide Web Consortium, 2021. W3.org: вебсайт. URL: https://www.w3.org/TR/xpath-31/ (дата звернення: 18.10.2025).

54. Левченко Ю. Архітектурні особливості великих мовних моделей у контексті парсингу даних. *Студентські роботи за науковою тематикою кафедри інформаційних систем та технологій*: матеріали XXII щорічного міждисциплінарного семінару, 25 листопада 2025 р. Полтава: ПДАУ, 2025 р. 120 с. С. 62-64.

55. Zhao H., Kim S. Advances in Web Scraping: Techniques and Applications. *Journal of Web Engineering*, 2022, Vol. 21(5), pp. 415–430.

56. Touvron H., Lavril T., Izacard G., et al. LLaMA: Open and Efficient Foundation Language Models. arXiv preprint, 2023. arXiv:2302.13971. DOI: 10.48550/arXiv.2302.13971.

57. Liu Y., Sun C. Evaluating Large Language Models for Structured Data Extraction. In Proceedings of ACL 2021, 2021.

58. Liu P., Zhang Y. Architectural Patterns in Data Parsing and Extraction. IEEE Access, 2022.

59. Флегантов Л., Левченко Ю. Принципи та архітектури LLM для парсингу даних. *The Future of Science, Technology and Economy: Collection of Scientific Papers with Proceedings of the 3rd International Scientific and Practical Conference*. International Scientific Unity. October 29-31, 2025. Sofia, Bulgaria. 164-169 p. *Isu-*

conference: вебсайт. URL: https://isu-conference.com/en/archive/the-future-of-science-technology-and-economy-29-10-25/ (дата звернення: 30.10.2025).

60. Флегантов Л., Масич А., Левченко Ю. Архітектурні та функціональні особливості провідних моделей Reasoning-LLM. *Progressive Approaches in Science and Engineering: Collection of Scientific Papers with Proceedings of the 2nd International Scientific and Practical Conference. International Scientific Unity*. November 26-28, 2025. Copenhagen, Denmark. 338-344 p. URL: https://isu-conference.com/arkhiv/progressive-approaches-in-science-and-engineering-26-11-25/ (дата звернення: 27.11.2025).

61. Ouyang L., Wu J., Jiang X., et al. Training language models to follow instructions with human feedback (InstructGPT / RLHF). arXiv preprint, 2022. arXiv:2203.02155. DOI: 10.48550/arXiv.2203.02155.

62. Chung H. W., Hou L., Longpre S., Zoph B., Tay Y., Fedus W., et al. Scaling Instruction-Finetuned Language Models. arXiv preprint, 2022. arXiv:2210.11416. DOI: https://doi.org/10.48550/arXiv.2210.11416.

63. Dettmers T., Pagnoni A., Holtzman A., Zettlemoyer L. QLORA: Efficient Finetuning of Quantized LLMs. *In Advances in Neural Information Processing Systems (NeurIPS)*, 2023. URL: https://arxiv.org/abs/2305.14314 (дата звернення: 22.10.2025).

64. Avinash M. S. R. Profiling LoRA/QLoRA Fine-Tuning Efficiency on Consumer GPUs: An RTX 4060 Case Study. MSR Avinash, 2024. URL: https://arxiv.org/pdf/2509.12229 (дата звернення: 22.10.2025).

65. Amazon Product Dataset. *Kaggle*: вебсайт. URL: https://www.kaggle.com/datasets/piyushjain16/amazon-product-data (дата звернення: 03.12.2025).

66. Yelp Dataset. Kaggle: вебсайт. URL: https://www.kaggle.com/datasets/yelp-dataset/yelp-dataset (дата звернення: 03.12.2025).

67. Yelp Open Dataset. Yelp: вебсайт. URL: https://business.yelp.com/data/resources/open-dataset/ (дата звернення: 03.12.2025).

68. Insider Threat Test Dataset. CMU KiltHub. DOI: https://doi.org/10.1184/R1/12841247. URL: https://kilthub.cmu.edu/articles/dataset/Insider_Threat_Test_Dataset/12841247 (дата звернення: 03.12.2025).

69. CERT Insider threat. Kaggle: вебсайт. URL: https://www.kaggle.com/datasets/nitishabharathi/cert-insider-threat (дата звернення: 03.12.2025).

70. Dominika T., Szostek P., Bolikowski Ł. GROTOAP2. Interdisciplinary Centre for Mathematical and Computational Modelling, University of Warsaw, 2014. DOI: https://doi.org/10.18150/8527338.