УДК 004.94

Р. Довгополюк, Ю. Олійник, М. Кувічка

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ ГЕНЕРАЦІЇ ТЕКСТІВ УКРАЇНСЬКОЮ МОВОЮ

Анотація. Зростання ролі систем обробки природної мови (NLP) створює високий попит на програмне забезпечення, здатне генерувати якісні тексти українською мовою. Однак складна морфологічна структура, багата словозміна, синтаксична гнучкість та брак достатніх мовних ресурсів роблять цю задачу вкрай нетривіальною. Наявні великі мовні моделі (LLM), як-от GPT або LLaMA, хоча і демонструють чудові результати в англомовному середовищі, часто не забезпечують високої якості при роботі з українською мовою. Ця проблема актуальна в таких сферах, як створення навчальних, офіційних і наукових текстів. Запропоноване рішення орієнтоване на усунення цих обмежень шляхом інтеграції LLM із морфологічним аналізом та інструментами виправлення граматичних помилок, зокрема LanguageTool, що дозволяє суттєво підвищити стилістичну й граматичну коректність згенерованих текстів. Робота присвячена розробці програмного забезпечення для генерації українськомовних текстів на основі інтеграції великих мовних моделей, морфологічного аналізу та тематичного моделювання, в результаті чого реалізовано масштабовану архітектуру. Проведено дослідження ефективності запропонованого підходу та здійснено fine-tuning моделей GPT-3.5 та LLaMA-3 для української мови. Програмне забезпечення реалізоване мовами Python та Java з використанням бібліотек spaCy, Gensim, LanguageTool, Airflow та OpenAI API.

Ключові слова: генерація тексту, Big Data, українська мова, NLP, великі мовні моделі, морфологічний аналіз, LDA, синтаксичний аналіз, тематичне моделювання, перевірка граматики, spaCy, LanguageTool.

Вступ

У сучасних дослідженнях значну увагу приділено архітектурам генерації тексту на основі трансформерів [1–3]. Відомі моделі GPT (Generative Pre-trained Transformer) від OpenAI, LLaMA від Meta та інші, що базуються на архітектурі Transformer, успішно застосовуються для генерації англомовних текстів. Проте українська мова через складну морфологію, нестачу якісних корпусів та обмежену представленість у фазі pre-training залишалася осторонь активного розвитку.

Метою дослідження є підвищення якості згенерованих текстів української мови при заданій тематиці та наборі ключових слів.

Матеріали та методи

Для генерації українськомовних текстів застосовуються два основні підходи: статистичні мовні моделі та великі трансформерні моделі (LLM). Перші вимагають ретельної побудови корпусів і ручного визначення граматичних залежностей, однак забезпечують контроль над стилем та структурою. Другий підхід – нейронний – базується на попередньому навчанні моделей на великих об'ємах текстових даних. Для досягнення якісної генерації українською мовою необхідна додаткова адаптація (fine-tuning) таких моделей, оскільки українська мова часто є недостатньо представлена в базовому pre-training. Окремої уваги потребує питання корекції граматики, яке вирішується за допомогою алгоритмічного інструмента LanguageTool, доповненого правилами на основі морфологічних і синтаксичних залежностей. У цій роботі реалізовано гібридний підхід – поєднання тематичного моделювання LDA для релевантної генерації контенту, використання LLM для побудови тексту, та алгоритмічної перевірки LanguageTool з метою забезпечення граматичної правильності. Така комбінація дозволяє досягти як семантичної, так і формальної якості результату. У даній статті особливу увагу приділено архітектурі системи, що дозволяє автоматизувати весь процес генерації текстів — від ключових слів до редагованого тексту, готового до публікації. Завданнями даного дослідження є:

- створення методу генерації текстів українською мовою;
- створення прототипу програмного забезпечення для реалізації методу генерації текстів українською мовою;
- дослідження ефективності запропонованого методу та програмного забезпечення.

Метод генерації текстів українською мовою

Серед інструментів морфологічного аналізу найбільш ефективним виявлено SpaCy, який має підтримку української мови. Інструмент LanguageTool відзначено як єдиний open-source засіб [15], здатний виявляти граматичні помилки українською мовою. Попри це, актуальним залишається завдання інтеграції всіх цих засобів в єдину систему для генерації текстів з покращеними морфологічними характеристиками.

Недостатньо дослідженим аспектом є саме поєднання LLM з морфологічним аналізом та LDA-моделюванням, а також створення нового типу правил для граматичної перевірки, що враховують залежності між словами. Це й стало основою даного дослідження.

Метод генерації текстів українською мовою, запропонований у дослідженні, реалізує інтеграцію трьох ключових технологічних компонентів:

1. Великої мовної моделі (LLM) — для генерації тексту.
2. Моделі тематичного моделювання LDA [9] — для розширення ключових слів і оцінки тематичної релевантності.

 ISSN 1560-8956

3. LanguageTool [8] із вдосконаленими правилами — для виявлення та виправлення граматичних помилок.

Цей підхід дозволяє забезпечити високу якість, змістовність та граматичну правильність згенерованих текстів українською мовою [4].

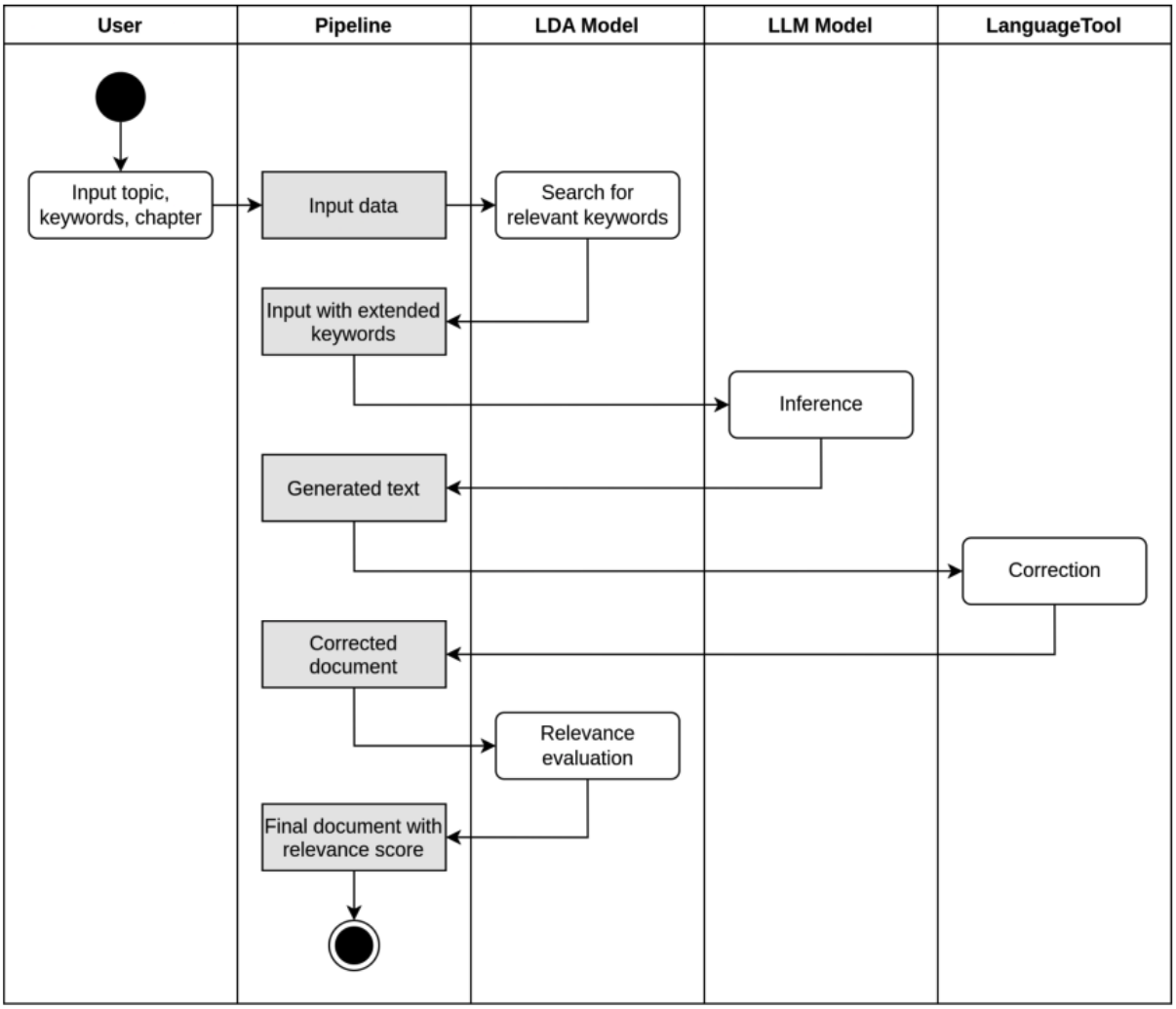

Рисунок 1. Схема запропонованого методу

Ключові слова, задані користувачем, подаються у модель LDA. Визначаються найбільш релевантні теми та слова з високою тематичною ймовірністю, які додаються до початкового списку. Основним призначенням цього етапу є розширення контексту, що буде використано під час генерації тексту. Навчена модель латентного розміщення Діріхле містить набір тем, кожна з яких, у свою чергу, складається з пар слів та їх вагових коефіцієнтів, що визначають їх приналежність до теми [4].

Для кожної теми T_i в моделі з N тем розраховується оцінка релевантності на основі попередньо заданих ключових слів K. Позначимо j-те слово у темі T_i як $W_{i,j}$, а як $p_{i,j}$ позначимо відповідну ймовірність (приналежність) цього слова у темі. Тоді оцінка релевантності S_i для теми T_i розраховується за формулою (1).

$$S_i = \Sigma_{W_{i,j} \in K} p_{i.j} \quad (1)$$

Вибір тем відбувається на основі попередньо встановленого порогового значення *topic_threshold*. Тема включається до набору кандидатів для розширення якщо її оцінка релевантності S_i є більшою за порогове значення, що виражено у формулі (2).

$$SelectedTopics = \{T_i \mid S_i > topic\_threshold\} \quad (2)$$

Наступним кроком є вибір слів із виділених тем для розширення списку ключових слів. Для кожної обраної теми T_i розглядаються всі її слова $W_{i,j}$ та їх ймовірності $p_{i,j}$. На цьому етапі також встановлюється порогове значення *word_threshold*, що використовується для фільтрації слів у межах теми. Набір розширених ключових слів E_i з теми T_i визначається за формулою (3).

$$E_i = \{W_{i,j} \mid p_{i,j} > word\_threshold\} \quad (3)$$

Таким чином формується список додаткових ключових слів E за формулою (4).

$$E = i \mid T_i \in SelectedTopics \cup E_i \quad (4)$$

Результуючий список ключових слів формується шляхом об’єднання початкового K та додаткового E наборів.

Фінальним етапом підготовки набору вхідних даних для тренування LDA моделі є перетворення у модель “Bag of words” [10]. Він включає створення структури даних, де текст представлений як набір слів без збереження інформації про порядок токенів у документі. [18]

На наступному кроці відбувається генерація тексту за допомогою великої мовної моделі на основі вхідних параметрів, що визначають контекст. Принцип роботи таких моделей полягає у генерації продовження для вхідного тексту. Таким чином, для виконання цього етапу потрібно привести набір вхідних параметрів до простого текстового представлення запиту (prompt).

Вхідні параметри:

- тема;
- ключові слова;
- назва розділу.

Вони трансформуються у структурований промпт “<TOPIC>topic</TOPIC>|<KEYWORDS>keywords</KEYWORDS>|<CHAPTER_NAME>chapter</CHAPTER_NAME>”, де TOPIC та CHAPTER - це відповідні значення вхідних параметрів теми та назви розділу, а KEYWORDS – це перелічені через кому ключові слова із результату виконання попереднього етапу.

Наприклад в реченні *«Хлопець, ступивши крок, побачила будівлю»*. виявляється порушення: присудок жіночого роду, підмет — чоловічого. Задане правило формує виправлення: заміна “побачила” на “побачив”.

На наступному етапі здійснюється оцінка відповідності згенерованого тексту набору ключових слів, що був заданий користувачем та розширений. Для обчислення цього значення використовується та ж сама натренована LDA модель, що й для розширення списку ключових слів.

 ISSN 1560-8956

Алгоритм знаходження релевантних тем можна виразити за допомогою математичної моделі. Нехай K - множина ключових слів, і T - множина тем, що містяться у LDA моделі. Своєю чергою, кожна тема T_i містить j слів $W_{i,j}$ з відповідними ймовірностями $p_{i,j}$. Функцію відображення ключового слова $k \in K$ на тему T_i у суму ймовірностей $p_{i.j}$ для кожного входження k у T можна виразити у вигляді формули (5).

$$f(k, T_i) = \sum_{(W_{i,j}, p_{i,j}) \in T_j, W_{i,j} = k} p_{i,j} \quad (5)$$

Використовуючи функцію f створено словник релевантних тем *keywords_relevant_topics*, як наведено у формулі (6).

$$keywords\_relevant\_topics[T_i] = \sum_{k \in K} f(k, T_i), \forall T_i \in T \quad (6)$$

Послідовність кроків для обчислення оцінки відповідності також записані у вигляді формул. Нехай D - документ, оцінку релевантності якого потрібно визначити. Тоді T_d є набором тем, які LDA модель визначила релевантними для документа, що виражено у формулі (7).

$$(T_d, w_d) = LDA(D) \quad (7)$$

Кожна з визначених релевантних тем T_{di} має відповідну вагу w_{di} для аналізованого документу. Тоді для кожної із визначених релевантних тем тексту визначаємо її вклад у загальну оцінку відповідності тексту за допомогою функції g, що виражено у формулі (8).

$$g(T_{di}, keywords\_relevant\_topics) = w_{di} * keywords\_relevant\_topics[T_{di}], \forall T_{di} \in keywords\_relevant\_topics \quad (8)$$

Таким чином, загальна оцінка S_D відповідності документу заданому набору ключових слів обчислюється за формулою:

$$S_D = \sum_{T_{di} \in T_d} g\left(T_{di,} keywords_{relevant_{topics}}\right) \quad (9)$$

Комбінація тексту з обчисленою оцінкою відповідності є результатами виконання останнього кроку та повинна бути повернена користувачу у якості фінального результату роботи системи [18].

Розробка програмного забезпечення генерації текстів

Архітектура реалізована за модульним принципом з мікросервісами, кожен з яких виконує окрему функцію [4].

Основні компоненти:

– LLM Server — REST-сервер генерації тексту з підтримкою моделей:
  1. GPT-3.5-turbo (через OpenAI API [14]),
  2. LLaMA 3-8B (локально на Google Colab [16] із оптимізацією PEFT + QLoRA).

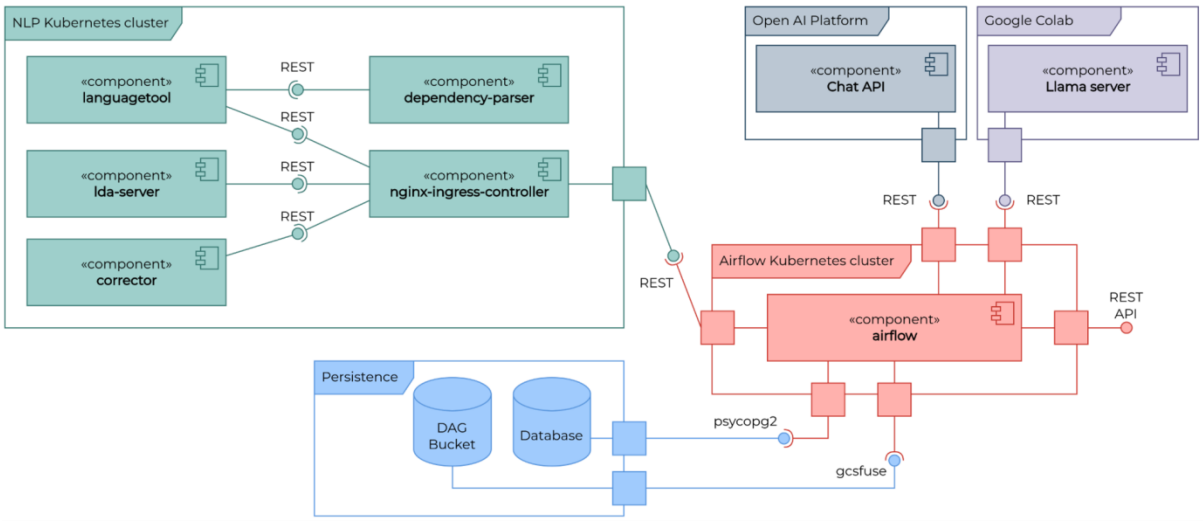

Рисунок 2. Архітектура програмного забезпечення

– LDA Server — сервер, створений за допомогою мови програмування Python 3 [5], що реалізує:

1. Розширення ключових слів за допомогою навченої LDA-моделі;
2. Обчислення тематичної релевантності згенерованого тексту.

– LanguageTool — модифікований Java-сервер для перевірки граматики, доповнений:

1. Новим типом правил, що базуються на синтаксичних залежностях;
2. Підключенням до Dependency Parser через API.

– Corrector — модуль, що вносить виправлення до тексту згідно з виявленими помилками.

– Dependency Parser — Python-сервер на основі spaCy [7] з моделлю uk_core_news_lg для виявлення синтаксичних залежностей у реченнях.

– Workflow Server (Apache Airflow [12]) — компонент, що координує виконання всіх етапів генерації.

– Database (PostgreSQL [17]) — для збереження запитів, згенерованих текстів, оцінок релевантності.

– DAG Bucket (Google Cloud Storage) — зберігає DAG-файли для Airflow.

Для генерації використовуються:

– GPT-3.5-turbo — через OpenAI API (формат: JSONL, role-based).

– LLaMA 3-8B — локально натренована з використанням transformers, peft, bitsandbytes, accelerate, trl.

Набір даних для fine-tuning сформований із понад 57 тис. документів із сумарним об’ємом понад 100 GB (до конвертації), отриманих із порталу ELA KPI [13]. Після обробки було створено JSONL-файли, де контекст і текст були розділені.

ISSN 1560-8956

Для виправлення граматичних помилок у згенерованому тексті використовується інструмент LanguageTool, що станом на поточний момент містить 1191 правило для перевірки та виправлень текстів української мови, підтримує новий тип правил на базі залежностей, використовує spaCy [6] для побудови дерева залежностей речення та використовує бібліотеку JEXL [11] для валідації граматичних відповідностей (наприклад, роду підмета і присудка). Правила для цього інструменту можна задавати декларативно у форматі XML або напряму, надавши Java імплементацію. Вхідними даними на цьому етапі є згенерований великою мовною моделлю текст, тоді як вихідними — виправлений відповідно до заданих правил текст [18]. Застосування даних бібліотек для обробки україномовних текстів та підходу до обробки надвеликих масивів даних добре зарекомендував себе в статті [19].

Для розширення можливостей інструменту LanguageTool було прийнято рішення про розробку нового типу правила, в основі якого лежить виявлення залежностей між членами речення, що є поширеною задачею обробки природної мови. Зокрема, інструмент spaCy підтримує виконання цієї задачі для текстів українською мови.

Експериментальні дослідження розробленого програмного забезпечення

Основною метрикою перевірки якості розробленого методу є оцінка відповідності згенерованого тексту набору ключових слів. Змінними параметрами під час дослідження ефективності є:

– використана модель: як у розрізі сімейств, так і за ознакою додаткового тренування (fine-tuning);

– включення кроку розширення набору ключових слів; Результати проведення експериментів приведено у табл. 1.

За результатами експериментального дослідження бачимо, що тренування моделей позитивно впливає на якість згенерованого тексту в розрізі його відповідності ключовим словами. Також покращенню результатів сприяє використання запропонованого у методі кроку розширення набору ключових слів. Важливо зауважити, що цей етап методу має позитивний вплив як для оцінки релевантності із використанням базового набору, так і розширеного.

Обидві моделі GPT і LLaMA продемонстрували ефективність, однак навіть базова модель GPT виявилась потужнішою за треновану модель LLaMA, проте друга виявилася більш гнучкою для локального використання та показала більший відносний приріст від процесу fine-tuning.

Що ж стосується кількостей виправлених помилок, то у більшості випадків додаткове тренування моделі провокує ріст цієї метрики. Це пояснюється даними, із використанням яких проводиться процес fine-tuning, адже модель здатна вбирати в себе не тільки корисні зв’язки із тренувального корпусу, а й інші його специфічні

особливості, серед яких можуть бути помилки різного роду. На рис. 3 приведено гістограму кількості помилок для кожної із досліджуваних комбінацій.

Таблиця 1.

Результати експериментів на відповідність

Сімейство моделей	**Модель**	**Наявність кроку розширення списку ключових слів**	**Час роботи, с**	**Кількість виправлених помилок**	**Оцінка релевантності / оцінка релевантності для розширеного списку**
GPT	gpt-3.5-turbo-1106	Так	50	2	0.0093 / 0.1614
		Ні	20	0	0.0080 / -
	gpt-3.5-turbo-1106-ft	Так	57	4	0.0124 / 0.1816
		Ні	30	0	0.0072 / -
Llama	Llama 3.8B Instruct	Так	96	7	0.0036 / 0.0638
		Ні	81	6	0.0038 / -
	Llama 3.8B Instruct FT	Так	108	15	0.0048 / 0.1061
		Ні	95	18	0.0023 / -

Як бачимо із наведених графічних матеріалів, немає явної кореляції кількості помилок із наявністю кроку розширення набору ключових слів. Натомість можемо помітити, що ця метрика є значно вищою для сімейства Llama, що знову підтверджує перевагу GPT у якості.

Також для дослідження ефективності розробленого прототипу програмного забезпечення було проведено ряд експериментів із різними моделями та конфігураціями з метою встановлення часу виконання як кожного етапу, так і процесу в цілому. Досліджені сценарії представлені у табл 2.

Варто зазначити, що для всіх етапів, тривалість яких не перевищує однієї секунди, час внеску в сумарну тривалість виконання вважається сталою величиною і складає 0.5 секунди. Результати всіх дослідів приведено в табл. 3.

 ISSN 1560-8956

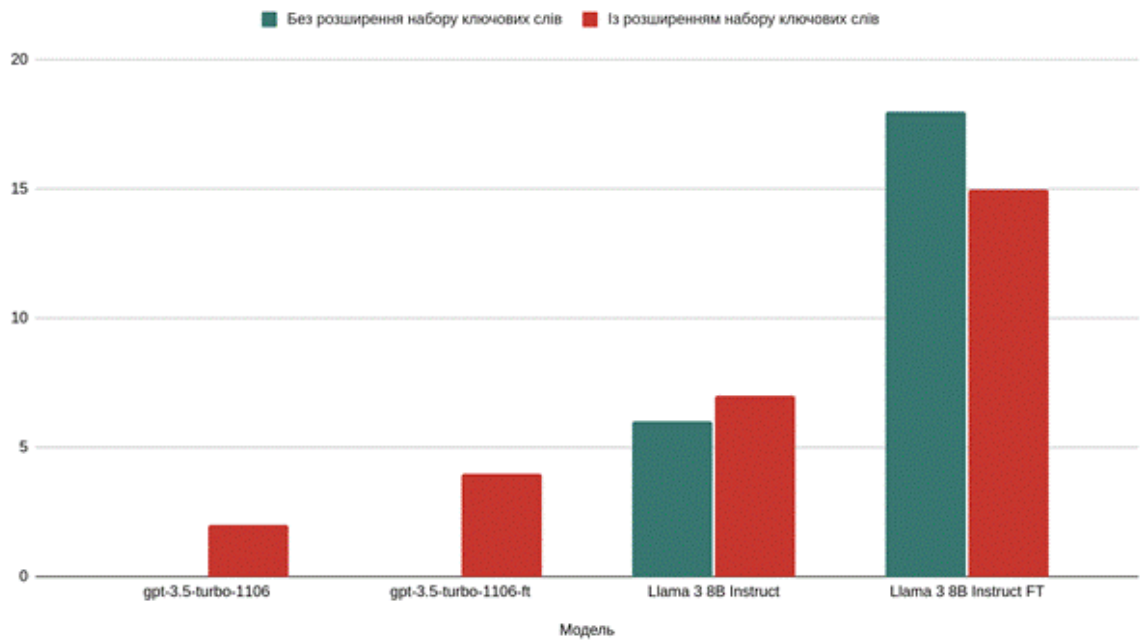

Рисунок 3. Залежність кількості помилок від моделі і набору ключових слів

Таблиця 2.

Досліджені сценарії

Номер сценарію	**Використана LLM**	**Розширення набору ключових слів**
1	GPT	вимкнене
2	GPT	увімкнене
3	Llama	вимкнене
4	Llama	увімкнене

Таблиця 3.

Результати проведених експериментів

Крок / Опис сценарію	**Тривалість сценарію №1, с**	**Тривалість сценарію №2, с**	**Тривалість сценарію №3, с**	**Тривалість сценарію №4, с**
Службовий крок десеріалізації параметру моделі	< 1	< 1	< 1	< 1
Розширення набору ключових слів	< 1	1	< 1	< 1

Закінчення табл. 3

Крок / Опис сценарію	**Тривалість сценарію №1, с**	**Тривалість сценарію №2, с**	**Тривалість сценарію №3, с**	**Тривалість сценарію №4, с**
Генерація тексту	11	12	62	64
Пошук та виправлення помилок	1	2	3	3
Оцінка відповідності	4	28	3	26
Збереження результатів	< 1	< 1	< 1	< 1
Сума виконання всіх кроків	17.5	44	69.5	95
Загальний час виконання	30	57	81	105

Із отриманих даних робимо висновок, що увімкнення кроку розширення набору ключових слів впливає насамперед на тривалість виконання етапу оцінки релевантності тексту. Окрім цього, помітна суттєва різниця в тривалості генерації тексту між GPT та Llama, що зумовлена відмінностями середовищ розгортання. Результати експерименту також вказують на наявність певних накладних витрат, що виражено різницею у сумі тривалостей виконання кожної із задач та загальним часом виконання процесу [18]. Ці величини, а також відношення між ними, представлено на рисунках (4, 5).

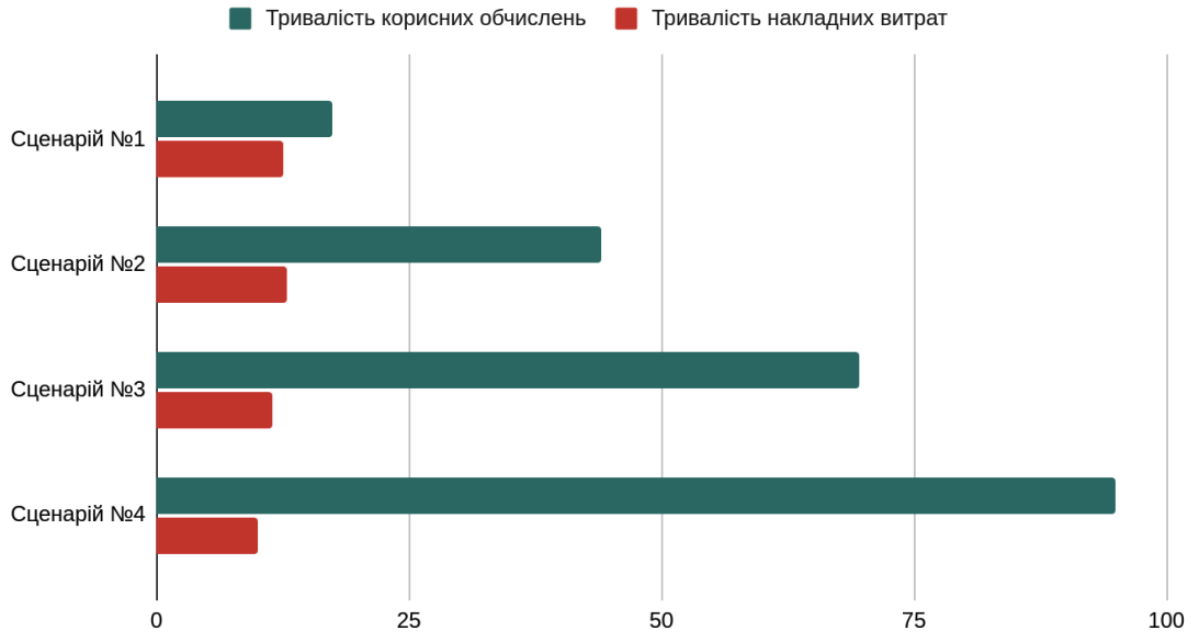

Рисунок 4. Тривалості корисних обчислень та накладних витрат

ISSN 1560-8956

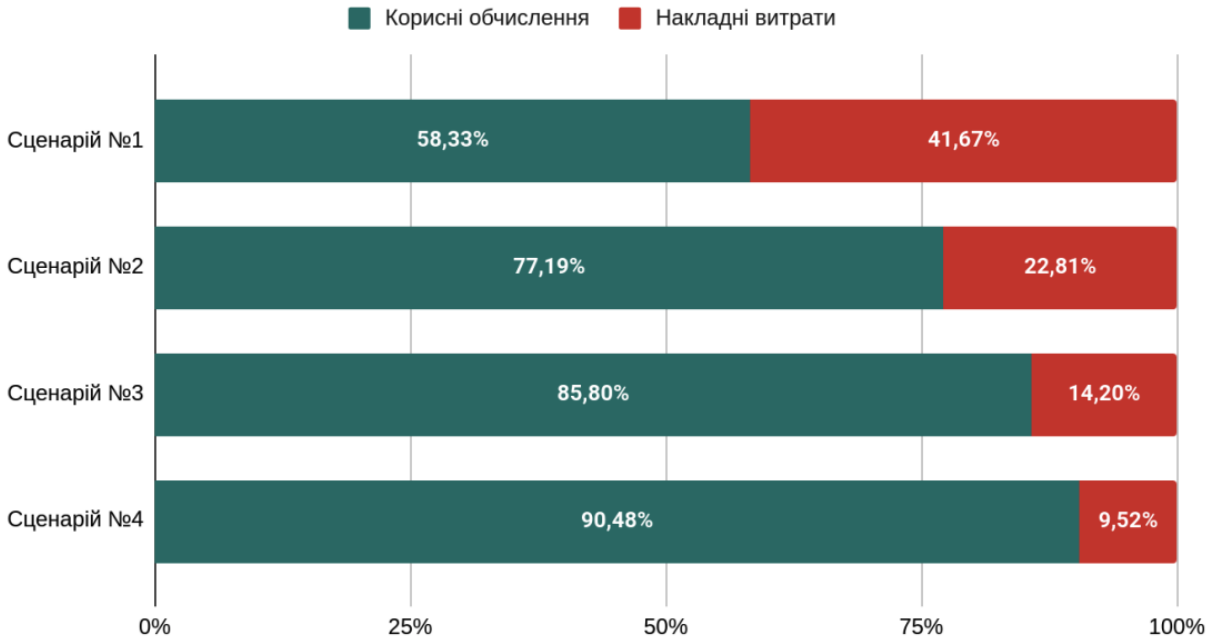

Рисунок 5. Відношення корисних обчислень до накладних витрат

З наведених візуалізацій було зроблено висновок, що хоч накладні витрати та присутні в системі, їх тривалість є здебільшого сталою, а відношення до корисних обчислень зменшується зі збільшенням навантаження на систему. Загалом такі витрати можна пояснити внутрішньою комунікацією між компонентами Apache Airflow в межах кластера та затратами на підготовку середовища до виконання процесу.

Висновки

У роботі розроблено метод генерації текстів українською мовою на основі таких етапів як розширення набору ключових слів, генерація первинного тексту, виправлення граматичних помилок та оцінки релевантності згенерованого тексту на основі LLM та LDA моделей.

Представлено архітектуру програмного забезпечення, засновану на мікросервісному підході та використанні Apache Airflow для керування робочими процесами. Розроблено програмне забезпечення, що автоматизує процес генерації тексту на основі заданої теми та ключових слів, із подальшим виправленням граматичних помилок. У межах дослідження виконано fine-tuning моделей GPT-3.5-turbo та LLaMA 3-8B на основі попередньо зібраного набору даних українською мовою. В якості засобу перевірки граматики текстів українською мовою використано LanguageTool, який було модифіковано додаванням нового типу правил, що враховують синтаксичні залежності між словами. У якості морфоаналізатора використано бібліотеку SpaCy. Програмне за безпечення продемонструвало високу якість генерації, релевантність створених текстів до тематики та здатність виявляти складні граматичні помилки, зокрема помилки узгодження.

Проведено дослідження ефективності використання великих мовних моделей у поєднанні з морфологічним аналізом та тематичним моделюванням для задачі генерації

українськомовних текстів. Виявлено, що тренування моделей позитивно впливає на якість згенерованого тексту в розрізі його відповідності ключовим словам. Також поліпшенню результатів сприяє використання розширення набору ключових слів

Отримані результати можуть бути використані для побудови інтелектуальних систем автоматизованого створення контенту українською мовою в сферах освіти, медіа та документообігу.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. *Radford A.* Improving Language Understanding by Generative Pre-Training [Електронний ресурс] / A. Radford, K. Narasimhan, T. Salimans. – 2021. – Режим доступу до ресурсу: https://cdn.openai.com/research-covers/language-unsupervised/language_ understanding_ paper.pdf.

2. *Vaswani A.* Attention Is All You Need [Електронний ресурс] / A. Vaswani, N. Shazeer, N. Parmar. – 2017. – Режим доступу до ресурсу: https://doi.org/10.48550/ arXiv.1706.03762.

3. *Touvron H.* Llama 2: Open Foundation and Fine-Tuned Chat Models [Електронний ресурс] / H. Touvron, L. Martin, K. Stone. – 2023. – Режим доступу до ресурсу: https://doi.org/10.48550/arXiv.2307.09288.

4. *Довгополюк Р. Р., Олійник Ю. О.*, Метод генерації текстів українською мовою, Матеріали VI Міжнародної науково-практичної конференції молодих вчених та студентів, 21-23 травня 2024 року, м. Київ, Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського», ФІОТ, 41-45 с.

5. Документація мови програмування Python. [Електронний ресурс] – Режим доступу до ресурсу: https://docs.python.org/3/.

6. Документація бібліотеки spaCy. [Електронний ресурс] – Режим доступу до ресурсу: https://spacy.io/usage.

7. *Honnibal M.* Introducing spaCy [Електронний ресурс] / Matthew Honnibal. – 2015. – Режим доступу до ресурсу: https://explosion.ai/blog/introducing-spacy.

8. LanguageTool [Електронний ресурс] – Режим доступу до ресурсу: https://languagetool.org/uk.

9. *Blei D.* Latent Dirichlet Allocation [Електронний ресурс] / D. Blei, A. Ng, M. Jordan. – 2003. – Режим доступу до ресурсу: https://www.jmlr.org/papers/volume3/ blei03a/blei03a.pdf.

10. *Олійник Ю. О.* Підхід до виявлення аномалій в потоках текстових даних / Ю. О. Олійник, О. Є. Афанасьєва, Г. Д. Аршакян. // Системні технології. – 2020. – №2. – С. 126–139.

11. Java Expression Language (JEXL) [Електронний ресурс] – Режим доступу до ресурсу: https://commons.apache.org/proper/commons-jexl/.

12. Apache Airflow [Електронний ресурс] – Режим доступу до ресурсу: https://airflow.apache.org/.

13. ELA KPI [Електронний ресурс] – Режим доступу до ресурсу: https://ela.kpi.ua/.

14. Open AI Platform [Електронний ресурс] – Режим доступу до ресурсу: https://platform.openai.com/docs/overview.

 ISSN 1560-8956

15. LanguageTool on GitHub [Електронний ресурс] – Режим доступу до ресурсу: https://github.com/languagetool-org/languagetool.

16. Google Colab [Електронний ресурс] – Режим доступу до ресурсу: https://colab.google/.

17. Документація PostgreSQL [Електронний ресурс] – Режим доступу до ресурсу: https://www.postgresql.org/docs/.

18. *Довгополюк, Р. Р.* Програмне забезпечення генерації текстів українською мовою : магістерська дис. : 121 Інженерія програмного забезпечення / Довгополюк Роман Русланович. - Київ, 2024. - 127 с. https://ela.kpi.ua/handle/123456789/72154.

19. *Д. Галайко, Ю. Олійник*. Застосування сховищ даних для виявлення плагіату в текстових документах / Адаптивні системи автоматичного управління. Том 2 № 45 (2024). – С. 100–108.