НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

Кафедра технічної кібернетики

«На правах рукопису»
УДК 004.8

«До захисту допущено»
Завідувач кафедри
_____ Ігор ПАРХОМЕЙ
(підпис)
“___”_____2020 р.

Магістерська дисертація

на здобуття ступеня магістра

зі спеціальності 126 «Інформаційні системи та технології»

на тему: Гібридний алгоритм автоматичної генерації тексту

Виконав: студент другого курсу, групи ІК-91мп
(шифр групи)

Кузнєцов Денис Вячеславович _____
(прізвище, ім’я, по батькові) (підпис)

Науковий керівник доцент, к.т.н., доцент Пасько В.П. _____ _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант з норм. контролю доцент, к.т.н., доцент Пасько В.П. _____
(назва розділу) (науковий ступінь, вчене звання, , прізвище, ініціали) (підпис)

Рецензент _____ _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2020 року

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

Кафедра технічної кібернетики

Рівень вищої освіти – другий (магістерський)

Спеціальність 126 «Інформаційні системи та технології»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ І.Р. Пархомей
(підпис)
«___»_____2020 р.

ЗАВДАННЯ
на магістерську дисертацію студенту
Кузнєцову Денису Вячеславовичу
(прізвище, ім'я, по батькові)

1. Тема дисертації Гібридний алгоритм автоматичної генерації тексту, науковий керівник дисертації: к.т.н., доцент Пасько Віктор Петрович,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом по університету від « 26 » жовтня 2020р. № 3132-с
2. Термін подання студентом дисертації _____
3. Об'єкт дослідження – Процес автоматичної генерації тексту за допомогою нейронних мереж
4. Предмет дослідження – Моделі і методи гібридних генераторів тексту на базі багатокомпонентних нейронних мереж
5. Перелік завдань, які потрібно розробити – дослідження архітектур генерації тексті; розробка гібридної архітектури автоматичного генератору тексту на базі нейронних мереж; розробка серверної частини демонстраційного

застосунку роботи моделі; розробка клієнтської частини демонстраційного застосунку роботи моделі

6. Орієнтовний перелік ілюстративного матеріалу – 4 плакати, 2 креслення

7. Орієнтовний перелік публікацій – 1 стаття

8. Консультанти розділів дисертації

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Норм. Контроль	доц. Пасько В.П.		
Перевірка на співпадіння	доц. Лісовиченко О.І.		

9. Дата видачі завдання – _____

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації	Примітка
1	Аналіз предметної області та постановка задачі	03.09.2020 – 11.09.2020	
2	Вибір технологій для розробки систем	12.09.2020 – 22.09.2020	
3	Проектування алгоритмів	23.09.2020 – 29.09.2020	
4	Розробка основних модулів	30.09.2020 – 10.10.2020	
5	Тестування роботи систем	11.10.2020 – 23.10.2020	
6	Написання та оформлення пояснювальної записки до дипломної роботи	24.10.2020 – 12.11.2020	
7	Висновки	12.11.2020 – 18.11.2020	

Студент _____ Кузнєцов Д. В.
(підпис) (ініціали, прізвище)

Науковий керівник дисертації _____ Пасько В.П.
(підпис) (ініціали, прізвище)

АНОТАЦІЯ

У даній дисертації розглянуто проблеми генераторів тексту, показано основні особливості існуючих рішень, їх класифікацію, переваги та недоліки.

Розглянуто основні методи обробки та генерації природної мови, проведено їх класифікацію, виявлено проблеми та сформульовано необхідність створення нової системи.

Проаналізовано основні етапи та методи обробки та генерації тексту, розглянуто інтегровані та модульні архітектури систем автоматизованої генерації тексту, виявлено можливі структури гібридних алгоритмів.

Розроблено гібридний алгоритм генерації тексту на основі комбінації нейронних мереж та програмне забезпечення для його випробування. Отриманий алгоритм натреновано на різних наборах даних різного розміру, тематики та інших характеристик. Алгоритм порівняно з існуючими поширеними алгоритмами генерації тексту, виявлено його сильні та слабкі сторони.

Ключові слова: генерація тексту, нейронні мережі, варіаційний автокодувальник, інтегрована архітектура нейронних мереж, модульна архітектура нейронних мереж.

Розмір пояснювальної записки – 93 аркушів, містить 15 ілюстрацій, 30 таблиць та 8 додатків.

ABSCTRACT

This paper considers the problems of text generators, showcases main characteristics of existing solutions, classification, advantages and disadvantages.

Main means and methods of processing and generation of natural language are considered, their classification is carried out, problems are revealed and the necessity of creation of new system is formulated.

The main stages and algorithms of text processing and generation are analyzed, integrated and modular architectures of automated text generation systems are considered, possible structures of hybrid algorithms are hypothesized.

A hybrid algorithm for text generation based on a combination of neural networks and software for its delivery has been developed. The developed algorithm is trained on different data sets of different size and other characteristics. The algorithm is compared with the existing common algorithms for generating text, its strengths and weaknesses are identified.

Keywords: text generation, neural networks, variational autocoder, integrated neural network architecture, modular neural network architecture.

The size of the explanatory note is 93 sheets, contains 15 illustrations, 30 tables and 8 appendices.

Пояснювальна записка

до магістерської дисертації

на тему: *Гібридний алгоритм автоматичної генерації тексту*

Київ – 2020 року

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

CV – Computer Vision;

NLP – Natural Language Processing;

NLG – Natural Language Generation;

RNN – Recurrent Neural network;

LSTM Long short-term memory;

GPT-3 Generative Pre-trained Transformer;

BLEU Bilingual Evaluation Understudy;

NIST метрика названа за Національним Інститутом Стандартів та Технологій;

ROUGE – Recall-Oriented Understudy for Gisting Evaluation;

METEOR – Metric for Evaluation of Translation with Explicit Ordering;

WER – Word Error Rate;

HLEPOR – автоматична метрика оцінювання якості згенерованого тексту;

RIBES – автоматична метрика оцінювання якості згенерованого тексту;

CIDEr – Consensus-based Image Description Evaluation;

MED – низка методів і оцінок, які використовуються в аналізі текстів медичної орієнтації;

TER – Translation Error Rate;

PYRAMID – автоматична метрика оцінювання якості згенерованого тексту;

SPICE – Semantic Propositional Image Caption Evaluation;

SPIDER – автоматична метрика оцінювання якості згенерованого тексту;

NoSQL – Збірний термін, який охоплює усі технології побудови баз даних, що не базуються на реляційній алгебрі.

ВСТУП

Однією з головних проблем інтеграції комп'ютерних систем з нашим життям є створення ефективних, швидких та простих інтерфейсів "спілкування" людини та комп'ютера. Ця задача набула останнім часом певної популярності через розробку інтерфейсів "комп'ютер – мозок", наприклад, компанію Neuralink. Ця розробка спрямована на безпосередній аналіз мозкової активності людини с ціллю інтерпретації цієї активності як команд і є одним з можливих напрямків розробки в цій галузі. Іншими, більш розвиненими засобами є аналіз згенерованої людиною інформації вже після її "висловлювання", дослідження в цьому напрямку охоплюють Computer Vision(CV), Natural Language Processing(NLP) та інші.

Основною перешкодою та випробовуванням в аналізі інформації, згенерованої людиною, будь то електронні хвилі мозку, текст або зображення, є той факт що ця інформація є неструктурованою, або слабко структурованою. Ця особливість даних робить розробку "традиційних" алгоритмів для їх аналізу неможливою або дуже складною. Саме тому великої популярності набули статистичні методи вирішення задач, які спрямовані не на "бездоганну" точність розпізнавання, аналізу та генерації даних, а на деякий, "досить добрий", результат. Саме тому найважливішими метриками математичних моделей які вирішують задачі аналізу даних за допомогою машинного навчання є їх швидкість, надійність, стабільність та відтворюваність результатів.

Проблема генерації тексту є одним з напрямків аналізу та машинного дослідження тексту та безперервно з ними пов'язана. Зазвичай нові підходи, які використовуються в генерації тексту, можна застосувати в його аналізі та, навпаки, прогрес в аналізі тексту майже завжди просуває прогрес і в галузі в цілому, що обумовлює актуальність досліджень саме в напрямку генерації тексту.

Метою дисертації є аналіз існуючих рішень автоматизації побудови текстів за допомогою машинного навчання і розробка гібридної моделі, що дала б можливість подолати виявлені в ході аналізу недоліки і порівняти утворене рішення з наявними за наступними метриками:

1. Розмір та простота збору бази текстів для отримання задовільних результатів;
2. Швидкість тренування моделей та їх “сходимість”;
3. Семантична та лексична коректність отриманих текстів;
4. Можливість адаптації рішення для аналізу тексту та для генерації текстів іншої тематики.

1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ І ПОСТАНОВКА ЗАДАЧІ

1.1. Об’єкт та предмет дослідження

Основною метою генерації природних мов (NLG) є дослідження знань та механізмів – як лінгвістичних, так і нелінгвістичних, – які використовуються мовцями та письменниками для того щоб поспілкуватися з цільовою аудиторією. Генерація тексту, таким чином, охоплює питання прийняття рішень: яка саме інформація буде доречною, а також як ефективно організувати та подати цю інформацію. Доповідачі та письменники також мають підбирати відповідні слова та формувати відповідні структури речень.

Ці проблеми проявляються у питаннях:

- Про що слід говорити?
- Коли слід говорити про це?
- Як ми повинні говорити про це?

Моделі для формування значущих та зв’язних речень є найважливішою частиною для багатьох систем обробки природної мови. Загальна ідея цих моделей полягає в тому, щоб оцінити та проаналізувати розподіл слів, речень та інших лексичних одиниць із бази знань, а потім використати цю інформацію для генерації реалістичних на вигляд речень. У відриві від інших напрямків аналізу природніх мов це завдання є важливим, з точку зору *новітньості* генерованого тексту. Результат цих алгоритмів зберігає семантичні та синтаксичні властивості вихідних даних, зібраних у реальному світі, і в той же час відрізняється від будь-якого з прикладів, використаних для тренування моделі. Наприклад, у контексті генерації діалогу бажано отримувати відповіді, які є більш різноманітними та менш загальними.

Очевидно, що існує безліч різних видів аналізу тексту з дуже різними цілями. Літературна критика – це аналіз тексту. Лектор, який оцінює студентські роботи, займається аналізом тексту. Бібліограф, який досліджує

лексичні конструкції в двох уривках тексту, щоб визначити, чи вони мають одного автора, займається аналізом тексту. Читання та перечитування приміток – це аналіз тексту. Використання відкритих питань в опитуванні – це аналіз тексту.

Більш того, всю спостережувану поведінку людей та систем можна представити як масив неструктурованої інформації, яка підлягає розбору за певним набором правил (або граматик у лінгвістиці). Таким чином, інтерпретацію поведінки можна проводити засобами обробки природної мови.

Різні засоби роботи з текстом можна категоризувати за кількома вимірами. Наприклад, один вимір – це формальність. Загалом, чим більш формальним є вислів, тим простіше його використати у різних контекстах, але, у той же час, тим складніше його складніше використати для описання декількох різних явищ, що робить вислів менш потужним.

Іншим виміром категоризації методів текстового аналізу є те, чи вони основані на базі знань чи на основі евристик. Методи, засновані на базах знань, беруть за вхідні дані масу текстового матеріалу, який аналізується як єдина сутність. Наприклад, примітки з дослідження спостереження за учасниками або стенограми засідань. В рамках такої сутності існує менша одиниця аналізу – ідея. У кожному окремому реченні може бути кілька різних ідей або не бути жодної ідею. Grounded theory[1], як правило, базується на базах знань.

Методи, що базуються на конкретних випадках, або евристиках, розглядають дані як набір порівнянних випадків, що відтворюються . в середовищі. Наприклад, можна поставити групі респондентів запитання з відкритою формою відповіді. Кожна відповідь аналізується, на їх основі утворюються групи відповідей що мають схожі метрики, які й утворюють евристики. Аналіз вмісту, як правило, базується на евристиках.

Аналіз на основі баз знань зазвичай є більш інтерпретаційним, менш формальним, менш відтворюваним і більш схильним до упереджень.

Як на основі випадків, так і на основі знань в аналізі тексту широко використовується кодування текстів. Кодування – це спосіб перетворення необроблених природних безформних даних в обмежений набір взаємопов'язаних символів, які утворюють основу для подальших міркувань.

1.2. **Огляд існуючих рішень**

Загалом кажучи, можна розрізнити три домінуючі підходи до побудови архітектури систем генерації природної мови:

- Модульні архітектури: такі архітектури передбачають досить чіткий розподіл між підсистемами що виконують різні елементи процесу генерації тексту, зі значними відмінностями між ними;
- Системи планування: реалізація системи аналізу тексту як системи планування дозволяє застосувати до неї відомі засоби та підходи науки штучного інтелекту. Такий підхід надає більш інтегрований, менш модульний погляд на окремі компоненти;
- Інтегровані або глобальні підходи: нині домінуюча технологія в генерації природної мови (як і в аналізі в цілому). Підходи цієї категорії, як правило, покладаються на вивчення статистичних вивчення відповідностей між (нелінгвістичними) входами та виходами певного алгоритму обробки тексту.

Далі у розділі будуть розглянуті системи саме третьої категорії як найбільш використовуваної в галузі. Інші підходи та їх можлива інтеграція будуть розглянуті в другому розділі.

1.2.1 Стохастичні системи генерації тексту

Підходи, розроблені останнім часом, спрямовані на отримання лексичних правил(граматик) через статистичні методи, прагнучі зменшити

обсяг ручної праці на формування цих правил. По суті, існує дві родини підходів для включення стохастичних засобів в процес реалізації систем генерації природної мови.

Один підхід, запроваджений через використання моделей, описаних у роботі Лангкілде та Найта щодо використання системи **HALogen** [2], спирається на дворівневий підхід, при якому невелике, сформоване вручну ядро граматики використовується для генерування альтернативних реалізацій цієї граматики, представлених у вигляді структури random forest, з яких стохастичний оцінювальний алгоритм вибирає оптимального кандидата. В оригінальній статті використовується база у формі N-грам, тоді як більш сучасні підходи експериментують з більш досконалими статистичними моделями для здійснення оцінювання[3].

Другий підхід не спирається на обчислювально дорогий підхід генерації та фільтрування, а використовує статистичну інформацію безпосередньо на етапі прийняття рішень щодо генерації. Прикладом цього підходу є система PCRU, розроблена у 2007[4], яка генерує найбільш вірогідне скінчення речення з урахуванням бази знань, використовуючи безконтекстну граматику. У цьому випадку статистичні дані використовуються для контролю поведінки генератора при оцінюванню, коли він шукає оптимальне рішення.

В обох підходах базовий генератор розроблюється вручну, тоді як статистична інформація використовується для фільтрації виходів. Очевидною альтернативою було б також покластись на статистичну інформацію і для генерації цього ядра. Повністю керовані даними граматичні підходи були розроблені шляхом виведення граматичних правил із Treebanks – баз знань, які було розмічено семантично. Наприклад, фреймворк OpenCCG[5] представляє англійськомовний аналізатор широкого застосування, заснований на комбінативній категоріальній граматиці[6], побудований на

базі знань, отриману з Penn Treebank, використовує статистичні мовні моделі для переоцінки.

Існує кілька інших підходів до реалізації, які використовують схожу ідею генерації граматики автоматично. У багатьох із цих систем базовий генератор використовує якийсь варіант алгоритму генерації діаграм[7], щоб покроково реалізувати певні частини вхідної специфікації та об'єднати їх в одну або кілька кінцевих структур, які потім можна оцінити. Існування стохастичних аналізаторів з граматиками широкого застосування спонукало дослідників більше зосередитись питаннях вдосконалення стохастичних аналізаторів, таких як уникнення структурної неоднозначності, або доповнення речень та слів англійською мовою. Дещо подібним чином статистичний підхід до мікропланування[8], фокусується на взаємодії між реалізацією граматики, агрегацією та сегментацією речень у комбінованій моделі.

1.2.2 Моделі машинного навчання для генерації тексту

Альтернативний підхід до аналізу проблем прийняття рішень для генерації природної мови – це розглядати їх як проблеми класифікації та оптимізації, наприклад, визначення вмісту[9] та реалізація наборів правил лексики та семантики[10]. Оскільки генерація, зрештою, полягає у виборі найбільш оптимального рішення на декількох рівнях, один із способів моделювання процесу полягає у використанні каскадних моделей, які будуються з класифікаторів, де дані будуються поступово та вихідні данні певного класифікатора будуть вхідними даними іншого.

Це приводить до розглядання певного алгоритму генерації тексту як послідовності поєднаних математичних моделей, або пайплайну машинного навчання. Іншим способом моделювання засобами машинного навчання буде побудова зваженої багатошарової решітки, де генерація є рішенням за

принципом first-best: на будь-якому етапі класифікатор дає найбільш вірогідний результат , що веде до наступного етапу по найімовірнішому шляху.

Це узагальнення концептуально пов'язане з поглядом аналізу природної мови з точки зору побудови політик та обмежень в рамках Reinforced learning, що визначає обхід через послідовності станів, які можуть бути ієрархічно організованими[11].

У згаданій вище роботі, дослідники починають з невеликої бази знань з анотованих вручну текстів описів маршрутів, та декомпозують проблему генерації подібних описів на серію з восьми класифікаційних проблем, від визначення лінеаризації термів, до визначення лексичної форми дієслів та типу їх аргументів. Рішення щодо генерації приймаються за допомогою алгоритму KStar, який, як показано, перевершує більшість інших розглянутих класифікаторів на всіх етапах прийняття рішень щодо генерації тексту. Підходи до генерації природної мови, що базуються на навчанні на прикладах, також є досить поширеними.

Подібна архітектура були розроблена в рамках вирішення задачі генерації статей на німецькій мові[12], знову взявши в якості вводу текстові дані, які анотуються представленням залежності членів речення.

В ній автори використовують послідовність класифікаторів для генерації правил та виразів, побудованих за цими правилами. Вони використовують модель оцінки та сортування, засновану на моделі Support Vector Machines, яка, приймаючі розмічені вхідні дані, виконує два завдання в довільному порядку:

- відображення тексту до синтаксичного дерева для лінеаризації;
- вставка виразів що посилаються на інші частини тексту.

Цікаво, що за результатами роботи було відзначено, що якість виконання будь-якого із завдань залежить від порядку, оскільки обидва

завдання з класифікації працюють гірше, коли вони займають друге місце в пайплайну. Спостерігається незначне покращення, коли завдання виконуються паралельно, найкращі результати були отримані в рекурсивній архітектурі, заснованій на редакції, де синтаксичне відображення супроводжується вставкою виразів, що посилаються, з подальшим переглядом синтаксису.

Таким чином, іншим підходом до генерації реалізації певної граматики є використання одного або кількох класифікаторів для покращення результатів. Одним з таких алгоритмів є підхід до лінеаризації складових речення, використовуючи двоступеневий підхід з класифікаторами максимальної ентропії[10]. Спочатку розраховується, яка складова речення має бути першою, потім усі інші складові оцінюються, сортуються та утворюють залишок речення.

Іншим засобом генерації реалізації певної граматики, є використання Support Vector Machines, які використовують в якості вхідних даних неорганізовані структури залежностей, які потім класифікуються за допомогою каскадних моделей машинного навчання. Початковий класифікатор декодує вхідні данні у відповідні синтаксичні ознаки, тоді як два наступні класифікатори спочатку лінеаризують синтаксис, а потім надають правильну морфологічну реалізацію для лексем компонентів[13]. Моделювання вибору за допомогою каскадних моделей з класифікаторів не обмежується лише задачею реалізації певної граматики. Одним із результатів використання моделей машинного навчання у всьому процесі генерації природної мови є можливість експериментувати зі вхідними даними: чим більше рішень приймається за допомогою моделей машинного навчання, тим менше лінгвістичним і більш абстрактним можуть бути вхідні данні, що дозволяє розробити системи генерації тексту які приймають ніяк не оброблені текстові дані.

1.2.3 Нейромережі для генерації тексту

Закінчити огляд сучасних рішень можна обговоренням застосувань архітектури глибоких нейронних мереж. Рішення не розглядати їх в рамках огляду стохастичних або моделей машинного навчання мотивоване великою популярністю саме цих моделей[14], а також великим (та постійно зростаючим) асортиментом архітектур та підходів, які використовують саме нейронні мережі для генерацію тексту, розроблених, опублікованих, та використовуваних промислово.

Власне кажучи, застосування глибокого навчання у в аналізі природної мови є досить старою розробкою[15], проте її не можна було використати промислово через слабкі обчислювальні потужності, які спонукали використовувати невеликі моделі та масиви даних. З початку 1990-х років, коли інтерес до підходів на базі нейронних мереж зменшився у спільнотах дослідників обчислювальної лінгвістики та штучного інтелекту, використання цих моделей змістилося у спільноти когнітивні досліджень. Нещодавнє пожвавлення інтересу до нейромереж частково зумовлене досягненнями апаратного забезпечення, яке може виконувати багато обчислень та, таким чином, підтримувати складні проблеми навчання мереж, притому використовуються не тільки реставровані старі моделі, а й принципово нові архітектури[16]. Іншою відзнакою використання нейронних мереж є наслідки алгоритмів їх навчання (зворотного поширення помилки). Внутрішня будова нейронних мереж, яку можна розглянути як відображення вхідних даних у простір з іншою вимірністю, є щільною, низьковимірною та розподіленою, що робить її оптимізованою для фіксації граматичних та семантичних узагальнень. Зазначені вище моделі також досягли помітних успіхів у послідовному моделюванні з використанням нейронних мереж прямого зв'язку, логарифмічно-білінійних моделей та рекурентних нейронних

мереж, включаючи RNN з одиницями довготривалої пам'яті (LSTM). Останні зараз є домінуючим типом RNN для завдань моделювання мови. Їх головна перевага перед стандартними лінгвістичними моделями полягає в тому, що вони обробляють послідовності різної довжини, уникаючи при цьому розрідженості даних та вибуху в кількості параметрів за допомогою проекції попередніх даних у низьковимірний простір, завдяки чому ці данні відображуються у наступних результатах генерації.

Демонстрацію потенційної корисності такого роду мереж для генерації природної мови надали дослідники з університету Торонто[20], які використовували модель LSTM на рівні символів для генерації граматично коректних англійських речень. Однак це зосереджувалось виключно на їхньому потенціалі для реалізації певної граматики. Моделі, які генеруються на основі семантичних або контекстних входів, об'єднуються в два пов'язані типи моделей, описані нижче.

Впливовою архітектурою також є підхід “Encoder-Decoder”, де нейронна мережа використовується для кодування вхідних даних у деяке векторне представлення, яке служить допоміжним входом для декодуючої нейронної мережі. Це розділення між кодуванням та декодуванням дозволяє в принципі поділити векторне представлення між кількома навчаючимося генераторами природної мови в умовах багатозадачного навчання[21]. Архітектури кодера-декодера особливо добре підходять для завдань типу “послідовність в послідовність” (seq2seq), таких як машинний переклад, що, як можна вважати, вимагає відображення вхідних послідовностей змінної довжини в мові-джерелі та послідовностей змінної довжини в мові-цілі. Цей підхід легко адаптувати і до подання даних до тексту. Наприклад, вже натреновані seq2seq моделі адаптують для генерування тексту з подань абстрактних значень (AMRs).

Подальшим важливим розвитком у парадигмі “Encoder-Decoder” є використання механізмів, заснованих на зважені уваги(Attention Mechanism), які змушують кодер під час тренування більше зважувати деякі частини вхідного кодування при прогнозуванні певних частин виводу під час декодування. Цей механізм усуває необхідність прямого вирівнювання вводу-виводу, оскільки моделі, орієнтовані на увагу, можуть засвоїти відповідності введення-виведення на основі вільних зв`язків вхідних уявлень та вихідних текстів.

У генераторах природної мови, які використовуються для генерації відповідей в інтерактивному контексті (наприклад, діалог чи публікації в соціальних мережах) застосовують цю архітектуру. Наприклад, у роботі “Semantically Conditioned LSTM-based Natural Language Generation for Spoken Dialogue Systems” використовують семантично обумовлені LSTM для генерації наступного акту в діалозі; подібний підхід застосовується у роботі “A Neural Network Approach to Context-Sensitive Generation of Conversational Responses”, в якій розглянуто використання RNN для кодування як вхідного висловлювання, так і контексту діалогу, з декодером для прогнозування наступного слова у відповіді.

Методи, що базуються на рекурентних нейронних мережах (RNN), такі як лінгвістичні моделі, є найпопулярнішими підходами до генерації тексту. Ці генератори тексту на основі RNN покладаються на рішення з оцінкою максимальної вірогідності (MLE), такі як teacher forcing (тобто модель навчена передбачати наступний пункт, враховуючи всі попередні спостереження); однак у добре відомо, що MLE занадто спрощеною метрикою для цього завдання. Методи, засновані на MLE, страждають від упередженості впливу, що означає, що під час навчання моделі надаються реальні данні, але під час тестування вона порівнюються лише власні прогнози.

Змагальні прогнозувальники(GAN), які базуються на функції змагальних втрат і мають генераторну та дискримінаційну нейронні мережі, менше страждають від вищезгаданих проблем. GAN можуть забезпечити кращу систему генерації зображень у порівнянні з традиційними методами, заснованими на MLE, і досягли значного успіху в галузі комп'ютерного зору для створення реалістичних та чітких зображень. Цей великий успіх спонукав дослідників застосувати цей підхід і до вирішення задач NLP.

GAN були нещодавно використані в різних застосунках NLP, таких як машинний переклад, моделі діалогу, відповіді на запитання та генерація природної мови. Однак застосування GAN у NLP є складним завданням через дискретну природу тексту. Отже, метод зворотного розповсюдження помилки буде неможливо застосувати для дискретних виходів, і ще складнішим є передача градієнтів через дискретні вихідні данні генератора. Існуючі рішення, засновані на GAN, можна класифікувати за методикою, яку вони використали для вирішення проблеми дискретного характеру тексту: методи, засновані на підкріпленому навчанні(RL), рішення на основі прихованого простору та підходи, засновані на постійному наближенні дискретної вибірки. Кілька версій методів, заснованих на RL, були представлені в літературі, включаючи SeqGAN, MaskGAN та LeakGAN. Однак вони часто потребують попередньої підготовки і обчислювально дорожчі порівняно з методами двох інших категорій. Приховані просторові рішення виводять прихований простір подання тексту за допомогою автокодувальника і намагаються дізнатись багатовимірні дані цього простору. Іншим підходом для генерації тексту за допомогою GAN є пошук безперервного наближення дискретної вибірки за допомогою методики Gumbel Softmax або апроксимація недиференційованого оператора argmax з неперервною функцією.

1.3. **Проблеми генерації тексту**

Найбільш впливовою розробкою останнього часу була архітектура GPT-3, заснована на зважені ваги, яка мала 175 мільярдів параметрів. Розмір самої моделі та кількість тестових даних, на яких вона була натренована, дозволили розглядати її як модель генерації тексту широкого застосування без необхідності дотренування на довільних даних. Модель виявилася настільки потужною, що окрім надзвичайно широкого кола “традиційних” задач генерації тексту, як продовження та генерація відповіді, вона могла генерувати абстрактну інформацію, як ноти для музичних інструментів та програмний код. Це значний прорив для напрямку автоматичної генерації тексту, проте проблеми, виявлені при використанні такої моделі, спонукають дослідників надалі вивчати інші підходи до генерації текстів.

Ця модель генерувала текст настільки ефективно, що не дивно, що багато хто поспішно почав говорити про штучний інтелект. Але реалістичний вивід GPT-3 і вражаюча універсальність – це результат проектування, а не справжній розум. З одного боку, модель все ще може згенерувати незв’язні тексти, які виявляють повну відсутність людського розуміння. З іншого, навіть реалістичні тексти не мають достатньої глибини, виглядаючи більше як компіляції, а не оригінальні композиції.

Навіть Сем Альтман, який був співзасновником OpenAI разом з Ілоном Маском, намагався приглушити ситуацію: “Ажіотаж з GPT-3 занадто великий. Модель вражає, але все ж має серйозні слабкі сторони і іноді робить дуже дурні помилки. ШІ збирається змінити світ, але GPT-3 – це лише дуже ранній погляд. Нам ще багато чого потрібно з’ясувати”.

Іншою проблемою цього підходу є прогресія обчислювальної складності. GPT-3 є на порядок більшою ніж її попередники, та повноцінне навчання її без допомоги корпорації Google вимагало б 4.5 мільйонів доларів

та 335 років часу, роблячи розробки в цьому напрямку неможливими для незалежних дослідників або інститутів. Також вражає швидкість ускладнення моделей: вона вже випередила темпи зростання потужності апаратних прискорювачів і можна сміливо казати що “безглузде” збільшення моделей у сподіванні на паралелізацію та обчислювальні потужності застаріло.

З іншого боку, штучний інтелект як технологія стає все ближчим. В останніх відеокартах NVIDIA широко представлена технологія тензорних ядер, обчилювальних підсистем що оптимізовані під матричні операції, які лежать в основі нейронних мереж[21]. Розвиток цієї технології призвів до того, що наразі кожна людина може купити для свого комп’ютера надзвичайно потужний обчислювач, який дозволяє вдома будувати моделі, які 10 років назад можна було побудувати лише розподіленими засобами. У той же час, стрімко розвивається хмарний напрямок штучного інтелекту. Microsoft Azure, Google Cloud, Amazon Web Services – усі ці постачальники надають інструменти для тренування моделей, які потребують оплати лише за використання, та які можна масштабувати безкінечно. Ті ж самі постачальники об’єднуються з закладами освіти щоб підготовити програми для спеціалістів зі штучного інтелекту, отже, в майбутньому кожна людина матиме можливість отримати відповідну освіту та досвід вдома.

Це обумовлює актуальність даної роботи: вивчення можливостей гібридних алгоритмів генерації тексту для зменшення розміру та складності моделей генерації тексту.

Висновки до розділу

Розробка архітектур генерації тексту з широкого погляду – це процес створення технологій для розробки програмного забезпечення, яке може функціонувати як автор, оратор чи співбесідник. Ця галузь дуже складна та високотехнологічна, бо письмо та розмова навіть з людського погляду вважаються складними мистецтвами, в яких неможливо досягнути

досконалості та які самі по собі є інтелектуальними досягненнями. На противагу цьому, існуючі програмні рішення спроектовані таким чином, щоб вирішувати певну вузьку задачу, а системи широкого застосування ще недостатньо розвинені та вимагають великих ресурсів для експлуатації.

Генерація тексту вивчається вже довгий час, але використання для її цілей алгоритмів на базі нейронних мереж серйозно вивчалася в обчислювальній лінгвістиці лише протягом останніх п’яти чи десяти років, і тому це все ще є відкритим простором для досліджень.

Частина різноманітності підходів у створенні тексту, безумовно, походить від проблеми, з якою дослідники стикаються наразі: попри те що сучасні системи досить легко генерують невеликі тексти або речення, незрозумілі механізми та алгоритми для побудови великих, складних та зв’язних письмових творів. Це може бути неочікуваним, бо люди читають та пишуть текст майже щодня і досить успішно маніпулюють ним, явних знань про те як створювати тексти “механічно”, дуже мало. Ця проблема присутня в усіх практичних застосуваннях штучного інтелекту: речі які для людини ясні, дуже важко формалізувати.

Одне з центральних питань стосується організації тексту. Очевидно, що природний текст організований, і що його організація є суттєвою для його функцій. Текст має частини, які систематизовані. Але яка природа організації або структури тексту? Які частини та які принципи розташування?

Загальновизнаних відповідей на ці питання немає. Відповіді, доступні за межами обчислювальної лінгвістики, є частковими та складними. Є відповіді спеціалістів з логіки, відповіді граматиків тощо, часто які представляють переважно припущення їх розробників, а не вимірювані і порівняльні результати.

Крім цієї високорівневої інформація, що має вирішальне значення для формування тексту, немає відомостей про організацію тексту на низькому

рівні, достатньому для розробки і підтримки програмного забезпечення. Потрібно набагато більше деталей.

В наслідок цього, аналіз та генерація тексту знаходяться лише на початку свого розвитку. Відомі архітектури та підходи до генерації тексту мають свої недоліки, що змушує дослідників звертатися до інших галузей науки, або знаходити рішення на стику вже існуючих досліджень.

2. АРХІТЕКТУРА ІНТЕЛЕКТУАЛЬНИХ СИСТЕМ АВТОМАТИЧНОЇ ГЕНЕРАЦІЇ ТЕКСТУ

2.1. **Аналіз тексту**

2.1.1 Видобуток інформації

Текст на природній мові містить багато інформації, яка безпосередньо не підходить для автоматичного аналізу. Однак комп'ютери можна використовувати для обробки великої кількості тексту та вилучення корисної інформації з окремих слів, фраз або уривків. Тому видобуток інформації можна розглядати як обмежену форму повного розуміння природної мови, коли ми заздалегідь знаємо, яку інформацію ми шукаємо. Основне завдання – витягти частини тексту та призначити їм певні ознаки.

Задача вилучення інформації, розкладається на низку етапів обробки, як правило, включаючи токенізацію, сегментацію речень, присвоєння частини мови та ідентифікацію названих сутностей, наприклад, імен осіб, назв населених пунктів організацій. Хоча найточніші системи вилучення інформації часто включають вручну виготовлені модулі обробки мови, досягнуто значного прогресу в застосуванні методів видобутку даних для частини цих етапів[17].

Вилучення сутності у технічному розумінні – це задача призначення позначки для слів: Слово, де сутність починається, отримує позначку "B", слова-продовження отримують позначку "I", а слова поза сутністю отримують позначку "O".

Отже, місце має задача класифікації ряду слів за ознакою "позначка", причому навколишні слова є вхідним вектором ознак. Найпростіше сформувати вектор ознак за допомогою двійкової схеми кодування. Кожний елемент цього вектору можна розглядати як підтвердження того, чи існує

певна закономірність у реченні чи ні. Наприклад, елемент вектору приймає значення 1, якщо попереднім словом є слово “Google”, та 0 у іншому випадку. Також ми можемо перевірити не тільки наявність конкретних слів, але й те, чи починаються слова з великої літери, мають певний суфікс чи є певною частиною мови.

Маючи постановку задачі класифікації, можна використовувати будь-який ефективний метод класифікації для класифікації позначок слів. Гарним варіантом буде метод підтримуючих векторів, завдяки його здатності ефективно обробляти великі розріджені вектори функцій. Це важливо в контексті того, що математично моделі текстів представляються як великі розріджені матриці або графи. Детально зберігання інформації буде розглянуто у наступному підрозділі.

2.1.2 Кодування тексту

Для побудови великих баз знань необхідно попередньо обробити текстові документи та зберегти інформацію у структурі даних, яка є більш доречною для подальшої обробки ніж звичайний текстовий файл. Незважаючи на те що існує кілька методів, які намагаються використати синтаксичну структуру та семантику тексту, більшість підходів до аналізу тексту базуються на ідеї, що текстовий документ може бути представлений набором слів, тобто текстовий документ описується на основі набір слів, що містяться в ньому (Bag of words). Однак для того, щоб мати можливість визначити принаймні важливість слова в даному документі, або його вагу, зазвичай використовується векторне подання, де для кожного слова зберігається числове значення “важливості”. На сьогодні переважаючими підходами, заснованими на цій ідеї, є векторна модель, імовірнісна модель та логічна модель[18].

Для того, щоб отримати всі слова, що використовуються в даному тексті, потрібен процес токенізації, під час якого текстовий документ розбивається на набір слів, та видаляються всі розділові знаки а також нетекстові символи замінюються одиничними пробілами . Потім це символьне подання використовується для подальшої обробки.

Сукупність різних слів, отриманих шляхом злиття всіх текстових документів колекції, називається словником колекції документів. Для того, щоб дати можливість більш формального опису алгоритмів, ми визначаємо спочатку деякі терміни та змінні: Нехай D – це набір документів, а $T = \{t_1, \ldots, t_m\}$ словник , тобто сукупність усіх різних термінів, що зустрічаються в D, тоді частота терму $t \in T$ у документі $d \in D$ задається $tf(d,t)$. Вектори термів позначаються $\overrightarrow{t_d} = (tf(d,t_1), \ldots, tf(d, t_m))$. Пізніше нам також знадобиться поняття центроїду множини X термінових векторів. Він визначається як середнє значення $\overrightarrow{t_X} := \frac{1}{|X|}\sum_{\overrightarrow{t_d} \in X} \overrightarrow{t_d}$ $\sum$ його термових векторів. У подальшому ми застосуємо tf також на підмножинах термінів: Для $T' \subseteq T$ нехай $tf(d,T') := \sum_{t \in T'} tf(d,t)$.

Для того, щоб зменшити розмір словника, а отже й розмірність опису документів у колекції, набір слів, що описують документи, можна зменшити за допомогою фільтрування, лематизації та стемінгу.

Методи фільтрування вилучають слова зі словника. Стандартним методом фільтрації є фільтрація через стоп-список. Ідея цього методу полягає у видаленні слів, які містять мало інформації про вміст або взагалі не містять її, наприклад, сполучники, прийменники тощо. Крім того, можна сказати, що слова, які трапляються надзвичайно часто, мають незначний інформаційний вміст для розрізнювання документів, а також слова, які трапляються дуже рідко, зазвичай не мають особливу статистичну значимість і можуть бути

вилучені зі словника. З метою подальшого зменшення кількості слів у словнику можна також використовувати методи вибору термів.

Методи лематизації намагаються зіставити форми дієслова з нескінченним часом, а іменники – з формою однини. Однак, щоб досягти цього, форма слова повинна бути відома, тобто повинна бути визначена частина мови кожного слова в текстовому документі. Оскільки цей процес визначення, як правило, досить трудомісткий і схильний до помилок, на практиці застосовується лише стемінг.

Стемінг – процес, під час якого будуються основні або невизначені форми слів. Стем – це група слів з однаковим (або дуже подібним) значенням. Після закінчення процесу кожне слово представляється своєю основою. Найбільш відомий алгоритм, що базується на правилах, був спочатку Портером. Він визначив набір правил для ітеративного перетворення (англійських) слів до їх основи. Для української мови існують алгоритми, розроблені компанією Grammarly та опубліковані в 2016 році.

Для подальшого зменшення кількості слів у словнику також можуть використовуватися алгоритми індексації або вибору ключових слів. У цьому випадку для опису документів використовуються лише вибрані ключові слова. Найпростіший спосіб вибору ключових слів – це вилучення ключових слів на основі їх ентропії. Наприклад для кожного слова t у словниковому запасі може бути обчислена ентропія (2.1.1).

$$W(t) = 1 + \frac{1}{log_2|D|}\sum_{d \in D} P(d,t)\ log_2 P(d,t)\text{, де } P(d,t) = \frac{tf(d,t)}{\sum_{l=1}^{n} tf(d_l,t)}. \qquad (2.1)$$

Тут ентропія визначає, наскільки слово підходить для розділення документів за допомогою пошуку за ключовими словами. Наприклад, слова, що трапляються у багатьох документах, матимуть низьку ентропію. Ентропію можна розглядати як міру важливості слова в даній предметній області. В якості слів-індексів можна вибрати ряд слів, що мають високу ентропію щодо

загальної частоти, тобто зі слів, що трапляються однаково часто, можна віддати перевагу тим, що мають вищу ентропію.

Незважаючи на свою просту структуру без урахування семантики, *векторна модель*(Vector Space Model) дозволяє дуже ефективно аналізувати величезні обсяги даних. Спочатку вона була введена для індексації та пошуку інформації, але зараз використовується також у багатьох алгоритмах аналізу тексту, а також у більшості доступних на даний час систем пошуку документів.

Векторна модель представляє документи як вектори в m-мірному просторі, тобто кожен документ d описується числовим вектором ознак $w(d) = (x(d, t_1), \ldots, x(d, t_m))$. Таким чином, документи можна порівнювати, використовуючи прості векторні операції, і навіть запити можна виконувати, кодуючи його терми, роблячи їх подібними до самих документів. Потім цей вектор можна порівняти з кожним документом, а список результатів можна отримати, впорядкувавши документи відповідно до чисельної міри подібності. Основним завданням подання документів у векторному просторі є пошук прийнятного кодування вектору ознак.

Кожен елемент вектора зазвичай представляє слово (або групу слів) колекції документів, тобто розмір вектора визначається розміром словника. Найпростіший спосіб кодування документа – це використання двійкових векторів термінів, тобто векторному елементу дається значення 1, якщо в документі використовується відповідне слово, і 0, якщо слово не використовується. Для підвищення ефективності зазвичай використовують термові схеми зважування, де ваги відображають важливість слова в конкретному документі розглянутої колекції. Великі ваги присвоюються термінам, які часто використовуються у відповідних документах, але рідко в усій колекції документів. Таким чином, вага $w(d, t)$ для терму t в документі d обчислюється за формулою

$$w(d,t) = \frac{tf(d,t)\log\left({}^{N}/_{n_t}\right)}{\sqrt{\sum_{j=1}^{m} tf\left(d,t_j\right)^2 (\log\left(N/n_{t_j}\right))^2}}, \tag{2.2}$$

де N – розмір колекції документів D, а n_t – кількість документів у D, які містять терм t.

На основі системи ваг документ d визначається вектором вагових значень $w(d) = (w\ (d,\ t_1), \ldots, w(d,t_m))$ та подібністю S двох документів d_1 і d_2 (або подібністю документа та вектору запиту). Його можна обчислити, виходячи з внутрішнього добутку (2.1.3)

$$S(d_1,d_2) = \sum_{k=1}^{m} w(d_1,t_K)\cdot w(d_2,t_K). \tag{2.3}$$

Часто використовуваною мірою відстані є евклідова відстань. Відстань між двома текстовими документами $d_1,\ d_2 \in D$ обчислюється наступним чином (2.1.4).

$$dist(d_1,d_2) = \sqrt{\sum_{k=1}^{m} |w(d_1,t_k) - w(d_2,t_k)|^2}. \tag{2.4}$$

Однак евклідову відстань слід використовувати лише для нормованих векторів, оскільки в іншому випадку різна довжина документів може призвести до меншої відстані між документами, які мають менше слів, ніж між документами, що мають більше спільних слів.

Зауважимо, що для нормованих векторів скалярний добуток мало чим відрізняється за поведінкою від евклідової відстані, оскільки для двох векторів $\vec{x}$ і $\vec{y}$ він буде обчислюватися за формулою (2.1.5).

$$\cos\varphi = \frac{\vec{x}\,\vec{y}}{|\vec{x}|\cdot|\vec{y}|} = 1 - \frac{1}{2}d^2\left(\frac{\vec{x}}{|\vec{x}|},\frac{\vec{y}}{|\vec{y}|}\right). \tag{2.5}$$

Часто методи видобутку тексту можуть застосовуватися без подальшої обробки. Однак іноді для покращення наявної інформації про терми та ознаки

може використовуватися додаткова лінгвістична Для цього часто застосовуються такі підходи:

- Part-of-speech tagging (POS) – розмічування частин мови;
- Text chunking – розбивання тексту;
- Word Sense Disambiguation (WSD) – розрішення лексичної багатозначності;
- Parsing – семантичний аналіз.

Однак виявилося, що для багатьох задач з видобутку тексту лінгвістична не є дуже корисною. Причина полягає в тому, що спільна поява термів у векторному поданні сама по собі вирішує лексичну багатозначність. Нещодавно був досягнутий певний прогрес у кластеризації та класифікації тексту завдяки розширенню мішка слів і додаванням до нього лінгвістичних ознак слів.

2.1.3 Класифікація

Класифікація тексту спрямована на присвоєння заздалегідь визначених класів текстовим документам. Прикладом може бути автоматичне позначення кожної новини новинами такою темою, як “спорт”, “політика” чи “мистецтво”. Незалежно від застосовуваного методу, класифікація даних починається з навчального набору $D = (d_1, \ldots, d_n)$ документів, які вже позначені класом $L \in \mathcal{L}$ (наприклад, спорт, політика). Потім завдання полягає у визначенні моделі класифікації

$$f: D \to \mathcal{L} \quad f(d) = L, \tag{2.6}$$

яка зможе присвоїти правильний клас новому документу d.

Для вимірювання ефективності моделі випадкову частку розмічених документів відкладають і не використовують для навчання. Ми можемо класифікувати документи цього тестового набору за моделлю класифікації та

порівняти оцінювані класи із справжніми. Частка правильно класифікованих документів відносно загальної кількості документів називається *точністю*.

Однак часто цільовий клас охоплює лише невелику частку документів. Тоді ми отримуємо високу точність, якщо присвоюємо кожен документ іншому класу. Щоб уникнути цього, часто використовуються різні міри успішності класифікації. *Точність* кількісно визначає частку отриманих документів, які дійсно мають відношення до цільового класу. *Відкликання* вказує, яка частина відповідних документів отримується.

$$\text{точність} = \frac{\#\{\text{відповідні} \cap \text{отримані}\}}{\#\text{отримані}}. \tag{2.7}$$

$$\text{відкликання} = \frac{\#\{\text{відповідні} \cap \text{отримані}\}}{\#\text{відповідні}}. \tag{2.8}$$

Очевидно, що існує компроміс між точністю та відкликанням. Більшість класифікаторів внутрішньо визначають певний “ступінь належності” до цільового класу. F-міра є компромісом між обома для вимірювання загальної ефективності класифікаторів.

$$F = \frac{2}{1/\text{відкликання} + 1/\text{точність}}. \tag{2.9}$$

Існує багато різних типів класифікаторів;

- Класифікатори на базі ключового терму;
- Наївні баєсові класифікатори;
- Класифікатори на базі алгоритму найближчого сусіда;
- Класифікатори на базі дерев рішень;
- Класифікатори на базі методу опорних векторів.

Протягом останніх років класифікатори тексту оцінювались за допомогою спільних масивів тексту. Виявляється, що рівень ефективності, здебільшого залежить від обраного масиву. У табл. 2.1 наведені деякі

репрезентативні результати, досягнуті для колекції груп новин Reuters (менше – краще).

Таблиця 2.1 Відносне порівняння ефективності класифікаторів

Тип	Значення F-міри
Наївний баєсів класифікатор	0.795
Класифікатор на базі дерев рішень	0.794
Класифікатор k найближчих сусідів	0.856
Класифікатор на базі опорних векторів	0.870
Покращене дерево рішень	0.878

2.1.4 Кластеризація

Кластеризація може бути використана для пошуку груп документів зі схожим змістом. Результатом кластеризації зазвичай є розділення , або кластеризація $\mathbb{P}$, та набір кластерів P. Кожен кластер складається з ряду документів d.[19] Кластера документів повинні бути подібними між собою та несхожими на документи інших кластерів. Зазвичай якість кластеризації вважається кращою, якщо вміст документів всередині одного кластера є більш подібним, а між кластерами – більш різним. Методи кластеризації групують документи, враховуючи лише їх розподіл у просторі документів (наприклад, n-мірний простір, якщо ми використовуємо модель векторного простору для текстових документів).

Алгоритми кластеризації обчислюють кластери на основі атрибутів даних та показників їх подібності. Однак уявлення про те, як повинен виглядати ідеальний результат кластеризації, різниться залежно від методу і може навіть відрізнятися у різних людей.

Загалом, є два способи для оцінки результатів кластеризації. З одного боку, статистичні показники можуть бути використані для опису властивостей в результаті кластеризації. З іншого боку, певна класифікація може розглядатися як свого роду стандарт, який потім, використовується для

порівняння результатів кластеризації з цією класифікацією. Найбільш використовуваними мірами для класифікації є:

- Середній квадрат помилки;
- Коефіцієнт силуету – Основною ідеєю коефіцієнта є з'ясування місця розташування документа в просторі щодо кластера документа та наступного подібного кластера;
- Компаративні міри: точність, чистота, F-міра.

Існують групування або кластери, які можуть бути представлені в вигляді ієрархії кластерів. Цю ієрархію можна отримати підходом "зверху вниз" або "знизу вгору". Зверху вниз означає, що ми починаємо з одного кластера, який містить усі документи. Кластеризація отримується шляхом його ітераційного розділення на підкластери. У цьому випадку можна говорити також про так званий "роздільний" алгоритм. Процедури типу "знизу вгору" або агломератні починаються з розгляду кожного документа як окремого кластера. Потім найподібніші кластери ітеративно зливаються, поки всі документи не опиняються в одному єдиному кластері. На практиці процедура розділення майже не використовується через її погані результати.

Агломератний алгоритм[20] розглядає спочатку кожен документ d усього набору документів D як окремий кластер. Це перша кластерізація. Передбачається, що кожен документ є членом рівно одного кластера. Далі визначається подібність між кластерами на основі цієї кластеризації і обирається два кластери p, q кластеризації $\mathbb{P}$ з мінімальною відстанню $dist(p, q)$. Обидва кластери потім об'єднується, і таким чином отримується чергова кластеризація. Алгоритм завершується, якщо залишився лише один кластер. Для ієрархічної кластеризації використовуються наступні алгоритми:

- Алгоритм k-середніх;
- Алгоритм k-середніх з бісекцією;

– Нейронна мережа архітектури самоорганізованої мапи;
– EM-алгоритм.

Також існують альтернативні методи кластеризації тексту.

Алгоритм **ко-кластеризації** позначає одночасну кластеризацію документів та термів. Тим самим вони працюють за іншим принципом, а ніж класичні алгоритми кластеризації, наприклад, k-середніх, який кластеризує лише елементи одного виміру на основі їх подібності до другого, наприклад документи за термінами.

Нечітка кластеризація. У той час як більшість класичних алгоритмів кластеризації призначають кожен елемент класифікації дату лише одному кластеру, утворюючи таким чином чіткий розділ даних, нечітка кластеризація дозволяє отримати ступені приналежності до кластерів.

2.2. Архітектура сучасних генераторів тексту

2.2.1 Архітектури процесу генерації

Високорівневе проектування відіграє вирішальну роль у розробці систем NLG, як і інших типів програмного забезпечення. Для NLG ця задача характеризується різними формами вхідних даних та каналів зв'язку. Сучасні системи використовують модифікації різноманітних архітектурних моделей. Ці моделі розділені на чотири різні кластери. Ця класифікація не означає, що нинішні системи суворо їх дотримуються, але вони використовують ці архітектури як базу та додають нові елементи там, де це доцільно[21].

Послідовна архітектура базується на послідовному потоці інформації через три основні компоненти, як показано на рис. 2.1.

Оскільки більшість попередньо розроблених систем NLG цього дотримувались, вважається, що послідовна архітектура – це консенсусна архітектура, яку можна використовувати для завдань генерації природної мови.

Рисунок 2.1 Послідовна архітектура генератора тексту

Кожний компонент в цій архітектурі виконує досить значну роботу. У табл. 2.2 наведено перелік завдань, класифікованих за трьома основними модулями в послідовній архітектурі.

Таблиця 2.2 Структура послідовної архітектури генераторів тексту

Компонент	Задача генерації	Задача структуризації
Планувальник документу	Визначення цілі генерації	Структуризація документу
Мікропланувальник	Лексикалізація	Агрегація
Реалізація граматики	Реалізації граматики	Реалізація структури

Визначення цілі генерації відповідає за вибір інформації, необхідної для вираження за допомогою сформованого тексту.

Структуризація документу визначає процес утворення структури поданої інформації.

Лексикалізація – це вибір того, які слова, терміни та поняття потрібно включати в текст.

Агрегація може бути виконана для структурування та специфікації структур речень для виділення значущих речень.

Реалізація граматики та **реалізація структури** відповідають за створення вихідного тексту.

Структура потоку, за якою побудована послідовна архітектура, широко використовується завдяки простоті розділу відповідальності за різні компоненти та простоті у визначенні завдань. Незважаючи на те, що ця простота в реалізації розглядається як перевага, є також відомі недоліки.

По-перше, оскільки існує лише односторонній інформаційний потік, результати, що генеруються певним компонентом, ніколи не переглядаються і не уточняються. По-друге, менший зв’язок між різними компонентами може погіршити якість тексту. Наприклад, не передбачений зв'язок між блоком визначення специфікації документу та лексикалізацією. Важливість такого зв'язку полягає у тому, що він може бути використаний для створення краще лексикалізованого тексту через використання контексту при генерації речень. Тим не менше, кожен компонент в послідовній архітектурі повинен генерувати проміжне представлення для того, щоб передати його наступним.

Модель архітектури з перевірками була запропонована для подолання одностороннього потоку взаємодії в послідовних архітектурах. Ця модель широко використовувалася певний час, однак наразі вона використовується лише у поєднанні з іншими архітектурними моделями.

Архітектура з перевірками може бути реалізована у двох формах, як показано на рис. 2.2. У першому підході перегляд – це просто повторення процесу, та кожен компонент викликається з оновленими вхідними даними. Це важко реалізувати на практиці, оскільки багаторазова обробка у повному циклі є обчислювально важкою. Другий підхід полягає у приєднанні окремого модуля ревізії для кожного компонента таким чином що ревізія проводиться негайно в тому самому компоненті. Завдяки простоті, цей підхід широко використовується в різних системах NLG, а також у поєднанні з іншими архітектурами.

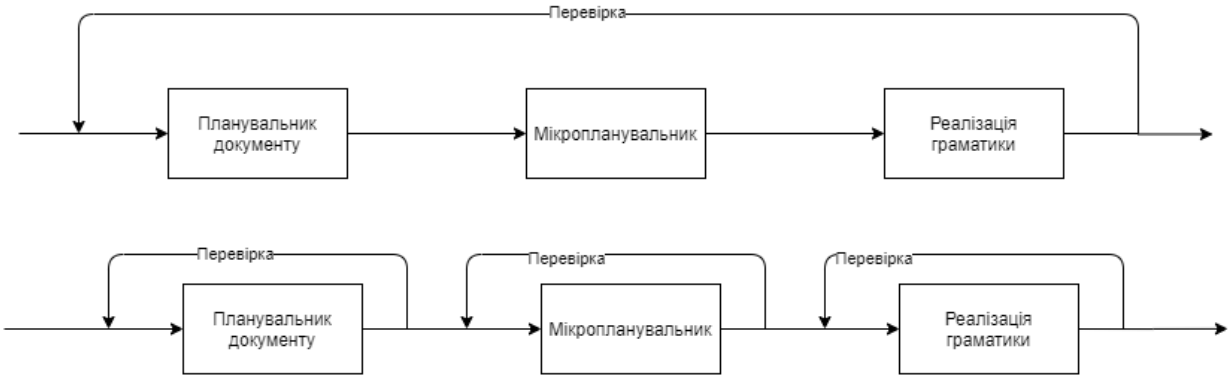


Рисунок 2.2 Архітектура з перевірками

Інтегрована архітектура визначає єдиний механізм для всіх етапів генерації тексту, за допомогою якого можна вирішити проблему розділення аналізу в послідовних архітектурах NLG.

Наразі інтегровані системи генерації набули максимальної популярності через те що нейронні мережі є імплементаціями саме інтегрованої архітектури.

Використання принципів проектування програмного забезпечення є сучасним трендом у проектуванні архітектур NLG. Встановлено, що використання архітектурних моделей програмного забезпечення в NLG є виграшним, оскільки такі моделі розвиваються шляхом вирішення теоретичних та практичних питань.

Архітектура RAGS є результатом практичного втілення подібних міркувань. У RAGS головними є два принципи: можливість використовувати моделі повторно та формальне визначення модулів в архітектурі NLG. Можна помітити, що ці концепції походять від усталених підходів до проектування програмного забезпечення, таких як повторне використання коду та формальні визначення програмних компонентів. Двома важливими елементами RAGS є визначення даних та різні оператори взаємодії, які RAGS представив опублікував як частину архітектури. Послідовна архітектура систем генерації тексту не забезпечує формальної моделі даних, яку можуть

використовувати системи NLG, і цей недолік усувається в RAGS за допомогою чітко визначеної моделі даних. Також послідовна архітектура не розглядає взаємодію між різними модулями як окремий процес. Це ускладнює повторне використання модулів в іншій системі. RAGS вирішила цю проблему, розглядаючи взаємозв'язок як оператор взаємодії, який можна визначити та охарактеризувати.

2.2.2 Визначення складу документа

Визначення змісту, який буде передано, має прямий вплив на результат генерації. Існує чотири аспекти, на яких потрібно зосередитися під час визначення вмісту:

1. Вибір даних на основі значущості.
2. Зведення, або узагальнення даних.
3. Виведення похідної інформації.
4. Додаткова персоналізація під область використання.

Як і в багатьох завданнях NLP, поява машинного навчання та розпізнавання образів у NLG принесли нові досягнення в вирішенні цих питань.

В останніх роботах представлено статистичний підхід, застосований до задачі отримання правил вибору вмісту. Це керований даними підхід, який генерує текст на основі виведених ним самим правил. Основою в системі є метрика, що генерується внаслідок кластеризації. Ця метрика використовується для виявлення співвідношення між вибором певного слова та вихідним текстом. Значення цієї метрики є основою для рішення, чи включати певний набір даних в текст, чи ні. Цей підхід є більш ефективним, ніж просто використання бази знань, яке застосовується у традиційних підходах. Це пов’язано з двома причинами.

По-перше, статистичний аналіз чутливий до даних та їх варіацій у порівнянні з вибором на основі правил із бази знань.

По-друге, цей метод можна легко узагальнити в різних сферах застосування порівняно з підходом, заснованим на знаннях.

Також існує можливість розглядати це як задачу виявлення закономірностей. Інтерпретація цих ідентифікованих закономірностей виконується шляхом асоціації абстракції та конкретних даних, де також можливий взаємозв'язок.

Використання правил у процесі визначення складу також є загальним підходом. Хоча є системи, що частково використовують правила у своїх процесах на основі традиційного підходу до визначення правил, існують системи, які використовують методи на основі правил та механізми пошуку як основну методологію визначення складу.

Також останнім часом технології семантичних Web-технологій стрімко зростали і наразі вони використовуються у багатьох сферах діяльності. Розглядуване питання також виграло від цієї нової тенденції. Кілька дослідників використовували семантичну мережу для задачі визначення складу документів, іноді в поєднанні з раніше обговореними підходами.

2.2.3 Структурування документу

Структурування документу – це процес перетворення отриманого складу в більш структурований формат, завдяки чому інформація може передаватися на наступні рівні для подальшої обробки. Є кілька ключових властивостей, яким повинен відповідати модуль структурування документів:

- групування елементів тексту;
- упорядкування елементів тексту;
- зв'язування елементів тексту.

Враховуючи ці властивості модуля структурування документів, його конкретна реалізація може відрізнятися в залежності від області застосування. Наприклад, реалізація структурування документів з семантичним змістом та реалізація структурування, що застосовується у системі, яка обробляє числові дані, матиме великі відмінності.

Схеми та риторичні відносини – це прийоми структурування, які використовують більшість систем NLG.

Механізм структурування документів схемами натхненний раніше розробленими схемами подання знань, такими як фрейми та сценарії[20].

Більшість раніше розроблених систем використовували матриці значень атрибутів (AVM) для структурування документів. Приклад AVM наведено нижче.

Однак використання необробленої структури AVM згодом було припинено через труднощі із застосуванням її для оперуючих числами систем NLG. На заміну цьому дослідники придумали більш гнучкі структури, які все ще базуються на початковій концепції, введеній з використанням атрибутів та значень.

$$\begin{bmatrix} \text{тип} & \text{Схема документу} \\ \text{діти} & \begin{bmatrix} \text{тип} & \text{Список фактів} \\ \text{відношення} & \text{послідовність викладення} \\ \text{діти} & \begin{bmatrix} \text{тип} & \text{повідомлення про дощ} \\ \text{період} & \begin{bmatrix} \text{місяць} & 11 \\ \text{рік} & 2020 \end{bmatrix} \\ \text{день} & \begin{bmatrix} \text{день} & 16 \\ \text{місяць} & 11 \\ \text{рік} & 2020 \end{bmatrix} \\ \text{дощ} & \text{сильний} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

Відомо, що системам NLG, які обробляють числові дані та абстрактні синтаксичні дерева, важко згенерувати структури для виконання цієї задачі. Системи, які оброблюють більш структуровані джерела даних, можуть визначати окремі елементи тексту простіше. Основна причина простоти

перетворення полягає в тому, що структуровані дані самі по собі представляють схему. Попри це, зазвичай системи генерації тексту не мають насамперед розмічених даних, отже досить часто отримання переваг від таких джерел даних неможливе.

Іншим способом структуризації документів є риторична структура(Rhetorical Structure Theory). На рис. 2.3 зображена основна структура відношення RST з ядром (центральний сегмент тексту або ядро) та сполученим текстом (периферійний сегмент). Окрім цієї базової структури, у RST існують багатоядерні відносини, які не мають центрального сегменту тексту. Наприклад, контраст – це відношення з двома ядрами.

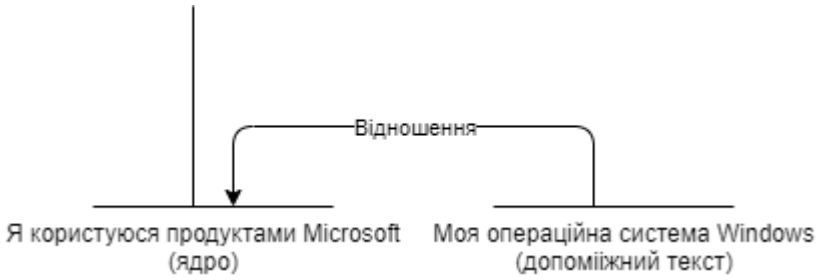


Рисунок 2.3 Структура RST

Ця структура може бути розширена як новими елементами, так і в масштабі генерації. Існують дослідження які адаптували цю систему до структури документа в цілому та збагатили ці відносини додатковою інформацією за допомогою технік отримання інформації, описаних в розділі 2.1.1 .

2.2.4 Лексикалізація

Під час лексикалізації структура документу перетворюється у лексичну структуру. Цей етап є важливим у процесі NLG, оскільки він безпосередньо впливає на те, наскільки “людяним” буде результуючий текст. Лексикалізація може здійснюватися у два різні способи:

– лексикалізація з низькою точністю;

– лексикалізація з високою точністю.

При лексикалізації низькою точністю основна увага приділяється формуванню лексичної структури найпростішим можливим способом. Зазвичай такий тип лексикалізації досягається за допомогою механізму заповнення шаблону, який дозволяє створити лексичну структуру механічно, але деталізація (наприклад, точність перекладу) не може бути гарантована.

Замість того, щоб розглядати лексикалізацію як єдиний процес, лексикалізація з високою точністю проводить більш детальний аналіз. Це досягається за допомогою наступних технік:

– колокація – перевірка, які слова в основному використовуються парами;
– зменшення надмірності видалення непотрібних даних;
– вибір слів – процес підбору слова, яке може максимізувати ефективність донесення мети тексту.

Існує декілька різних методів лексикалізації. Підхід, заснований на правилах – це найбільш використаний тип завдяки його простоті. Основна його перевага, окрім простоти, це те що отримані лексичні структури самі по собі виглядають природньо через те що набір правил зазвичай генерується людиною вручну. З цього виходять головні недоліки цього типу – залежність від області застосування, непереносність на інші області та необхідність роботи людини. Хоча участь людини в свою чергу має свої недоліки окрім необхідності праці. Люди мають різні лексичні вподобання. За участі обмеженого кола людей в генерації правил лексикалізація може бути упередженою щодо лексичних уподобань тих, хто бере участь у оцінюванні.

Міркування на основі конкретних випадків (Case based reasoning) працює у чотири основних етапи:

1. Отримання попередніх прикладів лексикалізації. Якщо в процесі ще не було схожих випадків, можна провести пошук в лексичній базі даних.

2. Повторне використання знайдені лексикалізації (якщо такі є) у новому випадку, зіставляючи цей випадок із минулим обробленим.
3. Оцінка отриманої лексикалізації.
4. Збереження лексикалізації для подальшого використання.

Підхід на основі баз знань є відносно поширеними в більшості розділів обробки природної мови. У контексті NLG, підходи, засновані на базі знань, наприклад, колокація, є поширеними засобами. Це досягнуто завдяки тому, що наразі існує безліч баз знань, які різняться за тематикою, мовою, ступенем обробки. Втім, сучасні підходи, засновані на базі знань, застосовують більш просунуті або поєднані методи лексикалізації, ніж прості колокації.

Застосування статистичних методів у лексикалізації – це також тенденція в останніх дослідженнях, у різних застосунках. Основні виміри тексту, які оптимізуються в таких методах – пізнаваність, доречність та двозначність. Пізнаваність – це ймовірність того, що слово зустрінеться в лексиконі користувача. Ця метрика є значним покращенням у порівнянні з раніше обговорюваною моделлю колокації – яка не враховує це, отже згенеровані тексти можуть виглядати менш природньо. Вимірювання пізнаваності досягається за допомогою розподілу Ципфа, модель якого базується на розподілі ймовірностей появи слова.

Онтологічні підходи також широко використовуються в нещодавно розроблених системах NLG для лексикалізації. Є безпосередні переваги, які можна отримати від залучення збірок під час лексикалізації:

- адаптованість: знайти збірку простіше, ніж готову базу знань для певної області застосування;
- охоплення: на відміну від баз знань, збірка забезпечує більш широке охоплення семантики тексту.

Методи на базі збірок, зазвичай виконують локалізацію через надання семантично схожих елементів тексту, які потім уточняються для отримання

необхідної точності. Наприклад, структурне подання дії “різати” можна лексикалізувати як “рубати”, якщо точність лексикалізації низька, або “нарізати”, якщо точність висока.

2.2.5 Агрегація

Агрегація головним чином відповідає за формування більш структурованих та згрупованих речень. Серед інших компонентів NLG агрегація є відносно менш чітко сформульованою складовою. Є кілька випадків, коли агрегування виконується під час останнього етапу реалізації граматики, але також вона може виконуватися як окремий етап генерації тексту.

Залежно від якості , необхідної для окремого випадку застосування NLG, рівень уваги до агрегації різниться. Вважається, що якісне та точне агрегування впливає на якість отриманого тексту.

Якість агрегування була формально визначена і може бути охарактеризована 4 рівнями. Візьмемо 4 простих речення:

1. Іван – хлопець;
2. Іван – студент;
3. Іван добре грає у шахи;
4. Іван швидко плаває.

На їх прикладі визначимо рівні агрегації:

- Просте сполучення, коли речення пов’язані за допомогою сполучника (Іван – студент і Іван швидко плаває);
- Об’єднання за спільним коренем, де спільний об’єкт використовується для комбінації (Іван – студент і добре грає у шахи);
- Спільна структура (Іван добре грає у шахи і плаває);
- Синтаксичне вбудовування, де речення інтегруються, створюючи цілісну структуру (Іван добре грає у шахи і плаває як для студенту).

Агрегація зазвичай виконується за допомогою алгоритмів машинного навчання, еволюційних алгоритмів або на основі обходу графу.

2.2.6 Реалізація граматики

Реалізація граматики є завданням відображення специфікації тексту синтаксичні структури, які в будуть представлені читачам. Реалізація іноді розглядається як два етапи: структурна та лексична реалізація.

Більшість ранніх застосувань NLG, таких як відображення повідомлення про помилку, були оснащені найпримітивнішим блоком реалізації. Цю реалізацію часто називають заготовленим текстом, і її використання все ще є доцільним для нескладних операцій генерації. Цей підхід пропускає всі попередні етапи и просто надає готовий, написаний раніше текст, за запитом системи прийняття рішень.

Схожим підходом є генерація тексту на основі шаблонів, яка теж ще використовується. Вона надає системі прийняття рішень певний узагальнений текст, який система прийняття рішень заповнює семантичними об’єктами.

В розрізі інтелектуальних систем генерації тексту методи реалізації граматики можна розділити на керовані людиною так керовані даними, як було зазначено в першому розділі.

2.3. Оцінка та тестування моделей генерації тексту

2.3.1 Засоби оцінки за допомогою людини

Незалежно від того, чи система генерує відповідь на запитання користувача, обґрунтування рішення класифікаційної моделі чи коротку історію, кінцевою метою NLG є створення цінного для людей тексту. З цієї причини оцінка людини зазвичай розглядається як кінцева і найважливіша форма оцінки систем NLG і вважається стандартом при розробці нових автоматизованих показників. Оскільки автоматичні показники все ще не

відтворюють людських рішень, багато робіт в галузі NLG включають певну форму оцінки людини.

Внутрішня оцінка або **власна оцінка**(Intrinsic evaluation) – це процес, під час якого дослідник пропонує людям оцінити якість сформованого тексту в цілому або за певним виміром (наприклад, плавність, узгодженість, граматична коректність тощо). Зазвичай це робиться шляхом генерації кількох зразків тексту з моделі та прохання оцінювачів оцінити їх якість.

Майже всі рішення з генерації тексту сьогодні оцінюються за власними оцінками людей. Машинний переклад – одна з задач генерації тексту, в якому справжні людські оцінки зробили величезний внесок у розробку більш надійних і точних систем перекладу, оскільки автоматизовані показники перевіряються шляхом кореляції з людськими судженнями.

Хоча вільність (оцінка властивостей перекладку без порівняння з похідним текстом) і адекватність (оцінка у порівнянні з похідним текстом) стали стандартними вимірами людського оцінювання для машинного перекладу, не всі алгоритми з генерації тексту мають усталений набір вимірів, які використовують дослідники. Тим не менше, є кілька вимірів, які є загальними в оцінках людини для сформованого тексту. Як і для адекватності, багато з цих вимірів фокусуються на змісті сформованого тексту. Фактичність важлива для завдань, які вимагають, щоб сформований текст точно відображав факти, описані в контексті. Наприклад, у таких завданнях, як генерація або узагальнення даних до тексту, інформація у вихідних даних не повинна суперечити інформації в таблиці вхідних даних або статті новин. Навіть якщо немає чіткого набору фактів, яких слід дотримуватися, дослідники можуть знати, наскільки сформований текст відповідає правилам здорового глузду або наскільки він логічний. Для завдань із генерації, які передбачають доповнення тексту, дослідники можуть попросити оцінювачів оцінити узгодженість тексту – наскільки він відповідає наданому контексту.

Зовнішня оцінка(Extrinsic evaluation) – це процес, під час якого люди оцінюють ефективність системи з погляду завдання, для вирішення якого вона була розроблена. Зовнішні оцінки є найбільш значущими оцінками, оскільки вони показують, як система фактично виконує завдання, але вони також є більш дорогими та складними у виконанні. З цієї причини внутрішні оцінки є більш поширеними, ніж зовнішні оцінки і стають дедалі популярнішими.

Зовнішні методи вимірюють, наскільки успішною є система в подальшому завданні. Цю успішність можна виміряти з двох різних точок зору: досягнення користувачем його мети у виконанні завдання та успіх системи у виконанні своєї мети.

Зовнішні оцінки людей зазвичай використовуються для оцінки ефективності систем, що генерують репліки діалогу. Для вимірювання продуктивності системи під час спілкування з людьми використовуються різні підходи, наприклад, вимірювання тривалості розмови або прохання людей оцінити систему.

Для багатьох завдань з оцінки NLG від оцінювачів не потрібні спеціальні знання, крім володіння мовою сформованого тексту. Це значно полегшує оцінювання згенерованих текстів коли метрикою успіху є “вільність”. Часто цільова аудиторія системи NLG є широкою, наприклад, читачі новин, або користувачі чат-ботів. У цих випадках оцінки на людях виграють від того, що їх проводять якомога ширшими.

Не всі завдання з оцінки NLG може виконувати будь-яка підмножина мовців певної мови. Спеціалізовані групи оцінювачів можуть бути корисними при тестуванні системи для певної категорії людей. Дослідники можуть набирати людей, які можуть бути потенційними користувачами системи, наприклад, студентів для освітніх інструментів або лікарів для систем генерації проскрипцій. Інші випадки, які можуть вимагати більш

спеціалізованої оцінки людей, – це проекти, де експертиза оцінювача є важливою для виконання завдання, або коли похідні тексти або створені тексти складаються з довгих документів або колекції документів.

Хоча оцінювачі часто проходять навчання для формалізації своїх оцінок, оцінка сформованої природної мови завжди включатиме певний ступінь суб'єктивності. Оцінювачі можуть не погодитися в своїх рейтингах, і рівень розбіжностей може бути корисним показником для дослідників. Високий рівень згоди між оцінювачами, як правило, означає, що завдання є чітко визначеним, а відмінності в сформованому тексті постійно помітні оцінювачам, тоді як низька згода може свідчити про погано визначене завдання або про відсутність достовірних відмінностей у сформованому тексті.

Для вимірювання цих розбіжностей (або збігів) існують кілька статистичних показників:

– частка випадків, у яких оцінювальники погодилися;
– Каппа($\varkappa$) Когена – частка погоджень, враховуючи вірогідність того, що пара оцінювальників погодилися випадково;
– Каппа Флейсса – частка погоджень, враховуючи вірогідність того, що пара оцінювальників погодилися випадково, враховуючи всі можливі пари оцінювальників;
– Альфа Кріппендорффа – міра схожа на каппа-подібні, проте вона формується в термінах непогодження та зважується в залежності від частки непогоджень.

2.3.2 Нетреновані метрики оцінювання

Попри те, що людське оцінювання систем NLG визнано найефективнішим, людська оцінка є дорогою та трудомісткою для проведення, і що більш важливо, результати не завжди можна повторити.

Таким чином, автоматичні показники оцінки використовуються як альтернатива як для розробки нових моделей, так і для порівняння їх із існуючими.

Метрики співпадіння n-грамів зазвичай використовуються для оцінки систем NLG та вимірювання ступеня “збігу” між машинними та авторськими текстами.

F-міра – Статистичний показник, який було розглянуто у розділі 2.1.3 . Зазначимо, що в контексті співпадіння n-грамів, точність – це частка грамів в згенерованому тексті, які є і в цільовому тексті. Відкликання – частка n-грамів в цільовому тексті, які є в згенерованому.

BLEU – метрика, яка була розроблена для оцінювання машинних перекладів, заснована на зваженому середньому оцінок точності, визначеному як

$$prec_n = \frac{\sum_s \min\left(c(s,\hat{y}), c(s,y)\right)}{\sum_s c(s,\hat{y})}, \tag{2.10}$$

де $\hat{y}$ це закодована згенерована послідовність, y – це закодовна похідна послідовність, s – послідовність n-грамів в $\hat{y}$, і c – функція підрахунку. Міра BLEU визначається наступним чином:

$$BLEU = BP \cdot \exp\left(\sum_{n=1}^{N} w_n \log prec_n\right), \tag{2.11}$$

де BP – це штраф для закоротких послідовностей.

NIST – це міра схожості текстів, схожа на BLEU, проте вона зважує найбільше менш часті співпадіння n-грамів, керуючись припущенням, що такі співпадіння є більш інформативними. Це виконується за допомогою показника $Info$ який рахується як логарифм співпадінь

$$Info(w_1, \dots, w_n) = log_n(\frac{c(w_1 \dots w_{k-1})}{c(w_1 \dots w_n)}), \tag{2.12}$$

де w_n – n-грам в цільовій послідовності. Далі схожість розраховується схожим з BLEU чином.

ROUGE це набір метрик для оцінювання згенерованих узагальнень текстів. Узагальнено, метрика розраховується наступним чином:

$$ROUGE \cdot N = \frac{\sum_r \sum_n match(gram_{n,r})}{\sum_r \sum_s count(gram_n)}, \quad (2.13)$$

де N – це зменшена на одиницю кількість слів у триграмах, перетин яких досліджується. Таким чином, існують метрики ROUGE-1, ROUGE-2, ROUGE-3, ROUGE-4, ROUGE-L(збіжність найдовших можливих підстрок).

METEOR – це метрика, яка базується не лише на точності, а й на відкликанні у співставленні вихідного та згенерованого текстів. Менш використовуваними є метрики **HLEPOR**, **RIBES** та **CIDEr**.

Також використовуваними метрики на основі відстані:

WER – це кількість слів, яку не обхідно додати, видалити або змінити для того, щоб згенерований текст став однаковим з примірником, або edit distance між примірником та результатом.

MED – більш тонка міра, яка вираховує відстань редагування між реченнями, враховуючи окремі літери.

TER – метрика, характерна для оцінювання перекладів, яка визначає нормалізовану за довжиною кількість виправлень, яку необхідно ввести для того щоб примірник та машинний переклад співпадали.

Метрики відповідності семантичності вмісту визначають подібність між написаним людиною та створеним моделями текстів, витягуючи явні семантичні інформаційні одиниці з тексту окремо від n-грамів. Ці показники діють на семантичному та концептуальному рівнях і, як визначено, добре корелюють із судженнями людини. Відомі наступні показники: **PYRAMID**, **SPICE**, **SPIDER**.

2.3.3 Треновані міри оцінювання

Багато нетренованих метрик оцінювання, описаних вище, припускають, що сформований текст має такий зразок, який є достовірним. І з яким згенерований текст має збігатися. Однак це припущення не є адекватним багатьом практичним застосуванням NLG де коректних відповідей може бути безліч.

Одним із вирішень цієї проблеми є використання метрик на основі вбудовування, які вимірюють семантичну подібність. Але методи, засновані на вбудовуванні, не працюють у ситуаціях, коли згенерований результат семантично відрізняється від зразка, наприклад, при генерації реплік діалогу. У цих випадках ми можемо побудувати моделі машинного навчання, та спробувати натренувати їх для того щоб вони імітували людей та надавали оцінки, розглянуті у розділі 2.3.1.

Підходи на базі машинного навчання для репрезентації речень прагнуть закодувати семантичний та синтаксичний зміст речень з різних точок зору та тем, а також відобразити ці речення на векторі за допомогою моделей глибокого навчання. Як і при вбудовуванні слів, моделі NLG можуть бути оцінені шляхом кодування кожного речення у згенерованому та зразковому текстах.

Висновки до розділу

В даному розділі розглянуті основні елементи архітектури генераторів тексту, підходи до їх побудови та оцінювання.

Обробка тексту традиційно є складним завданням для алгоритмічного розв’язання, та історично детерміновані засоби відомі тим що є досить обмеженими у використанні, потребують великого обсягу роботи з підготовки даних та самі дані мають бути в специфічних формах. З розвитком галузі обробки тексту були виділені певні її етапи, як кластеризація,

сегментування, кодування, токенизація, лематизація та інші, більш специфічні фази з перетворення тексту на структуровану інформацію.

Кожен з цих етапів пройшов великий шлях розвитку засобів його втілення, і стохастичні алгоритми наразі домінують в усіх цих задачах через те що стохастичні засоби є автономними та їх легко використовувати на різних даних.

Самі генератори текстів також розвивалися тривалий час і пройшли шлях від примітивних систем, що заповнюють проміжки у тексті до складних та розумних систем, які здатні генерувати унікальний, новітній та логічний текст.

В розділі були розглянуті тенденції розвитку архітектур як систем обробки природної мови взагалі, так і систем генерації тексту. Головним трендом є інтеграція усіх компонентів та етапів в одній моделі машинного навчання. Це дозволяє отримати по-справжньому зручні системи: вони не вимагають попередньої обробки або іншої підготовки та генерують готові тексти, параметри яких можна конфігурувати.

Одним із найскладніших питань генерації тексту є оцінювання. Найбільш старим та досі найякіснішим засобом є оцінка людиною, проте цей метод має значні недоліки та не завжди є адекватним з точки зору обсягу роботи. Це спонукало дослідників розробити низку метрик, які націлені на оцінку генерацій за допомогою статистики, від примітивних ознак, до більш складних похідних ознак. Перспективним напрямком досліджень також є використання нейронних мереж для імітації поведінки людини в оцінюванні.

3. РОЗРОБКА АЛГОРИТМІЧНОГО ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1. **Огляд архітектури проекту**

Практична реалізація проекту магістерської дисертації передбачає розробку класичного Веб-додатку з трьохрівневою структурою. Орієнтація застосунку передбачає свої, унікальні технічні виклики, але концептуально кінцева архітектура може вважатися класичною, тому що проблеми високого навантаження не є характерними для додатків, які використовують готові математичні моделі.

Традиційним способом створення такого роду додатків є проектування монолітної серверної системи з інтегрованим сервером БД, та розташування готового рішення на сервері. Проте, трендом останнього часу є використання хмарних рішень для розміщення та підтримки програмних рішень.

Хмарні обчислення – це парадигма обчислень, при яких усі обчислювальні потужності керуються стороннім постачальником, і в якому програміст керує лише певними, віртуалізованими одиницями обчислення. Так, наразі потужність хмарних серверів обчислюється у МГц процесорної потужності та обсязі пам’яті, без уточнення будь-яких деталей надання цих потужностей. Розробника не цікавлять кінцеві характеристики комп’ютера, на якому будуть виконуватися обчислення, або кількість цих комп’ютерів у кластері. Аналогічним чином, потужність серверу БД обчислюється у RWCU(Read/Write computing unit) а не в характеристиках безпосереднього кластера серверів БД.

Хмарні обчислення надають безперечні переваги у зручності використання та безпеці розміщення обчислювальних ресурсів. Відомі й значні недоліки цих систем: їх вартість та залежність від постачальника хмарних рішень. Ці недоліки є неподоланими для деяких компаній, які

змушені розміщувати свої потужності в своїх дата-центрах, проте для розробки прототипу реалізації певного рішення використання хмарних ресурсів є вдалим архітектурним рішенням.

Останніми роками, отримав популярності підхід безсерверних обчислень, який вважається еволюцією хмарного обчислення. Цей підхід передбачає декомпозицію цього додатку на частини, такі як авторизація, керування навантаженням, бізнес-логіку та використання для їх реалізації абстрактних хмарних сервісів. У цьому випадку обчислювальні потужності надаються навіть не у формі певного абстрактного кластеру, а повністю децентралізовані та абстраговані.

Цей підхід вважається розвитком хмарних обчислень через те що він ще більше акцентує переваги та недоліки віртуалізації обчислювальних ресурсів. Безсерверні архітектури дуже просто реалізувати через те що вони приховують певні системні рішення, та такі архітектури масштабуються потенційно безкінечно та є дуже надійними. Недоліки таких систем ще більш поглиблені: безсерверні архітектури є менш ефективними у рази та десятки разів ніж архітектури на власних серверах, та такі архітектури зазвичай неможливо експортувати від постачальника хмарних рішень.

Для розробки практичної реалізації генератора тексту цікавим є інша характеристика безсерверних архітектур: вони є гнучкими у керуванні ресурсами та легко адаптуються під нерівномірне навантаження, на відміну від серверних архітектур, у яких ресурси мають бути виділені заздалегідь та є фіксованими.

Для реалізації додатку використовується сервіс AWS (Amazon Web Service). Архітектуру проекту наведено у рис. 3.1. Розглянемо окремі частини додатку. В Amazon Simple Storage Service (S3) зберігаються файли веб-клієнту, доступ к яким керується за допомогою Amazon CloudFront.

S3 – це розподілене сховище файлів, і у поєднані з CloudFront можна досягти реплікації файлів та його географічне розподілення, що робить систему швидкою та надійною.

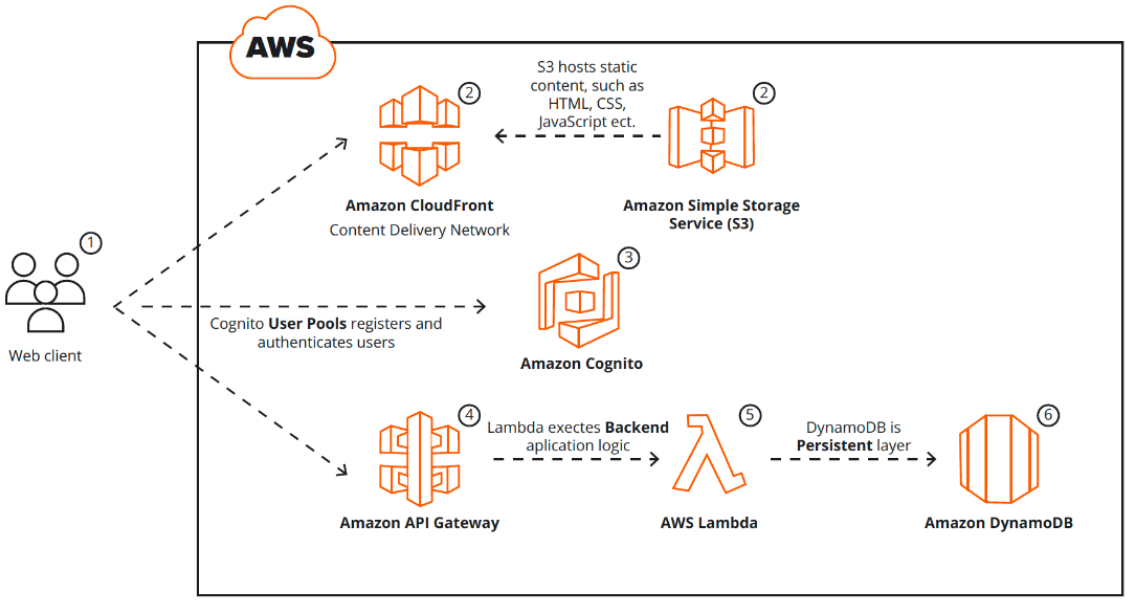


Рисунок 3.1 Архітектура безсерверного Web-додатку

Оскільки система є вразливою до високих навантажень, необхідно забезпечити обмеження кількості користувачів що можуть використовувати додаток. Це класична проблема аутентифікації користувача, і в нашій архітектурі для її вирішення виділено сервіс Amazon Cognito. Це універсальне рішення для надання доступу користувачам, яке може авторизувати їх як і внутрішньо, так і за допомогою інших сервісів (Google або Facebook). Це забезпечує авторизацію без стану: користувач отримує токен, у якому зашифровані його привілеї, дата авторизації та дата зупинення придатності, отже сама сесія зберігається також у токені.

Логіка додатку реалізується за допомогою AWS Lambda – сервісу середовища виконання (runtime), яке дозволяє виконувати код на усіх популярних мовах програмування та оброблювати запити. Ця платформа

підтримує багато ініціаторів запиту (поява нових файлів у сховищі або певний розклад) та має багато інтеграцій з іншими сервісами.

Дані будуть зберігатися у Amazon DynamoDB – базі даних, яка підтримує декілька парадигм проектування БД (SQL та NoSQL) та декілька реалізацій (таблиці, key-value, стовпчикові БД та графові БД).

Керувати навантаженням буде Amazon API Gateway. Це є елементом, який поєднує усі вищеописані сервіси у єдиний програмний інтерфейс та керує маршрутизацією усіх запитів як у самому додатку, так і зовнішніх.

3.2. Проектування та тестування генератору текст

У розділі 2 були розглянуті існуючі архітектури генераторів тексту, їх переваги та проблеми. Невідомо, чи можливо подолати ці проблеми в існуючих рішеннях, проте магістерська дисертація пропонує використати гібридні архітектури як засіб керування балансом між цими перевагами та недоліками. Магістерська дисертація досліджує архітектури на базі нейронних мереж, і вже існують дослідження, які використовують гібридні моделі на базі нейронних мереж для генерації тексту. В даній роботі пропонується гібридна модель з двох нейронних мереж (рис. 5).

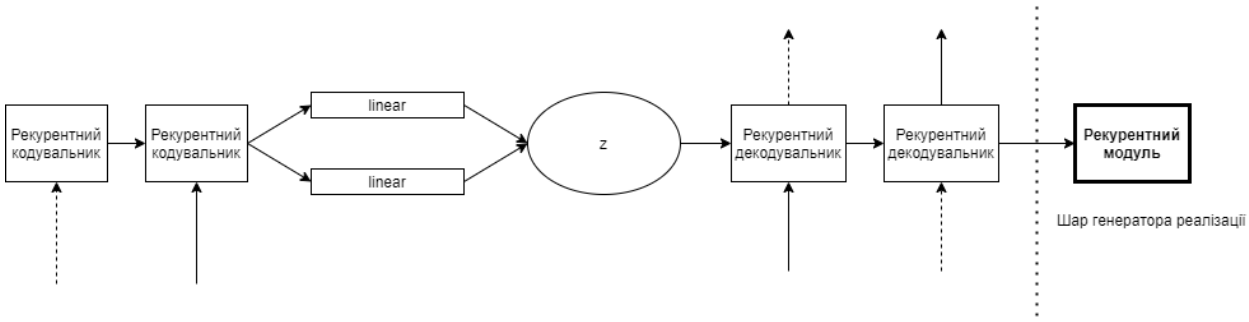


Рисунок 3.2 Гібридна модель генератору тексту

Модель містить дві нейронні мережі: варіаційний автокодувальник як шар планування змісту та рекурентну нейронну мережу як шар планування

складу та поверхневої реалізації. Для реалізації моделі використовується бібліотека PyTorch. Компоненти мережі надаються наступним чином:

```
encoder = nn.Sequential(
      nn.Conv2d(3, 32, kernel_size=4, stride=2),
      nn.ReLU(),
      nn.Conv2d(32, 64, kernel_size=4, stride=2),
      nn.ReLU(),
      nn.Conv2d(64, 128, kernel_size=4, stride=2),
      nn.ReLU(),
      nn.Conv2d(128, 256, kernel_size=4, stride=2),
      nn.ReLU(),
      Flatten()
)
decoder = nn.Sequential(
      UnFlatten(),
      nn.ConvTranspose2d(3, 128, kernel_size=5, stride=2),
      nn.ReLU(),
      nn.ConvTranspose2d(128, 64, kernel_size=5, stride=2),
      nn.ReLU(),
      nn.ConvTranspose2d(64, 32, kernel_size=6, stride=2),
      nn.ReLU(),
      nn.ConvTranspose2d(32, 3, kernel_size=6, stride=2),
      nn.Sigmoid(),
)
Lstm = nn.Sequential(
      nn.LSTM(204, lstm_size),
      nn.Linear(lstm_size, n_classes),
      SoftMax(),
)
```

Для тренування використовується база повідомлень на англійській мові. База була отримана з бази даних соціальної мережі Twitter та п’єс Шекспіра. Повідомлення були токенізовані та векторизовані. Базу знань було поділено на базу для тренування та валідації.

Для оптимізації використовується метрика розходження Кульбака — Лейблера. На її основі будуються нормалізований та денормалізований терми. В якості алгоритму оптимізації використовується ADAM, Для попередження

дефектів тренування використовується нормалізація шарів мереж через випадіння (Dropout layer), нормалізація пакетів навчання та використання зменшуючихся термів навчання.

Тренування моделі проходило на різних джерелах даних, різних розмірах тексту та різних значеннях параметру α. α – це гіперпараметр нормалізації, який відповідає за те, як сильно друга частина моделі (рекурентна мережа) впливає на процес навчання.

$$f_{\text{втрат}}(P, Q) = D_{KL}(P \mid Q) + \alpha \cdot H(P|Q).$$

$D_{KL}(P \mid Q)$ – розходження Кульбака — Лейблера, функція втрат варіаційного автокодувальника.

$H(P|Q) -$ перехресна ентропія, функція втрат рекурсивного генерувалника.

Рисунок 3.3 Графік функції втрат варіаційного автокодувальника $(S = 20, \alpha = 0.2, N = 400, n = 20)$

При α=0 другий компонент гібридної моделі не приймає ролі в процесі генерації, при α=1 другий компонент впливає на процес повною мірою.

Іншими параметрами, які використовуються при тренуванні є розмір частки тексту S , кількість ітерацій тренування N, кількість пакетів навчання n.

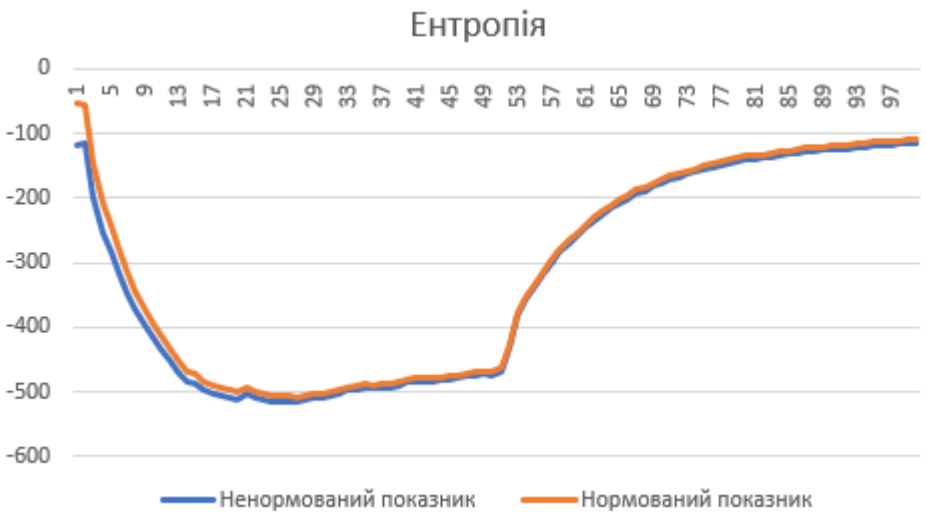


Рисунок 3.4 Графік функції втрат рекурентного генератора ($S = 20, \alpha = 0.2, N = 400, n = 20$)

Час тренування склав одну годину. Можна побачити характерну ознаку генераторів тексту на базі варіаційного автокодувальника – функція їх втрат не збігається.

Порівняємо характеристики тренування с класичним варіаційним автокодувальника. Код реалізації нейронної мережі наведено нижче:

```
encoder_decoder = nn.Sequential(
   OneHot(n_classes),
   LayoutRNNToCNN(),
   Convolution1d(3, 128, n_classes, pad=1, stride=2, causal=False),
   BatchNormalization(128),
   ReLU(),
   Convolution1d(3, 256, 128, pad=1, stride=2, causal=False),
   BatchNormalization(256),
   ReLU(),
   Flatten(),
   Linear(sample_size // 4 * 256, z * 2),
   Sampler(z),
   Linear(z, sample_size // 4 * 256),
   ReLU(),
```

```
    Reshape((-1, 256, sample_size // 4, 1)),
    Deconvolution1D(256, 128, 3, pad=1, stride=2),
    BatchNormalization(128),
    ReLU(),
    Deconvolution1D(128, 200, 3, pad=1, stride=2),
    BatchNormalization(200, name="deconv_bn1"),
    ReLU(),
    LayoutCNNToRNN(),
    Linear(200, n_classes, name="classifier"),
    SoftMax()
)
```

Характеристики тренування залишаються такими самими (окрім гіперпараметру α, оскільки він не має сенсу в контексті тренування).

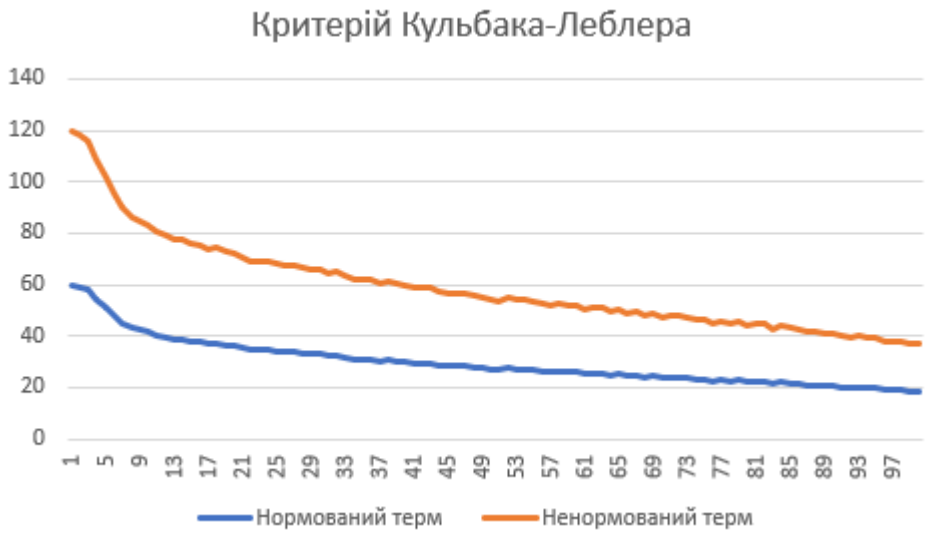


Рисунок 3.5 Функція втрат “звичайного” генератору тексту на базі кодувальника

Можна побачити, функція втрат збігається значно стабільніше, проте модель тренується довше: дві години. Це визвано тим що класична модель є значно складнішою для того щоб отримати схожі показники.

Для порівняння результатів генерації тексту, як було зазначено у другому розділі, існують чисельні та якісні показники. Одним із відомих чисельних показників є метрика порівняння n-грамів BLEU. Методика вимірювання полягає у наступному: формується початок речення, потім

генерується його решта та порівнюється з наявним в тексті продовженням. Чим більше співпадінь серед н-грамів поміж реченнями, тим вище метрика. В якості еталону використовуються продовження цих речень.

База даних з п’єсами Шекспіра є гарним джерелом даних тому що для генерації текстів на її основі доступні результати генерації.

Таблиця 3.1 Порівняння методів генерації тексту

Метод	BLEU-2	BLEU-3	BLEU-4
MLE	0.796	0.695	0.635
SeqGAN	0.887	0.842	0.815
RankGAN	0.914	0.878	0.856
Класична модель	0.704	0.655	0.551
Гібрид	0.658	0.621	0.451

Як можна побачити, запропонована модель залишається позаду лідерів, тому що лідери – це математичні моделі з мільярдами параметрів, які були оптимізовані під ці задачі.

Напроти, модель не дуже відстає від свого класичного співбрату, що доказує придатність моделі для генерації тексту.

Традиційно, генерація тексту в таких моделях проходить двома різними способами:

– Генерується вектор з розмірністю внутрішнього, найменшого шару автокодувальника, потім він декодується. Вважається, що окремі елементи цього вектору відповідають за різні характеристики тексту;
– Виконується рекурсивна генерація тексту, коли рекурсивна мережа отримує не тільки попередній результат, а й результати до них. Мережа генерує вірогідність появи різних символів.

Код генерації наведено нижче:

```
 input_eval = [char2idx[s] for s in start_string]
input_eval = tf.expand_dims(input_eval, 0)

text_generated = []
```

```
    temperature = 1.0

    model.reset_states()
    for i in range(num_generate):
        predictions = model(input_eval)
        predictions = tf.squeeze(predictions, 0)

        predictions = predictions / temperature
        predicted_id = tf.random.categorical(predictions, num_samples=1)[-
1,0].numpy()

        input_eval = tf.expand_dims([predicted_id], 0)

        text_generated.append(idx2char[predicted_id])

    return (start_string + ''.join(text_generated))
```

Зазначимо існування певного псевдопараметру t, або “температури” генерації. Зазвичай, чим вище значення цього параметру, тим більш випадкові результати.

Таблиця 3.2 Результати генерації текстів з п’єс Шекспіра

Початок речення	База для порівняння	Згенерований текст
I sin in…	*I sin in envying his nobility.*	*I sin in request!* *I sin in office to lead.* *I sin in me.*
Then his good report…	*Then his good report should have been my son*	*Then his good report I nivery!* *Then his good report should slpeece aughora*

Таблиця 3.3 Результати генерації повідомлень мережі Twitter за допомогою першого способу генерації

Довжина до 15 символів	Довжина до 50 символів
@ui @ui @ui @ui @ui	*@ui you should come to my house tomorrow*
@ui All the best!!	*How crazy is better in Notion take any credit*
@ui Thanks	*At some point it is a great but the culture*

3.3. Приклад роботи системи

Розглянемо приклад роботи гібридного генератору тексту на основі текстів роботехнічної тематики. Дана галузь може вважатися складною за кількома показниками: складністю самих текстів та складністю формування баз знань для тренування моделей.

Під час проектування розглядалася можливість розробити системи видобутку тексту, проте у відкритому доступі наявний датасет з великою кількістю наукових статей у тому числі з роботехнічної тематики.

Рисунок 3.6 Графік функції втрат під час навчання моделі

В цій базі знань наявні понад 1.7 мільйонів наукових статей на англійській мові, довжиною від 3 до 21 сторінки. Більшість статей мають довжину 4-6 сторінок.

Для тренування ця база знань була розділена звичним чином, довжина послідовності слів було збільшено.

Методика тренування, кількість ітерацій та значення гіперпараметрів обрано таким чином, щоб згенерований текст не втрачав цільності при великій довжині рядків.

Рисунок 3.7 Графік функції втрат під час навчання моделі

Приклади роботи моделі наведені у табл. 3.4.

Таблиця 3.4 Результати генерації тексту на роботехнічну тематику

Початок	Генерований результат
A mobile manipulator is a robot system…	*A mobile manipulator is a robot system programmed to activate the robot by building an injection* *A mobile manipulator is a robot system created as well along with software which can be programmed into the robots*
Balancing robots generally use a gyroscope…	*Balancing robots generally use a gyroscope normally very rigid — about 20cm in length* *Balancing robots generally use a gyroscope as a human skeleton is scaled up*

Під час експериментів було винайдено що на більш довгих послідовностях гібридна модель значно краще зберігає логічну та лексичну структуру речень ніж класичні рекурентні генератори тексту.

Такі генератори (усіх послідовностей) мають властивість швидко деградувати зі зростанням довжини послідовності, яку треба згенерувати. У цьому сенсі запропоновану модель можна використовувати як базу для подальших досліджень у генерації статей на різні тематики.

3.4. **Розробка серверної частини проекту**

Серверна частина проекту розроблюється в середовищі AWS, що позбавляє необхідності розробки проекту за допомогою традиційних бібліотек. Це означає, що код нейронної мережі розміщується в середовищі runtime самої функції. Середовище функцій AWS може бути розширене на необхідні бібліотеки, інші файли можуть бути додані до середовища. Заздалегідь тренована модель розміщується як ресурс окремої функції AWS Lambda.

Логічна частина серверу містить лише одну дію – безпосередню генерацію тексту. Попри те що це логічно одна дія, генерація тексту є вимогливою для ресурсів задачею та самий запит ініціює низку інших мережевих викликів, що призводить до великого часу виконання.

Проблеми довгих запитів у питанні аналізу даних є відомими та існує стандартний метод організації запиту, розроблений Microsoft – asynchronous pattern. Цей підхід полягає у наступному: при ініціації процесу генерації тексту клієнт робить запит на сервер, сервер, перевіривши особистість користувача, ініціює створення нової комірки у сховищі файлів та повертає результат з кодом статусу 201 та адресою цієї комірки. Клієнт, отримавши цю адресу починає з деякою періодичністю робити виклики цієї комірки, в яку сервер, в залежності від реалізації, може розміщувати інформацію про стан виконання, або лише флаг готовності результату. При закінченні роботи, сервер розміщує результат (згенерований текст) у комірку, яку згодом отримає клієнт. Схему роботи цього паттерну наведено у рис. 3.8.

Самий процес генерації тексту передбачає певний цикл генерації, який було розглянуто у попередньому розділі.

Веб додаток не зберігає велику кількість даних, проте він зберігає пул користувачів, їхні сесії та спроби генерації тексту. Цей список зберігається у

внутрішньому середовищі Amazon Cognito, специфікується стандартом OpenId та має наступну структуру:

Через те що внутрішнє сховище Amazon Cognito не є реляційною базою даних, структура сутності надається я виді схеми, специфікованої стандартом JSON Schema.

Таблиця 3.5 Структура сутності “користувач” в сервісі AWS Cognito

Рядок	Тип	Опис
sub	string	Ідентифікатор користувача
name	string	Повне ім’я користувача, яке може містити його посаду або звернення
given_name	string	Особисте ім’я користувача
family_name	string	Фамілія користувача
middle_name	string	Ім’я по-батькові або середнє ім1я користувача
nickname	string	Нікнейм користувача, який він обрав собі
preferred_username	string	Скорочений ідентифікатор користувача
profile	string	Посилання на сторінку з детальною інформацією користувача
picture	string	Посилання на зображення користувача
website	string	Посилання на веб-сайт користувача
email	string	Електронна адреса користувача
email_verified	boolean	Чи перевірена адреса користувача
gender	string	Стать користувача
birthdate	string	Дата народження користувача, в форматі ISO
zoneinfo	string	Часова зона користувача
locale	string	Мова користувача
phone_number	string	Телефонний номер користувача
phone_number_verified	boolean	Чи перевірений телефонний номер користувача
address	JSON object	Адреса користувача
updated_at	number	Дата останнього оновлення даних користувача

У сховищі S3, окрім комірок з веб-клієнтом додатку, містяться комірки з проміжними результатами виконання запиту на генерацію тексту, тренована модель, а також службові файли, необхідні для забезпечення функціонування додатку.

Доступ до цього сховища надає сервіс Amazon CloudFront – мережа швидкої доставки ресурсів (CDN), яка може ефективно надавати будь-які дані у будь-якому форматі користувачам, розподіленим географічно з низькою затримкою та високою швидкістю передачі.

CloudFront інтегрований з іншими системами AWS через вбудовані можливості та програмні інтерфейси. CloudFront працює із AWS Shield для захисту від кібер-атак, сховищами, балансувальниками навантаження тощо.

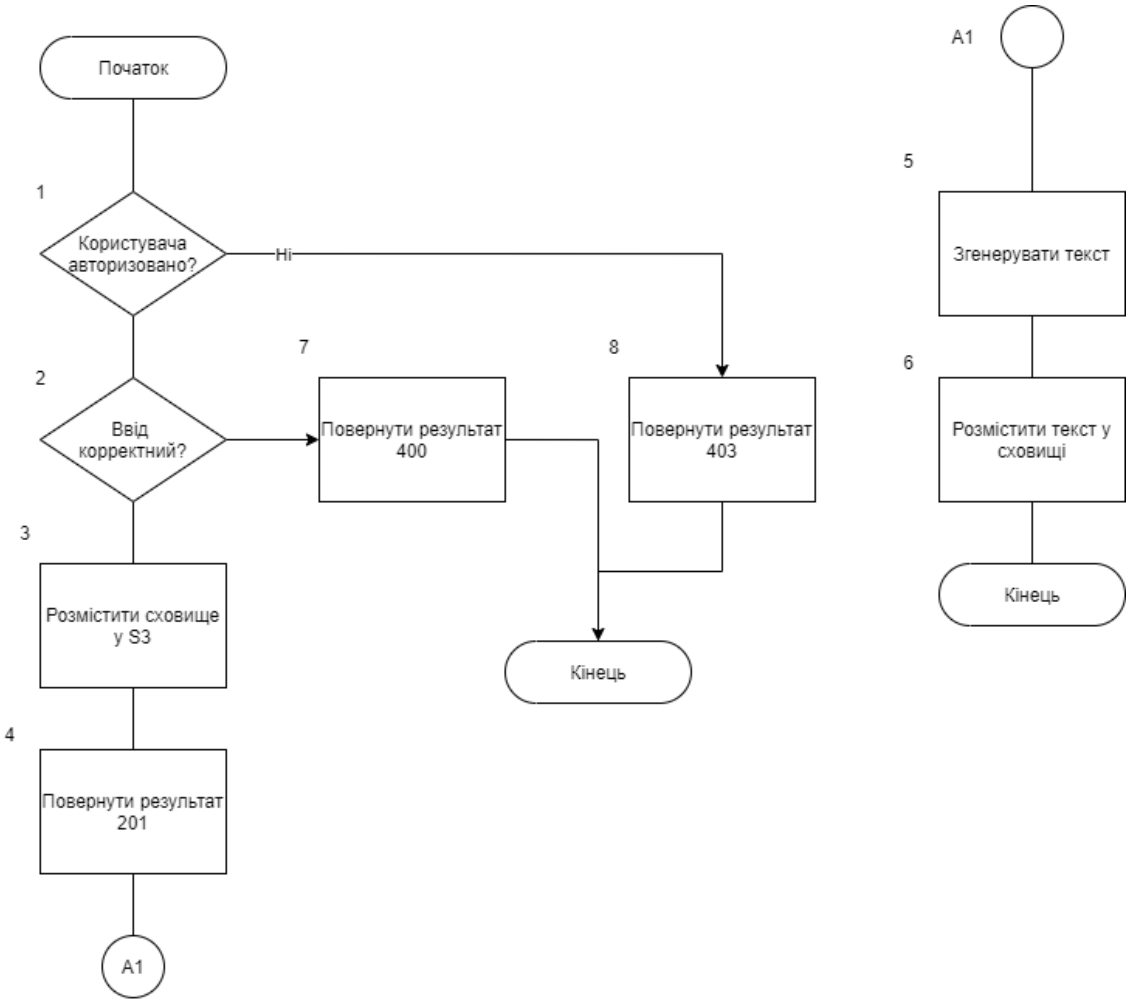


Рисунок 3.8 Асинхронний паттерн генерації тексту

Amazon DynamoDB – це база даних з декількома парадигмами збереження даних, яка може функціонувати як сховище “ключ-значення” та як документна база даних Це повністю керована багаторегіональна розподілена довговічна база даних із вбудованою системою безпеки, резервного копіювання та відновлення та кешування.

Ця база даних є noSQL базою даних що надає свої переваги, проте має суттєві недоліки. Найбільші недоліки таких баз даних:

- Відсутність схеми. База даних не має визначеної структури певної таблиці, тому для досягнення характеристик близьких до реляційних баз даних у швидкості, кожний запис має зберігати схему разом з даними, що призводить до значного зростання займаного простору.
- Відсутність реляційних відношень. Такі бази даних не надають жодних гарантій унікальності, необхідності або зв’язку рядків. Через це кожна окрема база даних реалізує ці гарантії по-різному, або не реалізує що призводить до проблем за надійністю таких баз даних.
- Відсутність гарантій ACID. ACID – це набір вимог до запису/читання у базі даних, які забезпечують однорідність даних у кожний проміжок часу та гарантують надійність транзакцій. Більшість NoSQL баз даних не відповідає частині або всім цим вимогам, що також потребує уваги від розробника.

Зазначимо що ці недоліки не є суттєвими для додатку, що реалізує запропонований генератор тексту та ці недоліки є прямим наслідком переваг таких баз даних, як: швидкість запису та читання, масштабування, надійність бази даних в цілому.

Вибір саме DynamoDB є вдалим рішенням тому що до даних, які зберігаються у цій базі, немає вимог, які втілюють SQL бази даних, отже, адміністрування і розвиток такої бази даних є простими задачами, які не потребують кваліфікації від розробника.

База даних зберігає журнал запитів, у форматі, специфікованому стандартом JSON Schema. Структура запису наведена нижче.

Таблиця 3.6 Опис структури журналу викликів генератора тексту

Рядок	Тип	Опис
`user`	`string`	Ідентифікатор користувача у Amazon Cognito
`input`	`object`	Стан форми вводу користувача, який містить початок фрази та необхідну кількість символів до кінця вводу
`date_start`	`string`	Дата початку обробки запиту
`date_end`	`string`	Дата скінчення обробки запиту
`result`	`string`	Результат генерації тексту
`url`	`string`	Посилання на результат генерації тексту у сховищі

Таку форму даних можна вважати денормалізованою, тобто, усі пов’язані дані зберігаються разом що призводить до їх дуплікації. Проте це є прийнятним для задач, описаних в цій дисертації. Зазначимо, що ця структура може використовуватися лише для дуже обмеженого кола задач, примітивної аналітики, або для накопичення бази знань з вводів користувача. Для розвитку цієї структури необхідно додати більше метрик запитів, а також розділити записи у БД для того щоб отримати більше можливостей для аналітичних запитів.

Для моніторингу стану системи в цілому використовується сервіс AWS CloudWatch. Це – комбіноване рішення з сховища даних (логів або метрик) та рішення інтеграції, яке дозволяє накопичувати усі згенеровані користувачами та системою дані в одне сховище. В усіх рішеннях AWS доступна інтеграція з цією системою та існують бібліотеки для інтеграції з іншими програмними продуктами.

Усі вищевказані ресурси поєднуються за допомогою сервісу AWS API Gateway. Це повністю керований сервіс, який забезпечує створення, публікацію, обслуговування, моніторинг та захист API в будь-якому

масштабі. API виступають як інтерфейси для програм для доступу до даних, бізнес-логіки або функціональних можливостей серверної частини додатку. API Gateway, дозволяє автоматично генерувати RESTful API та WebSocket API. Шлюз API підтримує контейнерні та безсерверні робочі навантаження, а також веб-додатки.

Шлюз API виконує всі завдання, пов'язані з прийняттям та обробкою запитів, включаючи управління трафіком, підтримку CORS, авторизацію та контроль доступу, регулювання, моніторинг та управління версіями API.

Використовуючи комбінацію зазначених сервісів, вдалося вирішити кілька проблем, пов’язаних з розробкою прототипів – доказів функціонування моделі:

– Швидкість розробки та розміщення. Безсерверні додатки за своєю природою розподілені та декомпозовані, отже можуть бути розгорнуті незалежно один від одного. Більш того, великий рівень абстракції дозволяє розгортання за парадигмою “інфраструктура як код” – конфігурація всіх сервісів може бути описана JSON файлом, що дозволяє автоматизувати процес оновлення версії додатку.
– Вартість розробки. Багато безсерверних додатків сплачують безпосередньо за навантаження. Така схема оплати гнучка і дозволяє зменшити кошти на підтримку додатку.

3.5. Розробка клієнтської частини проекту

Для розробки клієнтської частини додатку було обрано реалізація Web-клієнту, на базі бібліотеки ReactJS.

ReactJS – це бібліотека проектування інтерфейсів. Вона застосовує компонентний підхід та ключовим в архітектурі є поняття компоненту і життєвого циклу.

Під тим що розуміється під назвою “ReactJS” насправді існують дві незалежні підсистеми: середа побудови віртуального дерева компонентів та середа побудови фізичного дерева компонентів. Існує декілька реалізацій як першої, так і другої підсистем.

Наразі найбільш довершеною реалізацією першої підсистеми є архітектура Fiber. Ця архітектура передбачає що під час побудови абстрактного дерева та його порівняння з попереднім деревом можуть бути переривання на більш важливі задачі (наприклад, обробка дій користувача), чи сам процес може бути повністю відмінено. Це обумовлює різницю із попередньою реалізацією, яка виконувала весь процес побудови та порівняння одразу.

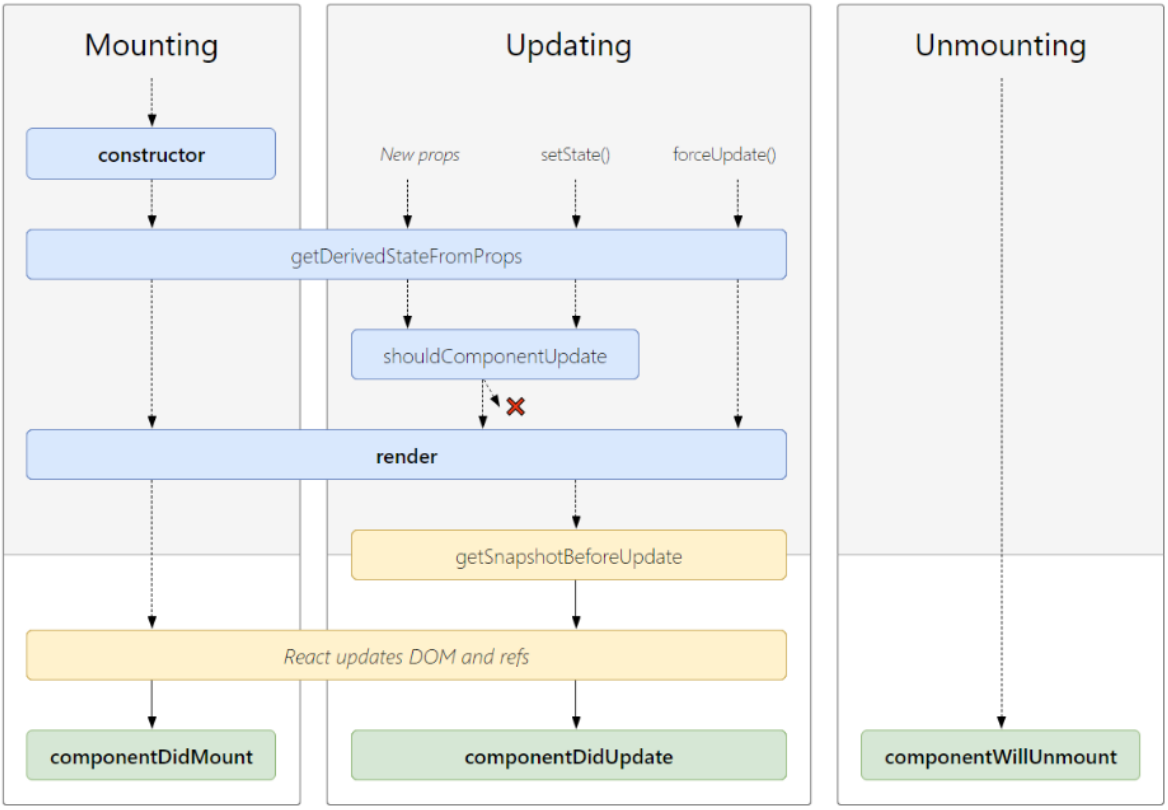


Рисунок 3.9Життєвий цикл компонента ReactJS

З боку реалізації фізичного дерева компонентів різниця реалізацій базується на різниці платформ, на які необхідно перенести дерево. Існують реалізації для побудови DOM-дерева, XML, PDF, SVG – дерев, мобільних платформ. Існує закрита розробка Facebook, яка транслює дерево у компоненти мови програмування Objective C.

React працює асинхронно. Обробники дій користувача, подій браузеру, подій ОС викликають програмні інтерфейси бібліотеки, які через будівельник розкладів запитують оновлення відображення. Запити на оновлення групуються та коли наступає наступний момент малювання, на основі описів компонентів будується дерево абстрактного відображення. Це дерево порівнюється з попереднім за оптимізованим алгоритмом, на основі порівняння будується лист змін. Процес від ініціації до побудови листа змін називається фазою рендеру.

Лист змін між абстрактними відображеннями компонентів передається другій підсистемі, яка у свою чергу оновлює інтерфейс через виклики системних API. Це розподілення процесу дозволяє розробнику описувати компоненти як готове для показу відображення, яке залежить від стану та параметрів.

У майбутніх версіях React мають використовуватися більш розвинені алгоритми порівняння дерев, які дозволять задіяти декілька ядер процесору або розтягнути процес малювання між кількома кадрами. Існують реалізації, які підтримують побудову сторінки ще до того, як вона буде показана користувачеві (технологія React Native та SSR).

Технічна зрілість, велика спільнота розробників, підтримка від великих технічних компаній та розвинена екосистема роблять ReactJS вдалим вибором для розробки проектів будь-якої складності та майже для будь-яких цілей (існують ситуації коли інтерфейси настільки інтерактивні, що їх неможливо втілити ефективно за допомогою React).

Окрім основної бібліотеки для побудови клієнтської частини додатку необхідні допоміжні засоби, а саме: бібліотека інтеграції з AWS, бібліотека управління станом додатку та реалізація HTTP-клієнту.

Для інтеграції з AWS використовується клієнт `aws-sdk`. Для інтеграції з CloudWatch використовується відповідний клас:

```
var cloudwatch = new AWS.CloudWatch({apiVersion: '2010-08-01'});
```

Після цього середа ініціалізується та замінює собою звичайний логер мови програмування JavaScript і клієнтом можна користуватись звичним чином.

Для управління станом додатку використовується бібліотека Redux. Redux є імплементацією архітектури Flux, розробленої у Facebook.

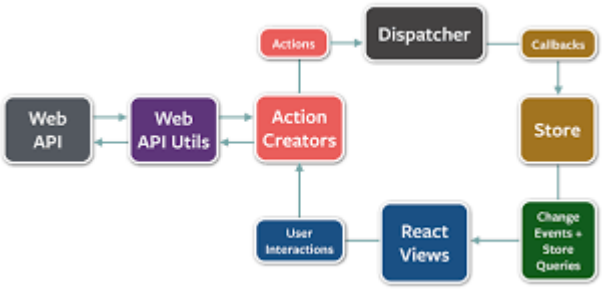


Рисунок 3.10 Архітектура Flux

Ця архітектура взагалі не є специфічною для будь-якої платформи, проте її було розроблено під потреби бібліотеки React, і проектування додатку з виконанням вимог Flux дозволяє ігнорувати більшість недоліків її реалізації.

Бібліотека Redux ще більш поглибила та обмежила вільний простір для розробника і її використання вимушує чітко слідувати наступним принципам

– Існує лише одне джерело даних для всього додатку.
– Це джерело не може буди змінено ніяким чином, окрім спеціальних об’єктів які містять в собі усю необхідну інформацію для зміни стану.

– Зміна стану додатку відбувається через відображення таких об’єктів на існуючий стан і генерацію нового. Зміненого стану. Функції, які генерують новий стан, не можуть мати жодних інших проявів.

Виконання цих вимог робить процес проектування бізнес-логіки додатку схожим на процес проектування самих компонентів: розробник описує певну множину можливих станів та процеси, які призводять до їх змін.

Для реалізації HTTP клієнту використано парадигму Stale While Revalidate (SWR). Ця парадигма має безпосереднє відношення то так званих оптимістичних оновлень інтерфейсу. Коли додаток виконує певний запит, інтерфейс оновлюється одразу, ще до відповіді серверу, а потім зберігає чи відміняє ці зміни.

Цей підхід маж свої недоліки та переваги, і можна навіть побачити в ньому збіжність з методами проектування розподілених баз даних, де окремий вузол спочатку сповіщає сусідів про зміни, потім оновлює свій стан і може відмінити зміни, якщо більшість вузлів вирішила що цей запис не є коректним.

3.6. **Керівництво користувача та розробника**

Веб-додаток є лише інтерфейсом до практичної реалізації магістерської дисертації, отже має просту структуру.

Нові користувачі реєструються через створення їхніх записів у базі даних та розміщенні у пулі Amazon Cognito.

Система авторизації підтримує вхід за допомогою Google акаунту або Github акаунту.

Після авторизації користувач потрапляє на основну сторінку проекту де він може випробувати модель.

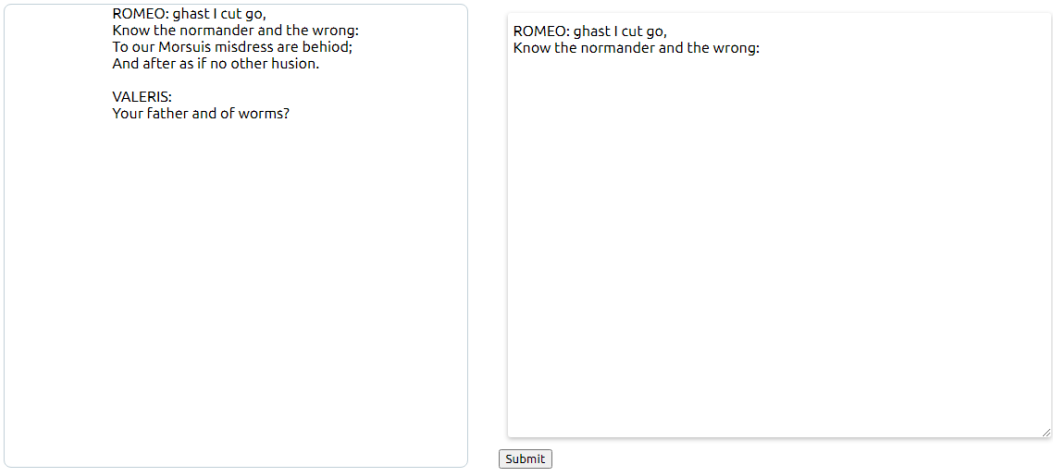


Рисунок 3.11 Сторінка генерації тексту

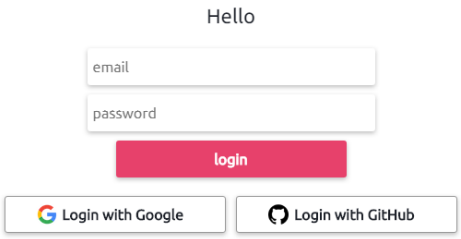


Рисунок 3.12 Сторінка авторизації

Усі ресурси, які використовуються для підтримки серверної частини проекту, можуть бути експортовані та розміщені за допомогою сервісів Amazon, які втілюють концепцію “Infrastructure-as-a-code” – можливість створювати та налагоджувати проекти за допомогою конфігураційних файлів.

API серверу

`POST provision-user` – Приймає параметри:

`auth-mode` – режим авторизації;

`g-token` – Id, який надає Google при використанні функції “Увійти за допомогою Google”;

`github-id` – Id, який надає Github при використанні функції “Увійти за допомогою Github”;

`login` – логін;

`password` – пароль;

`POST generate-text` – приймає параметри:

`start` – початок тексту

`len` – очікувана максимальна довжина тексту

Висновки до розділу

В даному розділі розглядалася реалізація програмного забезпечення для побудови інтелектуальної системи на базі штучного інтелекту.

Наразі для побудови інтелектуальних систем на основі нейронних мереж існує декілька конкуруючих програмних рішень. Найбільш впливовим та популярним є використання мови програмування Python. Для цієї мови існує декілька можливих середовищ розробки нейронних мереж, а саме PyTorch та Tensorflow. За низкою характеристик PyTorch краще відповідає вимогам до програмного забезпечення, яке реалізує запропонований в дисертації алгоритм. Для виконання поставленої задачі розроблено гібридну систему генерації тексту на основі нейронних мереж, побудованих за допомогою цієї бібліотеки.

Отриману в результаті досліджень математичну модель було вбудовано в Веб-додаток на базі хмарних безсерверних технологій від Amazon. Використання сервісів Amazon дозволяє отримати виграш у простоті керування та зменшити витрати на підтримку програмного забезпечення.

4. МАРКЕТИНГОВИЙ АНАЛІЗ СТАРТАП-ПРОЕКТУ

4.1. **Опис ідеї проекту**

Результати практичної реалізації магістерської дисертації можна використовувати як генератор продовження тексту в комерційних цілях

Таблиця 4.1 Опис ідеї стартап-проекту

Зміст ідеї	Напрямки застосування	Вигоди для користувача
Генератор тексту на основі гібридної нейронної мережі для створення текстів невеликого обсягу	1. Використання в генераторах невеликих повідомлень	1. Малий час тренування дозволяє розроблювати оптимізовані під певний сценарій генератори. 2. Використання компонентів генератору окремо для пришвидшення розробки нових генераторів.
	2. Розробка чат-ботів	1. Збільшення швидкості відповіді на повідомлення користувача. 2. Автоматизація найбільш простих сценаріїв взаємодії користувача та оператора служби підтримки.

Висновок: в табл. 4.1 наведені найбільш перспективні галузі для втілення гібридних генераторів текстів та використання їх переваг. Ці галузі – генерація невеликих за обсягом текстів – є затребуваними в галузі автоматизації контакту с клієнтом. Це можуть бути системи швидкої відповіді та системи генерації нотифікацій для клієнтів або системи будь-якої іншої комерційної взаємодії з людьми на базі текстових засобів передачі даних: переписка, чати, електронні письма тощо.

Таблиця 4.2 Визначення сильних, слабких та нейтральних характеристик ідеї проекту

№ п/п	Техніко-економічні характеристики ідеї	(потенційні) товари/концепції конкурентів				W	N	S
		Запропонований проект	GPT-3	chatbot.com	ebi.ai			
1	Швидкість тренування	Висока	Дуже низька	Не надає моделей	Не надає моделей			+
2	Генерація спеціалізованих моделей	Є	Є	Немає	Немає			+
3	Вартість розміщення	Низька	Дуже висока	Неможливо розмістити у себе	Неможливо розмістити у себе			+
4	Наявність панелі управління	Немає	Немає	Є	Є	+		
5	Можливості для інтеграції	Відкрите API	Закрите API	Надають засоби	Закрите API		+	
6	Наявність служби технічної підтримки	Лімітована	Немає	Повна підтримка	Повна підтримка	-		
7	Вартість	Низька	Не розголошується	Середня	Висока			+
8	Підтримка генерації великих текстів	Немає	Є	Лімітована	Є			

У порівнянні з головними конкурентами запропоноване рішення має переваги у швидкості розгортання та гнучкості у адаптації під потреби клієнтів.

Іншою перевагою є можливість швидко генерувати спеціалізовані моделі під певні задачі та можливість інтеграції з іншими платформами.

4.2. Технологічний аудит ідеї проекту

Для виконання технічного аудиту проекту необхідно здійснити генерацію та аналіз можливих варіантів реалізації проекту.

Таблиця 4.3 Технологічна здійсненність ідеї проекту

№ п/п	Ідея проекту	Технології її реалізації	Наявність технологій	Доступність технологій
1	Готова модель для генерації невеликих текстів	Математична тренована модель генерації тексту	Наявні	Доступні
		Web-клієнт для користувача	Наявні	Доступні
2	Система генерації спеціалізованих моделей	Автоматизований процес тренування моделей з певними характеристиками	Наявні	Малодоступні
		Автоматизований процес розгортання моделі для певного користувача	Наявні	Малодоступні
3	Інтегроване рішення для взаємодії з клієнтами	Рішення з автоматичної інтеграції та поєднання до сторонніх сервісів	Наявні	Недоступні
		Застосунок підтримки клієнтів повного циклу	Наявні	Недоступні
Обрана технологія реалізації ідеї проекту: 1, 2.				

За результатами аналізу створено три варіанти реалізації переваг рішення та були оцінені за трудозатратами на реалізацію та впровадження. Найбільш перспективними є варіанти 1 та 2, оскільки вони дозволяють випустити продукт до ринку.

4.3. **Аналіз ринкових можливостей запуску стартап-проекту**

Для оцінки доцільності входу на ринок стартапу, цей ринок оцінюється за основними характеристиками, які вказують його розмір та динаміку.

Таблиця 4.4 Попередня характеристика потенційного ринку стартап-проекту

№ п/п	Показники стану ринку	Характеристика
1	Кількість головних гравців, од	До 10
2	Загальний обсяг продаж, доларів США	1.250.000.000
3	Динаміка ринку (якісна оцінка)	Зростає
4	Наявність обмежень для входу (вказати характер обмежень)	Відсутні
5	Специфічні вимоги до стандартизації та сертифікації	Відсутні
6	Середня норма рентабельності в галузі (або по ринку), %	25%

Таблиця 4.5 Характеристика потенційних клієнтів стартап-проекту

№ п/п	Потреба, що формує ринок	Цільова аудиторія (цільові сегменти ринку)	Відмінності у поведінці різних потенційних цільових груп клієнтів	Вимоги споживачів до товару
1	Організація швидкої відповіді клієнту	Інтернет-магазини Кол-центри Інтернет-сервіси Веб-портали Інтернет-банки	Відмінності у тоні, структурі та змісту повідомлень	Гнучкість, простота впровадження, легкість налаштування, простота управління
2	Генерація контекстуальних повідомлень	Оператори мобільного зв’язки Системи взаємодії з клієнтами Системи організації воронки продажів	Тематика та структура повідомлень	Висока унікальність повідомлень, високі показники якості згенерованого тексту

Ринок розробки інтелектуальних рішень для спілкування з клієнтом вже займає значний розмір і за оцінками буде тільки зростати. Ринок не має природніх або штучних перешкод для входу, а середня рентабельність – висока.

Висновки: потенційними клієнтами стартап-проекту є комерційні підприємства, діяльність яких включає або цілком міститься з спілкування клієнтами у текстовій форми.

У першому випадку це можу бути будь-який бізнес, пов’язаний з наданням цифрових товарів або послуг, у другому – це можуть бути підприємства які забезпечують функціонал спілкування з клієнтами.

Таблиця 4.6можливостей

№ п/п	Фактор	Зміст можливості	Можлива реакція компанії
1	Інтеграція	Поява потреби впроваджувати оптимізовані моделі у сторонні сервіси	Відкриття програмних інтерфейсів додатку. Впровадження засобів інтеграції
2	Попит	Збільшення попиту на продукт	Автоматичне масштабування за допомогою безсерверних технологій
3	Інновації	Поява нових досліджень, які принципово змінюють технології генерації тексту	Імплементація запропонованих рішень

Таблиця 4.7 Фактори загроз

№ п/п	Фактор	Зміст загрози	Можлива реакція компанії
1	Конкуренція	Виникнення більш досконалих альтернатив	Пошук нових галузей застосування генерації текст
2	Функціонал	Відсутність необхідних компонентів для функціонування системи	Розробка необхідних компонентів

Висновок: інтелектуальні системи – це ринок з великими можливостями та ризиками. Постійно з'являються нові підходи, рішення так методи, використовуючи які можна отримати конкурентну перевагу, але ці рішення можуть бути використані конкурентами.

Таблиця 4.8 Ступеневий аналіз конкуренції на ринку

Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною)
1. Вказати тип конкуренції чиста	Велика кількість гравців без неконкурентних переваг	Слідувати стратегії просування та конкуренції Інформувати ринок про появу нового гравця
2. За рівнем конкурентної боротьби міжнаціональна	Багато гравців пропонують рішення у багатьох державах	Провести локалізацію продукту на інші мови. Будувати локалізовані та глобальні спеціалізовані моделі
3. За галузевою ознакою – міжгалузева	Спілкування з клієнтами відбувається в усіх галузях виробництва та надання послуг	Розвинути функціональність щодо впровадження спеціалізованих генераторів
4. Конкуренція за видами товарів: Товарно-видова	Усі гравці надають одну послугу з різним рівнем гнучкості та функціоналу	Можливість сфокусуватися лише на власному функціоналі
5. За характером конкурентних переваг Нецінова	Конкурентні переваги отримуються здебільшого завдяки вдосконаленню продукту	Можливість власноруч задавати ціни для своїх послуг
6. За інтенсивністю – Немарочна	Немає яскравих гравців, бо великі технологічні компанії пропонують лише загальні рішення	Розвивати свій бренд для отримання лідерства в цьому напрямку

Висновки: тип конкуренції на ринку чиста, ринок можна охарактеризувати як ринок, на якому багато гравців пропонують у багатьох галузях виробництва та державах, усі конкуренти пропонують один товар або послугу.

Конкурентні переваги отримуються завдяки вдосконаленню продукту та немає яскравих граців.

Таблиця 4.9 Аналіз конкуренції в галузі за М. Портером

Складові аналізу	Прямі конкуренти в галузі	Потенційні конкуренти	Постачальники	Клієнти	Товари-замінники
	GPT-3, chatbot.com, ebi.ai	Продукти, що забезпечують спілкування з клієнтом традиційним шляхом	Ширина потоку обробки платежів	Швидкість інтернет-з'єднання, якість згенерованого тексту	Наявні гравці на ринку Більш примітивні засоби генерації тексту
Висновки:	Інтенсивність конкуренції висока тому що багато гравців з неочевидними перевагами	Можливе злиття з гравцями з цієї галузі	Постачальники не впливають на умови роботи на ринку	Клієнти вимагають простоти у використанні продукту та якість генерованих текстів	Важкість у приваблені клієнтів через сформовані аудиторії конкурентів

Висновки: проаналізувавши динаміку ринку та можливості існування та конкуренції на ньому, можна зробити висновок що ринок є незрілим, дуже динамічним, так як існуючі гравці не впливають на нього значною мірою. Кожне з існуючих рішень має свої переваги, які не є очевидними та сформовану аудиторію.

Для функціонування у такому ринку необхідно пропонувати користувачам принципово нові рішення да намагатися збільшити ринок через приваблення нових користувачів.

Таблиця 4.10 Обґрунтування факторів конкурентоспроможності

№ п/п	Фактор конкурентоспроможності	Обґрунтування (наведення чинників, що роблять фактор для порівняння конкурентних проектів значущим).
1	Швидкість впровадження нових моделей	Через велику швидкість тренування нових моделей можливо швидко створювати нові генератори тексту.
2	Гнучкість у використанні	Інтерфейси продукту відкрити на хмарні, отже їх легко розвивати, отримуючи нові переваги.
3	Можливість адаптації під певного клієнта	Через невеликий розмір функціоналу його легко адаптувати під клієнта, проводити експерименти тощо.
4	Ціна впровадження	Завдяки безсерверним технологіям вартість впровадження рішення для окремого клієнта є низькою

Таблиця 4.11 Порівняльний аналіз сильних та слабких сторін системи

№ п/п	Фактор конкурентоспроможності	Бали 1-20	Рейтинг товарів-конкурентів у порівнянні з запропонованим						
			-3	-2	-1	0	1	2	3
Швидкість впровадження нових моделей	Швидкість впровадження нових моделей	13						+	
Гнучкість у використанні	Гнучкість у використанні	16							+
Можливість адаптації під певного клієнта	Можливість адаптації під певного клієнта	14				+			
Ціна впровадження	Ціна впровадження	10				+			

Висновки: оцінені основні переваги продукту. Через свої сильні сторони продукт є швидким у впровадженні, має низьку вартість та легко може бути адаптований під клієнта.

Завдяки правильному позиціюванні проекту та виділенню його сильних сторін він за сумою факторів конкурентоспроможності виграє у конкурентів.

Таблиця 4.12 SWOT-аналіз стартап-проекту

Сильні сторони:	Слабкі сторони:
Швидкість тренування Генерація спеціалізованих моделей Вартість розміщення Вартість	Наявність панелі управління Можливості для інтеграції Наявність служби технічної підтримки Підтримка генерації великих текстів
Можливості:	Загрози:
Інтеграція Попит Інновації	Конкуренція Функціонал

Таблиця 4.13 Альтернативи ринкового впровадження стартап-проекту

№ п/п	Альтернатива (орієнтовний комплекс заходів) ринкової поведінки	Ймовірність отримання ресурсів	Строки реалізації
1	Розробка готового рішення для генерації тексту з клієнтом для управління	Ймовірне	3 місяці
2	Розробка платформи для створення спеціалізованих моделей генерації тексту та створення панелі управління	Ймовірне	8 місяців
3	Розробка комплексного рішення для взаємодії з клієнтом на основі інтелектуальних систем з можливостю інтеграції	Неймовірне	18 місяці

Висновки: існують різні можливості практичної реалізації проекту, з різним рівнем надання послуг. Найбільш прийнятною за співвідношенням комплексу надання послуг до ймовірності впровадження.

4.4. **Розроблення ринкової стратегії проекту**

Таблиця 4.14 Вибір цільових груп потенційних споживачів

№ п/п	Опис профілю цільової групи потенційних клієнтів	Готовність споживачів сприйняти продукт	Орієнтовний попит в межах цільової групи (сегменту)	Інтенсивність конкуренції в сегменті	Простота входу у сегмент
1	Інтернет-магазини	Висока	Середній	Висока	Середня
2	Інтернет-сервіси	Висока	Високий	Висока	Низька
3	Веб-портали	Середня	Середній	Середня	Висока
4	Постачальники рішень взаємодії з клієнтом	Середня	Високий	Висока	Низька
5	Оператори зв’язку	Низька	Низький	Низька	Низька
Інтернет-магазини, Інтернет-сервіси, Веб-портали, Постачальники рішень взаємодії з клієнтом, Оператори зв’язку					

Висновки: за сукупністю факторів можна зробити висновок що всі напрямки розповсюдження є доступними для входу, проте найбільш сприйнятими є інтернет-магазини та постачальники інтернет-сервісів.

Інші, більш класичні напрямки розповсюдження є більш ризиковими або можуть принести менший прибуток. Таким чином, ці ринки не є характерними для стартапів.

Можна зазначити що такі ринки розповсюдження також характеризуються високою конкуренцією що робить їх непридатними для впровадження інноваційних рішень, які фокусуються на зміні ринку.

Таблиця 4.15 Визначення базової стратегії розвитку

№ п/п	Обрана альтернатива розвитку проекту	Стратегія охоплення ринку	Ключові конкурентоспроможні позиції відповідно до обраної альтернативи	Базова стратегія розвитку*
1	Розробка платформи для створення спеціалізованих моделей генерації тексту та створення панелі управління	Надання персоналізованих послуг для клієнтів, охоплення ринку через надання безкоштовної версії продукту	Швидкість впровадження, можливість спеціалізацій рішень, низька вартість	Стратегія диференціації

Таблиця 4.16 Визначення базової стратегії конкурентної поведінки

№ п/п	Чи є проект “першопрохідцем” на ринку?	Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?	Чи буде компанія копіювати основні характеристики товару конкурента, і які?	Стратегія конкурентної поведінки*
1	Ні	Компанія буде забирати користувачів у конкурентів	Так, компанія буде копіювати панелі керування системою генерації тексту	Стратегія виклику лідера

Таблиця 4.17 Визначення стратегії позиціонування

№ п/п	Вимоги до товару цільової аудиторії	Базова стратегія розвитку	Ключові конкурентоспроможні позиції власного стартап-проекту	Вибір асоціацій, які мають сформувати комплексну позицію власного проекту (три ключових)
1	Швидкість впровадження Якість тексту Відповідність текста контексту	Стратегія диференціації	Використання новітніх технологій в області розгортання систем та генерації тексту	Інноваційність Розумність Простота

Висновки: проект пропонується розвивати за стратегією диференціації. Конкурентну поведінку пропонується формувати за принципом виклику лідера.

Стратегія диференціації є одною із найбільш поширених та загальних конкурентних стратегій, основний принцип якої полягає у орієнтації діяльності, розвитку та виробництва підприємства на створення унікальних, маючих неповторні показники у будь-якому аспекті продуктів. Такі продукти потравляють на сформовані ринки з великою кількістю гравців та споживачів та отримують свою частку розповсюдження.

Стратегія виклику лідера – це тип конкурентної поведінки, характерний для нових гравців на ринку зі сформованими лідерами. Нове підприємство обирає свого основного конкурента та фокусується на створенні продукту, який буде кращим за існуючий продукт у деяких своїх проявах.

4.5. **Розроблення маркетингової програми стартап-проекту**

Таблиця 4.18 Визначення ключових переваг концепції товару

№ п/п	Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити
1	Організація швидкої відповіді клієнту	Використання моделей машинного навчання для генерації відповідей клієнту	Простота впровадження Можливість налаштування Цінова перевага
2	Генерація контекстуальних повідомлень	Генерація унікальних та доречних повідомлень клієнту	Можливість аналізувати попередні повідомлення клієнту Можливість використовувати різні алгоритми для різних моделей

Висновки: за результатами аналізу ринку та конкурентів були знайдені основні потреби користувачів, визначений рівень, на якому конкуренти задовольняють ці потреби та оцінено як стартап-проект задовольняє ці потреби у порівнянні з конкурентами.

Визначено що головними потребами у ніші яку планує зайняти проект є швидкість відповіді клієнту та надання правильного контексту повідомлень. Результат магістерської дисертації можна використати для цих цілей оскільки його перевагами є підвищена швидкість тренування моделей, а отже й можливість розробити одразу певну кількість цих моделей. Також використання хмарних технологій надає перевагу у швидкості доставки рішення користувачам.

Таблиця 5.19. Опис трьох рівнів моделі товару

Рівні товару	Сутність та складові		
I. Товар за задумом	Система купівлі та продажу рекламних повідомлень у мессенджері “Telegram”		
II. Товар у реальному виконанні	Властивості/характеристики	М/Нм	Вр/Тх /Тл/Е/Ор
	1. Можливості використання різних моделей	М	Тл/Е
	2. Кабінет користувача	М	Тл/Е
	3. Можливість розгорнути систему для кожного користувача	Нм	Тл/Е
	4. Інтеграція з іншими ресурсами	М	Тл/Е
	Параметр тестування – розходження Кульбака-Лейблера для моделей машинного навчання		
	Пакування – прогресивний веб-додаток		
	Марка: chatty.ai		
III. Товар із підкріпленням	До продажу: безкоштовна версія продукту, відкриті програмні інтерфейси		
	Після продажу: повний функціонал, можливості інтеграції та персоналізації		
За рахунок чого потенційний товар буде захищено від копіювання: Моделі машинного навчання, які розповсюджуються як сервіс, неможливо копіювати			

Висновки: даний проект позиціонується як гнучке та просте рішення для інтелектуальної організації спілкування з клієнтом. Його основною перевагою є можливість адаптувати продукт під кожного окремого клієнта та можливість доробки рішення.

Таблиця 4.19 Визначення меж встановлення ціни

№ п/п	Рівень цін на товари-замінники	Рівень цін на товари-аналоги	Рівень доходів цільової групи споживачів	Верхня та нижня межі встановлення ціни на товар/послугу
1	$ 200	$ 250	$ 2000-2000	$ 100 /$ 300

Таблиця 4.20 Формування системи збуту

№ п/п	Специфіка закупівельної поведінки цільових клієнтів	Функції збуту, які має виконувати постачальник товару	Глибина каналу збуту	Оптимальна система збуту
1	Клієнти цієї категорії схильні до оформлення підписок на різні сервіси, отже модель збуту є оптимальною	Можливість біллінгу Можливість рефанду	Глибина 1: лише для окремих користувачів без посередників	Розповсюдження за сервісною моделлю без альтернативних каналів збуту та постачальників

Таблиця 5.22. Концепція маркетингових комунікацій

№ п/п	Специфіка поведінки цільових клієнтів	Канали комунікацій, якими користуються цільові клієнти	Ключові позиції, обрані для позиціонування	Завдання рекламного повідомлення	Концепція рекламного звернення
1	Великий час знаходження в мережі інтернет	Соціальні мережі, портали, форуми	Інноваційність Новітність	Вмотивувати користувача перейти на сторінку додаток	Пропозиція, яка дійсно відповідає персональним вимогам

Висновки до розділу

Під час розробки маркетингової стратегії стартап проекту були сформовані концепція та форма монетизації результатів магістерської роботи, розроблені цілі, ідеї та візія майбутнього продукту.

Галузь генерації коротких повідомлень – це нова, швидко зростаюча галузь, яка використовує усі новітні розробки да досягнення у генерації тексту. У цьому ринку існує великий попит та велика кількість гравців, що надає можливість усім новим гравцям приєднуватися до конкуренції.

Майбутній стартап-проект буде використовувати знайдені переваги генерації тексту та переваги хмарних технологій як базу для створення швидких інтелектуальних рішень, оптимізованих під кожного клієнта, що надасть продукту перевагу над існуючими гравцями, які просувають універсальні рішення, які неможливо однаково ефективно використовувати всім користувачам.

ВИСНОВКИ

Під час виконання магістерської дисертації були розглянуті генератори тексту, проаналізовані їх характеристики, складові та відмінності між окремими варіантами реалізації генераторів тексту, їх сильні та слабкі сторони.

1. Виконано аналіз існуючих алогритмів автоматичної генерації тексту, виявлено їх недоліки та можливі покращення;
2. Сформульовано та поставлено задачу оптимізації часу тренування систем генерації тексту на базі нейронних мереж;
3. Запропоновано гібридний алгоритм генерації тексту, який дозволяє спростити модель на зменшити час тренування за допомогою декомпозиції класичних генераторів тексту на базі інтегрованої архітектури;
4. Запропонований алгоритм реалізовано засобами мови програмування Python та хмарного середовища AWS;
5. Реалізований алгоритм порівняно з базовим та іншими відомими алгоритмами генерації тексту за часом тренування та . Алгоритм надає значний виграш у часі тренування за рахунок незначної втрати якості.

Використано наступні технології: мова програмування Python, сервіс виконання коду AWS Lambda, розподілене сховище даних Amazon S3 та систему доставки даних AWS CloudFront, сервіс авторизації Amazon Cognito, базу даних DynamoDB та API Gateway для поєднання всього рішення. Для розробки системи генерації обрано бібліотеку PyTorch.

Для розробки Web-клієнту використано мову програмування JavaScript, ReactJS та aws-kit.

Розроблена новітня модель генератора тексту на основі поєднання двох відомих архітектур нейронних мереж, яку було натреновано на текстовій бази даних та відкрито у якості Web-додатку.

Був проведений маркетинговий аналіз стартап-проекту. Сформовано ідею та концепцію продукту, який би зміг використати переваги розробки, оцінено ринок збиту, оцінені сильні та слабкі сторони та можливі варіанти реалізації та виходу на ринок. Сформульовано стратегії розвитку та конкуренції, основні можливості та загрози для комерційної успішності проекту. Проаналізовані основні групи користувачів, за результатами аналізу згенеровано модель монетизації та маркетингу.

Результати досліджень в рамках магістерської дисертації опубліковані у виді статті.

Наукова новизна одержаних результатів магістерської дисертації полягає у наступному:

вперше:

- Вивчена можливість декомпозиції окремих елементів інтегрованих архітектур генераторів тексту для отримання гібридних моделей генерації тексту;
- Розроблена модель яка втілює декомпозицію генерації тексту на етапи формування змісту, формування складу та поверхневу реалізацію.

удосконалено:

- Зменшено час та складність отриманих моделей генерації тексту
- Полегшений процес генерації окремих інтелектуальних систем за допомогою безсерверних технологій.

здобуло подальший розвиток:

- Дослідження гібридних архітектур генераторів тексту як засобів отримати оптимізовані під певну задачу систему;

- Використання нейронних мереж в якості компонентів генераторів тексту.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Ylona Chun Tie, Melanie Birks, Karen Francis. Grounded theory research: A design framework for novice researchers [Електронний ресурс] – Режим доступу до ресурсу: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6318722/;

2. Srinivas Bangalore, Owen Rambow. Exploiting a Probabilistic Hierarchical Model for Generation [Електронний ресурс] – Режим доступу до ресурсу: https://www.researchgate.net/publication/2473335_Exploiting_a_Probabilistic_Hierarchical_Model_for_Generation;

3. OpenCCG: The OpenNLP CCG Library [Електронний ресурс] – Режим доступу до ресурсу: http://openccg.sourceforge.net/;

4. Combinatory categorial grammar [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Combinatory_categorial_grammar;

5. Martin Kay. Chart Generation grammar [Електронний ресурс] – Режим доступу до ресурсу: https://www.inf.ed.ac.uk/teaching/courses/nlg/readings/KayACL96.pdf;

6. Claire Gardent, Laura Perez-Beltrachini. A Statistical, Grammar-Based Approach to Microplanning [Електронний ресурс] – Режим доступу до ресурсу: https://www.aclweb.org/anthology/J17-1001.pdf;

7. Pablo A. Duboue, Kathleen R. McKeown. Statistical Acquisition of Content Selection Rules for Natural Language Generation

Microplanning [Електронний ресурс] – Режим доступу до ресурсу: http://www.cs.columbia.edu/nlp/papers/2003/duboue_mckeown_03.pdf;

8. Katja Filippova. Tree Linearization in English: Improving Language Model Based Approaches. [Електронний ресурс] – Режим доступу до ресурсу: https://www.researchgate.net/publication/220817081_Tree_Linearization_in_English_Improving_Language_Model_Based_Approaches;

9. Nina Dethlefs, Heriberto Cuayahuitl. Hierarchical Reinforcement Learning and Hidden Markov Models for Task-Oriented Natural Language Generation. [Електронний ресурс] – Режим доступу до ресурсу: https://www.aclweb.org/anthology/P11-2115.pdf;

10. Sina Zarrieß, Jonas Kuhn. Combining Referring Expression Generation and Surface Realization: A Corpus-Based Investigation of Architectures. [Електронний ресурс] – Режим доступу до ресурсу: https://www.aclweb.org/anthology/P13-1152.pdf;

11. Bernd Bohnet, Leo Wanner, Simon Mille, Alicia Burga. Broad Coverage Multilingual Deep Sentence Generation with a Stochastic Multi-Level Realizer [Електронний ресурс] – Режим доступу до ресурсу: https://www.aclweb.org/anthology/C10-1012.pdf;

12. Yoav Goldberg. A Primer on Neural Network Models for Natural Language Processing. [Електронний ресурс] – Режим доступу до ресурсу: https://arxiv.org/pdf/1510.00726.pdf;

13. Karen Kukich. Where do Phrases Come from: Some Preliminary Experiments in Connectionist Phrase Generation. [Електронний ресурс] – Режим доступу до ресурсу: https://link.springer.com/chapter/10.1007/978-94-009-3645-4_26;

14. Ian Goodfellow, NIPS 2016 Tutorial: Generative Adversarial Networks. [Електронний ресурс] – Режим доступу до ресурсу: https://arxiv.org/abs/1701.00160;

15. Yoshua Bengio, Réjean Ducharme, Pascal Vincent, Christian Jauvin. A Neural Probabilistic Language Model. [Електронний ресурс] – Режим доступу до ресурсу: https://jmlr.org/papers/volume3/tmp/bengio03a.pdf;

16. Andriy Mnih, Geoffrey Hinton. Three New Graphical Models for Statistical Language Modelling. [Електронний ресурс] – Режим доступу до ресурсу: https://www.cs.toronto.edu/~amnih/papers/threenew.pdf;

17. Stanislau Semeniuta, Aliaksei Severyn, Erhardt Barth. Convolutional Variational Autoencoder for Text Generation. . [Електронний ресурс] – Режим доступу до ресурсу: https://arxiv.org/abs/1702.02390;

18. Tomas Mikolov, Martin Karafiát, Lukas Burget. Recurrent neural network based language model. [Електронний ресурс] – Режим доступу до ресурсу: https://www.researchgate.net/publication/221489926_Recurrent_neural_network_based_language_model;

19. Ilya Sutskever, James Martens, Geoffrey Hinton. Generating Text with Recurrent Neural Networks. [Електронний ресурс] – Режим доступу до ресурсу: https://icml.cc/Conferences/2011/papers/524_icmlpaper.pdf;

20. Asli Celikyilmaz, Elizabeth Clark, Jianfeng Gao. Evaluation of Text Generation: A Survey. [Електронний ресурс] – Режим доступу до ресурсу: https://arxiv.org/abs/1511.06114;

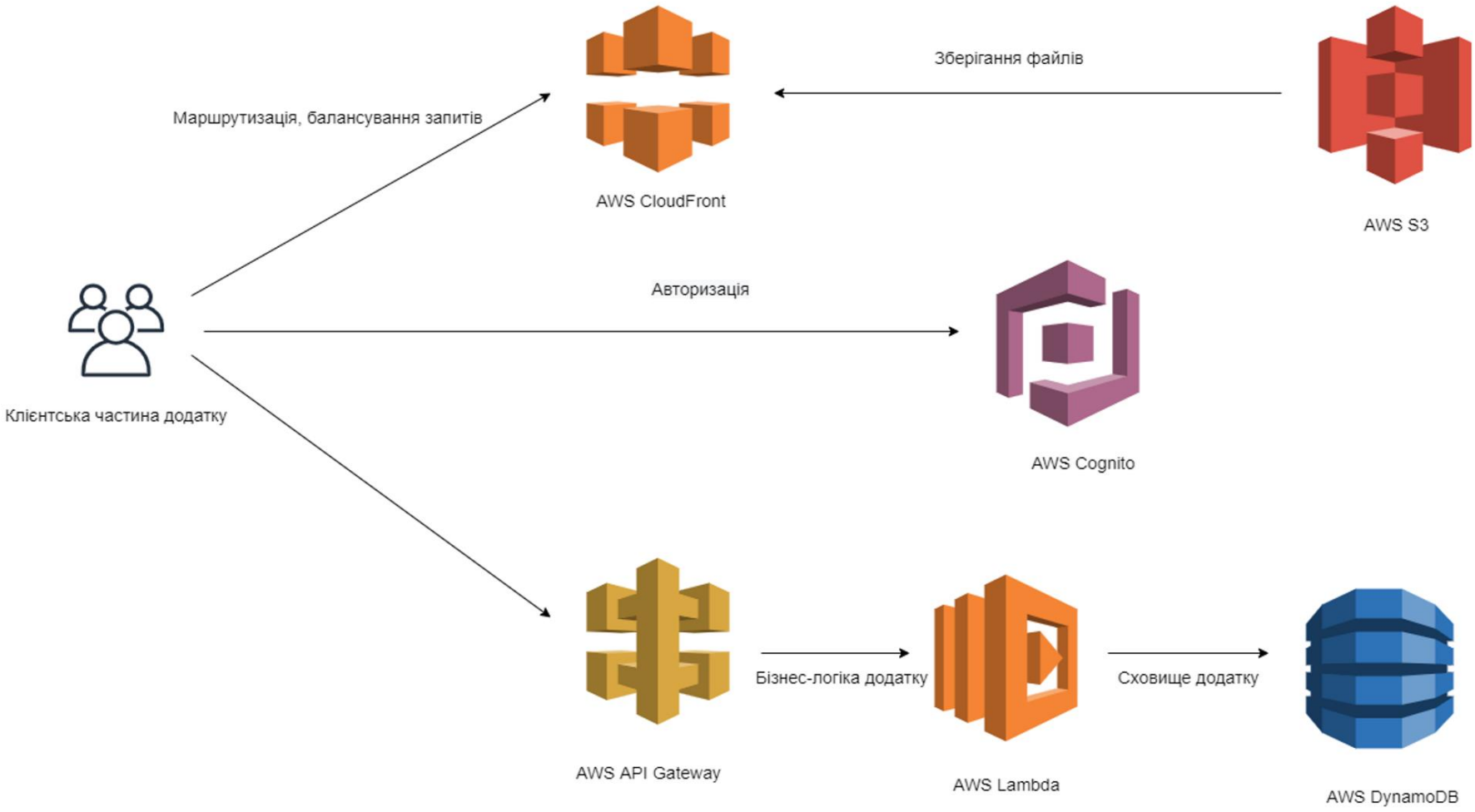
21. Diederik P. Kingma, Max Welling. An Introduction to Variational Autoencoders. [Електронний ресурс] – Режим доступу до ресурсу: https://arxiv.org/abs/1511.06114;

ДОДАТКИ

ДОДАТОК А

ІЛЮСТРАЦІЙНІ МАТЕРІАЛИ

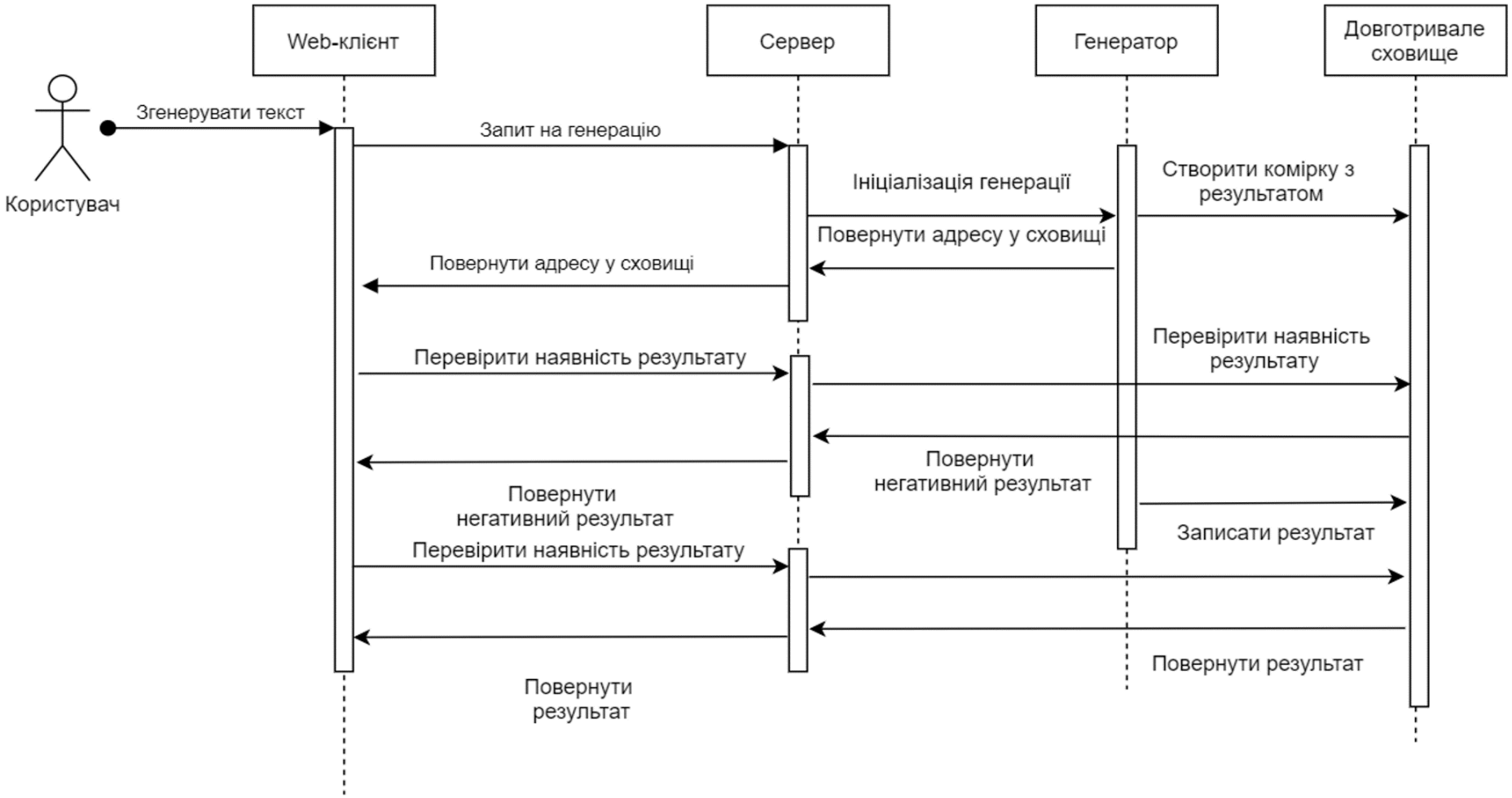
Бессерверна архітектура додатку



Демонстраційний плакат № 1
до магістерської дисертації на тему
„Гібридний алгоритм автоматичної генерації тексту"

Розробив: *Д.В. Кузнєцов*
Прийняв: *В.П. Пасько*

Діаграма послідовності запитів

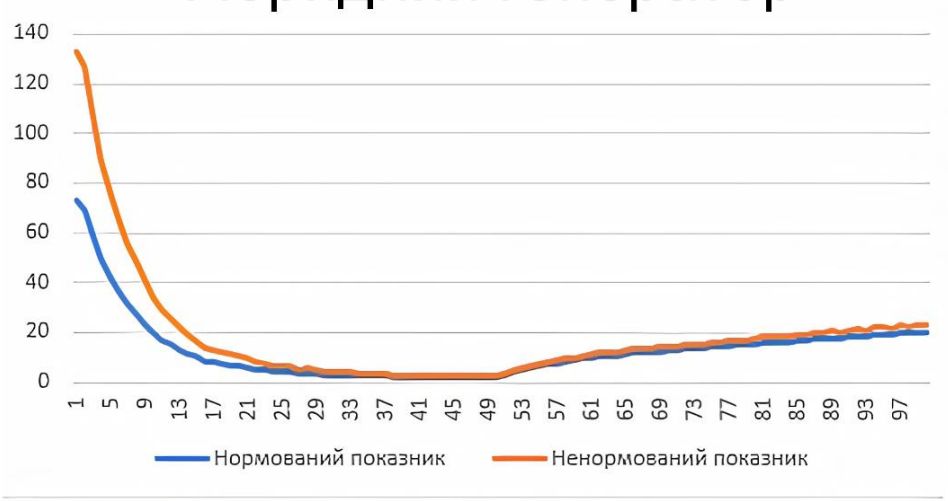


Демонстраційний плакат № 2
до магістерської дисертації на тему
„Гібридний алгоритм автоматичної генерації тексту”

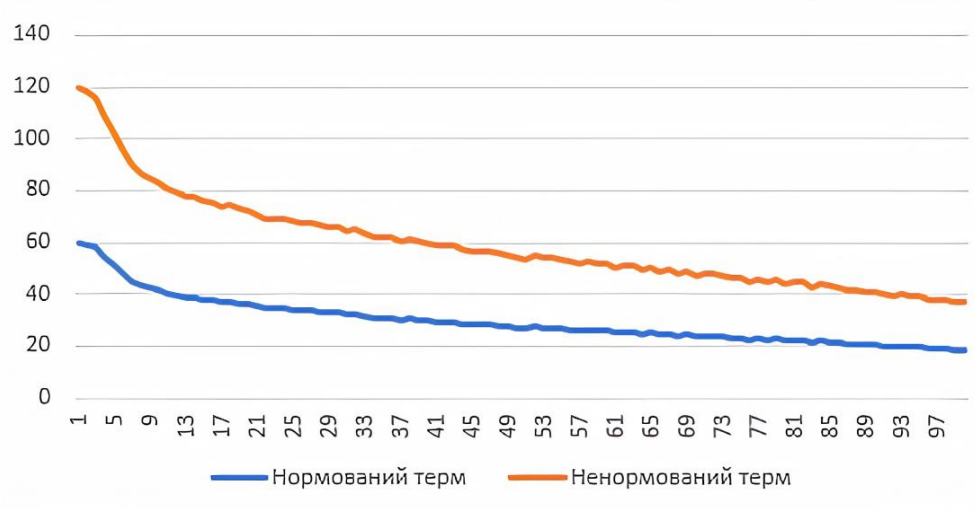
Розробив: *Д.В. Кузнєцов*
Прийняв: *В.П. Пасько*

Результати досліджень

Гібридний генератор



Класичний генератор



Обчислювальні метрики

Метод¤	BLEU-2¤	BLEU-3¤	BLEU-4¤
Класична· модель¤	0.704¤	0.655¤	0.551¤
Гібрид¤	0.658¤	0.621¤	0.451¤

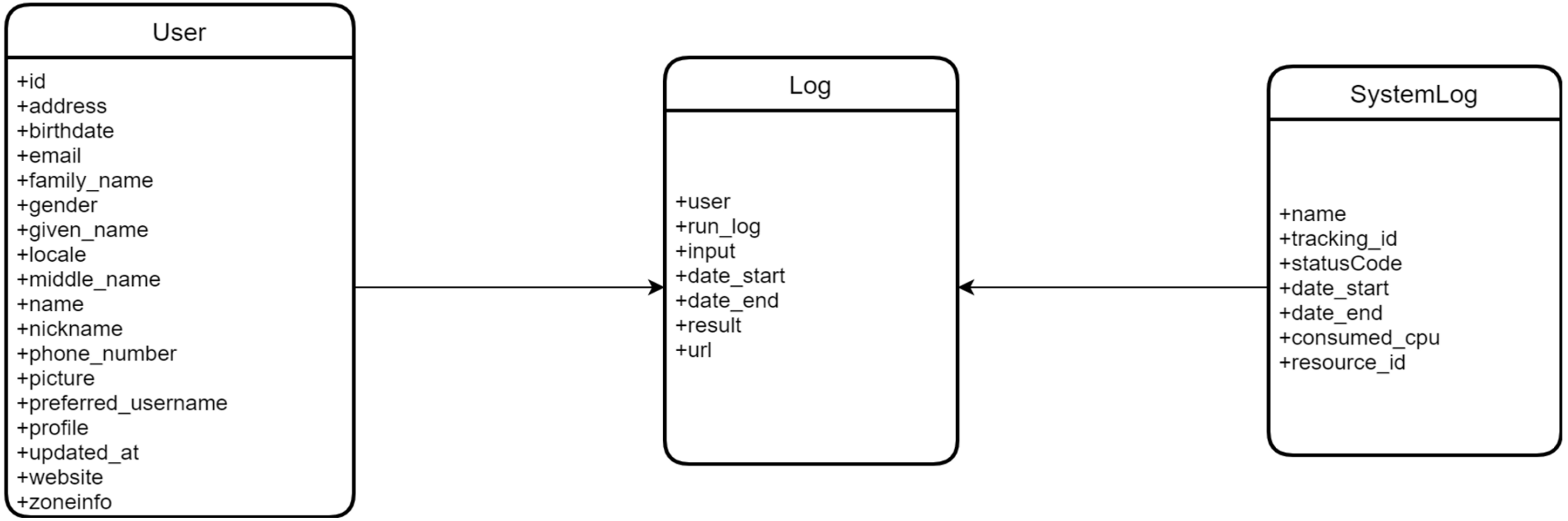
Приклад роботи моделі

Початок	Генерований результат
Robotics is an interdisciplinary research area at the interface…	Robotics is an interdisciplinary research area at the interface of engineering, computer science, and social science. Robotics is an interdisciplinary research area at the interface of engineering, computer science, biology and others.
The concept of creating robots that can operate autonomously…	The concept of creating robots that can operate autonomously has been explored for decades but has never been fully realized. The concept of creating robots that can operate autonomously from the human operator seems nearly a given now.

Демонстраційний плакат № 3
до магістерської дисертації на тему
„Гібридний алгоритм автоматичної генерації тексту”

Розробив: *Д.В. Кузнєцов*
Прийняв: *В.П. Пасько*

Схема баз даних



Демонстраційний плакат № 4
до магістерської дисертації на тему
„Гібридний алгоритм автоматичної генерації тексту”

Розробив: *Д.В. Кузнєцов*
Прийняв: *В.П. Пасько*

ДОДАТОК Б

КРЕСЛЕННЯ

					ІК-91.08 3132.003 Б1	
					Блок-схема архітектури генератору тексту	Літ. / Маса / Мірило
Зм.	Лист	№ докум	Підпис	Дата		
Розроб.		Кузнєцов Д.В				Лист 1 / Листів 1
Перев.		Пасько В.П				
Н.контр		Пасько В.П			Кафедра Технічної кібернетики	Група ІК-91мп
Затв.		Пархомей І.Р.				

Підпис і дата | Інв. № дубл. | Взам. інв. № | Підпис і дата | Інв. № ориг.

					ІК-91.08 3132.003 Б2	
Зм.	Лист	№ докум	Підпис	Дата	Модульна блок-схема генератору тексту	Літ. / Маса / Мірило
Розроб.		Кузнєцов Д.В				
Перев.		Пасько В.П				Лист 1 / Листів 1
Н.контр		Пасько В.П			Кафедра Технічної кібернетики	Група ІК-91мп
Затв.		Пархомей І.Р.				

ДОДАТОК В

АПРОБАЦІЯ РЕЗУЛЬТАТІВ ДОСЛІДЖЕНЬ

CERTIFICATE

is awarded to

Kuznietsov Denys

for being an active participant in
III International Scientific and Practical Conference

"WORLD SCIENCE: PROBLEMS, PROSPECTS AND INNOVATIONS"

24 Hours of Participation

TORONTO

25-27 November 2020

sci-conf.com.ua

ДОДАТОК Г

РЕЗУЛЬТАТИ ПЕРЕВІРКИ РОБОТИ НА СПІВПАДІННЯ

UNICHECK

Ім'я користувача:
Лісовиченко Олег Іванович

ID перевірки:
1005402399

Дата перевірки:
08.12.2020 15:31:54 EET

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
08.12.2020 15:33:49 EET

ID користувача:
76913

Назва документа: **Кузнєцов ІК-91мп_Співпад**

Кількість сторінок: **67** Кількість слів: **13900** Кількість символів: **105008** Розмір файлу: **173.67 KB** ID файлу: **1005694284**

1.11%
Схожість

Найбільша схожість: **0.52%** з Інтернет-джерелом (https://www.tensorflow.org/tutorials/text/text_generation?hl=ja)

0.88% Джерела з Інтернету	11	Сторінка 69
0.31% Джерела з Бібліотеки	35	Сторінка 69

0% Цитат

Вилучення цитат вимкнене

Вилучення списку бібліографічних посилань вимкнене

0%
Вилучень

Немає вилучених джерел

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Замінені символи	76