Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет комп'ютерних наук (або центр післядипломної освіти, або навчально-науковий центр заочної форми навчання)
(повна назва)

Кафедра програмної інженерії
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

Дослідження створення текстів згенерованих нейронною мережею та порівняння їх з природними
(тема)

Виконав:
студент (ка) 2 курсу, групи ІПЗм-22-2

Зуєтір М.Р.С.
(прізвище, ініціали)

Спеціальність 121 – Інженерія програмного забезпечення
(код і повна назва спеціальності)
Тип програми освітньо-наукова

Керівник доц. Вечур О.В.
(посада, прізвище, ініціали)

Допускається до захисту
Зав. кафедри _____ З.В.Дудар
(підпис) (прізвище, ініціали)

2024 р.

Харківський національний університет радіоелектроніки

Факультет комп'ютерних наук (або центр післядипломної освіти, або навчально-науковий центр заочної форми навчання)
Кафедра програмної інженерії
Рівень вищої освіти другий (магістерський)
Спеціальність 121 – Інженерія програмного забезпечення
Тип програми освітньо-наукова програма
Освітня програма Інженерія програмного забезпечення
(шифр і назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«___» _____ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Зуєтір Мухаммеда Рамі
(прізвище, ім'я, по батькові)

1. Тема роботи Дослідження методів створення текстів згенерованих нейронною мережею та порівняння їх з природними
Затверджена наказом по університету від 29.03. 2024р. № 250 Ст
2. Термін подання студентом роботи до екзаменаційної комісії 10.06.2024
3. Вихідні дані до роботи : наукові статті та публікації за темою роботи, відкриті наукові джерела в мережі Інтернет, приклади розробки нейронних мереж на тему роботи, програмна реалізація ПЗ для тестування і проведення аналізу на мові Python
4. Перелік питань, що потрібно опрацювати в роботі
мета роботи, аналіз предметної галузі і постановка задачі, огляд та аналіз літературних джерел з дослідження, дослідження теоретичне, дослідження практичне, тощо (може змінюватися відповідно до індивідуального завдання практики на розсуд керівника кваліфікаційної роботи та керівника практики)

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Аналіз предметної галузі та постановка задачі	23.01 – 14.02.24	*виконано*
2	Аналіз методів для дослідження	15.02 – 24.02.24	*виконано*
3	Аналіз існуючих методів та алгоритмів	17.02 – 28.02.24	*виконано*
4	Планування експериментів	25.02 – 28.02.24	*виконано*
5	Програмна реалізація кожної вибраної для дослідження нейронної мережі	25.02 – 01.04.24	*виконано*
6	Експериментальні дослідження	02.04 – 20.04.24	*виконано*
7	Аналіз результатів експериментальних досліджень та розробка рекомендацій	20.04 – 23.04.24	*виконано*
8	Написання та оформлення статті та тез доповіді	17.04 – 23.04.24	*виконано*
9	Підготовка пояснювальної записки	01.04 – 26.04.24	*виконано*
10	Підготовка презентації та доповіді	20.05 – 24.05.24	*виконано*
11	Нормоконтроль	25.05 – 28.05.24	*виконано*
12	Рецензування	28.05 – 30.05.24	*виконано*
13	Занесення диплома в електронний архів	31.05.2024	*виконано*
14	Попередній захист	2.06.2024	*виконано*
15	Допуск до захисту у зав. кафедри	5.06.2024	*виконано*

Дата видачі завдання 29 лютого 2024р.

Студент (ка) _____ Зуєтір М.Р.С.
(підпис)

Керівник роботи _____ доц. Вечур О..
(підпис) (посада, прізвище, ініціал)

.

ЗМІСТ

РЕФЕРАТ/ ABSTRACT

Пояснювальна записка до кваліфікаційної роботи містить 93 сторінки, 45 рисунків, 3 таблиці, 5 додатків, посилання на 19 джерел.

AI, CONTENT GENERATION, NEURAL NETWORKS, TEXT QUALITY, UNIQUENESS, COMPARATIVE ANALYSIS, HUMAN-AI INTERACTION, SURVEY, PYTHON, MACHINE LEARNING, QUALIFICATION WORK.

Об’єктом дослідження є процес автоматизованої генерації тексту за допомогою різноманітних моделей нейронних мереж. Робота зосереджена на оцінці якості та унікальності контенту, створеного цими моделями, порівняно з текстом, створеним людиною.

Мета дослідження полягає в дослідженні та навчанні різних нейронних мереж за допомогою навчального набору даних, а також у проведенні порівняльного аналізу здатності кожної мережі генерувати унікальний і якісний текстовий вміст у порівнянні з текстом, написаним людиною.

Дослідження використовує кількісні та якісні методи, включаючи використання Python для навчання та аналізу нейронної мережі, а також опитування для якісної оцінки. Дослідження також використовує декілька архітектур нейронної мережі та інструменти аналізу тексту.

Результати розкривають суттєве уявлення про можливості нейронних мереж у створенні тексту. Дослідження визначає найефективнішу модель нейронної мережі у створенні контенту, який конкурує з людською якістю та унікальністю. Ця робота представляє елементи наукової новизни, надаючи порівняльний аналіз різних моделей штучного інтелекту проти генерування людського контенту, відносно недослідженої області.

Ключові результати включають детальну оцінку продуктивності нейронних мереж у створенні тексту та сприйняття людиною контенту, створеного ШІ. Результати дослідження знайдуть практичне застосування в цифрових медіа, маркетингу та інших секторах, які покладаються на створення контенту.

Робота робить внесок у ширшу сферу штучного інтелекту та створення контенту, пропонуючи рекомендації щодо використання моделей нейронних мереж у завданнях створення контенту. Він закладає основу для майбутніх досліджень щодо покращення можливостей штучного інтелекту в цій галузі.

Економічна та соціально-економічна ефективність роботи полягає в її потенціалі для оптимізації процесів створення контенту, скорочення витрат часу та ресурсів при збереженні високої якості контенту.

Важливість цього дослідження полягає в його комплексній оцінці ролі штучного інтелекту у створенні контенту, сприяючи як академічним дослідженням, так і практичним застосуванням.

Висновки підкреслюють доцільність і бажаність продовження досліджень у цій галузі, зокрема в удосконаленні моделей штучного інтелекту для створення більш тонкого та контекстно-залежного контенту.

AI, CONTENT GENERATION, NEURAL NETWORKS, TEXT QUALITY, UNIQUENESS, COMPARATIVE ANALYSIS, HUMAN-AI INTERACTION, SURVEY, PYTHON, MACHINE LEARNING, QUALIFICATION WORK.

The object of research is the process of automated text generation using various models of neural networks. The work focuses on evaluating the quality and uniqueness of content generated by these models compared to human-generated text.

The purpose of the study is to investigate and train different neural networks using a training dataset and to benchmark each network's ability to generate unique and high-quality textual content compared to human-written text.

The research uses quantitative and qualitative methods, including the use of Python for neural network training and analysis, and surveys for qualitative assessment. The study also uses several neural network architectures and text analysis tools.

The results reveal a significant insight into the capabilities of neural networks in text generation. The study identifies the most effective neural network model in creating content that rivals human quality and uniqueness. This work presents

elements of scientific novelty, providing a comparative analysis of different models of artificial intelligence against human content generation, a relatively unexplored area.

Key findings include a detailed assessment of the performance of neural networks in text generation and human perception of AI-generated content. The results of the research will find practical applications in digital media, marketing and other sectors that rely on content creation.

The work contributes to the broader field of artificial intelligence and content creation by offering guidelines for the use of neural network models in content creation tasks. It lays the groundwork for future research to improve the capabilities of artificial intelligence in this field.

The economic and socio-economic efficiency of the work lies in its potential for optimizing content creation processes, reducing time and resource costs while maintaining high content quality.

The importance of this study lies in its comprehensive assessment of the role of artificial intelligence in content creation, contributing to both academic research and practical applications.

The conclusions emphasize the expediency and desirability of continuing research in this field, in particular in the improvement of artificial intelligence models for creating more subtle and context-dependent content.

Заява щодо самостійного виконання кваліфікаційної роботи та можливості її публікації в електронному архіві відкритого доступу ElArKhNURE.

Я, Зуєтір Мухаммед Рамі Самірович, студент гр. ІПЗм-22-2, здобувач вищої освіти на другому (магістерському) рівні кафедри «Програмна інженерія», заявляю: моя кваліфікаційна робота на тему «Дослідження методів створення текстів згенерованих нейронною мережею та порівняння їх з природними», що буде представлена в екзаменаційну комісію для публічного захисту, виконана самостійно, в ній не містяться елементи плагіату і вона може бути опублікована в електронному архіві відкритого доступу ElArKhNURE.

Всі запозичення з друкованих та електронних джерел мають відповідні посилання.

Я ознайомлений(на) з діючим положенням «Про протидію академічному плагіату в ХНУРЕ», згідно з яким виявлення плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту та застосування дисциплінарних заходів.

ПЕРЕЛІК СКОРОЧЕНЬ

ШІ – штучний інтелект

GPT- Generative Pre-trained Transformer

BERT - Bidirectional Encoder Representations from Transformers

LaMDA - Language Model for Dialogue Applications

LLama 2 - Large Language Model Meta AI

LLMs - Large language model

LSTM - Long short-term memory

NLP - Natural language processing

ВСТУП

У галузі штучного інтелекту (ШІ), яка швидко розвивається, здатність нейронних мереж генерувати текстовий контент стала центральною сферою досліджень. Це дослідження під назвою «Дослідження якості штучного інтелекту в питаннях формування текстового контенту та можливості конкуренції з людьми» заглиблюється в цю сферу. Суть цієї наукової проблеми полягає в оцінці того, якою мірою штучний інтелект може не тільки імітувати, але й потенційно перевершити людські можливості у створенні тексту.

Актуальність цього дослідження ґрунтується на зростаючій залежності від ШІ для створення контенту в різних секторах. Оскільки цифрові медіа та онлайн-комунікації експоненціально розширюються, попит на якісний, унікальний контент зростає. У цьому дослідженні розглядається критична потреба оцінити здатність ШІ відповідати цим вимогам, тим самим вносячи значний внесок у розвиток ШІ та його застосування у створенні контенту.

Ця робота узгоджується з поточними дослідницькими програмами у ПІ, зокрема тими, що вивчають передові програми ШІ та взаємодію людини з комп’ютером. Він розширює дослідницьку спрямованість відділу, надаючи емпіричне розуміння можливостей нейронних мереж у створенні тексту, таким чином усуваючи розрив між теоретичними моделями ШІ та їх практичним застосуванням.

Основна мета цього дослідження полягає в ретельному дослідженні та навчанні різних нейронних мереж за допомогою призначеного навчального набору даних. Цілі включають проведення порівняльного аналізу цих мереж на основі унікальності та якості генерації тексту в порівнянні з текстом, написаним людиною, оціненим за допомогою опитувань. Такий підхід забезпечує комплексну оцінку компетенції ШІ у створенні контенту.

Об’єктом дослідження є процес автоматизованої генерації тексту за допомогою нейронних мереж.

У цій роботі використовується поєднання кількісних і якісних методів дослідження. Кількісний аналіз передбачає навчання нейронних мереж і оцінку їх результату за допомогою встановлених показників якості та унікальності тексту. Якісний аналіз проводиться за допомогою опитувань, щоб оцінити сприйняття людьми контенту, створеного ШІ. Цей методологічний підхід забезпечує цілісне розуміння можливостей ШІ у створенні тексту.

Новизна цього дослідження полягає в його порівняльному аналізі багатьох нейронних мереж і подальшому визначенні найбільш ефективної моделі для генерації високоякісного унікального контенту. Це дослідження робить внесок у цю сферу, надаючи розуміння потенціалу ШІ конкурувати з людськими можливостями у створенні контенту. Його практичне значення підкреслюється його застосовністю в галузях, які залежать від генерації контенту, пропонуючи основу для стратегій контенту, керованих ШІ.

Ця робота ґрунтується на огляді поточної літератури, що охоплює дослідження архітектури нейронних мереж, алгоритмів генерації тексту та взаємодії людини та штучного інтелекту під час створення контенту.

Таким чином, це дослідження має на меті забезпечити оцінку можливостей ШІ у формуванні текстового контенту, пропонуючи значний внесок як в академічну, так і в практичну сфери штучного інтелекту та генерації контенту.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА АКТУАЛЬНІСТЬ ДОСЛІДЖЕННЯ

1.1 Аналіз предметної галузі

Швидкий розвиток штучного інтелекту, особливо в обробці природньої мови, започаткував нову еру генерації текстового контенту. Це відкрило можливості для використання штучного інтелекту в різних програмах, починаючи від автоматизованої журналістики та закінчуючи створенням персоналізованого контенту. Проте, здатність цих моделей штучного інтелекту генерувати текст, який відповідає людській якості, залишається предметом інтенсивних досліджень і дискусій.

Детальний огляд, як наукової літератури, показує значну увагу до розробки моделей нейронних мереж, таких як GPT (Generative Pre-trained Transformer), BERT (Bidirectional Encoder Representations from Transformers) та інших алгоритмів машинного навчання. Ці дослідження в першу чергу стосуються технічних аспектів генерації тексту штучним інтелектом, таких як навчання моделі мови, оптимізація алгоритмів для кращої взаємодії та зменшення лінгвістичної надмірності. Попри виняткові досягнення, існують суперечності в поточних теоретичних і експериментальних результатах. Деякі дослідження показують, що текст, створений ШІ, наближається до людського рівня розуміння та послідовності, тоді як інші вказують на помітні недоліки, особливо в чутливості до контексту та фактичній точності. Ця розбіжність є переконливою підставою для більш тонкого та порівняльного аналізу.

У сфері генерації тексту кілька моделей штучного інтелекту показали перспективні результати. Такі моделі, як GPT-3, продемонстрували здатність генерувати зв'язний і відповідний до контексту текст, що не має аналогів. Однак, існує потреба у порівняльному аналізі цих моделей, щоб оцінити їхню ефективність у різних сценаріях генерації контенту. Цей аналіз буде зосереджений на таких параметрах, як лінгвістична точність, узгодженість стилю та адаптованість до різних жанрів письма.

Всупереч досягненням, генерація тексту ШІ стикається з кількома проблемами. Однією з першочергових проблем є здатність моделі підтримувати фактичну точність і уникати створення оманливого або упередженого вмісту. Ще одним викликом є етичні наслідки створеного штучним інтелектом контенту, особливо у сферах, які значною мірою залежать від автентичності та людського досвіду[1].

Рисунок 1.1 – Роль навчальних даних у продуктивності ШІ (виконано самостійно)

Якість і різноманітність навчальних даних відіграють вирішальну роль у продуктивності моделей ШІ. Також треба врахувати потреби в більш інклюзивних і різноманітних наборах даних для навчання моделей штучного інтелекту, щоб гарантувати, що створений контент є не лише точним, але й вільним від упереджень. Включення людського зворотного зв’язку в процес навчання ШІ стає життєво важливим підходом до підвищення якості тексту, створеного ШІ. Методологія «людина в циклі» дозволяє вдосконалювати результати штучного інтелекту, забезпечуючи більшу відповідність очікуванням і стандартам людини. Застосування ШІ у створенні контенту має глибокі наслідки для різних галузей. У журналістиці ШІ може допомогти в написанні звітів і статей, а в маркетингу його можна використовувати для створення персоналізованого контенту. Розуміння впливу ШІ в цих секторах має вирішальне значення для оцінки його ширшої користі. Основним завданням є розробка нейронної мережі, навченої на визначеному наборі даних

для створення текстового вмісту. Наукове дослідження передбачає аналіз створеного тексту на такі якості, як точність, зв’язність і граматична правильність. Технічний аспект охоплює розгортання моделей ШІ та інструментів для аналізу продуктивності генерації тексту.

Мета полягає в тому, щоб подолати розрив у розумінні того, наскільки текст, створений штучним інтелектом, близький до контенту, написаного людьми. Це дослідження має на меті не лише оцінити можливості штучного інтелекту, але й вивчити ширші наслідки штучного інтелекту для створення контенту, включаючи етичні міркування та академічну доброчесність. Дослідження включає, як теоретичні дослідження, включаючи алгоритмічні основи та стратегії навчання моделей, так і експериментальні дослідження, які включають фактичне навчання нейронних мереж та аналіз їх результатів. Цей комплексний підхід забезпечує надійне розуміння можливостей генерації тексту ШІ[2]. Результати будуть проаналізовані, щоб визначити сильні та слабкі сторони та потенційні сфери вдосконалення генерації тексту ШІ. Це дасть змогу визначити напрямок подальших досліджень, зокрема щодо вдосконалення тонкощів контенту, створеного ШІ.

Дослідження охоплює технологію, яка використовується для навчання нейронної мережі, етапи розробки моделей штучного інтелекту та, якщо це можливо, опис розробленої системи програмного забезпечення.

Сфера генерації тексту ШІ готова до значного прогресу. Інтеграція більш просунутих алгоритмів разом з акцентом на етичних практиках штучного інтелекту сформує майбутнє цієї технології. Це дослідження вивчатиме ці майбутні напрямки та їхні потенційні наслідки. Цей аналіз закладає основу для комплексного дослідження текстового контенту, створеного ШІ. Вивчаючи поточний стан технологій, виклики та майбутній потенціал, це дослідження має на меті забезпечити цілісне уявлення про можливості та обмеження ШІ у створенні контенту. Кінцева мета полягає в тому, щоб зрозуміти, як штучний інтелект може доповнити людську креативність і ефективність у створенні контенту, і якою мірою йому можна довіряти для незалежного створення

контенту[3]. Результати цього дослідження можуть бути використані в багатьох сферах, зокрема в цифровій журналістиці, контент-маркетингу та наукових дослідженнях. Отримані результати дадуть цінну інформацію розробникам, дослідникам і творцям контенту щодо використання ШІ для ефективного створення тексту.

1.2 Аналіз аналогів та актуалізація рішень

Сфера генерації тексту штучним інтелектом засвідчила безліч досліджень і розробок, що потребує всебічного аналізу аналогів і актуалізації рішень у цій галузі. Дослідження генерації тексту штучним інтелектом були задокументовані, як у науковій літературі, так і в патентах. Ключові розробки включають різні архітектури нейронних мереж та їх застосування в обробці природної мови. Патенти виявляють увагу до інноваційних методів для підвищення ефективності та точності моделей генерації тексту. Сучасний стан розв'язування проблеми генерації тексту ШІ відзначається значним прогресом і певними обмеженнями. Дослідження показали прогрес у створенні зв’язного та відповідного контексту тексту, але залишаються такі проблеми, як дотримання фактичної точності та уникнення упереджень. Суперечності в результатах між різними моделями та підходами вказують на необхідність більш тонкого розуміння цих систем.

У сфері штучного інтелекту, зокрема в обробці природної мови, розробка моделей генерації тексту, таких як LaMDA і Llama 2, є значним кроком уперед. LaMDA, розроблений Google, базується на архітектурі Transformer, пристосованій для генерації розмовного тексту. Вона досягає успіху у створенні багатих на контекст і зв'язних діалогів, що є результатом її навчання на різноманітних і великих масивах даних. Набори даних, що використовуються для навчання LaMDA, є великими та діалогово-орієнтованими. Вони включають великомасштабні набори розмовних даних, отримані з різних діалогових ситуацій. Розмір і якість цих наборів даних

підібрані таким чином, щоб охопити широкий спектр розмовних тем і стилів, забезпечуючи майстерність моделі у створенні діалогів. Основна перевага LaMDA полягає в її тонкому розумінні нюансів розмови та здатності зберігати контекст протягом тривалої взаємодії. Вона вміє швидко обробляти та відповідати на запити користувачів, зберігаючи релевантність і контекст. Спеціалізоване навчання моделі дозволяє їй ефективно обробляти великий обсяг розмовних запитів з мінімальною затримкою. Однак ця спеціалізація на діалогах може обмежити його адаптивність і продуктивність у ширших завданнях генерації текстів, де потрібне створення різноманітного контенту.

З іншого боку, Llama 2 пропонує інший підхід. Її розроблено з урахуванням ширшої сфери застосування. Вона має на меті бути більш універсальною для різних завдань генерації тексту. Llama 2 вміє працювати з широким спектром тем і стилів: від наукового тексту до художніх творів. На відміну від Llama 1, Llama 2 навчається на більш неоднорідних наборах даних. Вони включають не лише діалоги, а й розповідні тексти, фактичний контент і зразки творчого письма. Різноманітність типів, розмірів і якості даних гарантує, що Llama 2 здатна впоратися з різними завданнями генерації текстів, а не лише з діалогами. Хоча вона може не зрівнятися з LaMDA за швидкістю реакції у специфічних для діалогу сценаріях, вона компенсує це своєю здатністю обробляти різні типи запитів, такі як текстові, аудіо та зображення. Її продуктивність у різних завданнях генерації тексту характеризується балансом між швидкістю та контекстною точністю.

Реалізація можливостей генерації тексту в цих моделях підкреслює властиві їм компроміси. Реалізація LaMDA добре налаштована для діалогів, що робить її високоефективною у сценаріях, які вимагають розмовного інтелекту, але потенційно менш ефективною у завданнях, які вимагають ширшого обсягу генерації контенту. І навпаки, підхід Llama 2, забезпечуючи широке застосування, може не досягти такого ж рівня майстерності у створенні контекстно-глибоких діалогів, як LaMDA[3] (див. табл. 1.1).

Таблиця 1.1 – Порівняльна характеристика генеративних можливостей LaMDA та llama 2 (виконана самостійно)

Характеристика	Llama 2	LaMDA
Мета	Генерація текстових повідомлень	Чат-бот
Данні для навчання	Понад 200 мільярдів текстових повідомлень	Текст і код, включаючи людські розмови
Сильні сторони	Природна та розмовна генерація текстових повідомлень, відповіді на запитання	Захоплива та інформативна генерація текстових повідомлень, відповіді на запитання
Розмір моделі (параметрів)	200 мільярдів	137 мільярдів
Розмір даних для навчання (токени)	200 мільярдів	137 мільярдів
Швидкість	Швидка	Швидка
Характеристика	Llama 2	LaMDA
Точність	Висока	Висока
Універсальність	Може використовуватися для різноманітних завдань, включаючи генерацію коду, наукових статей, математичних формул.	Створює різні типи діалогів, повідомлень та літературних творів різного характеру.

Таким чином, порівняння LaMDA і Llama 2 ілюструє різноманітні можливості та обмеження, притаманні моделям генерації тексту. Спеціалізація

LaMDA на завданнях, орієнтованих на діалог, дає явні переваги в розмовному контексті, тоді як універсальний підхід Llama 2 забезпечує ширше застосування до різних типів текстів.

1.3. Актуальність та мета дослідження

Поява великих мовних моделей (LLMs), таких як ChatGPT, LaMDA і Llama 2, зробила революцію в галузі обробки природної мови, дозволивши генерувати текст подібно людським можливостям. Актуальність цих моделей полягає в тому, що вони здатні докорінно змінити те, як ми взаємодіємо зі штучним інтелектом і використовуємо його в обробці природної мови.

Основна мета — оцінити та порівняти можливості генерації тексту моделями NLP, зокрема, в контексті їхньої ефективності, складності та застосовності порівняно з можливостями написання тексту людиною. NLP моделі мають кілька переваг над написанням текстів людиною, зокрема здатність швидко обробляти та аналізувати великі набори даних, послідовність у збереженні тону і стилю, а також можливість генерувати текст різними мовами та форматами. Їхні вдосконалені алгоритми дозволяють створювати контекстно-релевантний і стилістично різноманітний текст. Всупереч їхнім можливостям, реалізація цих моделей пов'язана з певними труднощами. Серед них — забезпечення етичного використання ШІ, уникнення упередженості при генерації тексту, а також управління складністю розуміння та інтерпретації людських емоцій і тонкощів[4].

Розробка та навчання таких складних моделей вимагає значних інвестицій. Витрати включають обчислювальні ресурси для обробки великих масивів даних і поточні витрати на вдосконалення та підтримку моделі. Навчання також вимагає великих і різноманітних наборів даних для забезпечення навчання та адаптивності. Методологія дослідження передбачає порівняльний аналіз NLP моделей у різних сценаріях. Це включає тестування їхньої точності відповідей, адаптивності до різних стилів письма та здатності

вирішувати складні лінгвістичні завдання. Підхід також включає оцінку ефективності навчання моделей та їхньої адаптивності до мовних тенденцій, що розвиваються. Потенційні результати цього дослідження включають покращення взаємодії користувачів із системами штучного інтелекту, підвищення ефективності створення контенту та вдосконалення автоматизованих завдань на основі мови.

Насамкінець, це дослідження має на меті забезпечити розуміння можливостей, і потенційних покращень генеративних текстових NLP моделей, створюючи основу для їхнього розширеного застосування в різних сферах обробки мови.

1.4 Постановка задачі

Основною метою цього дослідження є розробка та оцінка моделі обробки природної мови (NLP) за допомогою Python. Модель спрямована на вирішення конкретних лінгвістичних завдань і проблем, потенційно підвищуючи ефективність і точність аналізу та генерації текстів. Розробка моделей NLP є актуальною у сучасному світі, де обробка та розуміння великих обсягів текстових даних є критично важливими для різних застосувань. Це включає в себе аналіз настроїв, мовний переклад, узагальнення контенту. Python з його великою кількістю бібліотек та фреймворків для машинного навчання та NLP моделі є ідеальною мовою для цих цілей.

Цілі цього дослідження полягають у наступному:

- розробити власну модель на основі архітектури моделей NLP, яка може ефективно обробляти та інтерпретувати масиви даних текстової інформації;
- надати моделі можливість виконувати конкретні завдання, такі як категоризація тексту, розпізнавання об'єктів та генерування відповідей, подібних до людських;

- оцінити продуктивність моделі з погляду швидкості, точності та адаптивності до різних текстових жанрів і стилів.

Модель буде побудована з використанням сучасних методів, які можуть включати, але не обмежуватися токенізацією частин мови, розпізнаванням іменованих сутностей та генеруванням тексту на основі нейронних мереж. Для реалізації цих методів будуть використані бібліотеки Python, такі як NLTK, SpaCy та TensorFlow або PyTorch.

Набори даних будуть отримані з різних галузей, щоб забезпечити універсальність моделі. Це можуть бути загальнодоступні набори даних, такі як новини, пости в соціальних мережах, наукові статті та література. Дані будуть попередньо оброблені та анотовані за необхідності для навчання та тестування моделі. Продуктивність моделі буде порівнюватися з чинними моделями та бенчмарками NLP. Цей аналіз буде зосереджений на таких показниках, як точність, швидкість обробки та здатність обробляти нюанси мови. Крім того, буде оцінено здатність моделі адаптуватися до нових наборів даних і мов. Вибір моделі базуватиметься на кількох критеріях, включаючи:

- точність і надійність обробки тексту та виконання завдань;
- ефективність в обробці великих і різноманітних наборів даних;
- гнучкість і масштабованість для адаптації до різних завдань NLP і мовних патернів тексту;
- простота інтеграції в наявної системи та додатки.

На закінчення, це дослідження має на меті зробити внесок в область NLP моделей шляхом розробки власної моделі на основі мови програмування Python, яка має переваги в обробці та аналізі текстових даних.

2 АНАЛІЗ МЕТОДІВ ДЛЯ ДОСЛІДЖЕННЯ

2.1 Огляд та вибір датасету

Якість та представництво набору даних безпосередньо впливають на точність та ефективність моделі. Основа ефективного набору даних полягає у його здатності точно відображати задачу, яку має вирішувати нейронна мережа. Це означає, що набір даних має включати різноманітні типи даних, які можуть зустрічатися в реальних умовах. Наприклад, якщо мета — розпізнавання образів, набір даних має містити зображення різних об'єктів у різних умовах освітлення та під різними кутами. Величина набору даних є критичною. Великий об'єм даних забезпечує кращу узагальненість моделі та знижує ризик перенавчання. З іншого боку, малий набір даних може призвести до низької ефективності навчання та недостатньої точності моделі. Оптимальний розмір набору даних залежить від складності задачі та можливостей мережі[7].

Коректність міток у наборі даних є вирішальною. Некоректно позначені дані можуть спотворити процес навчання, викликаючи помилки в моделі. Тому необхідно забезпечити високу якість міток, щоб модель могла точно інтерпретувати вхідні дані. Для завдань класифікації та сегментації важливим є збалансований набір даних. Це означає, що кожен клас має бути представлений приблизно однаковою кількістю прикладів. Незбалансованість може призвести до упередженості моделі та зниження її точності у недостатньо представлених класах. Набір даних повинен містити широкий спектр варіацій: різні стилі, розміри, освітлення, пози та інше. Це дозволяє моделі навчитися адаптуватися до різних сценаріїв, які можуть виникнути у реальному світі. Така різноманітність підвищує універсальність моделі.

Актуальність набору даних гарантує, що модель буде ефективною на сучасних даних. Застарілі або нерелевантні дані можуть знизити її ефективність при роботі з новими даними[8]. Тому набір даних має постійно оновлюватися та відповідати сучасним вимогам. Збір та використання даних

повинні відповідати законодавчим нормам. Це включає захист конфіденційної та особистої інформації. Важливо використовувати набір даних, які не містять образливих чи неприйнятних матеріалів. Вибір набору даних є вирішальним для успіху нейронної мережі. Урахування цих факторів дозволяє створити набір даних, який ефективно підготує модель до реальних умов застосування. Не слід недооцінювати значення будь-якого з цих аспектів, оскільки вони взаємопов'язані та впливають на загальну ефективність та точність моделі.

2.2 Огляд обраного інструменту для дослідження

Наступним кроком буде аналіз використання нейронних мереж для генерації текстового контенту, який оцінює їхню здатність конкурувати з чинними моделями. Ми розглянемо різні моделі нейронних мереж, такі як LSTM, Seq2seq, KerasNLP, Біграм та Трансформер в контексті їх здатності генерувати якісний текст[5].

Важливим критерієм для нейронних мереж є якість генерованого контенту. Моделі оцінюються на основі оригінальності, стилістичної точності та релевантності тексту. Моделі, які використовують глибоке навчання, зазвичай демонструють високу здатність до генерації креативного та релевантного контенту (див. рис. 2.1).

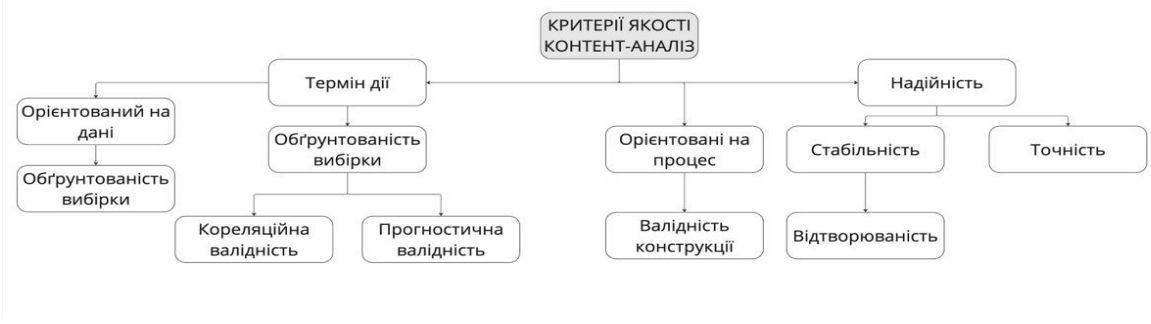


Рисунок 2.1 – Оцінка якості генерації контенту (виконано самостійно)

Моделі також оцінюються за їхню здатність імітувати стиль певного автора. Це включає в себе використання специфічного словникового запасу, граматичних конструкцій та тону. Наприклад, KerasNLP та Трансформер

відомі своєю здатністю точно адаптуватися до різних стилів письма (див. рис. 2.2).

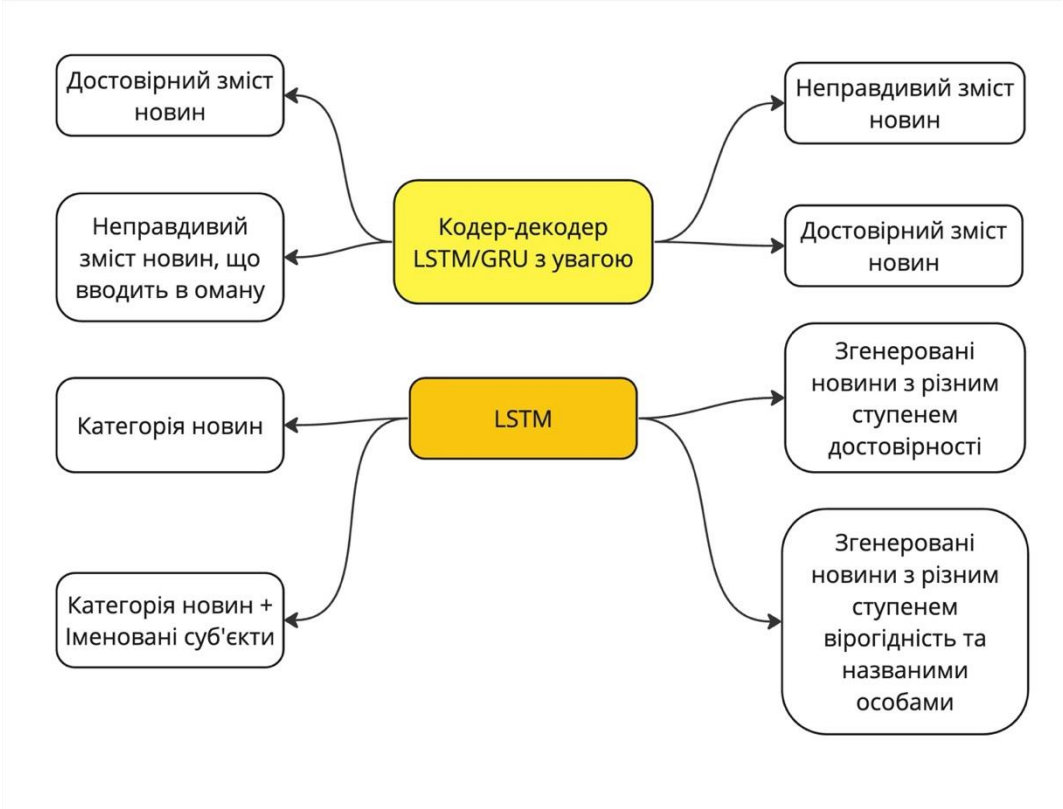


Рисунок 2.2 – Стилістична точність генерованого контенту (виконано самостійно)

Іншим важливим аспектом є різноманітність та оригінальність контенту, що генерується. Моделі повинні вміти створювати унікальний контент, який не є простим дублюванням вже існуючих текстів. В цьому випадку, моделі, які базуються на механізмах уваги, показують кращі результати (див. рис. 2.3).

Рисунок 2.3 – Різноманітність та оригінальність контенту (виконано самостійно)

Швидкість генерації контенту та адаптивність до різних тем є ключовими для використання у реальному часі. Моделі, такі як Трансформер, вирізняються своєю здатністю швидко адаптуватися та генерувати відповідні тексти за короткий час[5] (див. рис. 2.4).

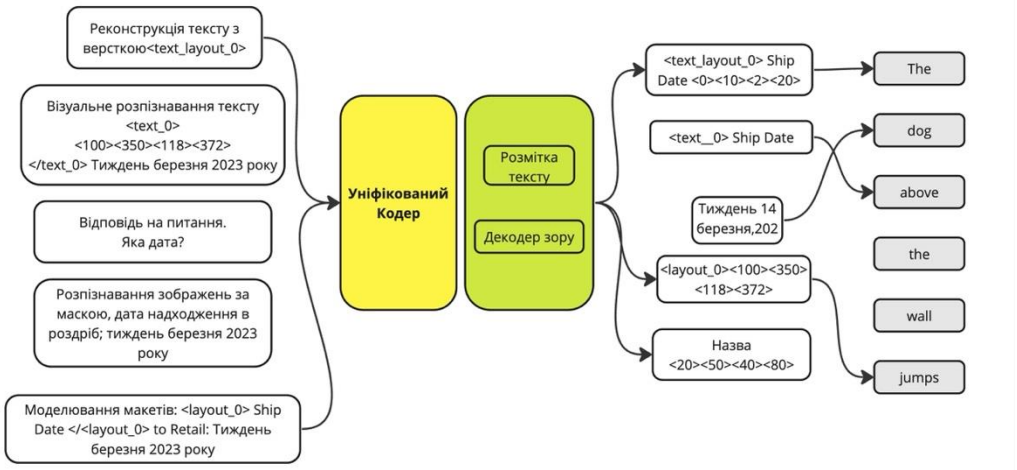


Рисунок 2.4 – Швидкість генерації та адаптивність текстової моделі (виконано самостійно)

Важливою характеристикою є взаємодія моделі з користувачем, зокрема її здатність розуміти запити та адекватно на них реагувати[6]. Моделі, які включають компоненти штучного інтелекту, демонструють значно кращу взаємодію з користувачами, розуміючи їх запити та надаючи підходящі відповіді. Це підвищує здатність моделі адаптуватися до різних сценаріїв використання (див. рис. 2.5).

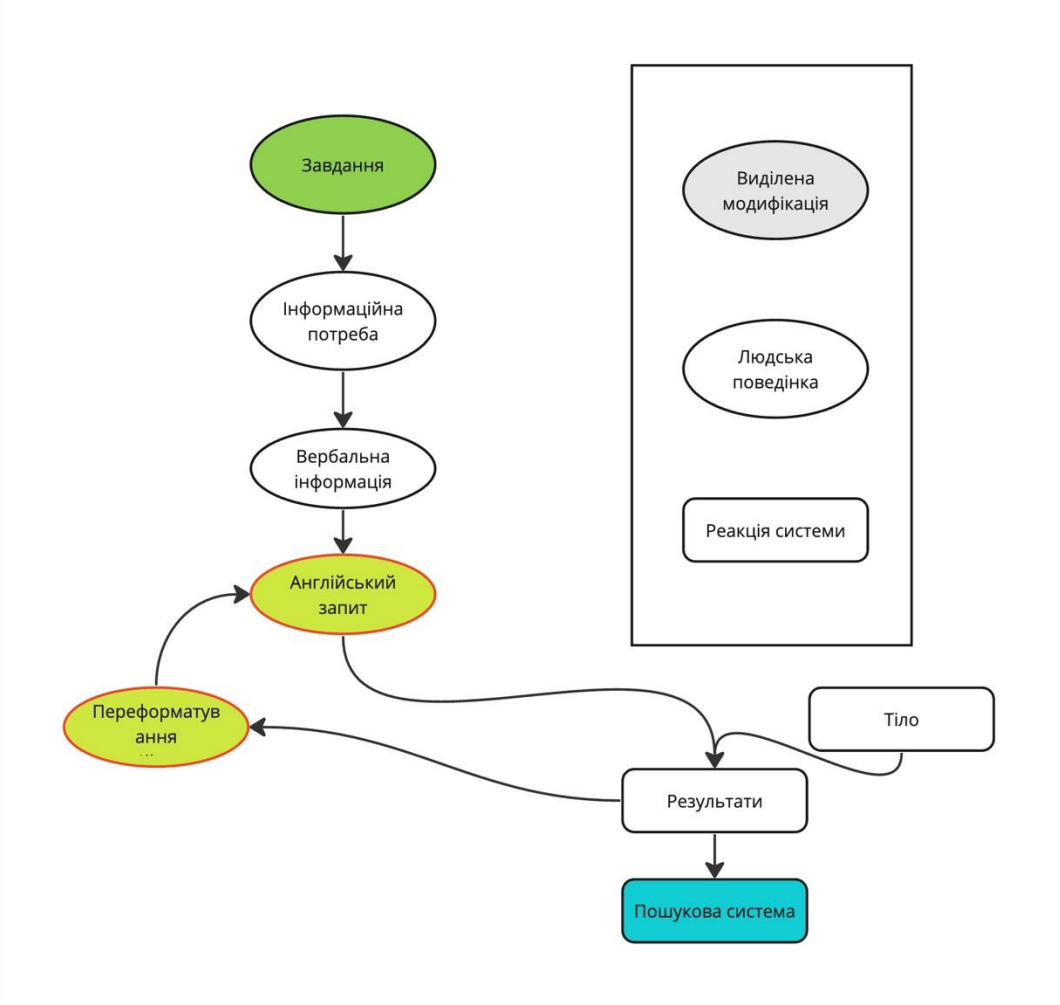


Рисунок 2.5 – Взаємодія з користувачем (виконано самостійно)

У сучасному світі ШІ існує безліч моделей, які вирішують різні завдання, пов'язані з обробкою природньої мови. Кожна з цих моделей має свої унікальні характеристики та застосування, що робить їх цінними для конкретних типів задач. Нижче розглянемо декілька ключових моделей, їх переваги та сферу застосування (див. рис. 2.6).

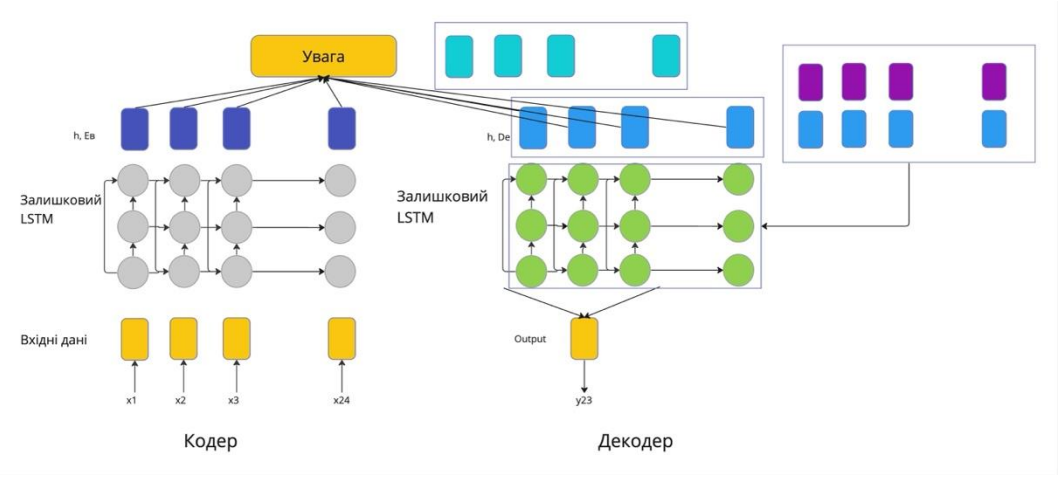


Рисунок 2.6 – Аналіз моделей LSTM (виконано самостійно)

- LSTM - Ці моделі ефективні для задач, пов'язаних з послідовністю даних, таких як переклад тексту. Вони добре адаптуються до контексту і забезпечують зрозумілість на рівні фрази;
- Біграм - Модель, націлена на швидке та ефективне оброблення великих об'ємів даних. Вона особливо корисна для задач, які потребують швидкого аналізу та відповідей;
- Трансформер - Одна з найбільш розширених моделей для генерації тексту, відома своєю здатністю до швидкої адаптації та високоякісної генерації контенту. Її функціональність робить її ідеальною для складних задач, таких як створення творчого контенту.

На основі нашого дослідження ми провели аналіз здатності різних моделей нейронних мереж до ефективної генерації тексту. Важливо відзначити, що кожна модель має свої особливості, які впливають на її придатність до конкретних задач. Розглядаючи моделі, як інструменти дослідження, ми оцінюємо їх здатність до швидкого реагування на запити, генерації контенту, який є релевантним та оригінальним, та їхню спроможність адаптуватися до різних стилів та тем[4].

Кожна з цих моделей має свої сильні та слабкі сторони, тому важливо обрати найбільш відповідну модель, залежно від специфіки досліджуваної задачі. Залежно від потреб, можна використовувати моделі для швидкої

обробки великих обсягів інформації, детального аналізу контексту, або ж для творчого створення контенту (див. таб. 2. 1).

Таблиця 2.1 – Критерії якості генеративних можливостей NLP моделей

	Опис	Метрики оцінки	Оцінювання	Ваговий коефіцієнт
Якість генерації контенту	Оцінка текстового матеріалу, на лексичну правильність, структуру та логіку висловлювань.	Такі метрики BLEU, ROUGE, або METEOR, які оцінюють подібність між згенерованим текстом та еталоном.	Людська експертиза	0,25
Схожість із стилем автора:	Визначення того, наскільки згенерований контент відповідає стилістиці.	Використання алгоритмів аналізу стилів тексту.	Метрики глибинного навчання	0,1
Різноманітність та оригінальність	Врахування різноманітності уникнення стереотипів та банальності.	Застосування метрик для оцінки різноманітності тексту, наприклад, використання N-gram аналізу	Людська експертиза	0,25

Кінець таблиці 2.1

	Опис	Метрики оцінки	Оцінювання	Ваговий коєф.
	мережа здатна створювати текст	для створення контенту	реальному часі	
Адаптивність до тематики	Перевірка здатності системи адаптуватися до різних тематик та генерувати відповідний контент.	Створення тестових сценаріїв та оцінка здатності системи генерації адаптуватися до різних тематик.	Людська експертиза	0,1
Взаємодія з користувачем	Оцінка можливості нейронної мережі взаємодіяти з користувачем, приймаючи відповіді на питання або реагуючи на коментарі.	Організація експериментів, де оцінюється зручність та ефективність взаємодії користувача з системою.	Людська експертиза	0,2

2.3 Аналіз обраних методів для дослідження

У рамках цього дослідження ми аналізуємо кілька методів нейронних мереж, включаючи Seq2Seq, LSTM, BERT, OpenAI Ada та GPT-3-TURBO. Кожен метод має свої характеристики та застосування, що дозволяє

забезпечити оптимальний підхід до вирішення завдань генерації тексту. Наведемо альтернативи прикладу моделей для оцінювання:

- seq2Seq: це нейромережа, яка здатна генерувати тексти на основі великого обсягу текстових даних. Вона використовує дві нейромережі – енкодер та декодер, які працюють спільно, щоб виробляти високоякісні тексти. Seq2Seq може використовуватися для створення статей, новин, книг, а також для роботи з будь-якими іншими текстовими матеріалами;
- LSTM (Long Short-Term Memory) – це нейромережа, яка здатна генерувати тексти будь-якої складності та будь-якою мовою. Вона використовує спеціальні алгоритми, щоб зберігати інформацію про попередні вхідні дані та використовувати її для генерації наступного елемента тексту. LSTM може використовуватися для створення книг, статей та інших текстових матеріалів[3];
- BERT (Bidirectional Encoder Representations from Transformers) – це нейромережа, яка здатна генерувати тексти, використовуючи структуру "transformer". Ця нейромережа здатна генерувати високоякісні тексти на будь-яку тему, використовуючи контекст інформації. BERT може використовуватися для створення текстів будь-якою мовою, а також для роботи з різними видами текстових матеріалів;
- OpenAI Ada є ще однією потужною нейромережею для створення текстів. Ця нейромережа використовує технологію глибокого навчання, що дозволяє швидко обробляти величезні обсяги даних. OpenAI Ada здатна генерувати якісні тексти на будь-яку тему, що робить цю нейромережу корисною для роботи в різних сферах;
- GPT-3-TURBO є однією з найвідоміших і найпотужніших нейромереж для генерації тексту. Ця нейромережа навчена великому обсягові текстового матеріалу, що дозволяє їй генерувати тексти будь-якої складності. Нейромережа здатна створювати статті,

новини, книги, а також редагувати та перекладати тексти. GPT-3-TURBO – це найкраща нейромережа для роботи зі складними текстами.

Вид моделі рішення:

Лінійна адаптивна згортка з ваговими коефіцієнтами та нормуючими показниками

Оцінювання:

- людська експертиза: від 1 до 10;
- метрики глибинного навчання: від 0,00 до 1,00;
- оцінка ефективності в реальному часі: від 0,00 до нескінченності;
- за формулою 2.1 для нормалізації усіх значень від 0 до 1.

$$\frac{x - ma\,x(x)}{\min(x) - \max(x)} \tag{2.1}$$

Початкові дані: (див. рис. 2.7)

w	Seq2Seq	LSTM	BERT	OpenAI Ada	GPT-3-TURBO	
0,25	9	6	7	7	8	Якість генерації контенту
0,1	4	6	7	8	10	Схожість із стилем автора:
0,25	0,44	0,9	0,96	0,6	0,55	Різноманітність та оригінальність
0,1	5,76	2,75	2,02	4,01	3,42	Час генерації
0,1	10	8	10	9	9	Адаптивність до тематики
0,2	7	8	8	7	10	Взаємодія з користувачем

Рисунок 2.7 – Початкові дані моделей (виконано самостійно)

На зображенні представлено порівняння п’яти нейронних мереж: Seq2Seq, LSTM, BERT, OpenAI Ada та GPT-3-TURBO за такими критеріями, як якість генерації контенту, схожість зі стилем автора, різноманітність та оригінальність, адаптивність до тематики та швидкість генерації. Також враховується взаємодія з користувачем. За діаграмою GPT-3-TURBO виділяється високими оцінками майже за всіма параметрами, особливо за якістю генерації контенту та адаптивністю до тематики. BERT має сильні позиції у схожості зі стилем автора. LSTM та Seq2Seq мають нижчі оцінки в порівнянні з іншими мережами, але всі вони показують добру адаптивність до

тематики. OpenAI Ada має збалансовані показники, але не виділяється в жодному з критеріїв (див рис. 2.8).

Рисунок 2.8 – Порівняння мереж (виконано самостійно)

Після нормалізації: (див. рис. 2.9)

Seq2Seq	LSTM	BERT	OpenAI Ada	GPT-3-TURBO	
1,00	0,00	0,33	0,33	0,67	Якість генерації контенту
0,00	0,33	0,50	0,67	1,00	Схожість із стилем автора:
0,00	0,88	1,00	0,31	0,21	Різноманітність та оригінальність
0,00	0,80	1,00	0,47	0,63	Час генерації
1,00	0,00	1,00	0,50	0,50	Адаптивність до тематики
0,00	0,33	0,33	0,00	1,00	Взаємодія з користувачем

Рисунок 2.9 – Результати після нормалізації даних (виконано самостійно)

Наведемо векторний опис та проведемо оцінювання за ним: (див. рис. 2.10)

Seq2Seq	LSTM	BERT	OpenAI Ada	GPT-3-TURBO	
1*0,25	0*0,25	0,33*0,25	0,33*0,25	0,66*0,25	Якість генерації контенту
0*0,1	0,33*0,1	0,5*0,1	0,66*0,1	1*0,1	Схожість із стилем автора:
0*0,25	0,88*0,25	1*0,25	0,30*0,25	0,21*0,25	Різноманітність та оригінальність
0*0,1	0,80*0,1	1*0,1	0,46*0,1	0,62*0,1	Час генерації
1*0,1	0*0,1	1*0,1	0,5*0,1	0,5*0,1	Адаптивність до тематики
0*0,2	0,33*0,2	0,33*0,2	0*0,2	1*0,2	Взаємодія з користувачем

Рисунок 2.10 – Оцінка даних на основі векторного опису (виконано самостійно)

Результати дослідження представлені нижче: (див. рис. 2.11)

Seq2Seq	LSTM	BERT	OpenAI Ada	GPT-3-TURBO	
0,25	0,00	0,08	0,08	0,17	Якість генерації контенту
0,00	0,03	0,05	0,07	0,10	Схожість із стилем автора:
0,00	0,22	0,25	0,08	0,05	Різноманітність та оригінальність
0,00	0,08	0,10	0,05	0,06	Час генерації
0,10	0,00	0,10	0,05	0,05	Адаптивність до тематики
0,00	0,07	0,07	0,00	0,20	Взаємодія з користувачем
0,35	0,40	0,65	0,32	0,63	

Рисунок 2.11 – Порівняльна характеристика результатів моделей (виконано самостійно)

Аналіз Парето: (див. рис. 2.12)

Seq2Seq	LSTM	BERT	OpenAI Ada	GPT-3-TURBO	
0,25	0,00	0,08	0,08	0,17	Якість генерації контенту
0,00	0,03	0,05	0,07	0,10	Схожість із стилем автора:
0,00	0,22	0,25	0,08	0,05	Різноманітність та оригінальність
0,00	0,08	0,10	0,05	0,06	Час генерації
0,10	0,00	0,10	0,05	0,05	Адаптивність до тематики
0,00	0,07	0,07	0,00	0,20	Взаємодія з користувачем

Рисунок 2.12 – Порівняльна характеристика результатів моделей (виконано самостійно)

Остаточний результат BERT > LSTM наведено нижче: (див. рис. 2.13)

Seq2Seq	BERT	OpenAI Ada	GPT-3-TURBO	
0,25	0,08	0,08	0,17	Якість генерації контенту
0,00	0,05	0,07	0,10	Схожість із стилем автора:
0,00	0,25	0,08	0,05	Різноманітність та оригінальність
0,00	0,10	0,05	0,06	Час генерації
0,10	0,10	0,05	0,05	Адаптивність до тематики
0,00	0,07	0,00	0,20	Взаємодія з користувачем

Рисунок 2.13 – Остаточний результат порівняння моделей (виконано самостійно)

На наступному графіку представлено порівняння потужності різних алгоритмів. Графік ілюструє ефективність кожного алгоритму в залежності від розміру вхідних даних (див. рис. 2.14).

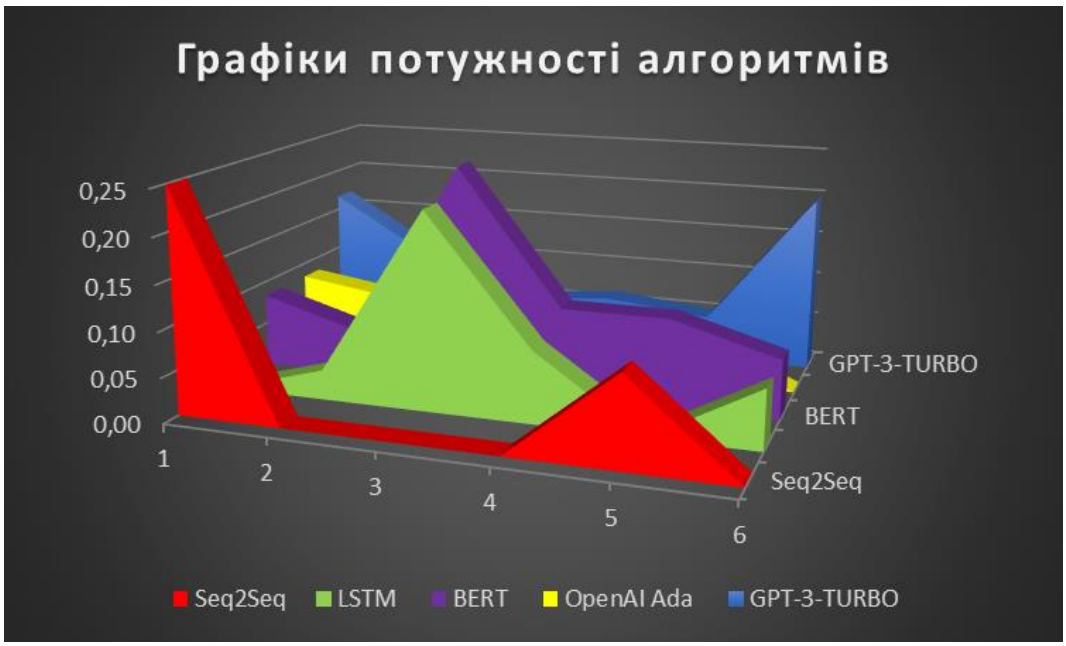


Рисунок 2.14 – Графік потужності алгоритмів (виконано самостійно)

Оцінки за нормалізацією з ваговими коефіцієнтами: (див. рис. 2.15)

Seq2Seq	LSTM	BERT	OpenAI Ada	GPT-3-TURBO
0,35	0,40	0,65	0,32	0,63

Рисунок 2.15 – Нормалізація вагових коефіцієнтів (виконано самостійно)

Результати моделей:

- перше місце: BERT;
- друге місце: GPT-3-TURBO;
- трете місце: LSTM.

Кожен із цих методів демонструє важливість у розвитку галузі штучного інтелекту та машинного навчання, пропонуючи різноманітні можливості для досліджень та розробок. Вибір методу залежить від конкретної задачі та вимог до точності, швидкості та стилістичної адаптації. Сформулювавши змістовну постановку багатокритеріальної задачі прийняття рішень в області магістерського дослідження, завдяки цьому дослідженню, було проведено етап інформаційної підготовки прийняття рішення, а також було наведено векторний опис цієї задачі та були знайдені її найкращі рішення на базі загорткової моделі теорії корисності.

3 АНАЛІЗ ІСНУЮЧИХ МЕТОДІВ ТА АЛГОРИТМІВ

3.1 Огляд моделей Біграм, Трансформер та KerasNLP

Серед різних підходів до моделювання мови значну увагу привернули три архітектури: модель Біграм, модель Трансформера та модель KerasNLP. Модель Біграм – це імовірнісна мовна модель, яка прогнозує ймовірність появи слова на основі попереднього слова. Вона відображає локальний контекст, враховуючи частоту вживання пар слів у навчальному корпусі. Попри свою простоту, модель Біграм довела свою ефективність у різних завданнях NLP, таких як генерація тексту та ідентифікація мови [2].

Однак залежність моделі від обмеженого контекстного вікна погіршує її здатність розуміти довгострокові залежності та генерувати зв'язний текст на довгих послідовностях [5]. З іншого боку, модель Трансформер, запропонована Васвані та ін. [3], зробила революцію в галузі NLP. На відміну від моделі Біграм, модель Трансформер використовує механізми самоуваги для фіксації залежностей між словами в послідовності, незалежно від відстані між ними. Це дозволяє моделі враховувати весь контекст при прогнозуванні, що дозволяє їй генерувати більш логічний і відповідний контексту текст.

Архітектура Трансформер (див. рис. 3.1) складається з кодера і декодера, які працюють разом для обробки вхідних послідовностей і генерування вихідних послідовностей.

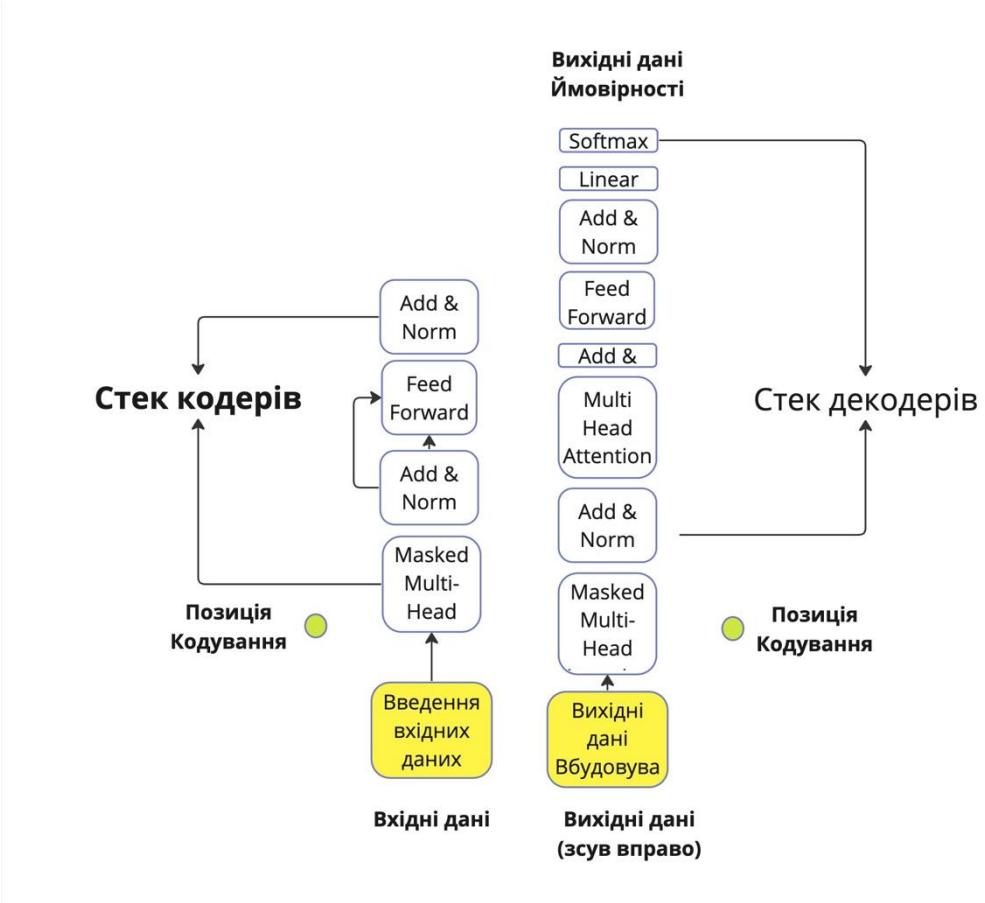


Рисунок 3.1 – Архітектура моделі Трансорфмер (виконано самостійно)

Кодер відображає вхідну послідовність і високорозмірне представлення, в той час, як декодер звертає увагу на вихід енкодера (див. рис 3.2) і генерує відповідну вихідну послідовність. Механізм самоуваги в моделі Трансформер обчислює релевантність кожного слова до кожного іншого слова в послідовності, що дозволяє моделі динамічно зважувати важливість різних слів контексту [4].

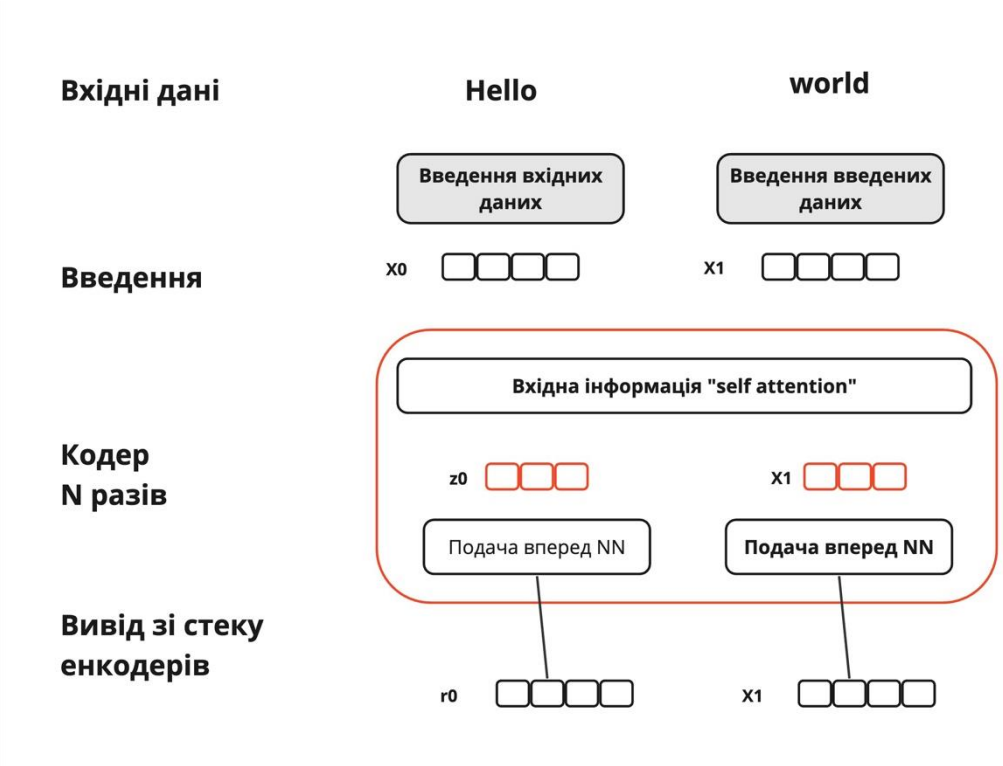


Рисунок 3.2 – Енкодер моделі Трансорфмер (виконано самостійно)

Розробка моделі Трансформер призвела до значного прориву в різних завданнях NLP, включаючи машинний переклад, узагальнення тексту та розуміння мови. Однак, успіх моделі досягається ціною збільшення обчислювальної складності та вимог до пам'яті порівняно з моделлю Біграм. Навчання моделі Трансформер на великих наборах даних вимагає значних обчислювальних ресурсів і може зайняти багато часу. Всупереч цим проблемам, модель Трансформер стала основою для численних сучасних моделей NLP, таких як BERT та GPT, які досягли чудової продуктивності в широкому діапазоні завдань.

Рекурентна нейронна мережа (KerasNLP) – це нейронна мережа, яка спеціалізується на обробці послідовності даних x(t)= x(1), . . . , x(τ) з індексом часового кроку t в діапазоні від 1 до τ. Для задач, які передбачають послідовність вхідних даних, таких як мова та мовлення, часто краще використовувати RNN. У задачах NLP, якщо ви хочете передбачити наступне слово у реченні, важливо знати слова, що стоять перед ним. RNNs називаються рекурентними, тому що вони виконують одне і те ж завдання для кожного

елемента послідовності, а результат залежить від попередніх обчислень. Узагальнюючи вищесказане, вони мають «пам'ять», яка фіксує інформацію про те, що було обчислено попередньо.

Вибір між моделями Біграм, Трансформер та KerasNLP залежить від конкретних вимог завдання NLP, що стоїть перед вами. У той час, як модель Біграм є обчислювально ефективною і може бути використана для завдань з обмеженими контекстними залежностями, модель Трансформер перевершує її в захопленні далекосяжних залежностей і генеруванні високоякісного тексту [5]. Однак, слід враховувати складність і ресурсомісткість моделі Трансформер, особливо в умовах обмежених ресурсів. Розробка моделей Біграм, Трансформер та KerasNLP значно просунула NLP уперед. У той час як модель Біграм забезпечує простий і ефективний підхід до моделювання мови, модель Трансформера зробила революцію в тому, як ми обробляємо і генеруємо природну мову. Оскільки дослідження продовжують розширювати межі мовного моделювання, важливо знайти баланс між складністю моделі та обчислювальною ефективністю, щоб розкрити весь потенціал цих потужних архітектур.

3.2 Обґрунтування вибору моделі Трансформер, Біграм та KerasNLP

Вибір відповідної моделі для генерації тексту має вирішальне значення для досягнення якісних та когерентних результатів. Наше завдання обґрунтувати вибір моделей Трансформер та Біграм для задачі генерації тексту на основі масиву даних українських фентезійних оповідань. Рішення про використання цих двох моделей ґрунтується на їхніх відмінних характеристиках, можливостях та придатності для поставленої задачі. На відміну від традиційних рекурентних нейронних мереж (RNN), які обробляють данні послідовно, модель Трансформер використовує механізми самоуваги для одночасної обробки різних частин вхідних даних. Це дозволяє моделі враховувати весь контекст при генерації кожного слова, що дозволяє їй

створювати більш цілісний і пов'язаний з контекстом текст [5]. Крім того, архітектура моделі Трансформер, що складається з декількох шарів мереж самоуваги та зворотного зв'язку, дозволяє їй вивчати складні патерни та взаємозв'язки в тексті [2]. Така глибока архітектура дозволяє моделі розуміти складні семантичні та синтаксичні структури, що особливо корисно для генерації текстів у певному стилі, такому, як українські фентезійні оповідання [3]. Здатність моделі Трансформер генерувати вільний і творчий текст робить її переконливим вибором для цього завдання.

Однак, важливо враховувати обчислювальну складність і вимоги до ресурсів моделі Трансформер та KerasNLP [7]. Навчання моделі Трансформер на великому наборі даних може бути обчислювально дорогим і трудомістким, вимагаючи значних ресурсів, таких як графічні процесори [4]. Саме тут модель Біграм використовується, як додатковий підхід.

Вибір моделі Біграм (див. рис. 3.3) поряд з Трансформером мотивований кількома факторами. По-перше, модель Біграм може бути базою для оцінки продуктивності моделі Трансформер [5]. Порівнюючи згенерований текст за допомогою обох моделей, ми можемо оцінити відносні покращення та переваги, які пропонує складніша архітектура Трансформер. Окрім того, простота та ефективність моделі Біграм робить її придатною для сценаріїв з обмеженими обчислювальними ресурсами або для генерації тексту в додатках, що працюють у режимі реального часу.

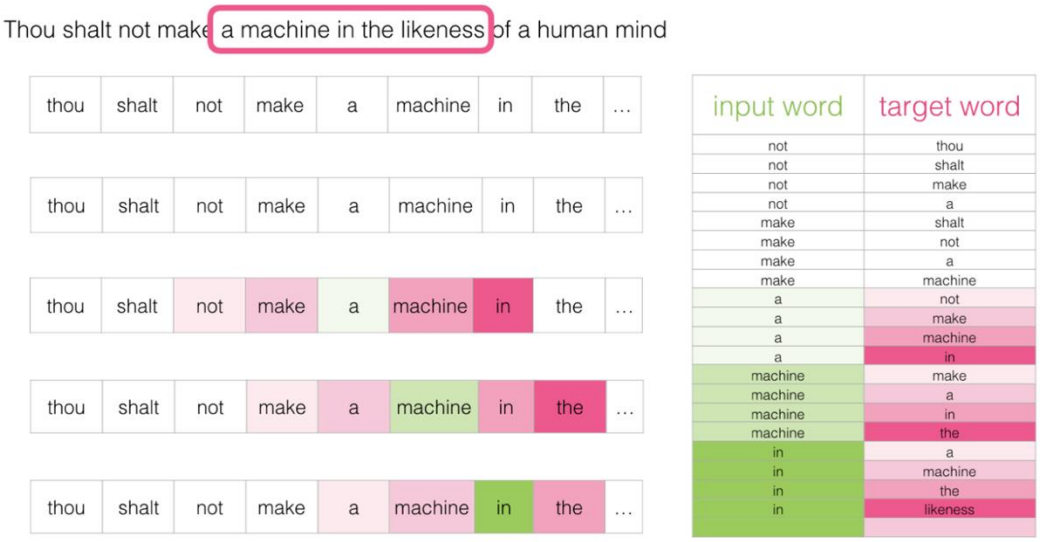


Рисунок 3.3 – Принцип роботи вікна n-gram для моделі Біграм (за даними [15])

Крім того, поєднання моделей Трансформер, Біграм та KerasNLP дозволяє проводити аналіз процесу генерації тексту. Вивчаючи згенерований текст за допомогою обох моделей, ми можемо отримати уявлення про сильні та слабкі сторони кожного підходу. Цей порівняльний аналіз може спрямовувати майбутні дослідження та розробки, висвітлюючи сфери, в яких моделі можуть бути вдосконалені або адаптовані до конкретних доменів чи мов. База даних українських фентезійних оповідань являє собою самостійно зібраний та згенерований датасет, який має багатий словниковий запас, різноманітну тематику та складну наративну структуру. Здатність моделі Трансформер розуміти довгострокові залежності та генерувати зв'язний текст робить її добре придатною для цього набору даних [5]. Потенційно вона може генерувати переконливі та образні історії, які відповідають стилю та темам, присутнім у навчальних даних [1]. З іншого боку, модель Біграм може служити надійною базовою лінією, генеруючи простіший і більш передбачуваний текст на основі локальних зв'язків слів. Варто зазначити, що вибір моделей Трансформер, Біграм та KerasNLP для генерації тексту не обмежується набором даних українських фентезійних оповідань. Ці моделі були успішно застосовані до різних мов, доменів та жанрів [2]. Модель Трансформер,

зокрема, досягла найкращих результатів у таких завданнях, як машинний переклад, реферування текстів та діалогові системи [1]. Універсальність і адаптивність цих моделей роблять їх цінними інструментами для широкого спектра застосувань для генерації текстів.

3.3 Збір та підготовка масиву даних для тренування

Набір даних відіграє вирішальну роль у розробці та оцінці моделей генерації текстів. У цьому розділі ми зосередимося на підготовці та зборі масиву даних українських фентезійних оповідань, який слугує основою для навчання та тестування моделей Трансформер та Біграм. Якість та різноманітність набору даних безпосередньо впливають на продуктивність моделей. Набір даних українських фентезійних оповідань[9] було зібрано з різних джерел, включаючи літературні онлайн-платформи, антології фентезі та створений користувачами контент. Основна мета полягала у створенні репрезентативної колекції українських фентезійних оповідань, що охоплює широкий спектр тем, стилів написання та наративних структур. Набір даних має на меті відобразити багатство та креативність українського жанру фентезі, забезпечуючи міцну основу для створення зв'язного та образного тексту. Щоб забезпечити якість та узгодженість набору даних, було здійснено кілька кроків попередньої обробки. По-перше, зібрані історії були ретельно переглянуті та відфільтровані, щоб виключити будь-який нерелевантний або неякісний контент. Потім оповідання були піддані низці процедур очищення та нормалізації тексту. Це включало видалення спеціальних символів, розділових знаків та будь-яких невідповідностей форматування (див. рис. 3.4). Текст також було перетворено на малі літери для збереження однорідності в усьому наборі даних. Крім того, всі зайві пробіли та символи нового рядка були видалені для забезпечення чистого та стандартизованого представлення тексту [3].

Рисунок 3.4 – Обробка масиву даних (виконано самостійно)

Однією з проблем, що виникли під час підготовки масиву даних, була наявність рідкісних або незвичних слів і словосполучень, характерних для українського жанру фентезі. Для розв'язання цієї проблеми було створено повний словник, що включає як часті, так і рідкісні слова (див. рис. 3.5). Цей словник служить основою для кодування тексту в числові представлення, придатні для введення в моделі.

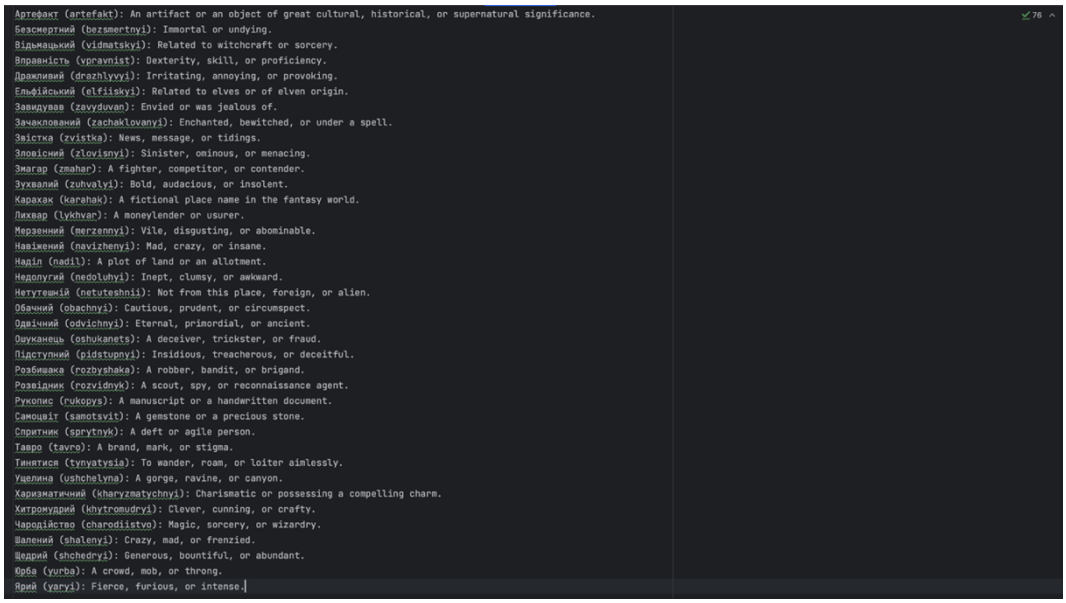


Рисунок 3.5 – Словник рідкісних слів (виконано самостійно)

Для полегшення ефективного навчання та оцінювання набір даних було розділено на дві підмножини: навчальну та валідаційну. Навчальна множина, що містить більшість історій, використовується для навчання моделей і

вивчення основних закономірностей і структур у тексті [1]. З іншого боку, валідаційна множина використовується для оцінки продуктивності моделей під час навчання та налаштування гіперпараметрів для отримання оптимальних результатів.

Розмір і склад навчальної та варіаційної множин були ретельно продумані, щоб забезпечити збалансоване представлення різних за довжиною, темами та стилями написання історій. Це допомагає запобігти надмірному пристосуванню і дозволяє моделям добре узагальнювати невидимі дані. Крім того, набір даних було перемішано, щоб усунути будь-які потенційні упередження або закономірності, які можуть виникнути через порядок історій.

Щоб полebigrгшити ефективне завантаження та обробку даних під час навчання, набір даних було перетворено у відповідний формат (див. рис. 3.6), наприклад, масиви NumPy або тензори PyTorch. Це забезпечує безперешкодну інтеграцію з фреймворками глибокого навчання, що використовуються для реалізації моделей.

```
chars = sorted(list(set(text)))
vocab_size = len(chars)
stoi = { ch:i for i,ch in enumerate(chars) }
itos = { i:ch for i,ch in enumerate(chars) }
encode = lambda s: [stoi[c] for c in s]
decode = lambda l: ''.join([itos[i] for i in l])
```

Рисунок 3.6 – Перетворення масиву даних у відповідний формат (виконано самостійно)

Одним з важливих моментів при підготовці наборів даних є робота з довгими послідовностями. Модель Трансформер, зокрема, має фіксовану довжину вхідної послідовності, що обумовлює необхідність усікання або доповнення історій до рівномірної довжини. Доповнення передбачає додавання спеціальних токенів до коротших послідовностей для досягнення бажаної довжини, тоді як усічення передбачає видалення зайвих токенів з довших послідовностей. Вибір довжини послідовності залежить від наявних обчислювальних ресурсів і компромісу між фіксацією довгострокових залежностей та ефективністю навчання. Іншим аспектом підготовки набору даних є процес токенізації. Токенізація передбачає розбиття тексту на менші

одиниці, такі як слова або під-слова, які слугують вхідними даними для моделей. Вибір стратегії токенізації може вплинути на здатність моделі обробляти слова, що не входять до словника, та враховувати морфологічні варіації в українській мові. У процесі підготовки набору даних особливу увагу було приділено збереженню цілісності та якості українських фентезійних оповідань. Метою було збереження унікальних характеристик та лінгвістичних нюансів жанру, забезпечивши при цьому сумісність з обраними моделями генерації текстів.

Підготовка та збір масиву даних українських фентезійних оповідань включали ретельний процес курації, попередньої обробки та форматування даних. Ретельно відібравши та очистивши оповідання, створивши вичерпний словник та розділивши набір даних на навчальний та валідаційний, ми забезпечили моделям доступ до високоякісних та різноманітних даних для навчання та генерації.

3.4 Оцінка якості генерації тексту

Зв’язок ШІ та генерації текстового контенту є важливою складовою досліджень у галузі опрацювання природної мови. Історично склалося так, що створення текстового контенту було винятково справою рук людини, оскільки потребувало тонкого розуміння смислового навантаження тексту та творчого підходу до його написання. Однак досягнення в галузі ШІ, зокрема в моделях нейронних мереж, таких як Трансформер, дали змогу автоматизувати цей процес до певної міри написання. Сам процес навчання є ресурсомістким, вимагає великих обсягів даних і обчислювальних потужностей, не гарантуючи при цьому створення справді унікального або якісного тексту. Щоб оцінити якість генерації тексту в нашому дослідженні використовується метрика Perplexity і accuracy.

Щоб проілюструвати застосування та ефективність запропонованих нами метрик, ми зосередились на розробці та порівнянні двох різних

нейромережевих моделей для генерації тексту: простої мовної моделі Біграм і складнішої мовної моделі на основі архітектури Трансформер. Обидві моделі було навчено на наборі створених на основі збірок есе, з метою оцінювання їхньої здатності генерувати зв'язний та унікальний текстовий контент[4]. Наприклад, perplexity демонструє міру того, наскільки добре імовірнісна модель прогнозує вибірку. Низький показник perplexity, який можна розрахуватись за формулою 3.1, вказує на те, що розподіл ймовірностей добре прогнозує вибірку. Вона визначається, як експонента ентропії розподілу ймовірностей.

$$PP = 2^{H(P)}, \tag{3.1}$$

де H(P) - ентропія розподілу P.

Ентропію розподілу P можна розрахувати за формулою 3.2:

$$H(P) = -\sum_{x} P(x) \log_2 P\,(x), \tag{3.2}$$

де P(x) - це ймовірність події x;

$\sum x$ - сумування по всіх можливих подіях x.

Perplexity, яка вимірює, наскільки добре розподіл ймовірностей, передбачений моделлю, збігається з істинним розподілом. На графіку (див. рис. 3.7) спостерігається спочатку різкий спад, а потім вирівнювання. Це очікувано, оскільки в міру того, як модель навчається, вона стає краще передбачати наступний токен слова, знижуючи таким чином коефіцієнт втрати[2].

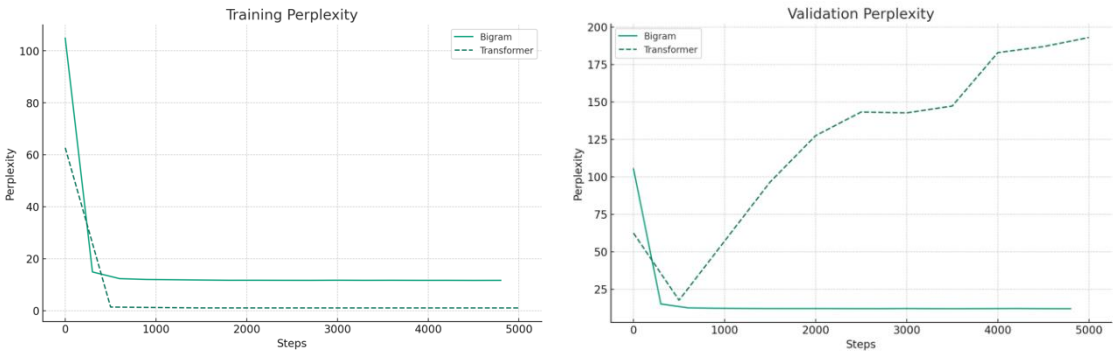


Рисунок 3.7 – Метрика оцінки perplexity для архітектури моделей Біграм та Трансформер (виконано самостійно)

Accuracy обчислюється за формулою 3.3:

$$\text{Accuracy} = \left(\frac{\text{Кількість правильних передбачень}}{\text{Загальна кількість зроблених передбачень}}\right) \times 100 \qquad (3.3)$$

Точність Generative pre-trained Трансформер та Біграм моделей з часом, як показано на графіках (див. рис. 3.8), демонструє швидке зростання точності як навчання, так і валідації на початкових етапах навчання.

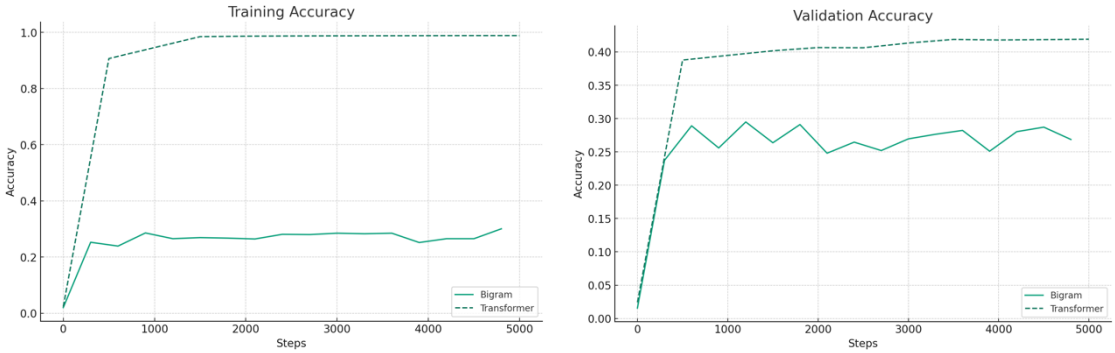


Рисунок 3.8 – Метрика оцінки accuracy для архітектури моделі Біграм та Трансформер (виконано самостійно)

Це типово для моделей на основі нейронних мереж, які швидко навчаються на початку, коли градієнтний спуск є більш помірним[3]. Після початкового сплеску обидві точності стають рівнозначними при точності 0,75 для Трансформер та 0,25 для Біграм, що вказує на те, що моделі досягли

свого потенціалу в навчанні на даних із заданою архітектурою та гіперпараметрами.

Остання метрика обчислює коефіцієнт втрати (див. рис. 3.9) для одного спостереження шляхом підсумовування всіх інших класів моделі. Метою навчання є Н функції втрат для всіх спостережень у наборі даних.

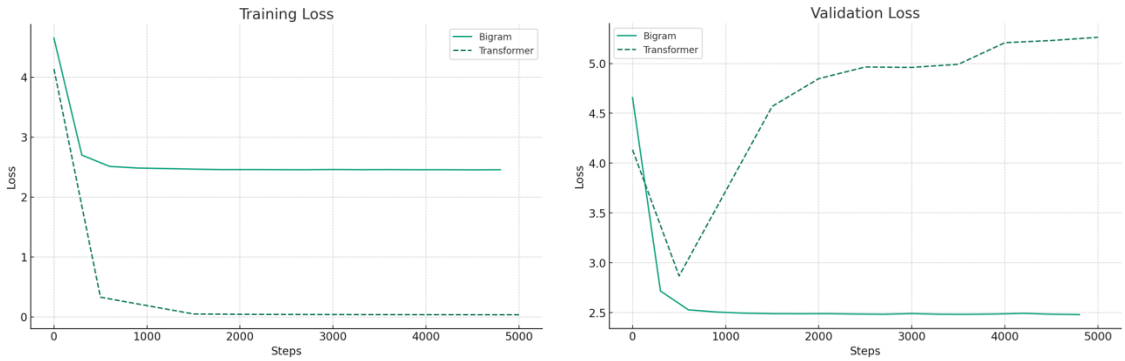


Рисунок 3.9 – Метрика оцінки втрат для архітектури моделі Біграм та Трансформер (виконано самостійно)

Оцінки втрат на цих графіках розраховувались за формулою 3.4:

$$L = -\sum_{c=1}^{M} y_{o,c} \log\left(p_{o,c}\right), \tag{3.4}$$

де L - втрати для одного спостереження;

M - загальна кількість класів;

$y_{o,c}$ - бінарний індикатор 0 або 1;

c - правильною класифікацією для спостереження;

$p_{o,c}$ - прогнозована ймовірність спостереження.

Підсумкова точність навчання[10] для моделі Біграм склала 0.3008, а для Transformer - 0.9887, що свідчить про значно вищу точність прогнозування другої моделі. Фінальне значення помилки моделі Біграм склало 11.6608, що свідчить про те, що йому було важче передбачити наступний елемент у послідовності, порівняно з помилкою моделі Трансформер 1.0355, що вказує на високу точність передбачення на навчальній вибірці.

Підсумовуючи результати дослідження у порівнянні моделей Трансформер а Біграм, текст, згенерований обома моделями враховував швидкість навчання, розміри навчального набору даних та особливості архітектури: кількість шарів, швидкість навчання, розмір батчів, кількість ітерацій та довжину вхідних токенів. Для навчання моделей було зібрано власний датасет на основі збірок есе на тему суспільствознавства, щоб поліпшити якість зібраного набору даних. Також було використано фреймворк для розмітки даних labelstud.io, за допомогою якого набір даних було поділено на категорії "технічна" і "гуманітарна" література.

Таблиця 3.1 – Результати генерації тексту моделей Біграм, Трансформер та KerasNLP (виконано самостійно)

Тип архітектури	Train Loss	Val. Loss	Train Accuracy	Val. Accuracy	Train Perplexity	Val. Perplexity	Згенеерований текст моделі
Трансформер (укр)	0.4607	3.2628	0.9887	0.4191	1.0355	193.0137	Одного разу в лісі з'явився загрозливий дракон, який прагнув підкорити
Біграм(укр)	1.1671	1.4853	0.6008	0.2686	1.6608	167.9474	Микола всі свої чарівні цій вежі зву чали зі свої м вогнем, що могли дом м.

Кінець таблиці 3.1

Тип архітектури	Train Loss	Val. Loss	Train Accuracy	Val. Accuracy	Train Perplexity	Val. Perplexity	Згенеерований текст моделі
KerasNLP (укр)	1.2312	0.1721	0.8231	0.2678	1.5231	182.1371	Одного разу в лісі Еріл. Його стріли лунали на поєдині, пошеру Марса в майстерність
Трансформер (англ)	0.1587	2.4871	0.7642	0.2262	1.1345	182.4453	Knave's utter warm thousand find the crown from when they were.
Біграм(англ)	1.1671	1.4853	0.8213	0.3125	1.3234	177.2341	How Sayish doth sit safe?

4 ПРОГРАМНА РЕАЛІЗАЦІЯ

4.1 Програмна реалізація методів

Програмна реалізація методів генерації текстів у цій роботі складається з чотирьох основних компонентів: моделі Трансформер, моделі Біграм, моделі KerasNLP та фреймворку Streamlit для взаємодії з користувачем. Кожен компонент є важливим у створені веб-додатку генерації зв'язного та лінгвістично точного українського тексту на фентезійну тему.

Основою системи генерації тексту є модель Трансформер, реалізована у файлі "transformer.py" (див. рис. 4. 1).

```python
import torch
import torch.nn as nn
from torch.nn import functional as F


batch_size = 64
block_size = 256
max_iters = 5000
eval_interval = 500
learning_rate = 3e-4
device = 'cuda' if torch.cuda.is_available() else 'cpu'
eval_iters = 200
n_embd = 384
n_head = 6
n_layer = 6
dropout = 0.2
# ------------

torch.manual_seed(1337)

with open('/Users/jhortresnystkyj/Documents/GPT_2/ukr_story.txt', 'r', encoding='utf-8') as f:
    text = f.read()


chars = sorted(list(set(text)))
vocab_size = len(chars)

stoi = { ch:i for i,ch in enumerate(chars) }
itos = { i:ch for i,ch in enumerate(chars) }
encode = lambda s: [stoi[c] for c in s]
decode = lambda l: ''.join([itos[i] for i in l])


data = torch.tensor(encode(text), dtype=torch.long)
n = int(0.9*len(data))
train_data = data[:n]
val_data = data[n:]


2 usages
```

Рисунок 4.1 – Лістинг коду файлу transformer.py (виконано самостійно)

Архітектура моделі Трансформер відповідає оригінальному дизайну архітектури, з кількома ключовими компонентами, які забезпечують ефективну обробку та генерацію послідовних даних. В основі моделі Трансформер лежить механізм Multi-headed attention (див. рис 4.2).

```
1 usage
class MultiHeadAttention(nn.Module):
    """ multiple heads of self-attention in parallel """

    def __init__(self, num_heads, head_size):
        super().__init__()
        self.heads = nn.ModuleList([Head(head_size) for _ in range(num_heads)])
        self.proj = nn.Linear(head_size * num_heads, n_embd)
        self.dropout = nn.Dropout(dropout)

    def forward(self, x):
        out = torch.cat([h(x) for h in self.heads], dim=-1)
        out = self.dropout(self.proj(out))
        return out
```

Рисунок 4.2 – Клас Multi-headed attention (виконано самостійно)

Цей механізм дозволяє моделі одночасно приділяти увагу різним позиціям вхідної послідовності, фіксуючи складні залежності та зв'язки між словами. Функція уваги обчислюється за допомогою матриць запитів, ключів і значень, які виводяться з вхідних вбудовувань за допомогою лінійних перетворень.

Ваги уваги потім використовуються для обчислення зваженої суми векторів значень, створюючи контекстно-залежне представлення кожного слова.

Клас Block (див. рис. 4.3), який є головним блоком моделі, складається з багатоголового шару уваги, за яким слідує позиційна мережа прямого поширення.

```
1 usage
class Block(nn.Module):
    """ Transformer block: communication followed by computation """

    def __init__(self, n_embd, n_head):
        super().__init__()
        head_size = n_embd // n_head
        self.sa = MultiHeadAttention(n_head, head_size)
        self.ffwd = FeedFoward(n_embd)
        self.ln1 = nn.LayerNorm(n_embd)
        self.ln2 = nn.LayerNorm(n_embd)

    def forward(self, x):
        x = x + self.sa(self.ln1(x))
        x = x + self.ffwd(self.ln2(x))
        return x
```

Рисунок 4.3 – Клас Block (виконано самостійно)

Багатоголовий шар уваги застосовує механізм уваги паралельно в декількох вхідних токенах, що дозволяє моделі розуміти різні аспекти вхідної послідовності. Мережа прямого поширення вносить нелінійність і розширює репрезентативність моделі. Позиційні кодування додаються до вхідних введень для включення позиційної інформації, оскільки сама архітектура Трансформатора є інваріантною до перестановок. Ці кодування дозволяють моделі розрізняти слова на основі їхніх позицій у послідовності.

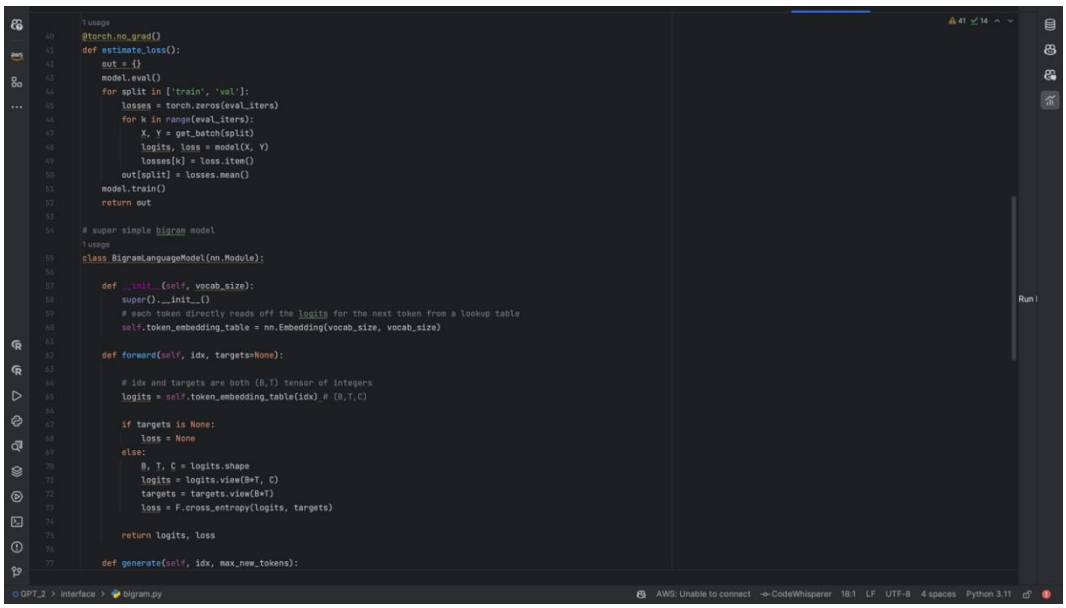
Модель Трансформер навчається за допомогою задачі мовного моделювання, мета якої – передбачити наступне слово, враховуючи попередні слова в послідовності. Параметри моделі оптимізуються за допомогою оптимізатора Adam (див. рис. 4.4) із заданою швидкістю навчання.

```
model = GPTLanguageModel()
m = model.to(device)
print(sum(p.numel() for p in m.parameters())/1e6, 'M parameters')
optimizer = torch.optim.AdamW(model.parameters(), lr=learning_rate)

for iter in range(max_iters):
    if iter % eval_interval == 0 or iter == max_iters - 1:
        losses = estimate_loss()
        print(f"step {iter}: train loss {losses['train']:.4f}, val loss {losses['val']:.4f}")

    xb, yb = get_batch('train')
    logits, loss = model(xb, yb)
    optimizer.zero_grad(set_to_none=True)
    loss.backward()
    optimizer.step()


context = torch.zeros((1, 1), dtype=torch.long, device=device)
print(decode(m.generate(context, max_new_tokens=500)[0].tolist()))
```

Рис 4.4 – Код оптимізатора Adam (виконано самостійно)

Процес навчання включає в себе ітерації над навчальними даними в пакетах, обчислення перехресної втрати ентропії між передбаченими та фактичними наступними словами та оновлення параметрів моделі за допомогою зворотного розповсюдження.

Модель Біграм, реалізована у файлі (див. рис. 4.5) "bigram.py", є більш простішою основою для генерації тексту.

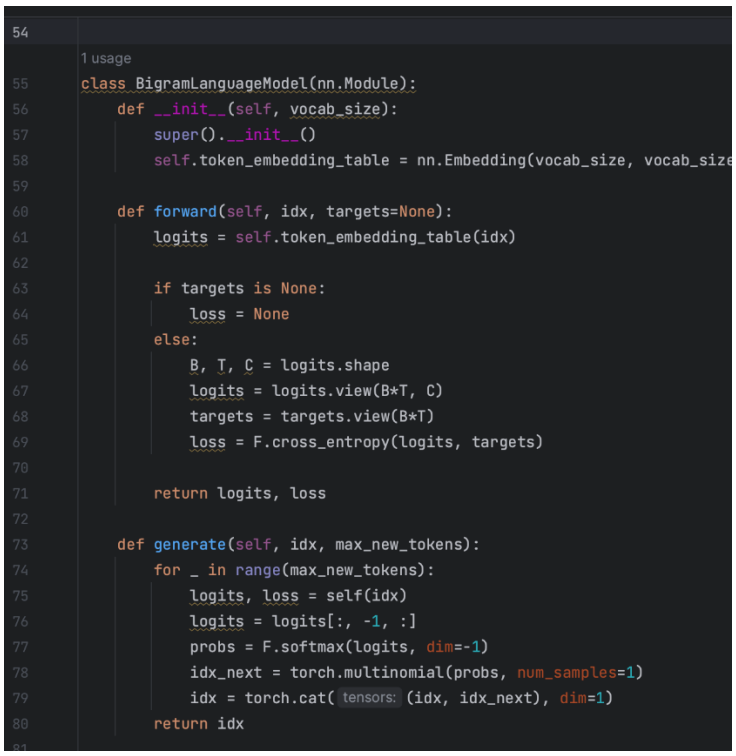


```
1 usage
@torch.no_grad()
def estimate_loss():
    out = {}
    model.eval()
    for split in ['train', 'val']:
        losses = torch.zeros(eval_iters)
        for k in range(eval_iters):
            X, Y = get_batch(split)
            logits, loss = model(X, Y)
            losses[k] = loss.item()
        out[split] = losses.mean()
    model.train()
    return out

# super simple bigram model
1 usage
class BigramLanguageModel(nn.Module):

    def __init__(self, vocab_size):
        super().__init__()
        # each token directly reads off the logits for the next token from a lookup table
        self.token_embedding_table = nn.Embedding(vocab_size, vocab_size)

    def forward(self, idx, targets=None):

        # idx and targets are both (B,T) tensor of integers
        logits = self.token_embedding_table(idx) # (B,T,C)

        if targets is None:
            loss = None
        else:
            B, T, C = logits.shape
            logits = logits.view(B*T, C)
            targets = targets.view(B*T)
            loss = F.cross_entropy(logits, targets)

        return logits, loss

    def generate(self, idx, max_new_tokens):
```

Рисунок 4.5 – Лістинг коду файлу bigram.py (виконано самостійно)

На відміну від моделі Трансформер, яка розглядає всю вхідну послідовність, модель Біграм генерує текст на основі ймовірностей пар слів (біграм).

Модель Біграм використовує таблицю вбудовування лексем для представлення кожного слова у словнику у вигляді вектора щільності. Під час прямого проходу модель отримує вбудовування для вхідних слів і безпосередньо використовує їх як логічні значення для передбачення наступного слова. Потім логічні значення пропускаються через функцію softmax (див. рис. 4.6) для отримання розподілу ймовірностей по словнику.

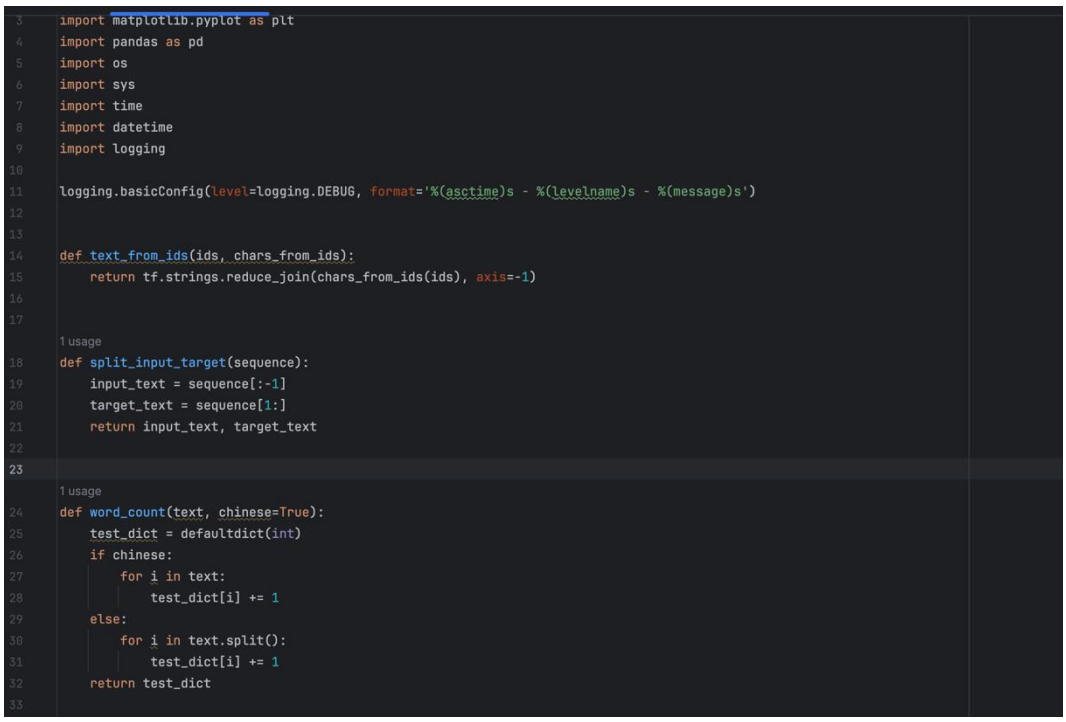


```
1 usage
class BigramLanguageModel(nn.Module):
    def __init__(self, vocab_size):
        super().__init__()
        self.token_embedding_table = nn.Embedding(vocab_size, vocab_size

    def forward(self, idx, targets=None):
        logits = self.token_embedding_table(idx)

        if targets is None:
            loss = None
        else:
            B, T, C = logits.shape
            logits = logits.view(B*T, C)
            targets = targets.view(B*T)
            loss = F.cross_entropy(logits, targets)

        return logits, loss

    def generate(self, idx, max_new_tokens):
        for _ in range(max_new_tokens):
            logits, loss = self(idx)
            logits = logits[:, -1, :]
            probs = F.softmax(logits, dim=-1)
            idx_next = torch.multinomial(probs, num_samples=1)
            idx = torch.cat( tensors: (idx, idx_next), dim=1)
        return idx
```

Рисунок 4.6 – Клас BigramLanguageModel з вбудуваною функцією softmax (виконано самостійно)

Навчання моделі Біграм передбачає мінімізацію перехресних втрат ентропії між передбаченим і фактичним наступним словом. Параметри моделі оптимізуються за допомогою оптимізатора Adam подібно до моделі Трансформер. Для генерації тексту за допомогою моделі Біграм, модель починає з вступного контексту та ітеративно прогнозує наступне слово на основі поточного контексту. Потім передбачене слово додається до контексту, і процес продовжується доти, доки не буде згенеровано потрібну кількість токенів. Streamlit – це бібліотека Python, яка дозволяє легко створювати веб-інтерфейси для моделей машинного навчання. Основа веб-застосунку Streamlit, реалізованої у файлі "str_app.py", надає зручний інтерфейс для взаємодії з навченими моделями Трансформер, KerasNLP та Біграм за допомогою завантажених навчених ваг моделі.

Модель KerasNLP, реалізована у файлі (див. рис. 4.7) " keras_kaggle.py ", головная ідея архітектури включає обробки природної мови, яка використовує алгоритм запам’ятовування послідовних даних.



```
import matplotlib.pyplot as plt
import pandas as pd
import os
import sys
import time
import datetime
import logging

logging.basicConfig(level=logging.DEBUG, format='%(asctime)s - %(levelname)s - %(message)s')


def text_from_ids(ids, chars_from_ids):
    return tf.strings.reduce_join(chars_from_ids(ids), axis=-1)


1 usage
def split_input_target(sequence):
    input_text = sequence[:-1]
    target_text = sequence[1:]
    return input_text, target_text


1 usage
def word_count(text, chinese=True):
    test_dict = defaultdict(int)
    if chinese:
        for i in text:
            test_dict[i] += 1
    else:
        for i in text.split():
            test_dict[i] += 1
    return test_dict
```

Рисунок 4.7 – Лістинг коду файла keras_kaggle.py (виконано самостійно)

Завдяки цьому модель можу пам'ятати свої попередні вхідні дані завдяки внутрішній пам'яті моделі. Побудовані на KerasNLP, ці моделі, шари, метрики та токенізатори можна навчати та серіалізувати в будь-якому форматі даних без використання повторної міграції моделі.

На відміну від моделі Трансформер, яка розглядає всю вхідну послідовність, модель KerasNLP генерує текст на основі запам’ятовування послідовності слів.

Додаток Streamlit[11] завантажує навчені моделі з відповідних файлів контрольних точок. Він визначає функцію під назвою "generate_text" (див. рис. 4.8), яка приймає вхідні дані користувача (підказку) та назву обраної моделі як параметри.

```
1 usage
def generate_text(prompt, model_name, max_new_tokens=200):
    if model_name not in loaded_models:
        model_path = models[model_name]
        if model_name == "Bigram":
            loaded_models[model_name] = load_bigram_model(model_path, device)
        elif model_name == "Transformer":
            loaded_models[model_name] = load_transformer_model(model_path, device)
        else:
            raise ValueError(f"Unsupported model: {model_name}")

    model = loaded_models[model_name]

    if model_name == "Bigram":
        generated_text = model.generate(prompt, max_new_tokens)
    elif model_name == "Transformer":
        encoded_prompt = torch.tensor( data: [encode(prompt)], dtype=torch.long).to(device)
        generated_ids = model.generate(encoded_prompt, max_new_tokens=max_new_tokens)
        generated_text = decode(generated_ids[0].tolist())
    else:
        raise ValueError(f"Unsupported model: {model_name}")

    return generated_text
```

Рисунок 4.8 – Функція generate_text (виконано самостійно)

На основі обраної моделі завантажується відповідна модель, яка використовується для генерації тексту. Для моделі Трансформер запит користувача кодується в ідентифікатори токенів за допомогою функції "encode"[12], а згенерований текст декодується назад у читабельний формат за допомогою функції "decode". Метод "generate" моделі Трансформер викликається з закодованим запитом (див. рис. 4.9) і бажаною кількістю нових токенів, які потрібно згенерувати.

```
1 usage
def load_transformer_model(model_path, device):
    model = GPTLanguageModel()
    model.load_state_dict(torch.load(model_path, map_location=device))
    model = model.to(device)
    model.eval()
    return model
```

Рисунок 4.9 – Метод load_transformer_model (виконано самостійно)

Для моделі Біграм метод "generate" викликається безпосередньо з підказкою користувача і бажаною кількістю нових токенів. Згенерований текст повертається і відображається в інтерфейсі Streamlit. Користувачі можуть

вводити власні підказки і вибирати потрібну модель для генерації тексту на основі наданого контексту. Додаток Streamlit забезпечує зручний спосіб демонстрації можливостей навчених моделей і дозволяє користувачам взаємодіяти з ними в режимі реального часу, генеруючи текст на основі їхніх підказок.

Таким чином, програмна реалізація методів генерації тексту в цьому проекті включає розробку моделі Трансформер, моделі Біграм та додатку Streamlit. Модель Трансформер використовує можливості multihead attention[13] та позиційного кодування для виявлення довготривалих залежностей та генерації зв'язного тексту. Модель Біграм є простішою базовою лінією, генеруючи текст на основі ймовірностей пар слів. Додаток Streamlit[14] надає зручний інтерфейс для взаємодії з навченими моделями та генерування тексту на основі підказок користувача. Разом ці компоненти утворюють комплексну систему для генерації фентезійних розповідей українською мовою.

4.2. Контрольний приклад

Щоб продемонструвати функціональність і продуктивність розробленого додатка для генерації тексту, надається детальний контрольний приклад, який демонструє використання інтерфейсу Streamlit і порівняння результатів, отриманих за допомогою моделей Трансформер, Біграм та KerasNLP. Інтерфейс Streamlit, як показано на рисунку 4.10, пропонує зручний та інтуїтивно зрозумілий макет для взаємодії з моделями генерації тексту.

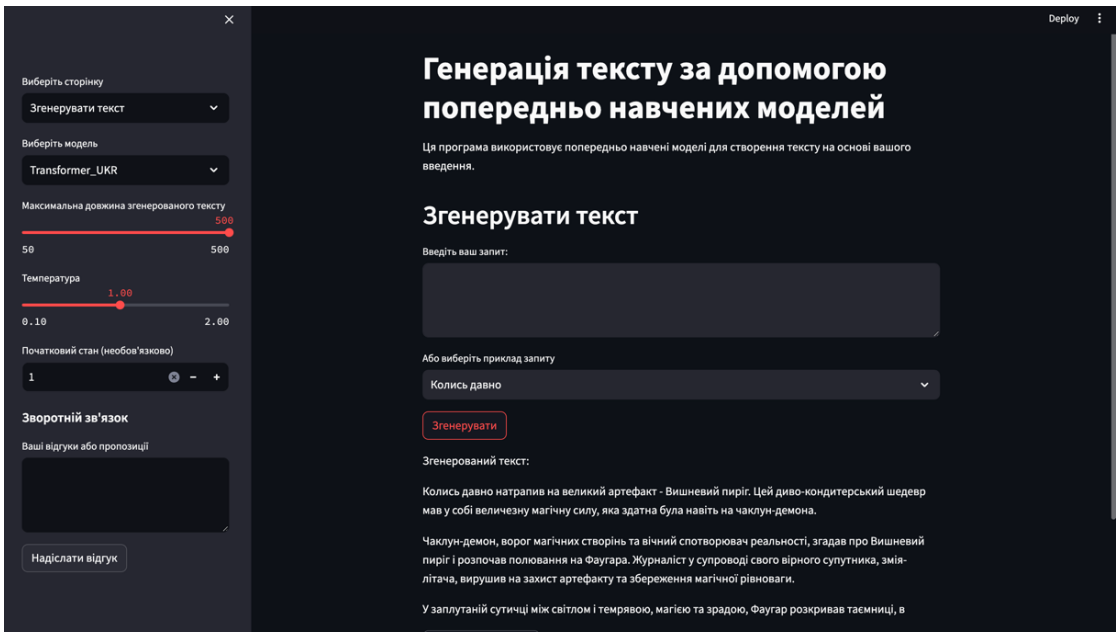


Рисунок 4.10 – Інтерфейс Streamlit (виконано самостійно)

Інтерфейс складається з текстового поля для введення, де користувачі можуть ввести бажаний запит, випадаючого меню для вибору потрібної моделі (Трансформер, Біграм або KerasNLP) і кнопки "Згенерувати" для запуску процесу генерації тексту. Згенерований результат відображається у вихідному текстовому полі нижче, що дозволяє користувачам легко переглядати та аналізувати результати.

Щоб розпочати процедуру генерації тексту, до текстового поля вводиться зразок запиту. У цьому прикладі використовується запит "Одного разу в лісі", який являє собою підказку для моделей генерації тексту для продовження розповіді. Модель Трансформер вибирається з випадаючого меню і натискається кнопка "Згенерувати”, щоб згенерувати вихідний текст.

Модель Трансформер (див. рис. 4.11) обробляє вхідний запит і генерує наступний результат: "У давні часи в глибині лісу жив король Артур - відважний півросли к, який славився своїм майстерним володінням луком і стрілою. У давні роки він знайшов величезний і могутній меч, який він вважав за свій найважливіший артефакт - Меч Північної Зорі".

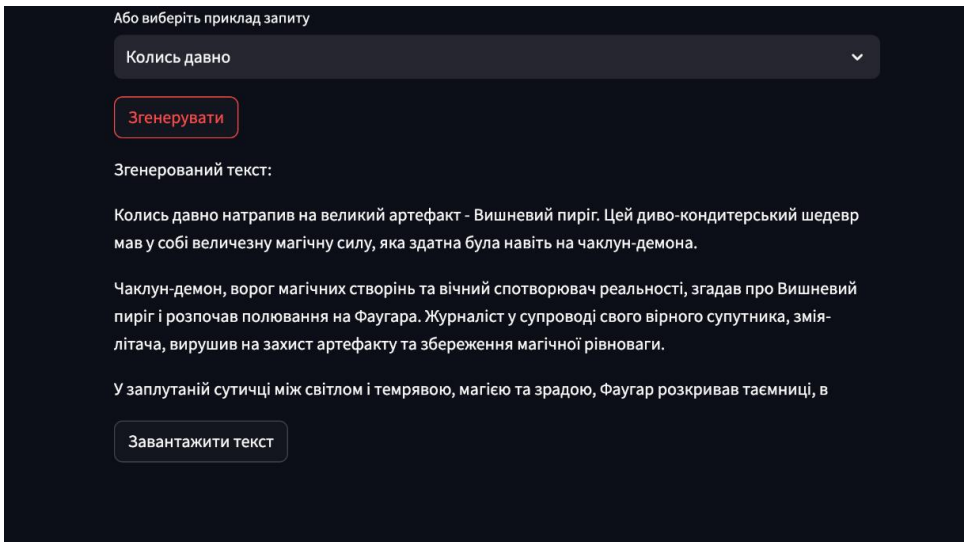


Рисунок 4.11 – Результат генерації тексту жанру фентезі (виконано самостійно)

Згенерований текст демонструє здатність моделі Трансформер продовжувати розповідь у зв'язний і лінгвістично точний спосіб. Модель ефективно розуміє контекст підказки і генерує правдоподібне продовження історії. Згенерований текст зберігає послідовний стиль написання і використовує відповідну лексику та граматику, підвищуючи загальну лінгвістичну якість вихідного тексту.

Щоб порівняти продуктивність моделі Трансформер з моделлю Біграм (див. рис. 4.12), використовується той самий запит, а модель Біграм вибирається з меню переліку моделей.

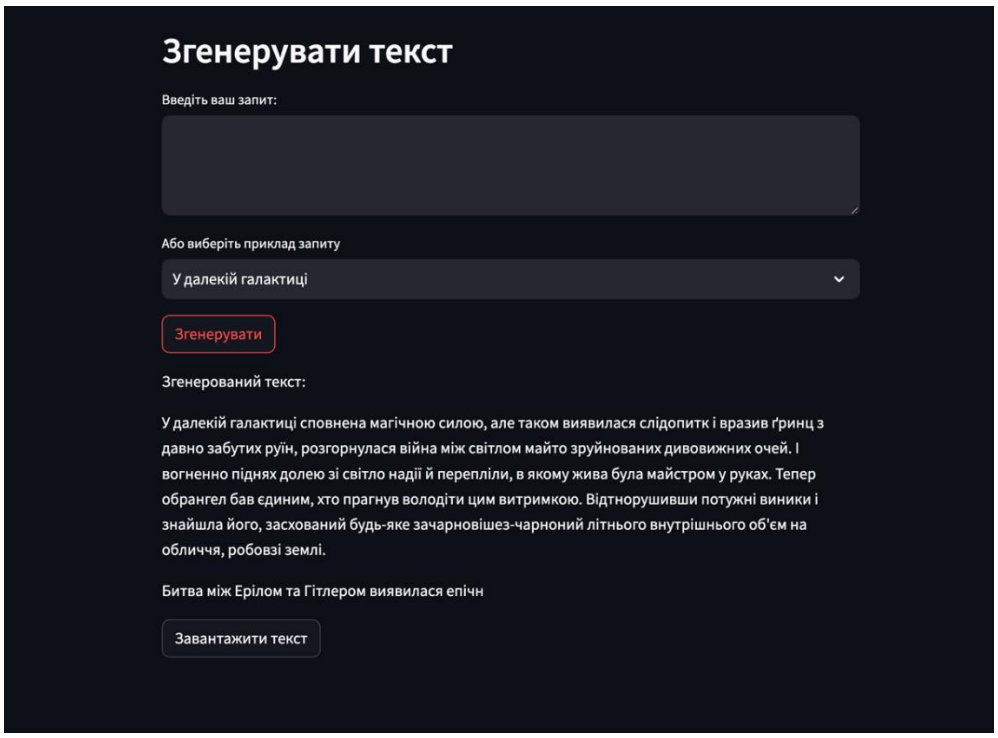


Рисунок 4.12 – Результат генерації тексту жанру фентезі моделю Bigram (виконано самостійно)

Після натискання кнопки "Згенерувати" отримуємо наступний висновок: "У далекій галактиці сповнена магічною силою, але таком виявилася слідопитк і вразив ґринц з давно забутих руїн, розгорнулася війна між світлом майто зруйнованих дивовижних очей. І вогненно піднях долею зі світло надії й перепліли, в якому жива була майстром у руках." Модель Біграм також намагається продовжити розповідь, але їй бракує зв'язності та креативності, які спостерігаються у вихідних даних моделі Трансформер. Модель Bigram генерує текст на основі вірогідності пар слів, що призводить до більш спрощеного і менш обумовленого контекстом продовження слів. Хоча згенерований текст згадує елементи з оригінальної історії, такі як "вежі" та "мудрість", він не вносить нових та значущих елементів в оповідь, що призводить до менш зв'язного результату.

Щоб проаналізувати обрану модель та її гіперпараметри для тренування, в додатку створена окрема сторінка для аналізу обраної моделі (див. рис. 4.13).

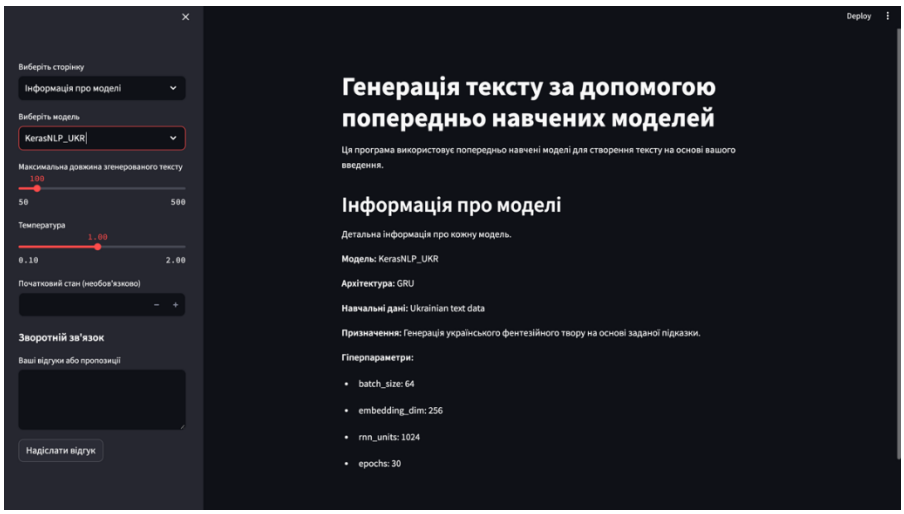


Рисунок 4.13 – Сторінка опису моделі (виконано самостійно)

Аналізуючи результати роботи усіх трьох моделей, стає очевидним, що модель Трансформер генерує більш зв'язний, креативний і лінгвістично точний текст порівняно з моделлю KerasNLP та Біграм. Можливість моделі Transformer розуміти довгострокові залежності та враховувати всю вхідну послідовність дозволяє їй генерувати речення, що відповідають ситуації в конкретному контексті. Модель успішно підтримує послідовний стиль написання, використовує відповідну лексику та граматику, а також вводить нові та релевантні елементи в оповідь, підвищуючи загальну якість та зв'язність згенерованого тексту. З іншого боку, залежність моделі Біграм від ймовірностей пар слів обмежує її здатність генерувати зв'язний і творчий текст. Модель Біграм намагається зрозуміти загальний контекст і генерує текст, який може не відповідати оригінальній історії, що призводить до не такого точного лінгвістично результату.

Важливо зазначити, що хоча модель Трансформер демонструє непогані результати у створенні зв'язного та креативного тексту, вона іноді може відхилятися від контексту оригінальної історії або вносити невідповідності. Це може бути пов'язано з труднощами, які виникають через підтримання довготривалої зв'язності та послідовності створеного тексту. Сильні сторони моделі Особливість Transformer полягає в його здатності враховувати складні мовні патерни, генерувати доречні в конкретному випадку тексти та

забезпечувати належний рівень лінгвістичної якості. На противагу цьому, сильними сторонами моделі Біграм є її простота та ефективність у створенні тексту на основі ймовірностей пар слів. Втім, його слабкі сторони проявляються у відсутності зв'язності, творчості та усвідомлення змісту в створеному тексті. Підсумовуючи, можна сказати, що розроблена програма генерації тексту показала свою ефективність, особливо підкресливши вищу продуктивність моделі Трансформер порівняно з моделлю Біграм та KerasNLP. Інтерфейс Streamlit забезпечує зручний спосіб взаємодії з моделями та аналізу їхніх результатів. Подальші вдосконалення можуть бути зроблені для покращення здатності моделей підтримувати довгострокову узгодженість і відповідність контексту оригінальної історії, що призведе до ще більш удосконаленого тексту, згенерованого моделлю.

4.3. Результати порівняння моделей на зібраному датасеті

Реалізація моделей Трансформер, Біграм та KerasNLP для генерації українських фентезійних текстів дала змогу зрозуміти їхню продуктивність, якість згенерованого тексту, обмеження та потенційні можливості застосування на практиці. Метрики продуктивності моделей оцінювалися за допомогою втрат і незрозумілості, як на навчальних, так і на валідаційних наборах. Модель Трансформер продемонструвала вищу продуктивність порівняно з моделлю Біграм за всіма показниками. Як показано на графіках (див. рис. 4.14), модель Трансформер досягла остаточної втрати при навчанні 0,0574 і втрати при валідації 3,3474 після 20 000 кроків навчання. На противагу цьому, модель Біграм досягла остаточної втрати при навчанні 2,5265 і втрати при валідації 2,5418 після 20 000 кроків.

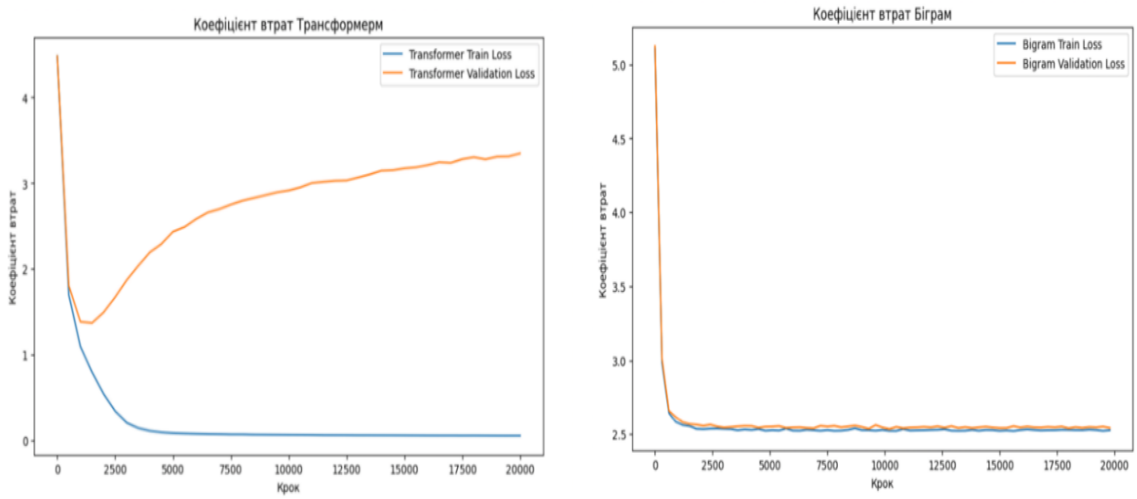


Рисунок 4.14 – Графік залежності коефіцієнта втрат для моделей Біграм та Трансформер (виконано самостійно)

Значно менші втрати при навчанні та валідації моделі Трансформер вказують на її здатність краще розуміти основні закономірності та генерувати більш точні прогнози.

Метрика розгубленості, яка вимірює середню кількість однаково ймовірних наступних слів у послідовності, ще більше підкреслює перевагу моделі Трансформер. Нижчий показник розгубленості свідчить про кращі результати моделювання мови. Загальна складність моделі Трансформер на апробаційному наборі була значно нижча, ніж у моделі Біграм, що свідчить про її здатність генерувати більш зрозумілий і конкретний для даного випадку текст. Модель KerasNLP навчалася на наборі даних з наперед визначеною архітектурою та гіперпараметрами. Процес навчання відстежувався протягом 29 епох, при цьому значення втрат фіксувалися в кінці кожної епохи. В якості функції втрат використовувалася категоріальна перехресна ентропія, що є стандартним вибором для задач класифікації в НЛП.

Результати (див. рис. 4.15) демонструють послідовне зменшення втрат при навчанні, що свідчить про ефективне навчання та оптимізацію параметрів моделі.

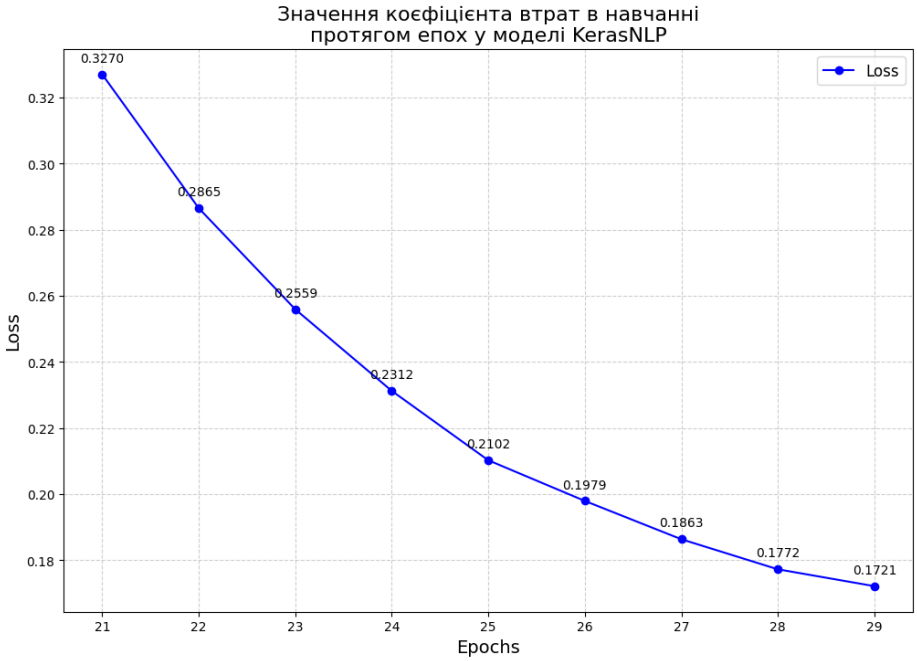


Рисунок 4.15 – Графік залежності коефіцієнта втрат для моделі KerasNLP (виконано самостійно)

Значення втрат зменшилося з 0.3270 на епосі 21 до 0.1721 на епосі 30, що відображає покращення продуктивності моделі з часом.

Оцінка згенерованого тексту виявила помітні відмінності між трьома моделями. Моделі Трансформер та KerasNLP стабільно генерували текст з більшою плавністю, зв'язністю та релевантністю до заданих підказок. Згенеровані речення демонстрували граматичну правильність, семантичну значущість і плавний перехід від одного речення до іншого. Моделі добре аналізували контекст підказок і генерували текст, який добре відповідав заданій темі. На противагу цьому, текст, згенерований моделлю Біграм, часто не мав зв'язності та демонстрував різкі переходи між реченнями.

Хоча модель Біграм генерувала граматично правильні речення, загальна зв'язність і відповідність підказкам поступалася моделям Трансформер та KerasNLP.

4.4. Порівняльний аналіз з існуючими аналогами

Поточний аналіз порівняння оцінює результати навчання двох різних моделей, навчених на наборі даних Шекспіра: моделі KerasNLP та моделі RNN[15]. Обидві моделі мають на меті генерувати текст у стилі Шекспіра, навчених на наборі даних збірника сонет та творів Шекспіра. Основними метриками для порівняння є коефіцієнти втрат при навчанні та якість згенерованого тексту. Модель KerasNLP навчалася протягом 30 епох. Коефіцієнти втрат (див. рис. 4.16) при навчанні послідовно зменшувалися з кожною епохою, що свідчить про ефективне навчання. Нижче наведено втрати при навчанні для вибраних епох:

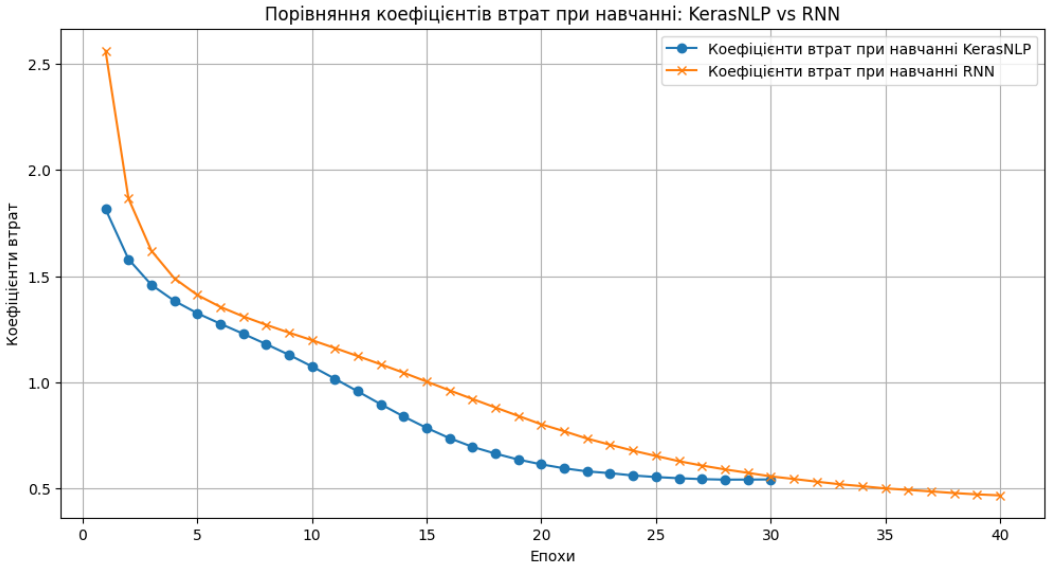


Рисунок 4.16 – Графік кривої залежності коефіцієнта втрат від кількості епох для моделей KerasNLP та RNN (виконано самостійно)

Згенерований текст покращував свою зв'язність і структуру в міру проходження тренування. Наприклад, при значенні епохи 22 текст був більш граматично вірним і з точки зору семантики, порівняно з попередніми епохами. RNN-модель було навчено на 30 епохах. Коефіцієнти втрат при навчанні також послідовно зменшувалися з кожною епохою, що свідчить про ефективне навчання. Нижче (див. рис. 4.17) зображено коефіцієнти втрат при навчанні для обраної кількості епох.

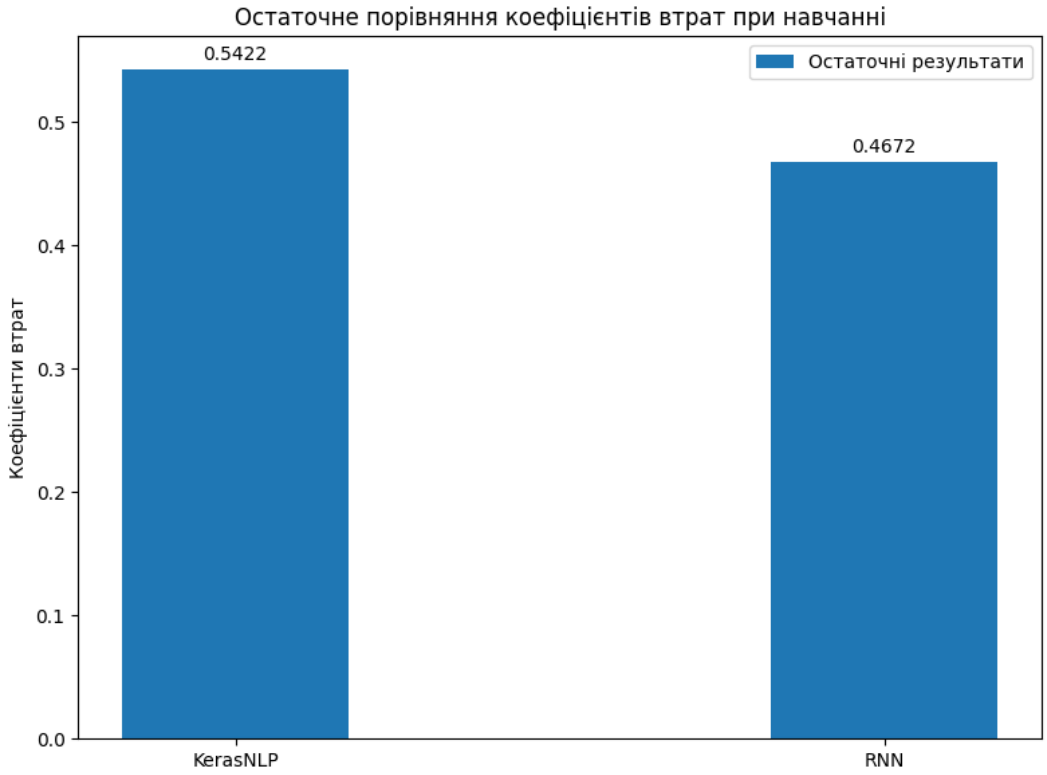


Рисунок 4.17 – Діаграма остаточних результатів тренування моделей KerasNLP та RNN (виконано самостійно)

Згенерований текст за допомогою моделі RNN також продемонстрував значне покращення з часом: кінцеві результати стали більш узгодженими та більш наближеними до контексту.

Модель KerasNLP продемонструвала постійне зменшення втрат навчання з плином часу, що свідчить про ефективне навчання. Фінальна втрата навчання після 30 епохи склала 0,5422. RNN модель також продемонструвала значне зменшення втрат, з кінцевою втратою навчання 0,4672 після 30 епох. Модель RNN досягла менших значень коефіцієнтів втрат при навчанні порівняно з моделлю KerasNLP, що свідчить про те, що вона більш ефективно навчалась на наборі даних.

Тривалість навчання:

– модель KerasNLP навчалася протягом 30 епох, кожна епоха займала приблизно 12-13 секунд;

- модель RNN навчалася протягом 30 періодів, кожен з яких тривав приблизно 29-31 секунду.

Модель KerasNLP потребувала менше часу для навчання на кожну епоху порівняно з моделлю RNN.

Якість генерації тексту:

- текст, згенерований моделлю KerasNLP, покращувався з кожною епохою, ставши більш зв'язним та логічним
- якість генерації тексту моделлю RNN також зросла з часом, а отримані результати стали більш точними з погляду відповідності контексту.

Обидві моделі продемонстрували здатність генерувати текст у стилі Шекспіра, але менший остаточний коефіцієнт втрат при навчанні моделі RNN свідчить про те, що вона може генерувати більш якісний та довгий текст.

Як KerasNLP, так і RNN моделі ефективно навчалися на наборі даних Шекспіра, про що свідчить послідовне зменшення втрат при навчанні та покращення якості згенерованого тексту. Модель RNN досягла нижчих кінцевих втрат при навчанні та продемонструвала кращу здатність до навчання, хоча і з більшою тривалістю навчання на епоху. Модель KerasNLP, з іншого боку, потребувала меншого часу навчання на кожну епоху, але при цьому забезпечувала високу якість тексту. Загалом, модель RNN виявилася кращою з погляду ефективності навчання, тоді як модель KerasNLP пропонує швидший процес навчання.

4.5. Аналіз впроваджених результатів

У цьому дослідженні ми порівнюємо продуктивність трьох різних моделей, навчених на текстових наборах даних: моделі KerasNLP, моделі Біграм та моделі Трансформер. Використані набори даних включають українське фентезійне оповідання та англійський текст Шекспіра[15].

Основними метриками для порівняння є втрати при навчанні, втрати при валідації та якість згенерованого тексту.

Модель KerasNLP навчалася протягом 7 епох. Коефіцієнти втрат при навчанні послідовно зменшувалися з кожною епохою, що свідчить про ефективне навчання. Нижче наведено коефіцієнти втрат при навчанні для кількох епох:

- епоха 1: Втрати = 2.3728;
- епоха 2: втрата = 2.1487;
- епоха 3: Втрати = 1.9260;
- епоха 4: Втрати = 1.7096;
- епоха 5: Втрати = 1.5410;
- епоха 6: Втрати = 1.4146;
- епоха 7: Втрати = 1.3170.

Остаточні результати тренування (див. рис. 4.18) : val loss – 0.1721, train loss – 1.317 для датасету українських фентезійних творів. Результати на основі тренування датасету Шекспіра: val loss – 0.5417, train loss – 1.021.

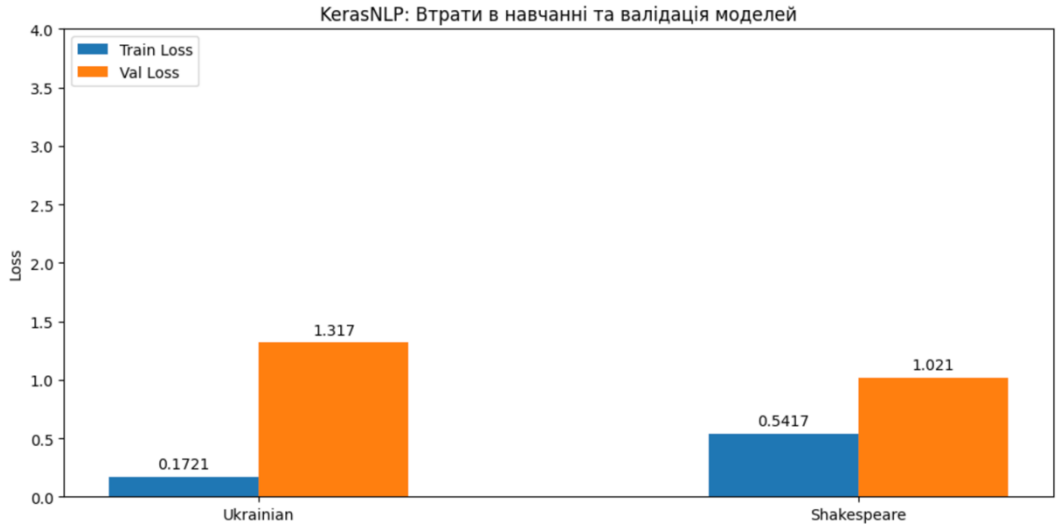


Рисунок 4.18 – Порівняльна діаграма результатів KerasNLP (виконано самостійно)

Згенерований текст покращував свою зв'язність і структуру в міру того, як тривало навчання. Наприклад, при значенні епохи 7 текст був більш

синтаксично правильним, змістовним та складним порівняно з попередніми епохами. Модель Біграм навчалася, як на українських фентезійних оповіданнях, так і на тексті Шекспіра[16]. Втрати під час навчання та валідації фіксувалися через регулярні проміжки часу (див. рис. 4.19).

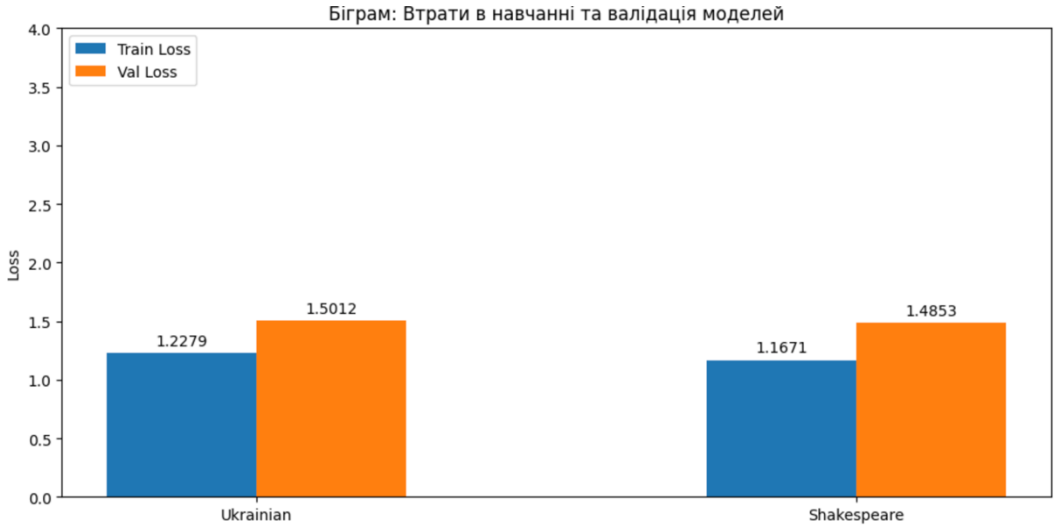


Рисунок 4.19 – Порівняльна діаграма результатів Біграм (виконано самостійно)

Для набору даних Шекспіра коефіцієнти втрат були наступними:

– початкові значення коефіцієнтів втрат: Train = 4.1694, Val = 4.1689;
– після 5000 ітерацій: Train = 1.4937, Val = 1.6586;
– остаточні значення коефіцієнтів втрат: Train = 1.1671, Val = 1.4853;
– для української фентезійної історії остаточні втрати були такими;
– коефіцієнти втрат під час тренування: 1.167;
– коефіцієнти втрат під час валідації: 1.4853.

Згенерований текст за допомогою моделі Біграм з часом значно покращився. Модель Трансформер також навчалася на обох наборах даних (див. рис. 4.20).

Рисунок 4.20 – Порівняльна діаграма результатів Трансформер (виконано самостійно)

Коефіцієнти втрат під час навчання та валідації для тексту Шекспіра[17] були такими:

– початкові коефіцієнти втрат: Train = 4.1694, Val = 4.1689;
– після 5000 ітерацій: Train = 1.4937, Val = 1.6586;
– остаточні коефіцієнти втрат: Train = 1.0705, Val = 1.4928.

Для української фентезійної історії остаточні втрати були такими:

– коефіцієнти втрат тренування: 0.4607;

коефіцієнти втрат валідації: 3.2628.

Модель Трансформер продемонструвала чудову продуктивність з погляду зменшення коефіцієнтів втрат і якості генерації тексту. KerasNLP продемонструвала стабільне зменшення втрат при навчанні в різні епохи, що свідчить про ефективне навчання[18]. Модель Біграм також продемонструвала значне зменшення коефіцієнтів втрат, але модель Трансформер показала кращі результати за показником остаточних коефіцієнтів втрат навчання. Трансформер досягла найнижчих коефіцієнтів втрат навчання, що свідчить про її вищу здатність до навчання.

- коефіцієнти втрат при валідації моделі KerasNLP не були зафіксовані в явному вигляді, але тенденція втрат при навчанні свідчить про хороше узагальнення;
- модель Біграм мала більший показник коефіцієнтів втрат валідації в порівняні з показниками при навчанні, що вказує на потенційну надмірну пристосованість;
- модель Трансформер мала високий показник коефіцієнтів втрат валідації для українського набору даних, що свідчить про надмірну адаптацію.

Модель Трансформер продемонструвала кращі результати в порівнянні з моделями KerasNLP та Біграм за показниками зменшення коефіцієнтів втрат при навчанні та якості генерації тексту. Однак на українському наборі даних вона показала ознаки перенавчання[19]. Модель KerasNLP продемонструвала ефективне навчання зі стійким зменшенням коефіцієнтів втрат навчання, тоді як модель Біграм показала значне покращення, але була менш ефективною в генерації тексту порівняно з моделями Трансформер та KerasNLP. Загалом, модель Трансформер є найефективнішою для задач генерації тексту, за нею йдуть моделі KerasNLP та Біграм.

ВИСНОВКИ

Метою роботи є розробка та впровадження моделей генерації текстів для української мови, зосереджених на архітектурах Трансформер, Біграм та KerasNLP. У роботі використано комплексну методологію, що охоплює попередню обробку даних, навчання моделей, оцінювання та розробку зручного інтерфейсу з використанням бібліотеки Streamlit. Модель Трансформер, заснована на революційній архітектурі, запропонованій Vaswani використовувала увагу кількох голів і позиційні кодування для фіксації довгострокових залежностей та генерування зв'язного тексту.

Модель Біграм є простішою базовою моделлю, яка генерує текст на основі ймовірностей пар слів. Для оцінки продуктивності обох моделей на навчальних і валідаційних наборах використовувалися метрики оцінки втрати та незрозумілість. Моделі Трансформер та KerasNLP перевершили модель Біграм, досягнувши нижчих значень коефіцієнтів втрат і незрозумілості, що свідчить про їх здатність краще розуміти лінгвістичні закономірності та генерувати більш точні тексти. Аналіз згенерованого тексту виявив, що модель Трансформер здатна створювати цілісні та зрозумілі речення, випереджаючи модель Біграм за загальною точністю створення текстів.

Здатність моделі генерувати граматично правильний і семантично змістовний текст може допомогти письменникам, творцям контенту та тим, хто вивчає мову. Інтеграція моделі в чат-боти та віртуальних асистентів може покращити користувацький досвід та уможливити більш природні та цікаві розмови українською мовою. Однак, дослідження в процесі також виявило обмеження та сфери для подальшого вдосконалення, такі як випадкове генерування повторюваних або загальних фраз та проблема збереження довготривалої зв'язності у довгих уривках. Рекомендації щодо усунення цих обмежень включають використання сучасних методів вибірки, точне налаштування моделі на більших і різноманітніших наборах даних, а також вивчення механізмів пам'яті для фіксації довгострокових залежностей.

Подальші дослідження і розробки в цій галузі мають потенціал для революційного оновлення мовних задач і додатків не лише для української, а й для інших мов. Науковий апарат, використаний у цьому дослідженні, включаючи архітектуру Трансформер, механізми уваги та метрики оцінювання, відповідає сучасним практикам у галузі генерації текстів. Аналіз отриманих результатів демонструє ефективність моделей Трансформер та KerasNLP у створенні якісних фентезійних українських історій, перевершуючи показники більш простої моделі Біграм. Результати цієї роботи сприяють розвитку методів обробки природнього мовлення для української мови та відкривають шлях для подальших досліджень та інновацій у цій галузі.

Використовуючи механізми уваги та фіксуючи довгострокові залежності, модель Трансформер генерує вільний, логічно цілісний та взаємопов'язаний з конкретним змістом текст, відкриваючи при цьому нові можливості для створення різноманітного контенту. Використаний науковий апарат і проведений аналіз демонструють надійність і певність дослідницької роботи. Оскільки сфера обробки природньої мови продовжує розвиватися, висновки та ідеї, отримані в цьому дослідженні, можуть посприяти розробці більш досконалих та ефективних моделей генерації текстів.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Ahmed M. Elkhatat, Khaled Elsaid & Saeed Almeer: Evaluating the efficacy of AI content detection tools in differentiating between human and AI-generated text - 19, Article number: 17 (дата звернення: 10.10.2023)

2. Ismail Dergaa,1,2,3 Karim Chamari,4 Piotr Zmijewski,5 and Helmi Ben Saad: From human writing to artificial intelligence generated text: examining the prospects and potential threats of ChatGPT in academic writing - 2023 Apr; 40(2): 615–622 (дата звернення: 05.12.2023)

3. Sisith Ariyaratne, Karthikeyan. P. Iyengar, Neha Nischal, Naparla Chitti Babu & Rajesh Botchu: A comparison of ChatGPT-generated articles with human-written articles - Volume 52, pages 1755–1758, (дата звернення: 20.12.2023)

4. Yongqiang Ma, Jiawei Liu, Fan Yi: Is This Abstract Generated by AI? A Research for the Gap between AI-generated Scientific Text and Human-written Scientific Text. (дата звернення: 10.01.2024)

5. Yongqiang Ma, Jiawei Liu, Fan Yi, Qikai Cheng, Yong Huang, Wei Lu, Xiaozhong Liu: AI vs. Human -- Differentiation Analysis of Scientific Content Generation - 12 Feb 2023 (дата звернення: 21.02.2024)

6. Attention Is All You Need. (2023). Computer Science. URL - https://arxiv.org/abs/1706.03762 (дата звернення: 01.03.2024)

7. Feasibility of Improving BERT for Linguistic Prediction on Ukrainian corpus. URL: https://ceur-ws.org/Vol-2604/paper40.pdf (дата звернення: 26.05.2024).

8. Introducing UberText 2.0: A Corpus of Modern Ukrainian at Scale. URL - https://aclanthology.org/2023.unlp-1.1/ (дата звернення: 26.05.2024).

9. The Second Ukrainian Natural Language Processing Workshop (UNLP 2023). URL - https://aclanthology.org/volumes/2023.unlp-1/ (дата звернення: 26.05.2024).

10. Зуєтір М.Р.С.. Дослідження методів створення текстів згенерованих нейронною мережею та порівняння їх з природними. Proceedings of the IX International Scientific and Practical Conference. Prague, Czech Republic. 2024. Pp. 308-312 URL: https://isg-konf.com/theoretical-and-practical-aspects-of-the-development-of-science-and-education/ (дата звернення: 10.03.2024)

11. Python 3. Python 3. URL: https://python-scripts.com/ (дата звернення: 05.04.2024)

12. PyCharm. https://www.jetbrains.com/help/pycharm/quick-start-guide.html. URL: https://python-scripts.com/ (дата звернення: 05.04.2024)

13. What is Pycharm. https://intellipaat.com/blog/what-is-pycharm/. URL: https://python-scripts.com/ (дата звернення: 05.04.2024)

14. Import module. https://docs.python.org/3/reference/import.html. URL: https://python-scripts.com/: (дата звернення: 05.04.2024)

15. Shakespear dataset Kaggle: https://www.kaggle.com/datasets/kewagbln/shakespeareonline: (дата звернення: 18.04.2024)

16. Researching and training text generation model by using a dataset Shakespeare: https://colab.research.google.com/github/trekhleb/machine-learning-experiments/blob/master/experiments/text_generation_shakespeare_ (дата звернення 20.04.2024)

17. Jay Alammar The Illustrated Word2vec URL: https://jalammar.github.io/illustrated-word2vec/ (дата звернення: 26.04.2024)

18. Суяргулова Е.Б. Вечур О.В: Статистична кластеризація новин з урахуванням абзаців, як смислових одиниць тексту: https://scholar.google.com.ua/citations?view_op=view_citation&hl=uk&user=Ew8uERcAAAAJ&citation_for_view=Ew8uERcAAAAJ:pqnbT2bcN3wC – Січень 2009

19. Вечур О.В, Ляпота В.М, Євгенія Суяргулова: Побудова мережевої моделі новинного веб-контенту з використанням методу для визначення плагіату:https://scholar.google.com.ua/citations?view_op=view_citation&hl=uk&

user=Ew8uERcAAAAJ&citation_for_view=Ew8uERcAAAAJ:M3NEmzRMIkIC:
- Березень 2013.