**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**
**Західноукраїнський національний університет**
**Факультет комп'ютерних інформаційних технологій**
Кафедра комп'ютерної інженерії

**СИЧ Тарас Андрійович**

# Алгоритми формування науково-технічних текстів на основі генеративного інтелекту / Algorithms for generating scientific and technical texts based on generative intelligence

спеціальність: 123 - Комп'ютерна інженерія
освітньо-професійна програма - Комп'ютерна інженерія
Кваліфікаційна робота

Виконав студент групи Кім-21
Т.А. Сич
_____

Науковий керівник
д.т.н., професор, О. М. Березький

_____

Кваліфікаційну роботу допущено
до захисту:

"___" _____ 20___ р.

Завідувач кафедри
_____О.Л. Дубчак

# АНОТАЦІЯ

Сич Т.А. Алгоритми формування науково-технічних текстів на основі генеративного інтелекту. – Рукопис.

Кваліфікаційна робота на здобуття освітнього ступеня «магістр» за спеціальністю 123 «Комп'ютерна інженерія», освітньо-професійна програма. Західноукраїнський національний університет, Тернопіль, 2024.

Робота написана обсягом 89 сторінок і містить 24 ілюстрації, 1 таблицю, 2 додатків та 63 джерела за переліком посилань.

Метою роботи є розроблення алгоритмів формування науково-технічних текстів на основі генеративного інтелекту.

Методи досліджень. Для розв'язання поставлених задач у кваліфікаційній роботі використано:методи аналізу наявних штучних інтелектів (перегляд chat gpt, bingo, gemini); аналізу наукових текстів (науково технічні статті); програмування (для проектування програми, яка буде генерувати науково технічні тексти із наявних текстів).

Результати дослідження: алгоритми опису, аналізу та генерування наукових текстів, на основі наявної бази, програмна ситсема аналізу згенерованного тексту та аналізу запитів для генерування тексту.

Результати роботи можуть бути використані в науковій практиці, для генерування наукових текстів згідно бази даних, яка характеризує конкретну тематику.

Орієнтовні напрямки розвитку дослідження: розроблення спеціалізованої локальної системи генерування наукових текстів та аналізу навчання з наявної бази; розвиток та покращення якості генерування наукових текстів для наукових статей.

КЛЮЧОВІ СЛОВА: АЛГОРИТМ, NLP, АНАЛІЗ, ШТУЧНИЙ ІНТЕЛЕКТ, ПРОГРАМНА СИСТЕМА.

# ANNOTATION

Sych T.A. Algorithm for the formation of scientific and technical texts based on generative intelligence. – Manuscript.

Qualification work for obtaining the educational degree «Master» in specialty 123 «Computer Engineering», educational and professional program. West Ukrainian National University, Ternopil, 2024.

The work is 89 pages and contains 24 illustrations, 1 table, 2 appendices and 63 sources according to the list of references.

The purpose of the work is to develop algorithms for the formation of scientific and technical texts based on generative intelligence.

Research methods. To solve the tasks in the qualification work, the following methods were used: analysis of existing artificial intelligences (viewing chat gpt, bingo, gemini); analysis of scientific texts (scientific and technical articles); programming (for designing a program that will generate scientific and technical texts from existing texts).

Research results: algorithms for description, analysis and generation of scientific texts, based on the existing database, software system for analysis of generated text and analysis of requests for text generation.

The results of the work can be used in scientific practice, to generate scientific texts according to a database that characterizes a specific topic.

Indicative directions of research development: development of a specialized local system for generating scientific texts and analyzing learning from the existing database; development and improvement of the quality of generation of scientific texts for scientific articles.

KEY WORDS: ALGORITHM, NLP, ANALYSIS, ARTIFICIAL INTELLIGENCE, SOFTWARE SYSTEM.

ЗМІСТ

# ВСТУП

Штучний інтелект (ШІ) є однією з найбільш перспективних галузей комп'ютерних наук, що займається створенням систем, здатних виконувати завдання, які традиційно вимагають людського інтелекту. В останні роки ШІ суттєво вплинув на різні сфери життя, зокрема на медицину [1-10], фінанси та промисловість. Однак порівняння ШІ з людським інтелектом (ЛІ) залишається складним завданням, яке включає вивчення як зовнішніх проявів інтелекту, так і внутрішніх механізмів, таких як когнітивні функції, емоції та інтуїція. Це питання має велике значення для розвитку технологій та їхнього впровадження у сучасному суспільстві.

Актуальність теми полягає у вивченні та порівнянні ШІ і ЛІ, що є особливо важливим в умовах швидкого розвитку ШІ. Багато аспектів людського інтелекту, таких як емоційне сприйняття та творчість, досі складно емулювати в ШІ, що ставить перед наукою нові виклики й етичні питання щодо впровадження таких технологій.

Мета роботи : Дослідження є порівняння ШІ з ЛІ через призму когнітивних функцій, таких як розуміння мови, прийняття рішень і адаптація до нових умов. Завдання включають визначення основних когнітивних функцій людини та їх аналогів у ШІ, а також розробку рекомендацій щодо вдосконалення технологій.

Об'єкт дослідження: локальна мережа генерації науково тексту.

Предмет дослідження: когнітивні функції, такі як розуміння мови, прийняття рішень та адаптація до нових умов.

Задачі досліджень:

1. Провести теоретичний аналіз літератури, математичне моделювання, кросс-дисциплінарний підхід і етичний аналіз.
2. Розробити алгоритми та реалізувати систему, що виконує аналіз тексту та його відтворення згідно запитів.

3. Розробити програмне забезпечення яке підтримує генерацію данних, із постійним поновленням даних та збільшення бази інформації.
4. Провести експерементальне тестування алгоритму генерування науково тексту для оцінки його ефективності, використовуючи технології NLT.

Наукова новизна одержаних результатів: Детальне порівняння когнітивних функцій людини і ШІ та виявлення нові можливості для вдосконалення алгоритмів ШІ у локальній мережі. Також розробка етичної рекомендації для використання ШІ в суспільстві.

Практичне значення результатів. Отримані результати можуть бути використані для вдосконалення ШІ-систем у галузях, що потребують адаптивних рішень, таких як медицина, безпека та фінанси. Також розроблені етичні рекомендації сприятимуть безпечному впровадженню ШІ. Публікації та апробація магістерської роботи :

Апробація роботи. Отримані результати опубліковані в межах всеукраїнськаї наукової конференції «Інтелектуальні комп'ютерні ситсеми та мережі», опубліковані тези доповіді [11, 12].

1. Сич Т.А. Алгоритими та програмний засіб формування науково-технічної статті / Т.А. Сич // Всеукраїнська науково-практична конференція студентів, аспірантів та молодих вчених «Інтелектуальні комп'ютерні системи та мережі», 5 листопада 2024, м. Тернопіль. – С. 113.

2. Сич Т.А. Аналіз програмних засобів для генерації тексту в текст / Т.А. Сич // IX науково-практична конференція молодих вчених та студентів «Інтелектуальні комп'ютерні системи та мережі», 21 травня 2024 року. – С. 18.

Кваліфікаційна робота містить три розділи, висновки, список використаних джерел та додатки [13].

Перший розділ присвячений розгляданню основного поняття штучного інтелекту. Порівнянню роботи штучних нейронних мереж, також розглянуто розвитку штучного інтелекту.

Другий розділ, присвячений науковій літературі, поняттю наукової статті, формою подання результатів дослідження, характеристика наукових статей,

мету їх публікації, структуру та компоненти, що повинні бути присутні у кожному науковому дослідженнію

Третій розділ присвячений розробці програмного засобу для генеративного інтелекту в умовах локальних мереж з локальною базою даних та проведенню експериментальних досліджень.

# 1 АНАЛІЗ ВИДІВ ГЕНЕРАТИВНОГО ІНТЕЛЕКТУ

## 1.1 Поняття штучного та генеративного інтелекту

Одним із самих сучасних, найцікавіших, найактуальніших напрямів в науці, є розвиток штучного інтелекту. За останні десятки років дослідження і експерименти із різних галузь дозволили просунутись в розвитку штучного інтелекту і уже навіть зараз вплинуло на сучасний світ. Протягом всієї історії людства усі задавались питанням здатністю мислити і генерувати щось нове із пройденого досвіту або вивченого. Сьогодення так і не дало однозначної відповіді на те як насправді відбувається процес мислення докорінно і як працює мозок. В загальному мозок людини працює наступним чином, шляхом передавання інформації по нейронах. Сам механізм роботи мозку описується як найскладніший орган, який являється найскладнішою живою структурою відомою у всесвіті. У людському мозку міститься близько ста мільярдів нейронів і всі використовуються. Кожен нейрон повинен спілкуватись із багатьма іншими нейронам. Він утворює зв’язки і таким чином обмінюється інформацією. Це скоординована дія відбувається у багатьох частинах мозку, що і називається правильною функціональністю всієї нервової системи. Самі нейрони спілкуються за допомогою електричних та і хімічних сигналів. Саме сенсорні стимули своєю чергою перетворюються в сигнали електричного типу. Електричний сигнал перетворюється по нейронах. Відповідні ці електронні сигнали викликають активність мозку, можливість думати та працювати належним чином. Кожна дія людини зумовлена роботою нейронів, точніше його зв’язками. В залежності від діяльності на даний момент людини сполука між нейронами посилюється чи послаблюється.

Нейрони, або нервові клітини, є основними функціональними одиницями людського мозку. Їх налічується близько 86 мільярдів, і вони утворюють нейронну мережу, що відповідає за всі наші думки, почуття та дії.

Нейрони мають довгасте тіло з відростками, які називаються дендритами та аксоном. Дендрити приймають сигнали від інших нейронів, а аксон передає сигнали далі. Нейрони обробляють інформацію, передаючи електричні та хімічні сигнали. Ці сигнали генеруються, коли дендрити отримують збудження від інших нейронів. Нейрони з'єднуються між собою за допомогою синапсів.

Синапс - це місце, де аксон одного нейрона контактує з дендритом іншого. Існує багато типів нейронів, які відрізняються за формою, розміром, функцією та типом хімічних сигналів, які вони використовують. Нейрони збирають інформацію від сенсорів, інших нейронів та навколишнього середовища, а потім обробляють її, щоб сформувати відповіді.

Нейрони передають інформацію один одному за допомогою електричних та хімічних сигналів. Нейронні зв'язки та синапси містять інформацію про наші спогади, знання та навички. Нейронні зв'язки можуть зміцнюватися або слабшати залежно від досвіду, що дозволяє нам навчатися та адаптуватися до нового середовища.

Нейрони – це складні та цікаві клітини. Їхня робота лежить в основі всього, що ми робимо, думаємо та відчуваємо. Вивчення нейронів може допомогти нам краще зрозуміти себе та те, як ми працюємо.

Зв'язки між нейронами формують нейронні мережі, які обробляють інформацію в мозку. Коли нейрон активується, він генерує електричний сигнал, який передається по аксонах до інших нейронів через синапси. У процесі передачі сигналу між нейронами використовуються хімічні речовини, які називаються нейромедіаторами. Цей процес дозволяє мозку обробляти інформацію, регулювати функції організму і контролювати поведінку.

Нейронні мережі мозку працюють у великій мірі паралельно, що дозволяє мозку швидко обробляти складну інформацію. Здатність мозку до навчання і адаптації базується на його здатності модифікувати сполучення між нейронами відповідно до нових досвідів і отриманої інформації. Цей процес відомий як нейропластичність і відіграє ключову роль у формуванні пам'яті, навчанні і розвитку. Нейронні мережі мозку відповідають за виконання різноманітних

функцій, включаючи сприйняття інформації з навколишнього середовища, регулювання рухів та внутрішніх органів, управління емоціями та формування пам'яті. Нейрони в мозку можуть утворювати складні мережі з великою кількістю зв'язків між собою, що дозволяє виконувати різноманітні завдання. Поняття нейропластичності вказує на здатність мозку змінювати свою структуру і функції відповідно до досвіду. Це означає, що мозок може перебудовувати свої нейронні мережі і змінювати зв'язки між нейронами відповідно до нових вимог і умов. Наприклад, у процесі навчання мозок формує нові зв'язки між нейронами, що дозволяє відображати нові знання і вміння. Таким чином, робота мозку людини є складним і динамічним процесом, який базується на взаємодії між нейронами у вигляді нейронних мереж. Ці мережі здатні до нейропластичності, що дозволяє мозку навчатися, адаптуватися і розвиватися протягом життя людини.

Штучний інтелект вже активно впливає на побутове життя людей і має потенціал змінити його ще більше в майбутньому. Однією з ключових областей застосування штучного інтелекту є автоматизація рутинних завдань, яка дозволяє відчутно зекономити час і зусилля людей. Наприклад, голосові асистенти, такі як Siri, Google Assistant або Alexa, дозволяють керувати побутовими пристроями, шукати інформацію в Інтернеті, ставити напоминання та планувати події за допомогою голосових команд.Ще однією важливою сферою є розпізнавання образів і машинне зорове сприйняття, що дозволяє розвивати автономні автомобілі, системи безпеки та відеоспостереження, а також допомагає в діагностиці медичних зразків і вирішенні проблем виробництва.

Штучний інтелект відіграє значну роль у сфері освіти, впливаючи на навчальний процес, методи навчання та доступ до знань. Однією з головних переваг є можливість індивідуалізації навчання. Системи штучного інтелекту можуть аналізувати інформацію про успішність кожного учня, їхні здібності та потреби, і розробляти персоналізовані програми навчання, що відповідають їхнім індивідуальним потребам. Важливою областю є підтримка вчителів у

проведенні уроків та оцінці знань учнів. Системи штучного інтелекту можуть аналізувати великі обсяги даних, щоб надати вчителям рекомендації щодо удосконалення програми навчання, а також допомагати в оцінці знань учнів і наданні їм індивідуального фідбеку. Додатково, штучний інтелект допомагає у підготовці навчальних матеріалів та засобів. Він може створювати інтерактивні навчальні програми, відеоуроки та інші матеріали, які роблять навчання більш цікавим і ефективним для учнів. В цілому, штучний інтелект має великий потенціал для трансформації сфери освіти, забезпечуючи індивідуалізований підхід до навчання, покращуючи якість навчання та підтримуючи вчителів у їхній роботі.У сфері медицини штучний інтелект вже знаходить застосування у вирішенні різних завдань, від діагностики захворювань до розробки нових лікарських засобів і лікувальних методів. Крім того, штучний інтелект може бути використаний для управління домашніми пристроями та системами, такими як енергоефективність, безпека та зручність управління, що робить побутове життя людей більш комфортним і ефективним.

Генеративний інтелект (Generative AI) в штучному інтелекті - це галузь, що створює системи, які здатні генерувати новий контент, що схожий на людський. Це може бути зображення, звуки, тексти або навіть код. Генеративний інтелект використовується для створення різних видів контенту, від мистецтва до музики, від текстів до фотографій.

Одним з основних методів генеративного інтелекту є використання глибоких навчальних моделей, таких як генеративні згорткові нейронні мережі (GAN) або автокодувальні нейронні мережі (autoencoders). GAN складається з двох нейронних мереж - генератора і дискримінатора. Генератор створює нові зразки, які намагаються виглядати як можна більше схожими на реальні дані, тоді як дискримінатор відрізняє справжні дані від фальшивих. Генеративний інтелект використовується в різних галузях. У мистецтві він може бути використаний для створення нових художніх робіт або інтерактивних виставок. У музиці - для створення нової музики або навіть інструментів. У кіноіндустрії - для створення реалістичних спецефектів або анімації. У наукових дослідженнях

- для створення нових хімічних сполук або матеріалів. Він також може бути використаний у відеоіграх для створення віртуальних світів або персонажів. Генеративний інтелект має великий потенціал у багатьох сферах, але водночас він стикається з викликами, такими як етичні питання щодо використання його в різних контекстах, а також з технічними обмеженнями, пов'язаними з навчанням моделей на великих обсягах даних.

## 1.2 Аналіз особливостей науково-технічних статей

Наукова стаття – це текст, що містить результати дослідження, аналізу або обговорення конкретної проблеми або питання у науці. Такий текст має специфічну структуру та стандарти оформлення, що допомагають чітко викласти інформацію та зробити її доступною для наукової спільноти.

Вступ. У першому абзаці зазвичай вказується проблема, яку досліджує стаття, її актуальність та мета. Він може містити короткий огляд літератури та пояснення вибору методів дослідження.

Методи. У цьому розділі автор(и) описують методику, що використовувалася у дослідженні. Це може включати опис експерименту, збір даних, статистичний аналіз тощо.

Результати. Тут представлені результати дослідження, часто у вигляді таблиць, графіків або інших візуалізацій. Результати можуть бути супроводжені коротким поясненням.

Обговорення. Автор(и) аналізують отримані результати, порівнюють їх зі старішими даними та іншими дослідженнями, обговорюють їхні можливі наслідки та вказують на напрямки подальших досліджень.

Висновки. У цьому розділі наводяться основні висновки статті, підсумовуючи основні результати та їх значення для науки або практики.

Література. Всі джерела, використані при написанні статті, перераховуються у цьому розділі відповідно до встановленого стандарту цитування (наприклад, APA, MLA тощо).

Наукові статті у технічних галузях мають давню та багату історію, що сягає коріння періодичних видань у XVII столітті. Важливим кроком у цьому розвитку став заснування перших наукових журналів, які стали платформами для обміну знаннями та ідеями. Наприклад, "Journal des sçavans", заснований у 1665 році у Франції, та "Philosophical Transactions of the Royal Society", заснований у 1666 році в Англії, стали одними з перших видань, що публікували наукові дослідження та сприяли розвитку наукового мислення.

У XX столітті з появою перших технічних журналів, таких як "Engineering", "IEEE Transactions on Information Theory", "Journal of Fluid Mechanics" та інших, почали з'являтися наукові статті, спрямовані на дослідження у галузі інженерії, технологій, комп'ютерних наук, фізики, хімії та інших технічних дисциплін. Такі статті зазвичай містять результати нових досліджень та розробок, вивчення принципів та законів природи, що лежать в основі технічних систем, а також аналіз і вирішення практичних проблем, що виникають у виробництві та наукових дослідженнях. Таким чином, технічні наукові статті є важливим джерелом знань та інформації для спеціалістів у цих галузях, а також сприяють розвитку технічних наук та технологій.

Науково-технічні статті відрізняються від інших видів наукових публікацій своєю спрямованістю на конкретні дослідження та розробки у галузі техніки, технологій, інженерії та природничих наук. Вони включають у себе об'єктивне викладення результатів досліджень, аналізів і практичних розробок, що базуються на конкретних фактах та експериментальних даних. Основним завданням таких статей є внесення вкладу в наукову спільноту, розширення наукового знання та вдосконалення технічних рішень і технологій.

Однією з ключових особливостей науково-технічних статей є їхня об'єктивність та відповідність науковим стандартам. Автори повинні дотримуватись точності та відкритості у поданні інформації, уникаючи

підкреслення власних думок або прикрашання фактів. Важливо також уникати конфлікту інтересів та недостовірної інформації. Структура науково-технічних статей має чітке визначення, що допомагає читачам швидко зорієнтуватися у змісті та знайти необхідну інформацію. Зазвичай вона включає в себе вступ, де формулюється мета дослідження, огляд літератури та існуючих підходів до проблеми, методологію, що описує методи дослідження, результати, де представлені отримані дані, обговорення, де аналізуються результати, та висновки, де формулюються основні висновки та можливі напрямки подальших досліджень. До важливих характеристик науково-технічних статей також відноситься їхня наукова новизна та оригінальність. Це означає, що результати дослідження повинні бути новими і відмінними від попередніх робіт у відповідній області. Оригінальність дослідження може виявлятися у впровадженні нових підходів, методів, концепцій або технологій. Окрім цього, науково-технічні статті часто включають в себе детальний аналіз отриманих результатів і їхнє порівняння з результатами попередніх досліджень. Це дозволяє читачам зрозуміти важливість та вплив дослідження на відповідну наукову галузь. Також важливою особливістю науково-технічних статей є наукова обґрунтованість і аргументованість висновків. Автори повинні чітко демонструвати логіку своїх доводів та використовувати відповідні методи дослідження для підтвердження своїх тез.

Загалом, науково-технічні статті відіграють важливу роль у розвитку науки та технологій, сприяючи обміну знаннями та інноваціям у відповідних галузях. Їхня структурованість, об'єктивність, наукова обґрунтованість та оригінальність дозволяють забезпечити високий рівень інформаційної цінності та достовірності результатів досліджень [19-22].

## 1.3 Аналіз програмних засобів

Генеративні нейронні мережі (ГНМ) представляють собою тип штучного інтелекту, що здатен створювати нові дані, які подібні до тих, на яких вони були навчені. Ці мережі складаються з нейронів, організованих у шари, і вони здатні навчатися та адаптуватися, подібно до людського мозку.У сфері освіти ГНМ можуть мати значний вплив. Наприклад, вони можуть допомагати в індивідуалізації навчання, створюючи програми, що відповідають індивідуальним потребам кожного учня. Також вони можуть бути корисними для вчителів, допомагаючи в проведенні уроків та оцінці знань учнів. ГНМ можуть аналізувати великі обсяги даних та надавати рекомендації щодо удосконалення навчальної програми.Додатково, в сфері створення контенту ГНМ можуть бути корисними для створення текстів, зображень, музики та іншого контенту. Вони можуть створювати нові матеріали або стилізувати існуючі, що може бути корисним для навчальних матеріалів, медіа-контенту та інших областей.

Штучні нейронні генеративні мережі (ШНГМ) є однією з найбільш інноваційних та цікавих галузей штучного інтелекту. Їхня ідея полягає в тому, щоб навчити комп'ютер генерувати нові дані, які є схожими на ті, що були використані для навчання. Це може включати генерацію зображень, текстів, музики та іншого контенту. Ідея створення ШНГМ виникла з бажанням створити систему, яка могла би творити нові дані, що не були б просто копією вхідних даних, а мали би нову, творчу цінність. Перші дослідження у цій області почалися приблизно в 2014 році, коли була запропонована модель генеративних змагальних мереж (GAN). GAN складаються з двох нейронних мереж - генератора і дискримінатора, які конкурують між собою. Генератор створює нові дані, а дискримінатор намагається розрізнити, що це за дані - реальні чи сгенеровані. Ця конкурентна динаміка дозволяє моделі навчатися створювати дедалі більш реалістичні дані. Злети ШНГМ пов'язані з їхнім успіхом у різних галузях. Наприклад, вони використовуються для створення реалістичних

зображень людей, об'єктів та пейзажів, для генерації текстів, створення музики та іншого контенту. Ці технології мають великий потенціал у таких галузях, як ігрова індустрія, дизайн, медіа та інші.

Проте разом із злетами ШНГМ також відомі невдачі. Наприклад, іноді моделі можуть генерувати контент, який має негативний вплив на суспільство, такий як фейкові новини або deepfakes. Також існують проблеми з етичним використанням цих технологій, зокрема у сферах приватності та безпеки даних.

Усупереч цим викликам, ШНГМ залишаються однією з найбільш захоплюючих галузей штучного інтелекту, що має великий потенціал для творчого використання в майбутньому.

На даний момент розвиток генеративних штучних нейронних мереж досягнув багато успіхів. Практичні використання, які вище вказувались стали буденністю для впевнений користувачів. Особливого розвитку в цій галузі можна виділити декілька програмних продуктів, які можуть впевнено назвати себе штучним інтелектом.

Одним із революційних кроків у галузі генеративного інетлекту можна приписати компанії OpenAI. OpenAI - це дослідницька компанія зі штучного інтелекту, заснована в 2015 році з метою просування та розробки штучного інтелекту на користь всього людства. Одним з ключових напрямків роботи OpenAI є створення штучного загального інтелекту (AGI), тобто інтелекту, що перевершує людський у всіх аспектах. Компанія відома своїми великими досягненнями в галузі машинного навчання та штучного інтелекту. Наприклад, в 2016 році вони представили систему OpenAI Gym, яка стала стандартом для тестування алгоритмів з reinforcement learning. Також OpenAI стояла за розробкою деяких з найбільш вражаючих та передових моделей глибокого навчання, таких як GPT (Generative Pre-trained Transformer) та DALL-E, які відомі своєю здатністю до генерації людино-подібних текстів та зображень відповідно. Одним з основних принципів OpenAI є збереження безпеки та етичності у використанні штучного інтелекту. Вони активно досліджують можливі наслідки та ризики впровадження AGI та розробляють стратегії для їхнього управління.

Крім того, OpenAI прагне зробити свої розробки доступними для всіх, публікуючи більшість своїх досліджень та розвитку відкритою формою. Особливо популярність серед людства і всього світу набув саме штучний інтелект GPT .


Рисунок 1.1 – Логотип генеративного інтелекту gpt від open ai

Чат GPT (Chatbot на базі Генеративно-Підсилених Трансформерів) - це програмне забезпечення, яке використовує модель глибокого навчання, зокрема технологію трансформерів, для створення відповідей на запитання користувачів у формі текстового чату. Ці системи здатні взаємодіяти з людьми, сприймаючи текстові вхідні дані від користувачів та генеруючи відповіді, які намагаються бути зрозумілими та змістовними. Основним принципом роботи чатботів на базі ГПТ є здатність моделі аналізувати вхідний текст, розуміти його семантику та контекст, і генерувати відповіді, що відповідають на запитання або коментарі користувачів. Це досягається за допомогою навчання моделі на великих корпусах текстів, що дозволяє їй засвоювати мовні правила та структури, а також

розуміти зв'язки між словами та фразами [30]. Чатботи на базі ГПТ можуть бути використані для різних цілей, включаючи підтримку клієнтів, відповіді на запитання, консультації з певних питань, розваги та багато іншого. Вони можуть працювати в реальному часі, спілкуючись з користувачами через текстовий інтерфейс, або в складі інших програм, наприклад, веб-сайтів або додатків, де вони виконують певні функції або завдання.

Одним із ключових викликів у розвитку чатботів на базі ГПТ є досягнення балансу між автономністю та точністю відповідей. Важливо, щоб чатбот був здатний генерувати відповіді, які не лише коректні з граматичної точки зору, але й відповідали на питання або коментарі користувачів з врахуванням їхнього контексту та потреб. Для того щоб зрозуміти роботу чат гпт і як він генерує текст, можна поставити запитання, щодо генераціх тексту. Для теми можна взяти статтю із джерела «Радіо свобода». Штучний інтелект зможе згенерувати текст на основі даних, які він здобув у роботі із даними, які отримував із різних джерел. На рисунку зображено як генеративний штучний інтелект згенерував статтю на тему: «Штучний інтелект – загроза для людства?».

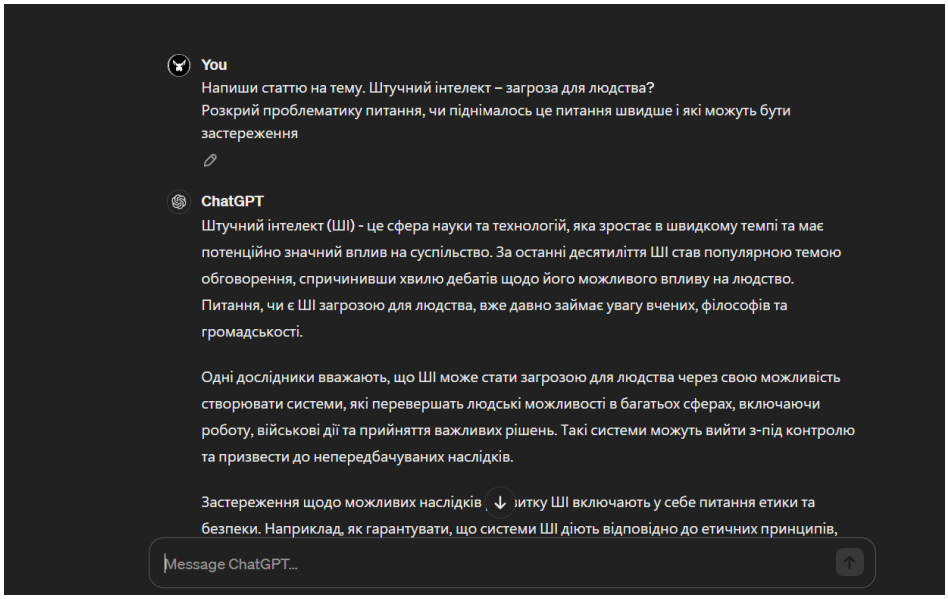


Рисунок 1.2 – Результат генерації тексту чат ГПТ

Отже, можна спостерігати що штучний інтелект зрозумів запитання і зміг коректно описати питання, шкоди штучного інтелекту та його користь.

Одним із найпопулярніших штучних інтелектів, окрім попередньо згаданий можна назвати ще від компанії Microsoft. Актуальність цієї теми набула настільки великої популярності, що це заставило світових гігантів, які є провідні в розвитку технологій та її модернізації вивчати це питання із великою серйозністю.

Microsoft — це одна з найбільших технологічних компаній у світі, заснована у 1975 році. Вона відома своїми операційними системами, програмними продуктами та хмарними послугами. Розвиток штучного інтелекту є однією з ключових областей, в якій Microsoft активно працює.

Деталізуємо внесок Microsoft у розвиток штучного інтелекту:

1. Інвестиції в дослідження: У 2019 році Microsoft інвестувала 1 мільярд доларів у провідну лабораторію OpenAI, спеціалізовану у дослідженні штучного інтелекту. Це партнерство розширюється, і сума інвестицій стала багатомільярдною. Microsoft прагне лідерства в галузі штучного інтелекту.

2. Штучний інтелект у бізнесі: Microsoft розробляє інноваційні рішення для бізнесу, використовуючи штучний інтелект. Один з таких інструментів — Copilot. Це інтелектуальний помічник, який використовує великі мовні моделі для допомоги розробникам та користувачам у складних завданнях, від написання коду до створення презентацій.

3. Застосування штучного інтелекту: ШІ дозволяє бізнесам ефективніше прогнозувати попит, адаптувати стратегії продажу, планувати ресурси та оптимізувати процеси. Вона знаходить застосування в різних галузях, таких як маркетинг, медицина та навчання.

4. Інтелектуальний помічник: Microsoft має доступ до великих мовних моделей, включаючи LLM (Large Language Model), і постачає моделі Chat GPT. Copilot — це один з результатів цієї роботи. Він допомагає розробникам та користувачам у різних завданнях, від написання комерційних пропозицій до створення зображень для презентацій.

На рисунку 1.3 зображенно логотип який описує штучний інтелект Microsotft Bing.


Рисунок 1.3 – Логотип генеративного інтелекту Bing

Також для того щоб перевірити можливість цього програмного продукту генерувати текст задамо теж саме запитання, яке ставили попередній штучній мережі. Як він зможе згенерувати проблематику питання так як опише застереження, щодо небезпечності цього. На рисунку 1.4 зображено приклад генерації тексту Bing на тему : «Штучний інтелект – загроза для людства?».

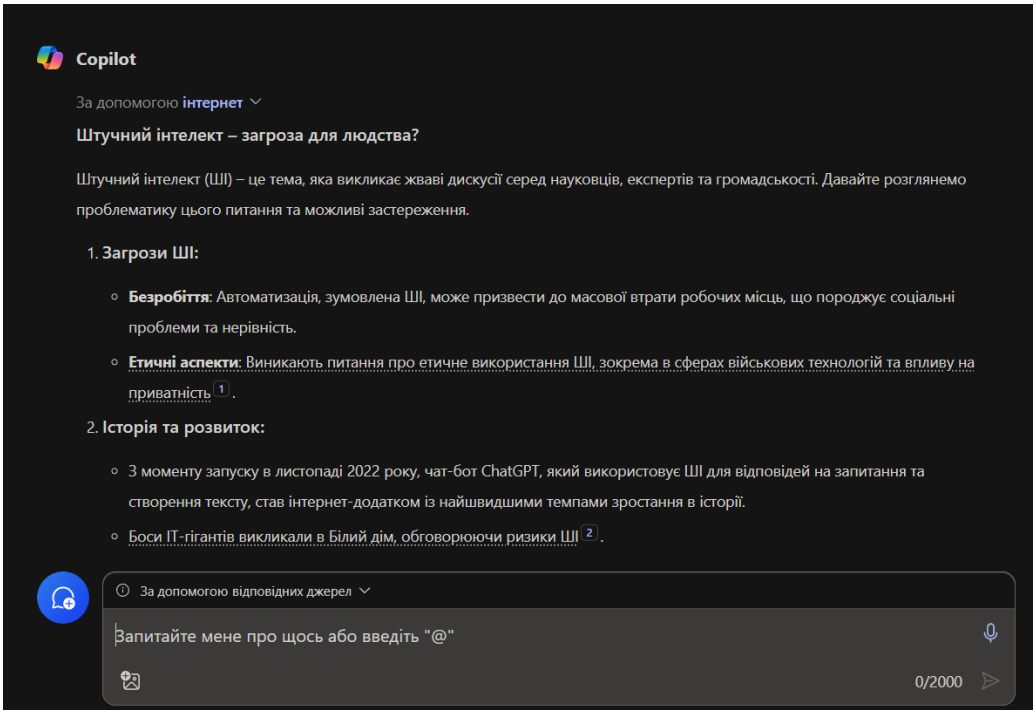


Рисунок 1.4 – Результат генерації штучного інтелекту bing

Отже, згідно попередніх даних можна отримати результат. Штучний інтелект також зміг докладно зрозуміти питання, щодо шкоди штучного інтелекту та застереження.

Наступним програмним продуктом, який розглянеться є продуктом від компанії Google. Провідна компанія, яка рахується також світовим монополістом в галузі штучного інтелекту розвивала питання і дослідження цієї галузі і також змогла отримати певні результати, які світ уже теж може використовувати. Компанія славиться своїми результатами та внеском в науку в цілому. Тому на даний момент розвиток ШІ в цій компанії є дуже правильним рішенням. На рисунку 1.5 зображено штучний інтелект від компанії гугл Gemini.


Рисунок 1.5 – Програмний продукт від Google Gemini

Google – це американська транснаціональна технологічна компанія, що спеціалізується на пошукових системах, онлайн-рекламі, хмарних обчисленнях, комп'ютерному програмному забезпеченні, квантових обчисленнях, штучному інтелекті та електроніці. Її штаб-квартира знаходиться в Маунтін-В'ю, штат Каліфорнія, а материнською компанією є Alphabet Inc. Google активно розвиває штучний інтелект (ШІ) з 2006 року, коли була заснована дослідницька група

Google Brain, що зосередилась на глибокому навчанні. З того часу компанія зробила значні кроки в цій галузі, випустивши ряд продуктів та сервісів, що використовують ШІ.

Фірма Google у генеративному інтелекті має такі досягнення:

1. 2011: Запуск Google+, соціальної мережі, яка використовує ШІ для персоналізації контенту.
2. 2012: Запуск Google X, дослідницької лабораторії, яка розробляє нові технології, включаючи ШІ.
3. 2014: Запуск Google Translate, системи машинного перекладу, яка використовує ШІ.
4. 2015: Запуск Google Photos, фотоальбому, який використовує ШІ для автоматичної організації фотографій.
5. 2016: Запуск Google Assistant, віртуального помічника, який використовує ШІ для розпізнавання мови та відповідей на запитання.
6. 2017: Запуск Google Cloud Platform, хмарної платформи, яка пропонує послуги ШІ.
7. 2018: Запуск TensorFlow, платформи машинного навчання з відкритим кодом.
8. 2019: Запуск Google AI Platform, платформи для розробки та розгортання моделей ШІ.
9. 2020: Запуск LaMDA, моделі ШІ, яка може вести розмови.
10. 2021: Запуск PaLM, моделі ШІ, яка може виконувати багато завдань.

Google прагне використовувати ШІ для того, щоб зробити світ кращим. Компанія використовує ШІ для вирішення таких проблем, як зміна клімату, бідність та хвороби. Google також прагне зробити ШІ доступним для всіх, а не лише для обраних.

Сьогодні Google є одним із лідерів у галузі ШІ. Компанія має команду дослідників світового рівня, які працюють над розробкою нових технологій ШІ. Google також інвестує значні кошти в дослідження та розробку ШІ.

Для заключення всіх досліджень також здійсниться перевірка на правильність генерації тексту. Розкриється теж саме питання що і швидше. На рисунку 1.6 зображено генерація тексту на тему: «Штучний інтелект – загроза для людства?».

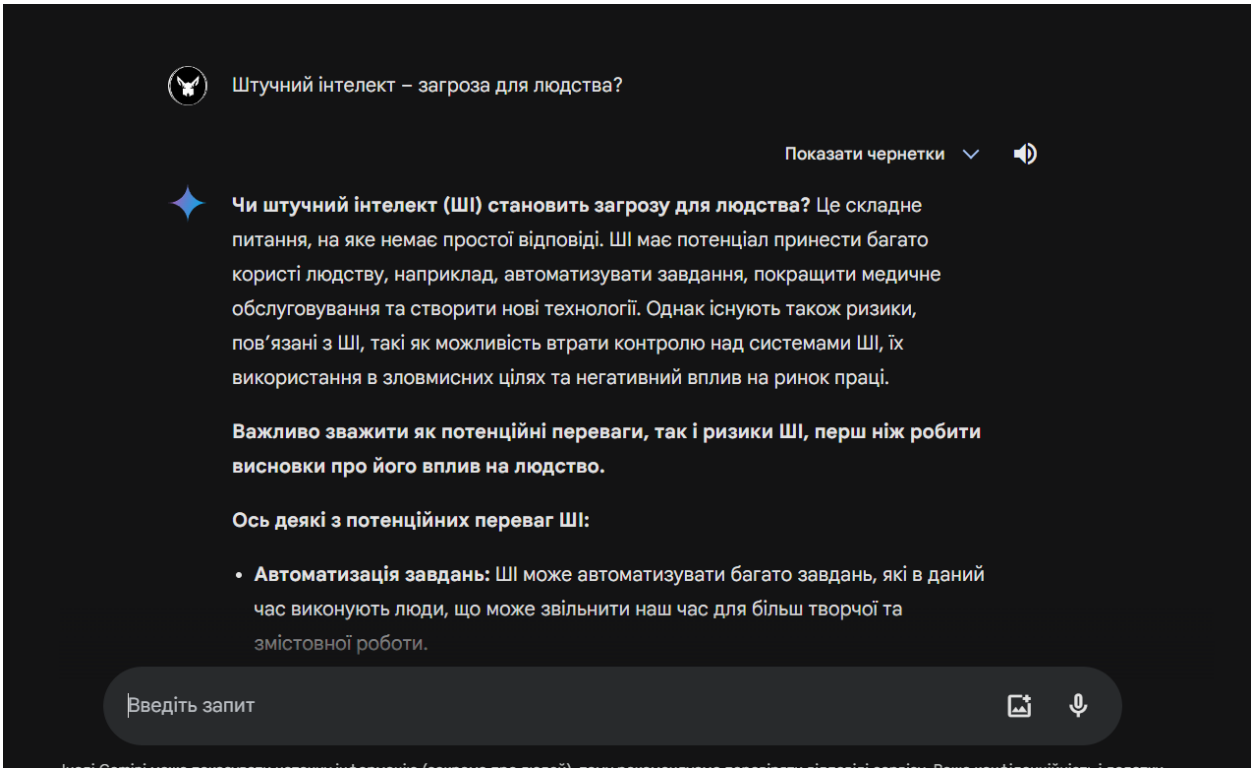


Рисунок 1.6 – Результат генерації тексту від Gemini

На рисунку 1.6 також можна зрозуміти що штучний інтелект зміг опрацювати питання та розгорнуто описати відповідь, щодо поставленого питання. Також якщо звернутись до статті, яка була опублікована, на сервісі «Радіо Свобода» можна побачити схожіть поглядів, та його аргументацію. Згідно цих досліджень можна зрозуміти, що розвиток ШІ настільки сильно уже розвинувся що можна уже уявляти новітні технології щодо повного залучення до багатьох сфери діяльності людини.

## 1.4 Висновки до розділу 1

В даному розділі розглядались феномени штучного інтелекту та інтелекту людини. Подібність роботи та аналізу інформації. Також розглядались наявні шутчні інтелекти від провідних компаній та їхній розвиток. На даний момент є 3 основні компанії які вносять вагомий вклад в наукову історію та історію розвитку людства. Наведені приклади роботи штучних інтелектів, їхні відмінності та результати роботи в експерементальних умовах. Задавши одне завдання, усі по своєму змогли виконати його успішно. Відповідно це доказує що розвиток штучного інтелекту в очах сьогодення є прогресивним та багатообіцяючим.

# 2 АЛГОРИТМИ ФОРМУВАННЯ НАУКОВО ТЕХНІЧНИХ ТЕКСТІВ

## 2.1 Алгоритм структурування науково-технічних текстів

Алгоритми структурування науково-технічних текстів відіграють критично важливу роль у формуванні логічної, послідовної та доступної для розуміння наукової роботи [20-25]. Головне завдання таких алгоритмів – забезпечити узгодженість тексту, дозволяючи читачу швидко і чітко зрозуміти основні ідеї, методи, результати та висновки. Найважливішими етапами структурування наукового тексту є створення чіткої архітектури документа, планування послідовності змісту та визначення ключових розділів, що відповідають вимогам наукового стилю. Нижче наведено алгоритм статті, який складається з наступних кроків:

Вступ: Створення контексту та обґрунтування теми. Вступ – це перший і найважливіший розділ наукової роботи, який має на меті забезпечити належний контекст для подальшого викладу. Для структурування цього розділу варто використовувати алгоритми, що базуються на конкретній послідовності: (1) опис загальної тематики та її актуальності, (2) огляд існуючих проблем або викликів, (3) виявлення розривів у знаннях або відсутності певних досліджень, і (4) формулювання основної мети статті та її завдань. Алгоритми допомагають розміщувати цей зміст у логічному порядку, підтримуючи фокус на ключових питаннях, щоб читач міг відразу зрозуміти, чому саме ця робота важлива. Важливо також дотримуватися послідовності між кожною секцією, використовуючи механізми перевірки логіки (наприклад, логічні блок-схеми), щоб уникнути неоднозначності та забезпечити зосередженість на меті дослідження.

Аналіз літератури: Аналіз попередніх досліджень та виявлення прогалин. Огляд літератури в науково-технічних текстах має бути організований за допомогою алгоритмів, що допомагають систематизувати великий обсяг інформації з літературних джерел. Цей процес часто включає побудову

тематичних карт та діаграм зв'язків для класифікації ідей, методів, досліджень та висновків інших авторів. Завдяки цьому можна структурувати огляд літератури у вигляді тематичних підрозділів, де кожен підрозділ охоплює певну тему, підхід або метод. Такі алгоритми також полегшують пошук і аналіз ключових джерел, забезпечуючи надійний фундамент для подальшої розробки власних досліджень. Використання програмних засобів, наприклад, бібліографічних менеджерів та аналітичних платформ для синтезу літератури, дозволяє автоматизувати багато процесів структурування огляду.

Методи: Логічна послідовність і деталізація кроків.Розділ методів є основою для відтворюваності дослідження, тому він повинен бути структурованим чітко і логічно. Тут доцільно застосовувати алгоритми структурованого опису процедур, що включають детальну послідовність кроків та обґрунтування вибору методів. Один із популярних підходів – використання алгоритмів блок-схем та моделей робочих процесів для ілюстрації етапів дослідження, що дозволяє не тільки описати, але й візуалізувати процедуру. Кожен підрозділ повинен фокусуватися на конкретній методологічній техніці, яка була застосована, з обґрунтуванням вибору кожного методу, що підвищує довіру до наукових результатів.

Експерименти: Експерименти в наукових статтях є основою для підтвердження або спростування гіпотез, перевірки методів чи дослідження нових явищ. У цьому розділі мають бути детально описані всі проведені дослідження, умови їх виконання, отримані результати та їхня інтерпретація. Зокрема, автори можуть перевіряти ефективність розроблених алгоритмів структурування тексту, застосовуючи їх до реальних наукових документів і порівнюючи з існуючими методами за критеріями, такими як швидкість роботи, якість структурування або рівень зрозумілості тексту для читача. Також важливо аналізувати вплив різних параметрів на результати роботи алгоритмів, наприклад, зміни критеріїв для визначення структури тексту або вибору послідовності секцій. Крім того, доцільним є порівняльний аналіз кількох підходів або методів на одному наборі даних, що дозволяє обґрунтувати вибір

найкращого з них. Окрему увагу можна приділити перевірці відтворюваності, тобто використанню методів на незалежних наборах даних для підтвердження, чи дозволяють вони отримати схожі результати. Експерименти із залученням користувачів для оцінки практичності алгоритмів також мають велике значення – наприклад, це може бути аналіз часу виконання завдань, зручності інтерфейсу чи якості отриманих текстів. Також можна моделювати реальні сценарії використання, як-от автоматизація підготовки звітів або статей із дотриманням певних вимог. Результати експериментів бажано аналізувати статистично, щоб оцінити значущість отриманих даних і довести надійність висновків. На завершення доцільно представити дані у формі графіків, таблиць чи діаграм, що сприятиме кращому розумінню результатів. Такі експерименти дозволяють глибше проаналізувати розроблені методи, виявити їхні сильні сторони та обмеження, а також запропонувати шляхи вдосконалення.

Результати: Організація даних та інтерпретація висновків.Алгоритми структурування результатів важливі для ефективної передачі отриманих даних. Зазвичай результати подаються у вигляді таблиць, графіків та діаграм, які доповнюються поясненнями. Для ефективного структурування цього розділу можна застосовувати алгоритми автоматичного побудови графіків і зведених таблиць, що полегшує розуміння інформації та допомагає акцентувати увагу на основних висновках. Варто уникати перевантаження читача надмірними деталями, тому варто застосовувати алгоритми вибіркової деталізації – вони дозволяють зосередитися на ключових даних і уникнути зайвої інформації, яка може відволікати від основних висновків.

Висновки: Логічне завершення і підсумки. Останнім етапом у структурі наукового тексту є висновки, що повинні логічно завершувати роботу, наголошуючи на досягненні поставлених цілей і результатах. Алгоритми допомагають структуровано викласти висновки так, щоб вони не лише підсумовували отримані дані, але й пропонували можливі напрями для подальших досліджень. Це досягається за допомогою алгоритмів логічного підсумування, які чітко організовують висновки відповідно до основних цілей

дослідження, дозволяючи читачу легко співвіднести отримані результати з початковими завданнями роботи.

Програмні засоби для структурування тексту [14-18]. Існує низка інструментів для підтримки структурування науково-технічних текстів. Наприклад, LaTeX широко використовується для форматування наукових документів, оскільки дозволяє організувати текст з математичними формулами, графіками, таблицями та посиланнями на літературні джерела. Такі програми, як EndNote, Mendeley чи Zotero, значно полегшують управління літературними джерелами. Інші інструменти, як-от Microsoft Word зі спеціальними шаблонами та макросами, допомагають підтримувати послідовність стилю в документі.

## 2.2 Алгоритм автоматизації написання та аналізу наукових текстів

Алгоритм автоматизації написання та аналізу наукових текстів включає цілу низку технологій та підходів, які значно полегшують і пришвидшують процес створення, редагування, оцінки якості та аналізу наукових робіт. Від класичних методів написання до сучасних підходів, які інтегрують штучний інтелект та обробку природної мови, автоматизація наукового процесу допомагає значно покращити ефективність і точність досліджень. Сучасні інструменти не лише допомагають автоматично генерувати тексти, а й виконують складні завдання з перевірки стилістичних і граматичних помилок, аналізу літератури, виявлення плагіату та створення наукових цитувань, що дозволяє дослідникам зосередитись на змісті роботи, мінімізуючи технічні аспекти.

Однією з основних складових алгоритму автоматизації є системи автоматичного генерації текстів, які використовують алгоритми обробки природної мови (NLP). Ці інструменти здатні значно полегшити процес створення наукових робіт, автоматично генеруючи частини тексту або

доповнюючи інформацію в уже написаних розділах. Завдяки таким системам, як OpenAI GPT або Cohere API, можна автоматично формулювати вступи або створювати огляди літератури, що особливо корисно для науковців, які працюють із великими обсягами інформації. Ці інструменти дозволяють сформулювати базові розділи статті, наприклад, вступ або обговорення літератури, на основі простих ключових слів або коротких фрагментів, після чого автор може адаптувати та відредагувати текст відповідно до вимог наукового стилю. Однак створення тексту — це лише перший етап. Для того, щоб науковий текст був готовий до публікації, необхідно провести ретельну перевірку стилю та граматики. Тут на допомогу приходять різноманітні інструменти автоматичної перевірки, такі як Grammarly, ProWritingAid або LanguageTool. Вони автоматично виявляють граматичні помилки, орфографічні неточності та стилістичні недоліки, що дозволяє автору значно зменшити час, витрачений на редагування. Такі програми допомагають забезпечити відповідність тексту академічним стандартам, підтримуючи точність і ясність викладення, що критично важливо для наукових публікацій [21, 30,38, 50, 56, 62].

Перевірка на оригінальність є важливою частиною підготовки наукових текстів, адже публікації повинні бути унікальними та не містити плагіату. В сучасному науковому середовищі для цієї мети активно використовуються програми, як-от Turnitin, iThenticate та Plagscan. Вони автоматично порівнюють текст із великими базами даних, визначаючи можливі збіги з іншими публікаціями. Це не лише допомагає уникнути плагіату, а й дозволяє дослідникам точніше визначати джерела, що використовуються, та забезпечувати коректне цитування, що має вирішальне значення для дотримання етичних стандартів у науці.

Ще одним важливим етапом є аналіз великих обсягів наукових текстів. У світі, де з кожним роком публікується все більше наукових статей, традиційні методи огляду літератури стають малоефективними. Для цього створено спеціалізовані платформи, такі як VOSviewer або Publish or Perish, які допомагають науковцям автоматизувати процес аналізу літератури. Ці

інструменти дозволяють візуалізувати зв’язки між різними дослідженнями, класифікувати їх за темами, методами або впливовістю, що дає змогу ефективно організовувати огляд літератури, виділяти ключові наукові напрямки та тенденції. Такі платформи значно полегшують процес пошуку важливих статей, що дозволяє науковцям зосередитись на головних темах їхніх досліджень.

Технології обробки природної мови (NLP) відіграють важливу роль у більш глибокому аналізі наукових текстів. Сучасні алгоритми NLP, такі як BERT або RoBERTa, здатні автоматично визначати основні теми, ключові слова та навіть емоційне забарвлення тексту. Ці технології дозволяють швидко обробляти великі обсяги інформації, виділяючи важливі аспекти та допомагаючи автору структурувати текст відповідно до основних вимог статті. Вони також можуть допомогти у виявленні прогалин у знаннях або напрямків для подальших досліджень, що є корисним для створення якісних оглядів літератури.

Для управління бібліографією та коректного оформлення посилань активно використовуються спеціалізовані програми, такі як EndNote, Mendeley та Zotero. Вони дозволяють науковцям зберігати, організовувати та автоматично генерувати посилання на використані джерела, значно полегшуючи процес підготовки списку літератури. Це також зменшує ймовірність помилок при оформленні цитувань та дозволяє зосередитись на змісті роботи, замість того, щоб витрачати час на технічні аспекти оформлення.

Використання інтелектуальних систем для рецензування та редагування текстів також є важливою складовою сучасного процесу автоматизації написання наукових статей. Ці системи здатні не тільки перевіряти правильність та чіткість тексту, а й аналізувати його логічність і послідовність. Вони можуть вказувати на можливі логічні прогалини, недостатню чіткість викладу або пропонувати альтернативні варіанти для покращення структури статті. Це значно спрощує процес рецензування і редагування, оскільки дозволяє автоматично виявляти проблеми, які можуть бути важко помітними для самого автора [26-29].

## 2.3 Алгоритм адаптації та персоналізації наукових текстів

Алгоритм адаптації та персоналізації наукових текстів включає використання передових технологій та підходів для індивідуалізації створення та редагування наукових робіт, враховуючи особливості кожного автора, галузі досліджень та цільової аудиторії. Сучасні методи адаптації дозволяють не лише покращувати стиль і граматику, але й створювати тексти, які краще відповідають вимогам певної наукової спільноти чи видавництва, а також оптимізувати процес рецензування та публікації. Інтеграція новітніх технологій, таких як обробка природної мови (NLP), машинне навчання та метадані, дає змогу значно підвищити ефективність написання наукових статей, враховуючи індивідуальні вимоги й тренди в науковому середовищі. Нижче наведені основні аспекти, які полягають у адаптації та персоналізації наукових текстів:

Адаптація наукових текстів до вимог конкретної наукової спільноти — це перший важливий етап персоналізації. Для того, щоб текст був сприйнятий і зрозумілий цільовою аудиторією, автори повинні враховувати специфічні вимоги до стилю, структури та змісту, які прийняті в конкретних наукових журналах чи середовищах. Інструменти, які здійснюють адаптацію тексту, можуть автоматично пропонувати змінити структуру або формулювання для того, щоб відповідати стандартам певної публікації. Наприклад, алгоритми можуть підказати, як краще організувати вступ, методологію, результати дослідження або висновки, з огляду на стиль, характерний для певної галузі або конкретного видання.

Персоналізація наукових текстів також включає використання алгоритмів, які адаптують текст до потреб і рівня розуміння певної цільової аудиторії. Використання систем на основі обробки природної мови (NLP) дозволяє виявляти унікальні стилістичні особливості автора, а потім підлаштовувати текст під конкретні вимоги читачів або рецензентів. Це важливо, оскільки наукові тексти можуть бути орієнтовані на різні типи аудиторії — від фахівців до ширшої

громадськості. Алгоритми здатні пропонувати варіанти, які зберігають академічну точність, але водночас роблять текст більш доступним для менш спеціалізованих читачів, що може бути особливо корисно при написанні оглядів літератури або популяризації наукових ідей.

Один з основних аспектів алгоритмів адаптації наукових текстів — це покращення якості через персоналізовані рекомендації. Системи, які використовують машинне навчання, можуть вивчати попередній досвід авторів і адаптувати свої поради, враховуючи індивідуальні потреби та стилі письма. Це дозволяє науковцям отримувати персоналізовані підказки щодо структури тексту, вибору термінів або використання спеціалізованої лексики. Наприклад, програми можуть адаптувати рекомендації залежно від того, чи пише автор для міжнародної аудиторії, чи для вузькопрофільної наукової спільноти. Це дає можливість підвищити точність і відповідність вимогам, що забезпечує вищу ймовірність прийняття статті до публікації.

Адаптація стилю і мови є ще одним важливим елементом персоналізації наукових текстів. Стилістичні алгоритми можуть допомогти науковцям вибирати правильні формулювання, відповідно до наукового контексту, знижувати кількість зайвих складних конструкцій і, в той же час, допомагати уникати використання неформальних виразів або сленгу. Програми, такі як Grammarly, ProWritingAid або Hemingway Editor, дають можливість не лише виправляти граматичні помилки, а й аналізувати текст на наявність повторів, надмірно складних виразів або неправильного використання термінів. Алгоритми можуть також автоматично підказати, як покращити читабельність тексту для цільової аудиторії, зберігаючи при цьому його академічний стиль.

Інтеграція метаданих для адаптації та персоналізації є ще одним важливим етапом процесу. Метадані включають інформацію про контекст публікації, цитовані джерела, ключові слова та інші елементи, що визначають видимість і вплив роботи. Алгоритми можуть використовувати метадані для того, щоб оптимізувати структуру тексту та покращити його індексацію в наукових базах даних та пошукових системах. Вони можуть пропонувати рекомендації щодо

вдосконалення списку літератури, покращення пошукових запитів або використання актуальних термінів, що сприятиме кращому розповсюдженню та цитуванню роботи.

Машинне навчання і персоналізація рекомендацій допомагають вдосконалювати алгоритми з часом. Системи, які використовують машинне навчання, можуть навчатися на основі величезної кількості публікацій, вивчаючи специфічні стилі написання, частоту використання термінів, а також тенденції в певних галузях науки. Це дозволяє не тільки адаптувати текст до сучасних наукових стандартів, але й автоматично впроваджувати нові тренди в стилі написання та форматуванні, що актуальні для конкретної галузі. Наприклад, у деяких наукових напрямках можуть з’являтися нові терміни чи концепти, і алгоритми можуть автоматично виявляти ці зміни і пропонувати автору відповідні корективи в тексті.

Логічна адаптація та структурна персоналізація також важливі для того, щоб забезпечити цілісність та логічну послідовність тексту. Алгоритми, що аналізують структуру і логіку наукових статей, можуть допомогти авторам покращити викладення своїх аргументів, пропонуючи рекомендації щодо перегрупування абзаців чи додавання пояснень для кращого розуміння. Ці системи здатні також визначати проблеми, які можуть виникнути внаслідок неструктурованого викладу або відсутності чіткої логічної послідовності.

## 2.4 Висновки до розділу 2

Алгоритм автоматизації написання та аналізу наукових текстів поєднує різноманітні технології, що дозволяють зменшити час і зусилля, необхідні для створення високоякісних наукових публікацій. Ці інструменти не лише допомагають науковцям покращувати якість своїх робіт, а й забезпечують ефективну організацію дослідницького процесу, що сприяє розвитку науки.

Також алгоритм адаптації та персоналізації наукових текстів має на меті не лише покращення стилістичних та граматичних аспектів, але й оптимізацію самого процесу написання, враховуючи індивідуальні вимоги дослідника та специфіку наукової спільноти. Завдяки інтеграції сучасних технологій, таких як машинне навчання, обробка природної мови та метадані, цей процес стає все більш точним і ефективним, що дозволяє науковцям створювати високоякісні публікації, що відповідають актуальним вимогам наукових видань та цільової аудиторії.

# 3 ПРОГРАМА ГЕНЕРУВАННЯ НАУКОВО-ТЕХНІЧНИХ ТЕКСТІВ

## 3.1 Локальне середовище для генерування науково- технічних текстів

Для того щоб була змога створити науково-технічні статті необхідно використовувати технології які можуть створити навчальне середовище. В першу чергу щоб статті генерувались у локальні мережі потрібна база із якої буде відбуватись навчання. Також з цієї бази буде генеруватись текст який буде працювати по ключових словах. Щоб навчання було швидше необхідно вибрати тему, яка буде фігурувати для навчання локальної мережі. Так як інформації повинно бути багато відповідно потрібно використовувати різноманітні джерела, також різноманітну тематику, щоб інформативність згенерованої статті була також наповнена корисною інформацією. Створення даної мережі буде відбуватись на операційній системі Linux Ubuntu 23.04. Вибір Linux Ubuntu для створення локальної мережі та системи генерації науково-технічних текстів є логічним кроком через його численні переваги. Перш за все, Ubuntu відомий своєю стабільністю та високим рівнем безпеки, що є особливо важливим при роботі з конфіденційними науковими матеріалами. Постійні оновлення та багаторівневі механізми безпеки забезпечують надійність системи, а також захист даних та інструментів, зокрема тих, які використовуються в обробці природної мови. Окрім цього, Ubuntu пропонує значну підтримку для Python та його бібліотек, таких як `nltk`, `spacy` та інших, які широко застосовуються для текстового аналізу та генерації. Завдяки відкритій архітектурі, Ubuntu легко інтегрує ці бібліотеки та надає доступ до пакетів, що спрощує налаштування середовища для виконання складних обчислювальних задач у межах локальної мережі. Система також підтримує безліч інших інструментів обробки тексту та машинного навчання, що дозволяє зібрати універсальне середовище для створення та обробки наукових текстів. Велика та активна спільнота користувачів Ubuntu надає численні ресурси для підтримки та розв’язання проблем, що робить процес налаштування мережі простішим. Це важливо, адже

при роботі зі складними задачами, такими як обробка великих обсягів текстової інформації, швидкість і простота отримання допомоги значно підвищують ефективність роботи. Ubuntu також сумісний із популярними серверними технологіями та підтримує LAMP-стек, що дає можливість налаштувати внутрішній веб-інтерфейс для відображення результатів генерації текстів. Сумісність, стабільність, легкість у налаштуванні та широкі можливості роботи з науковими текстами роблять Ubuntu чудовим вибором для наукових досліджень, де потрібна надійна інфраструктура та гнучка система управління обчислювальними ресурсами [17, 18, 63].

В першу чергу потрібно створити робочу теку, в які буде і відбуватись увесь процес, зберігання даних, написання програмного коду. Такі методи сприяють організації та структурованому підходу до розробки. Коли всі файли зберігаються в одній папці, їх легше знайти та керувати ними, що мінімізує час на пошук окремих елементів і значно підвищує продуктивність. Зберігання проєктних файлів у єдиній папці також спрощує процеси резервного копіювання та передачі проєкту на інші машини або середовища, оскільки весь проєкт можна перемістити чи скопіювати як єдиний блок. Також потрібно створити робочі файли в яких і буде зберігатись інформація та опрацьовуватись данні. Для зручності буде використовуватись вбудований термінал системи. Він дозволяє швидко виконувати необхідні дії та встановлювати необхідне програмне забезпечення. Для того щоб почати проєкт необхідно створити теку. На рисунку 3.1 відображено створення теки, та створення файлу, який буде містити в собі базу статей які будуть використуватись для генерування текстів.

Після створення текстового файлу потрібно заповнити його. Алгоритм роботи буде відбуватись наступним чином: усі статті які будуть задіяні при генерації тексту будуть перебувати в одному місці. Статті будуть генеруватись українською мовою але за допомогою програмних засобів можна використовувати будь яку мову. На рисунку 3.2 відображається інформація щодо завантаження статей. Було обрано файл у форматі txt так як цей формат

підтримується всіма операційними системами і уже по замовчуванню вбудований у можливість відкриття та читання.

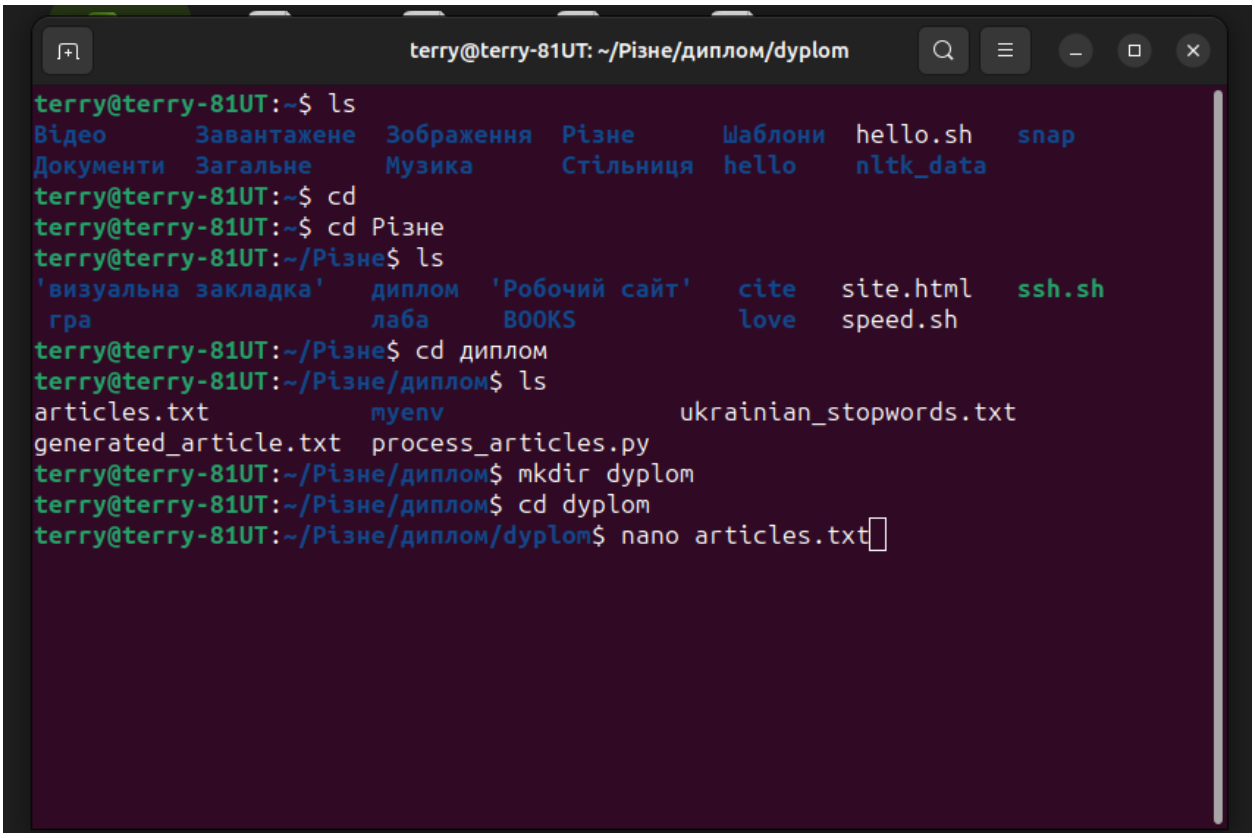


Рисунок 3.1 – Створення нової теки та створення текстового файлу в якому буде зберігатись вся інформація щодо статей

Наступними діями будуть обробка тексту щоб система не видавала помилок і також для читабельності. Відбувається попередня обробка текстів. Для того щоб не виникали помилок потрібно видалити лишні символи та непотрібні слова, які є не ключовими і не несуть ніякої інформації. Також необхідно очистити метадані та структурувати тексти. Для аналізу тексту та приведення його до уніфікованого вигляду під час створення наукових статей корисно застосовувати бібліотеки обробки природної мови, такі як NLTK та spaCy. Ці бібліотеки дозволяють виконувати різноманітні операції з текстами, такі як лематизація, видалення стоп-слів, токенізація тощо.

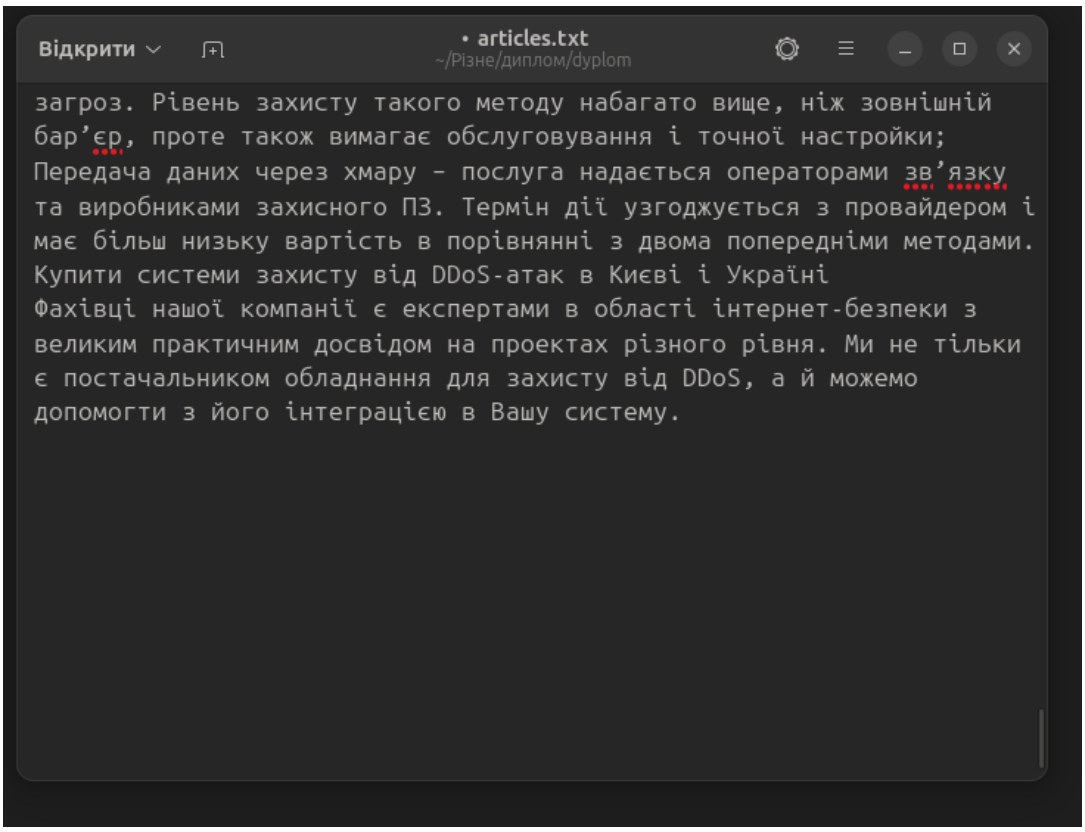


Рисунок 3.2 – Завантаження інформації у майбутню базу, за допомогою якої потім буде генеруватись текст

NLTK (Natural Language Toolkit) — це одна з найбільш популярних бібліотек для обробки природної мови в Python. Вона містить велику кількість інструментів для аналізу тексту, таких як токенізація (розбиття тексту на окремі слова або речення), видалення стоп-слів (зайвих слів на зразок "і", "на", "але", які не несуть смислового навантаження), лематизація (перетворення слів до базової форми), а також робота з корпусами текстів і алгоритмами класифікації. NLTK є корисною для попередньої обробки тексту, яка є необхідною перед проведенням аналізу чи генерацією нового тексту, що дозволяє структурувати вхідні дані, виділяти важливу інформацію та позбавлятись зайвих елементів [15, 16, 36].

spaCy — це інша потужна бібліотека для обробки тексту, яка також підтримує лематизацію, токенізацію і видалення стоп-слів, але надає додаткову функціональність для роботи з великими обсягами текстів. Вона орієнтована на більш швидку та ефективну обробку даних і містить готові до використання моделі для різних мов, що дозволяє значно спростити процес аналізу тексту.

spaCy активно використовується для складніших задач, таких як витягування іменованих сутностей (імен людей, географічних назв, компаній тощо) і частин мови, що особливо корисно в обробці наукових і технічних текстів, де потрібно точно аналізувати термінологію [15, 32].

Для того щоб підготувати текст для обробки в даних бібліотеках необхідно виконати в терміналі наступний код. Потрібно відкрити середовище python3 на терміналі та ввести наступний код. На рисунку 3.3 відображається обробка тексту, що міститься в тільки що створенному файлі articles.txt.

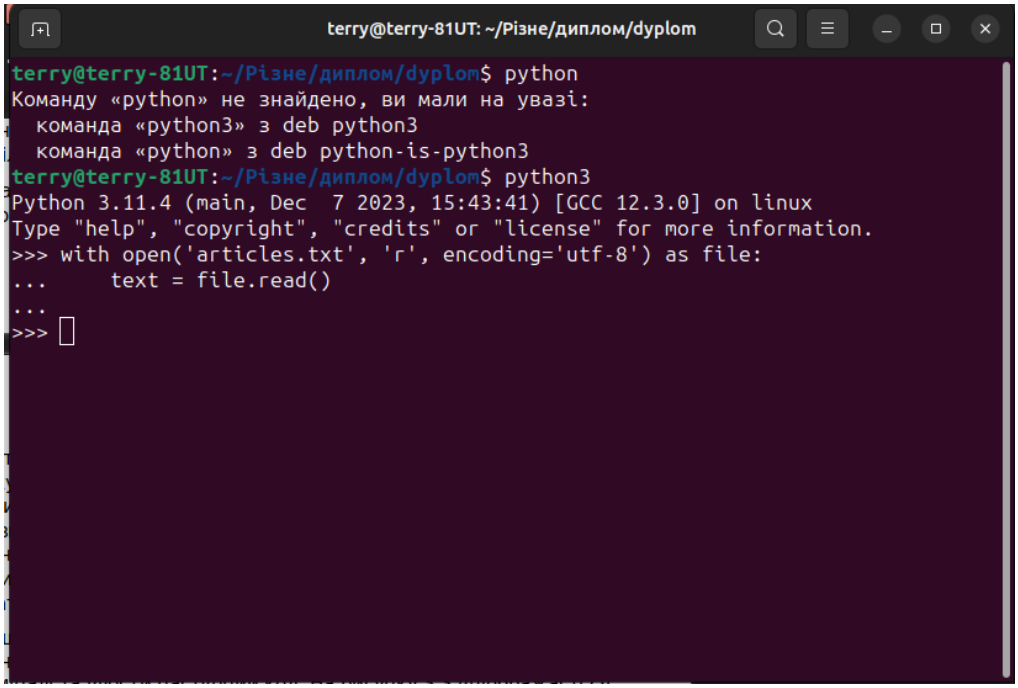


Рисунок 3.3 – Обробка тексту за допомогою python3

Цей код відкриває текстовий файл із назвою articles.txt в режимі читання ('r') з використанням кодування utf-8. Використання encoding='utf-8' гарантує коректне відображення тексту, особливо якщо він містить неанглійські символи, такі як українські літери. Конструкція with open(...) as file відкриває файл у так званому контекстному менеджері, що забезпечує автоматичне закриття файлу після завершення роботи з ним, навіть якщо станеться помилка під час виконання коду.

Після відкриття файлу за допомогою file.read() зчитується весь вміст файлу

і зберігається у змінній text, яка міститиме повний текст із articles.txt у вигляді одного рядка.

Наступними діями буде очищення тексту від зайвих символів та метаданих, таких як імена авторів, заголовки статей чи інші дані, які можуть не знадобитися для подальшого аналізу. Це допоможе спростити текст і виділити лише важливий зміст для аналізу чи генерації. Щоб видалити непотрібні елементи, можна використовувати регулярні вирази (regex) або бібліотеки для обробки тексту, які автоматично очищують текст від зайвих елементів, таких як HTML-теги чи спеціальні символи [37]. На рисунку 3.4 відображається процес очищення текст. Процес досить тривалий оскільки тексту є багато і потрібно зачекати деякий час щоб весь текст був проаналізований та усі непотрібні символи були видалені.

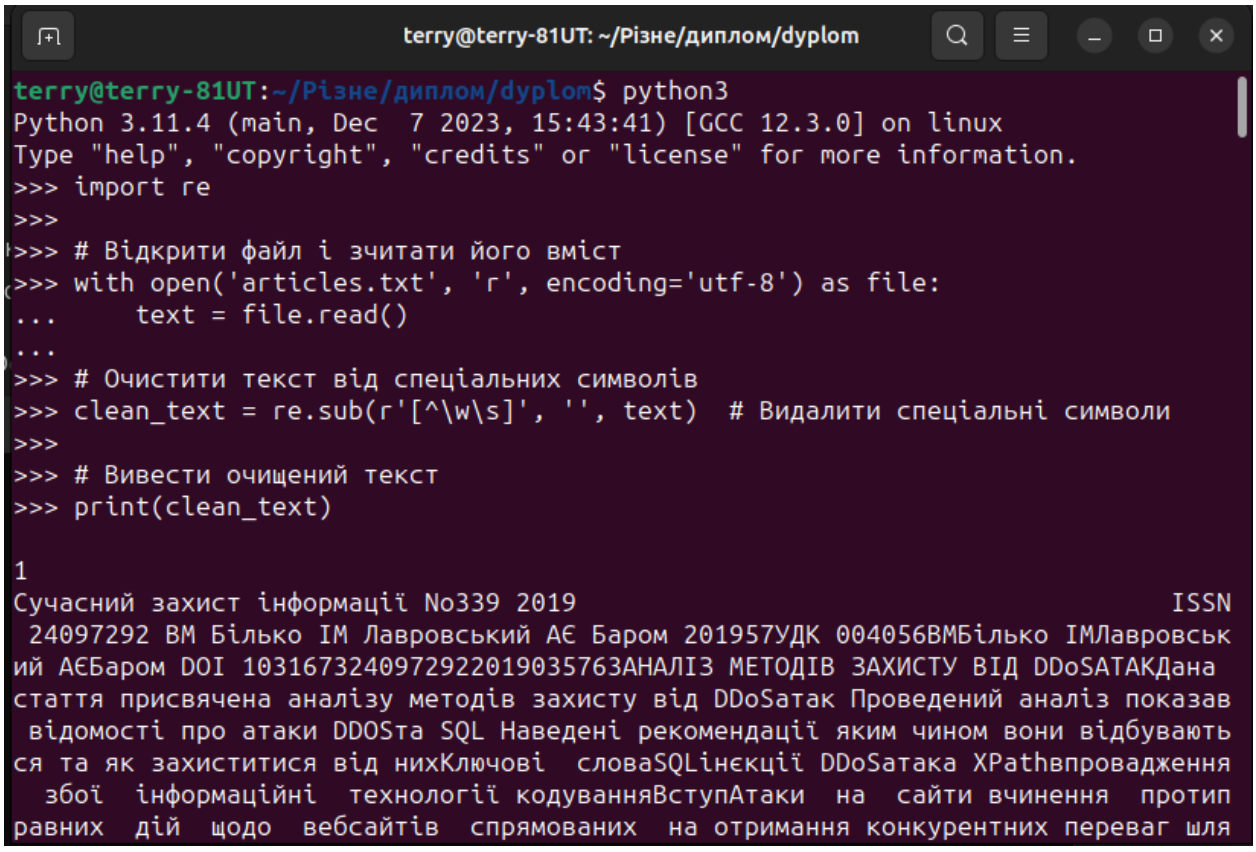


Рисунок 3.4 – Обробка тексту за допомогою мови python3

Цей код починається з того, що відкриває текстовий файл `articles.txt` у режимі читання, вказуючи кодування `utf-8`. Це кодування підтримує українські

символи, що важливо для роботи з текстами українською мовою. Файл відкривається за допомогою конструкції `with`, яка автоматично закриває його після завершення роботи з ним. Увесь вміст файлу зчитується і зберігається в змінній `text`. Це дозволяє подальшу обробку тексту без повторного доступу до файлу. Далі код очищує зчитаний текст від спеціальних символів, використовуючи регулярний вираз `re.sub(r'[^\w\s]', '', text)`. Вираз `r'[^\w\s]'` вибирає всі символи, які не є буквами або пробілами, і видаляє їх. Наприклад, такі символи, як пунктуація або спеціальні символи, будуть видалені з тексту, залишаючи лише слова та пробіли. Цей крок робить текст більш уніфікованим, що є корисним для подальшого аналізу, особливо в задачах з обробки природної мови. Кінцевим буде  очищений текст зберігається в змінній `clean_text` і виводиться на екран за допомогою команди `print(clean_text)`. Це дозволяє переглянути результат очищення тексту і переконатися, що видалення зайвих символів пройшло успішно.

Текст готовий, для подальшої обробки тексту необхідно використати бібліотеки які, будуть розуміти змістову нагрузку тексту, фільтрувати його та використовувати наявну оброблену базу, щоб виводити інформацію. Необхідно завантажити бібліотеку nltk.

spaCy і NLTK — це потужні інструменти для обробки природної мови (NLP), які дозволяють ефективно працювати з текстовими даними. spaCy є сучасною бібліотекою, яка пропонує швидкі й точні рішення для аналізу тексту. Вона розроблена для масштабованих проектів, підтримує попередньо навчені моделі для багатьох мов і дозволяє виконувати задачі, такі як токенізація, лематизація, визначення частин мови, розпізнавання іменованих сутностей. Однією з ключових переваг spaCy є її швидкість та інтуїтивний інтерфейс, що робить її ідеальною для роботи з великими обсягами тексту.

З іншого боку, NLTK (Natural Language Toolkit) відома своєю багатофункціональністю та великою кількістю навчальних ресурсів, таких як готові корпуси текстів і бази даних слів. Вона широко використовується для навчальних цілей і досліджень, оскільки включає інструменти для токенізації,

синтаксичного аналізу, видалення стоп-слів і роботи з граматикою. Хоча NLTK поступається spaCy у швидкості й сучасності, її гнучкість і доступність роблять її популярною серед дослідників і початківців. На рисунку 3.8 відображається введення команди, за допомогою якої можна завантажити бібліотеку spaCy і NLTK. Так як ці бібліотеки уже завантажені то відповідно повідомлення буде відповідне. Також перед тим як встановлювати ці бібліотеки потрібно перевірити чи встановлений pip – це менеджер пакетів для Python, який дозволяє встановлювати додаткові бібліотеки. На рисунку 3.5 відображається встановлення або шлях до наявної версії застосунку, який дозволяє завантажувати нові бібліотеки на мові python.

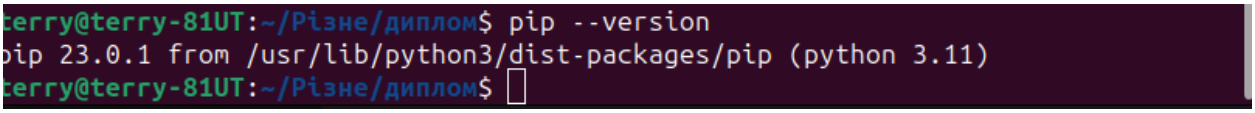


Рисунок 3.5 – Шлях до застосунку, який дозволяє завантажити усі бібліотеки, та додатки до python

Якщо не встановити ці елементи не буде змоги завантажити бібліотеки до python. На рисунку 3.7 відображається помилка, яка не дозволяє завантажити додатки. Відповідно термінал сам повідомляє що необхідно завантажити певні програмні засоби які дозволяють продовжувати створення локальної мережі. На рисунку можна спостерігати поради та розгорнуту відповідь та рішення яке може допомогти вирішити проблему із завантаженням.

Після того як усі програмні засоби завантажені, необхідно завантажити ще віртуальне середовище. Оскільки ми працюємо з базою даних відповідно потрібно середовище яку буде власне і опрацьовувати всі данні з бази і витягувати їх щоб складати текст. В мові програмування python це середовище має назву myenv. myenv — це назва віртуального середовища, яке зазвичай створюють для організації Python-проектів. Віртуальне середовище дозволяє ізолювати залежності проекту, щоб уникнути конфліктів між бібліотеками, які використовуються в різних проектах. Наприклад, якщо один проект потребує

однієї версії певної бібліотеки, а інший проект — іншої, використання віртуальних середовищ допомагає зберігати ці конфігурації незалежно одна від одної. Назва myenv не є специфічною або обов'язковою — її можна змінювати на будь-яку іншу, наприклад, project_env або nlp_project.

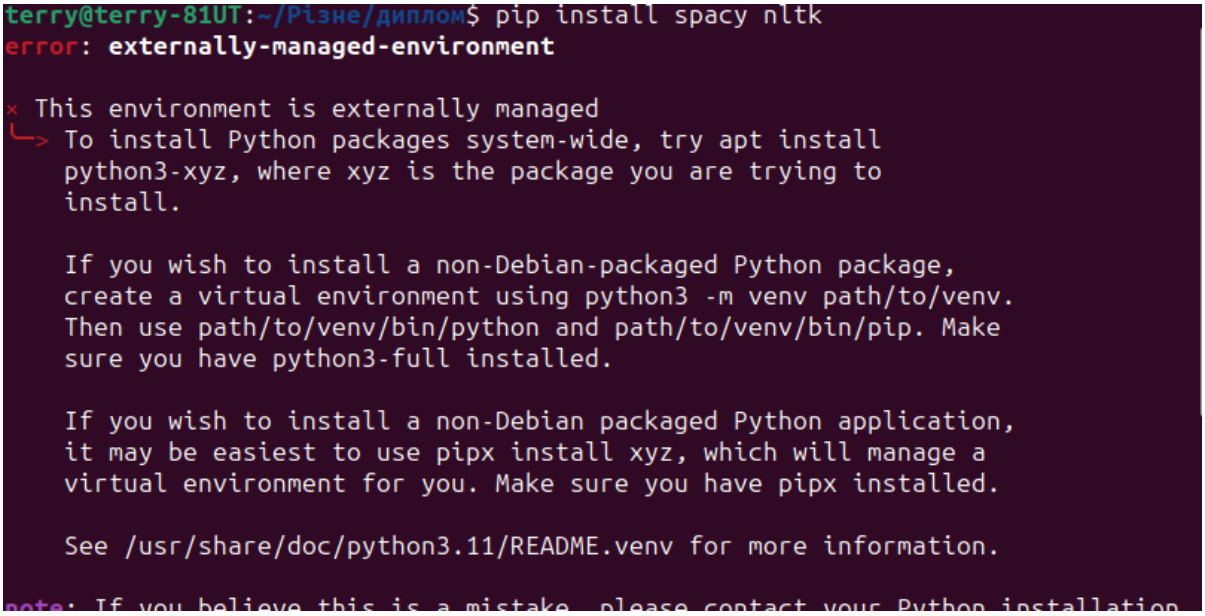


Рисунок 3.6 – Відображення проблеми із завантаження бібліотеки nltk із помилками

Середовище створюється за допомогою таких інструментів, як venv або virtualenv. Створене середовище містить власний ізольований інтерпретатор Python і каталог для встановлених бібліотек, завдяки чому ваш проект стає більш автономним і керованим. На рисунку 3.8 відображається встановлення бібліотек та середовища яке допоможе створити проєкт. Активація середовища дозволяє працювати з бібліотеками й залежностями, які будуть встановлені саме для цього проекту. Це забезпечує чистоту системного Python і зменшує ризики виникнення помилок через несумісність бібліотек.

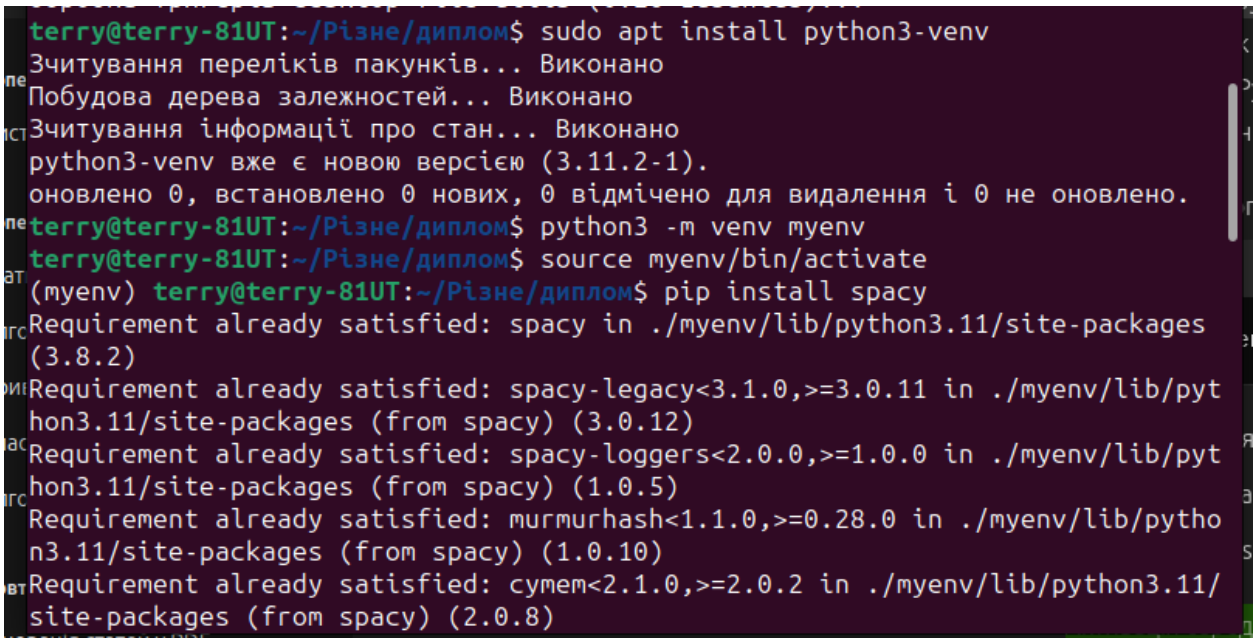


Рисунок 3.7 – Створення віртуального середовища

Ще одним дуже важливим аспектом генерації тексту заключається мова. Не всі програмні засоби можуть розуміти один і теж текст на різних мовах [39]. Для того щоб наявна база статей була зрозуміла для читання потрібно додатково встановити модель, яка буде розуміти символі українського походження. Це встановлюється однією командою і на рисунку 3.9 відображається процес встановлення пакунків та завантаження самої моделі.

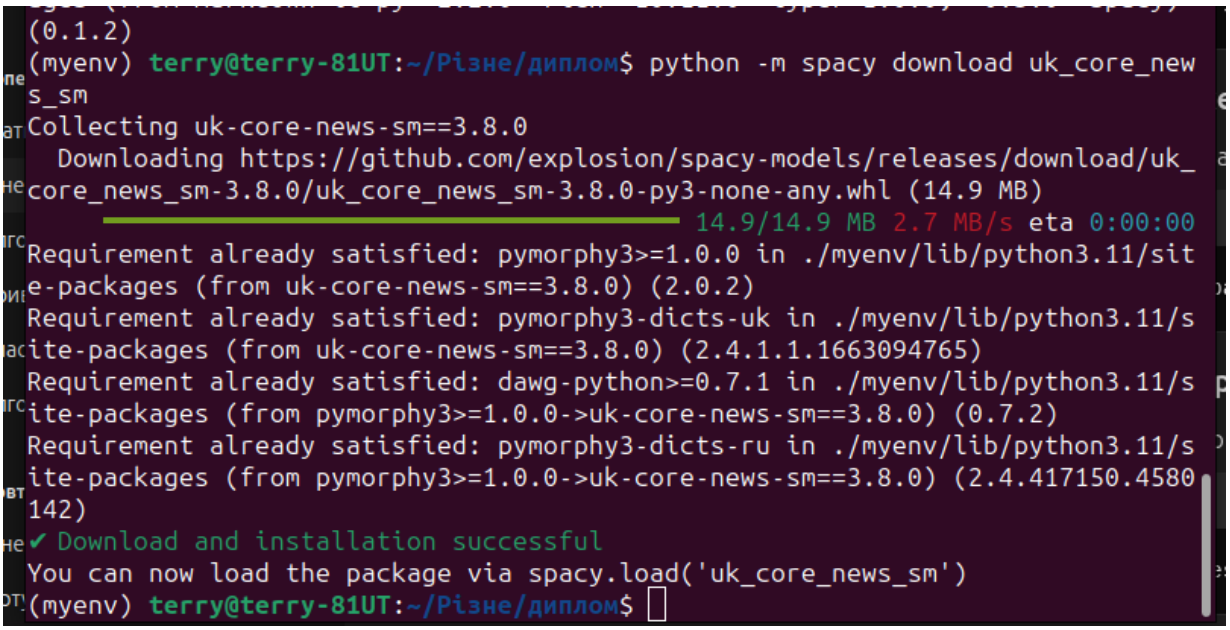


Рисунок 3.8 – Встановлення мовної моделі

Практично усе готово щоб створювати проект який буде генерувати текст але потрібно впевнитись, чи всі програмні засоби встановлені для того щоб створювати програму, яка буде і логічною складовою всього проєкту. Командою pip list можна зрозуміти які пакунки встановлені чи точно немає помилок, які будуть свідчити про те що програма встановилась некоректно. На рисунку 3.10 відображається список програмних засобів, які будуть відображати весь функціонал віртуального середовища, та його складових. Також можна спостерігати версію яка була встановлена. В подальших діях якщо ще раз ввести усі команди які завантажують бібліотеки чи програмні засоби то вони автоматично їх оновлюють, тобто якщо потрібно оновити лише один елемент то достатньо наново ввести команду для завантаження, а система уже автоматично довантажить все що потрібно.

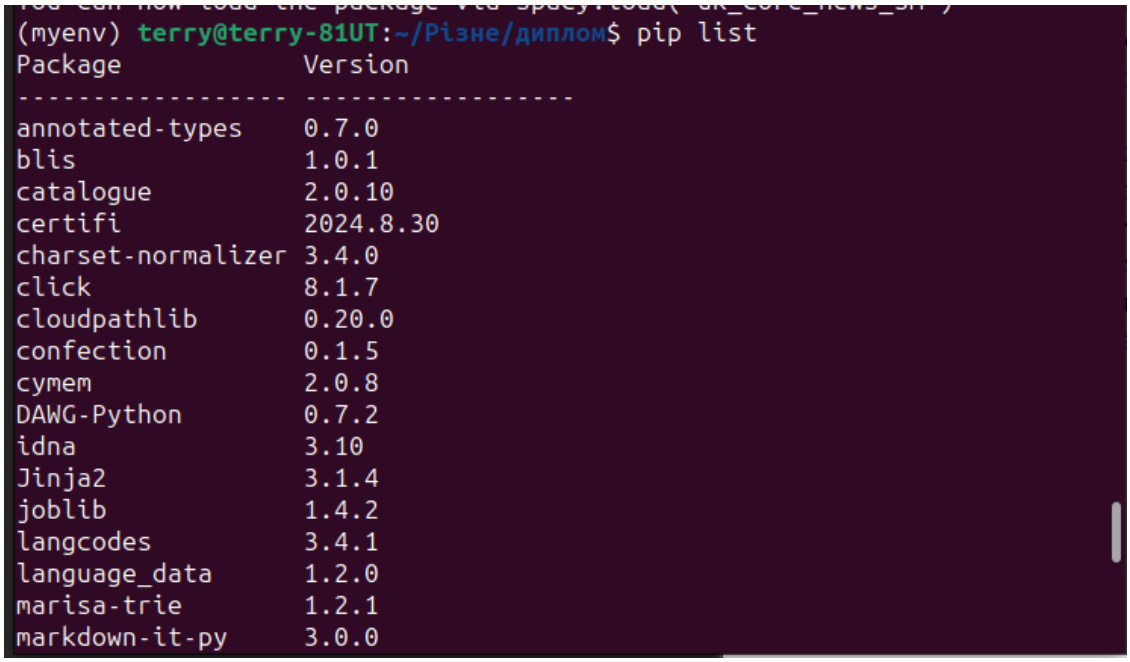


Рисунок 3.9 – Відображення усіх програмних засобів для коректної роботи віртуального середовища

Згідно даних можна зрозуміти що усе працює і можна приступати уже до написання коду, програмний код повинен також бути на мові програмування python щоб завантажені бібліотеки могли бути використані.

## 3.2 Система генерування наукових текстів

Розробка програмного забезпечення є основно проєкту який буде генерувати статті. Оскільки необхідні встановлені бібліотеки мають необхідний матеріал який допоможе опрацьовувати текст і відповідно виводити необхідну інформацію. Яка би велика база даних не була, якщо немає необхідного механізму, який буде розуміти що із цією базою буде робити то вона буде просто непотрібною. За допомогою програмного забезпечення користувач може користуватись цією програмою не прибігаючи до необхідності відкривати автоматично віртуальні середовища та автоматично запускати файли. Так як це буде не зручно і відповідно потребує багато знань. В першу чергу потрібно орієнтуватись на простоту в користуванні та зручність.

Першим кроком для запуску програми необхідно створити файл із стоп словами, які будуть видалятись із наукових текстів. Будь яка стаття яка має науковий нахил повинна бути наповнена лише сухими фактами, де немає місця словам: я, моя і тд. Однією проблемою, яка має бібліотека nltk це відсутність визначати ці слова. Вона не зможе їх видалити і при генеруванні цього тексту буде видаватись художнє творіння. У таблиці 3.1 відображається список стоп слів, які не несуть великого смислового навантаження в тексті. Також видалення цих слів допоможе спростити аналіз тексту для програмного коду.

Таблиця 3.1 – Стоп слова

| Стоп слова |
|---|
| в |
| На |
| Що |
| Як |
| З |
| Ми |
| ви |

Цей набір стоп-слів можна легко доповнити чи змінити залежно від ваших потреб у текстовому аналізі. Стоп-слова зазвичай використовуються для того, щоб усунути з тексту слова, які не несуть значущої інформації або не допомагають у виконанні конкретних завдань, таких як класифікація або пошук інформації. Це дозволяє зосередитись на важливих частинах тексту, таких як іменники, дієслова та інші значущі терміни. Якщо потрібно більше слів для аналізу, можна завантажити список готових українських стоп-слів із відкритих джерел [35, 51, 54]. Наприклад, існують публічні бази даних та ресурси, де можна знайти готові списки, що регулярно оновлюються. Такі списки можуть включати не лише звичайні стоп-слова, а й спеціалізовані терміни, які допоможуть налаштувати обробку текстів для конкретних задач.

Після створення файлу необхідно в програмному коді розписати логіку, яка буде завантажувати стоп слова з файлу.

```
# Завантаження стоп-слів з файлу
def load_stopwords(filepath):
with open(filepath, 'r', encoding='utf-8') as f:
return set(line.strip() for line in f)
stop_words = load_stopwords('ukrainian_stopwords.txt')
```

Ця функція відкриває файл, що містить стоп-слова (слова, які не несуть великої інформації в контексті текстового аналізу), і створює набір цих слів. Кожен рядок з файлу перетворюється на окреме слово, яке буде використовуватись для фільтрації під час обробки статей. Програмний код, який уже здійснив фільтрацію слів буде за допомогою функції що зчитує файл з статтями та кожну статтю обробляє за допомогою бібліотеки **spaCy**. Стаття розбивається на токени, фільтруються стоп-слова та знаки пунктуації, і результат додається до списку оброблених статей. Таким чином, кожна стаття очищується від непотрібної інформації. Наступна функція описує процес, який буде опрацьовувати текст.

```
def process_articles(file_path, stop_words):
    with open(file_path, 'r', encoding='utf-8') as file:
```

```
        articles = file.readlines()
    processed_articles = []
    for article in articles:
        doc = nlp(article.strip())
        filtered_tokens = [token.text for token in doc if token.text
not in stop_words and not token.is_punct]
        processed_articles.append(' '.join(filtered_tokens))
    return processed_articles
```

Функція process_articles виконує кілька важливих кроків для обробки текстових статей: завантаження статей з файлу, токенізація тексту за допомогою **spaCy**, фільтрація стоп-слів і знаків пунктуації. Ось детальний розбір кожної частини цієї функції. На початку функції відкривається файл, вказаний через параметр file_path, за допомогою контекстного менеджера with. Це забезпечує автоматичне закриття файлу після завершення роботи з ним. Кожен рядок файлу (кожна стаття) зчитується за допомогою методу readlines() і зберігається в список articles. Таким чином, кожна стаття буде окремим елементом списку. Далі створюється порожній список processed_articles, в який будуть додаватися оброблені статті. Кожна стаття з articles обробляється в циклі for. Спочатку, перед обробкою, із статті видаляються зайві пробіли з обох сторін за допомогою методу strip(). Використовуються потужні можливості бібліотеки spaCy для токенізації тексту. Текст статті передається в nlp() — об'єкт, який відповідає за лексичний аналіз та токенізацію. spaCy розбиває текст на окремі одиниці, які називаються токенами (наприклад, слова, цифри, знаки пунктуації). Опис функції process_articles. Функція process_articles виконує кілька важливих кроків для обробки текстових статей: завантаження статей з файлу, токенізація тексту за допомогою spaCy, фільтрація стоп-слів і знаків пунктуації. Ось детальний розбір кожної частини цієї функції.

```
python

with open(file_path, 'r', encoding='utf-8') as file:
    articles = file.readlines()
```

На початку функції відкривається файл, вказаний через параметр file_path,

за допомогою контекстного менеджера with. Це забезпечує автоматичне закриття файлу після завершення роботи з ним. Кожен рядок файлу (кожна стаття) зчитується за допомогою методу readlines() і зберігається в список articles. Таким чином, кожна стаття буде окремим елементом списку.

```
processed_articles = []
for article in articles:
    doc = nlp(article.strip())
```

Далі створюється порожній список processed_articles, в який будуть додаватися оброблені статті. Кожна стаття з articles обробляється в циклі for. Спочатку, перед обробкою, із статті видаляються зайві пробіли з обох сторін за допомогою методу strip(). Використовуються потужні можливості бібліотеки **spaCy** для токенізації тексту. Текст статті передається в nlp() — об'єкт, який відповідає за лексичний аналіз та токенізацію. **spaCy** розбиває текст на окремі одиниці, які називаються **токенами** (наприклад, слова, цифри, знаки пунктуації). Наступним кроком виконується фільтрація токенів. Створюється список filtered_tokens, в який додаються лише ті токени, що не є стоп-словами та знаками пунктуації.

- token.text — це текстовий рядок, що представляє токен (слово чи знак).
- stop_words — це набір стоп-слів, які були завантажені раніше у функції load_stopwords().
- token.is_punct перевіряє, чи є токен знаком пунктуації. Якщо токен є пунктуацією, він не додається в список.

Така фільтрація дозволяє отримати чистий текст, який складається лише з важливих для аналізу слів. Після фільтрації токенів, вони з'єднуються в одну строку за допомогою методу ' '.join(), де кожен токен відокремлюється пробілом. Очищена та відфільтрована стаття додається до списку processed_articles. Після того як всі статті пройшли через обробку, функція повертає список processed_articles, що містить очищені та токенізовані версії кожної статті.

Обробка тексту насправді може досить довго відбуватись, оскільки база

може бути велика то в залежності від пристроїв час може відрізнятись. Також для того щоб генерувати текст необхідні ключові слова, по яких і буде відбуватись генерація. Наступний програмний код через функцію буде реалізовувати це все . Вона відбирає статті, що містять хоча б одне з введених ключових слів, і випадковим чином вибирає з них кілька для створення згенерованої статті. Якщо відповідних статей немає, програма виводить повідомлення про помилку.

```
def generate_article(processed_articles, keywords):
    filtered_articles = [article for article in processed_articles
if any(keyword in article for keyword in keywords)]
    if not filtered_articles:
        return "Немає статей, що відповідають заданим ключовим
словам."
        selected_articles = random.sample(filtered_articles, min(3,
len(filtered_articles)))
    return ' '.join(selected_articles)
```

Функція generate_article створена для генерації нового тексту на основі заданих ключових слів і оброблених статей. Вона працює таким чином, щоб зібрати тексти, що відповідають запитам користувача, і створити з них новий згенерований документ. Ось докладний розбір кожного етапу цієї функції. На початку функція отримує список processed_articles (оброблених раніше статей) та keywords — список ключових слів, які користувач хоче знайти в текстах. Функція фільтрує статті так, щоб залишити тільки ті, які містять хоча б одне з заданих ключових слів.

– any(keyword in article for keyword in keywords) перевіряє, чи є хоча б одне з ключових слів у статті. Якщо хоча б одне ключове слово є в статті, вона потрапляє до списку filtered_articles.

– Це дозволяє забезпечити, щоб тільки ті статті, які стосуються заданих тем, були відібрані для подальшого об'єднання.

Якщо після фільтрації не залишилось жодної статті, яка відповідала б хоча б одному з ключових слів, функція повертає повідомлення про те, що не знайдено жодної відповідної статті. Це забезпечує зручний зворотний зв'язок для

користувача, щоб він розумів, що його запит не знайшов відповідних результатів. Якщо є відфільтровані статті, наступним кроком є вибір випадкових статей. З допомогою функції random.sample() вибирається випадковий підмножина статей з відфільтрованого списку filtered_articles. Обрана кількість статей обмежена максимальним значенням 3 (це забезпечується за допомогою min(3, len(filtered_articles))), тобто, якщо статей менше трьох, буде вибрано стільки, скільки їх є.

Вибрані статті з'єднуються в один текстовий рядок за допомогою методу ' '.join(selected_articles). Це дозволяє створити єдиний згенерований текст з кількох статей, який містить найбільш релевантні частини на основі ключових слів.

За допомогою наступного коду можна зберігати файли. Інформація, яка згенерована локальною системою повинна зберігатись у файлі, який потім можна відкрити.

```
def save_article_to_file(generated_article, filename):
    with open(filename, 'w', encoding='utf-8') as file:
        file.write(generated_article)
```

Функція save_article_to_file призначена для збереження згенерованої статті у текстовий файл. Вона працює просто, але виконує важливу роль, дозволяючи користувачеві зберегти результат своєї роботи для подальшого використання. Давайте розглянемо її роботу покроково.

Функція приймає два аргументи:

– generated_article: текст, який потрібно зберегти. Це рядок, зазвичай сформований на основі ключових слів із попередніх кроків програми.

– filename: ім'я файлу, в який буде записаний текст. Користувач може вказати будь-яку назву, наприклад, "generated_article.txt".

Цей рядок відкриває файл для запису:

– filename вказує назву файлу, який потрібно створити або відкрити. Якщо файл із такою назвою вже існує, його вміст буде перезаписаний.

– 'w' означає, що файл відкривається в режимі запису (write).

– encoding='utf-8' гарантує, що текст буде записаний у форматі UTF-8, що важливо для роботи з текстами українською мовою або будь-якими іншими мовами з нелатинськими символами.

Режим with забезпечує автоматичне закриття файлу після завершення операцій, навіть якщо в процесі виникає помилка. Це робить код більш надійним. Метод file.write() записує текст generated_article у відкритий файл. Якщо текст досить довгий, він записується повністю, без обмежень на довжину. Усе, що передається як аргумент, буде збережено в зазначений файл. Основна частина коду, яка опрацьовує весь текст та підбір інформації побудована на умовних операторах.

```
if __name__ == "__main__":
    nlp = spacy.load('uk_core_news_sm')
    stop_words = load_stopwords('ukrainian_stopwords.txt')
    file_path = 'articles.txt'
    processed_articles = process_articles(file_path, stop_words)
    keywords = input("Введіть ключові слова через кому:
").strip().split(',')
    generated_article = generate_article(processed_articles,
keywords)
   print("\nЗгенерована стаття:\n")
    print(generated_article)
    save_article_to_file(generated_article,
'generated_article.txt')
    print("\nЗгенерована стаття збережена у
'generated_article.txt'.")
    display_results(keywords, generated_article)
```

Цей код є ключовим фрагментом програми, який забезпечує виконання всіх етапів обробки статей, їхнього аналізу, генерації тексту на основі ключових слів, а також збереження й відображення результату. Умовний оператор дозволяє визначити, чи виконується скрипт напряму чи імпортований як модуль у інший код. У нашому випадку, якщо файл є головним, виконується весь код у цьому блоці. Це дозволяє безпечно розділяти функціонал, уникаючи автоматичного виконання при імпорті. nlp = spacy.load ініціалізує модель **spaCy** для обробки природної мови, яка спеціально розроблена для української мови.

Модель uk_core_news_sm містить інструменти для розпізнавання текстових структур, таких як токени, частини мови та інше. Вона є базовим ресурсом для роботи з текстом, надаючи можливість виконувати складний аналіз. Функція load_stopwords завантажує перелік стоп-слів, які зберігаються у текстовому файлі ukrainian_stopwords.txt. Ці слова (наприклад, "і", "та", "в") не несуть змістового навантаження, тому їх необхідно виключити з тексту для подальшого аналізу. Збереження стоп-слів у вигляді множини забезпечує швидкий пошук і фільтрацію. file_path = 'articles.txt' задає шлях до файлу articles.txt, який містить статті, кожна з яких записана окремим рядком. Програма використовує цей файл як джерело даних для подальшої обробки. Функція process_articles відкриває файл і обробляє кожну статтю:

– Зчитує рядки.

– Видаляє стоп-слова та пунктуацію.

– Зберігає очищені статті у список. Цей етап є важливим для підготовки тексту до пошуку ключових слів і генерації.

Користувач вводить ключові слова, які використовуються для пошуку релевантних статей. Метод strip() видаляє пробіли з початку й кінця введення, а split(',') розбиває рядок на список окремих слів. Це забезпечує точну обробку введення користувача.

Функція generate_article виконує два основних завдання:

– Вибирає статті, які містять хоча б одне ключове слово.

– Випадково об'єднує до трьох статей із результатів у єдиний текст. Це дозволяє створити новий зміст на основі найбільш релевантних матеріалів.

Згенерована стаття відображається в консолі для перегляду. Цей етап важливий для швидкої оцінки результату користувачем. Функція save_article_to_file записує згенерований текст у файл generated_article.txt. Це забезпечує збереження результату для подальшого використання або редагування. Користувач отримує повідомлення, що текст успішно записаний у файл. Це підвищує зручність і зрозумілість роботи програми. Функція display_results створює графічний інтерфейс за допомогою tkinter, де

відображаються ключові слова та згенерована стаття. Це робить результат більш зручним для читання та презентації.

Практично логічна частина програмного коду готова, якщо спробувати запустити код можна отримати уже результат згенерованої статті. Відповідно весь функціонал без графічної складової буде відбуватись в терміналі. Що не є дуже зручним для звичайного користувача. На Рисунку 3.2.1 приклад роботи коду, без графічного інтерфейсу. Також потрібно зауважити що завантаження тексту і функціоналу буде займати декілька секунд, так як база тексту є велика.

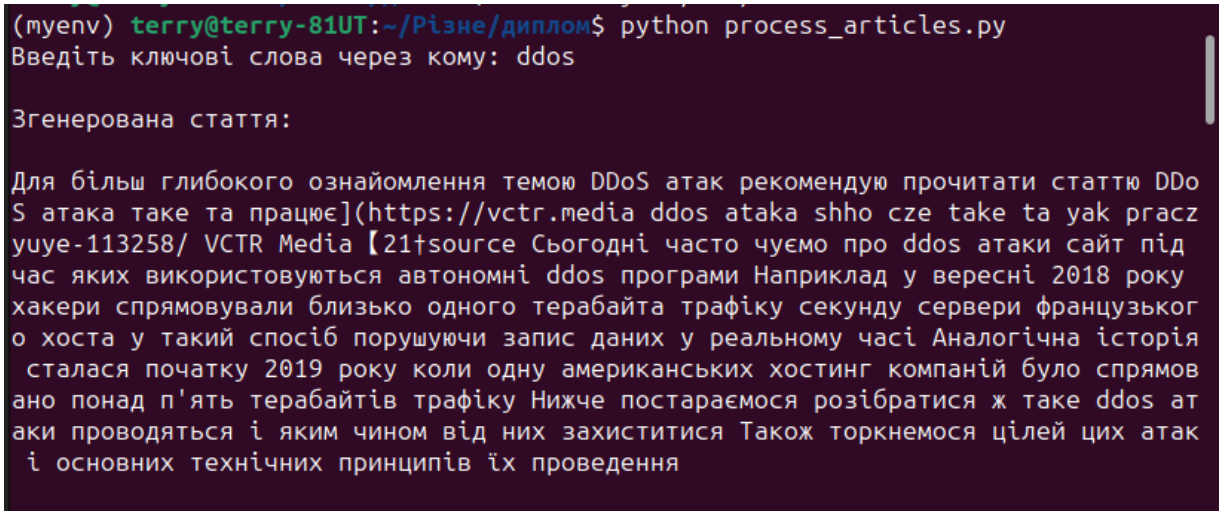


Рисунок 3.10 – Приклад генерації тексту без графічного інтерфейсу

Як можна спостерігати, генерація тексту відбулась за допомогою одного ключового слова ddos. Також можна спостерігати що генерація тексту відбулась успішно. Щоб згенерувати більш складнішу статтю, необхідно додати ще декілька ключових слів, які будуть також складовими теми статті. На рисунку 3.11 — відображення генерації тексти із декількома ключовими словами. На перших етапах тексту може бути небагато так як потрібно декілька разів генерувати текст щоб програма могла уже швидше опрацьовувати відомий їй текст.

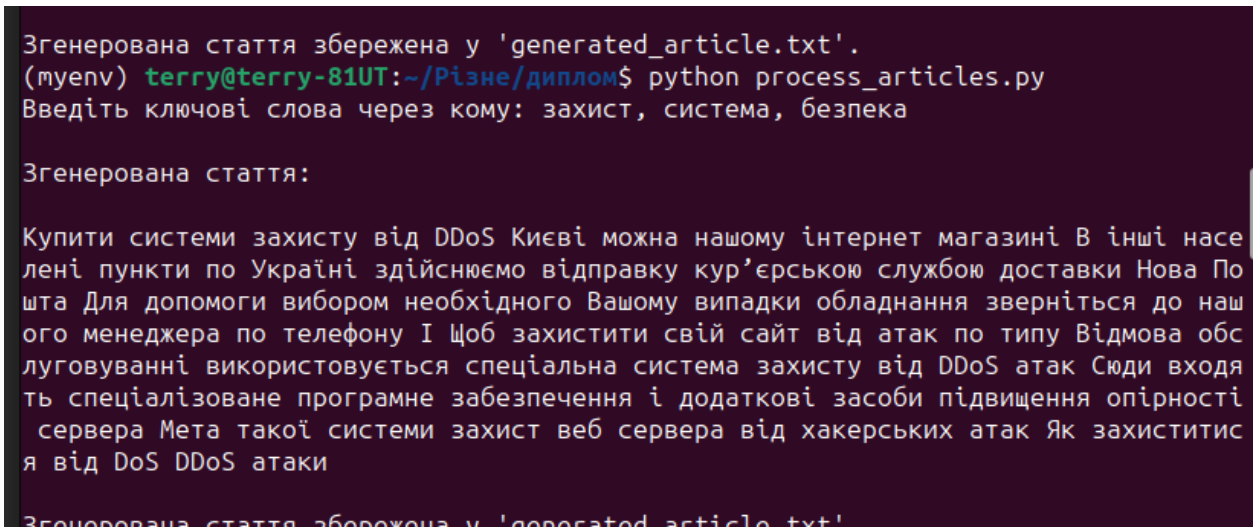


Рисунок 3.11 – Генерація тексту із декількома ключовими словами

Згідно даних, які відображаються на сторінці можна зрозуміти що програма працює коректно. Текст, який система згенерувала має свою логічну послідовність та має змогу відтворювати запити, які вимагає користувач. На даний момент система буде генерувати не великі тексти оскільки потрібен час багато спроб, також правильна підбірка ключових слів щоб система працювала і видавала текст великого розміру. База яка використовується є невеликою і налічує в собі близько 20 статей. Якщо взяти набагато більшу базу на конкретну тематику то інформації також буде в рази більше. На рисунку 3.2.3 можна спостерігати генерацію тексту коли ввести декілька разів один і теж текст. Якщо згенерувати один раз текст по ключових словах, то в теорії він має бути не змінним. словах.

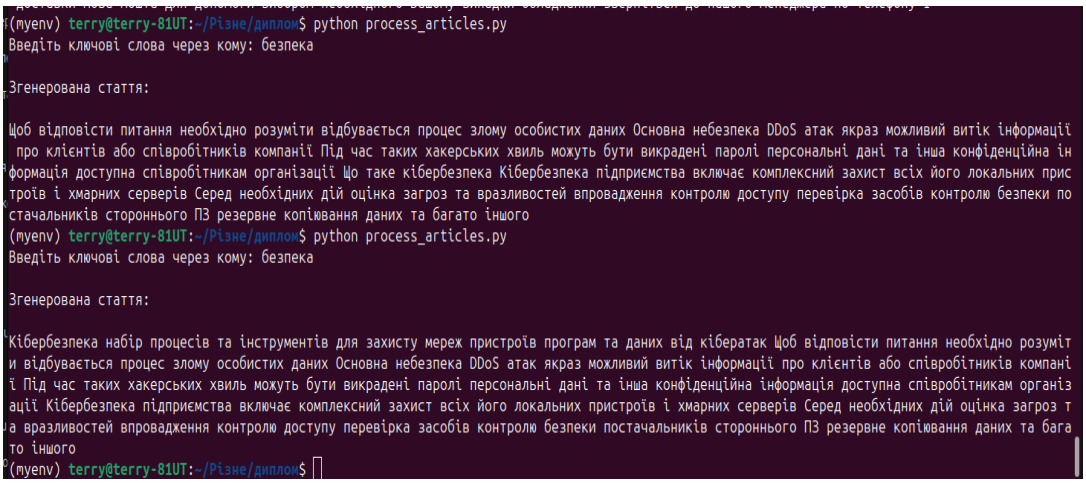


Рисунок 3.12 – Генерація тексту декілька разів по одних і тих самих ключових

Генерація тексту також відбулась успішною і можна використовувати, як матеріал, який буде генерувати матеріали в локальних мережах. Звичайно з невеликою базою можна добитись невеликих результатів але якщо збільшувати її все більше то система буде мати матеріал, який опрацьовувати і відповідно буде мати можливість аналізувати все більше і більше наукового тексту.

## 3.3 Інтерфейс програмної системи та комп’ютерні експерименти

Графічна частина коду реалізована за допомогою бібліотеки Tkinter — стандартного інструменту Python для створення GUI (графічних інтерфейсів). Кожен елемент у програмі виконує певну функцію. Розглянемо всі елементи та функції максимально детально. Програма починається зі створення головного вікна за допомогою функції tk.Tk(). Це основа графічного інтерфейсу, в якому будуть розміщені всі інші елементи. Заголовок вікна встановлюється методом title() — у нашому випадку це "Генератор статей". Розміри вікна задаються через метод geometry() з параметрами "700x650", що означає ширину 700 пікселів і висоту 650. Метод configure() встановлює світло-сірий фон для вікна, що створює сучасний вигляд програми.

```
root = tk.Tk()
root.title("Генератор статей")
root.geometry("700x650")
root.configure(bg="#f5f5f5")
```

Заголовок створюється за допомогою текстового віджета Label, який розташовується у верхній частині вікна. Він виконує роль візуального акценту. Текст стилізується шрифтом "Helvetica" розміром 20, з жирним накресленням. Для фону використовується зелений колір #4CAF50, а текст стає білим, що створює контраст.

header_label = tk.Label(root, text="Генератор статей", font=("Helvetica", 20, "bold"), bg="#4CAF50", fg="white", padx=10, pady=10) header_label.pack(fill=tk.X) Використання методу pack() з параметром fill=tk.X дозволяє заголовку займати всю ширину вікна, додаючи програмі професійний вигляд. Для організації елементів у вікні використовується Frame. Це контейнер, який дозволяє згрупувати віджети разом. У ньому розміщуються текстова мітка для підказки та текстове поле для вводу ключових слів.

```
frame = tk.Frame(root, bg="#f5f5f5")
frame.pack(pady=10)
keyword_label = tk.Label(frame, text="Ключові слова:",
font=("Helvetica", 14), bg="#f5f5f5")
keyword_label.grid(row=0, column=0, padx=5, pady=5)
keyword_entry = tk.Entry(frame, font=("Helvetica", 14), width=40)
keyword_entry.grid(row=0, column=1, padx=5, pady=5)
```

Мітка Label відображає текст "Ключові слова:", а поле Entry дозволяє користувачам вводити текст. Елементи розміщуються за допомогою методу grid(), який дозволяє організувати їх у вигляді таблиці. Кнопка для запуску генерації тексту створюється за допомогою віджета Button. Вона розміщується нижче поля вводу і стилізується зеленим кольором із білим текстом. При натисканні кнопки викликається функція generate_article_action.

```
generate_button = tk.Button(root, text="Згенерувати статтю",
font=("Helvetica", 14), bg="#4CAF50", fg="white",
command=generate_article_action)
generate_button.pack(pady=10)
```

Метод pack() додає вертикальний відступ у 10 пікселів, щоб кнопка не притискалася до інших елементів. Для відображення згенерованого тексту використовується віджет Text. Він дозволяє користувачам прокручувати та читати текст, навіть якщо його обсяг перевищує розмір вікна. Віджет розташовується у рамці Frame, яка також містить смугу прокрутки.

```
text_frame = tk.Frame(root)
```

```
text_frame.pack(pady=10, fill=tk.BOTH, expand=True)
scrollbar = tk.Scrollbar(text_frame)
scrollbar.pack(side=tk.RIGHT, fill=tk.Y)
text_area = tk.Text(text_frame, wrap=tk.WORD, font=("Helvetica",
12), yscrollcommand=scrollbar.set)
text_area.pack(fill=tk.BOTH, expand=True)
scrollbar.config(command=text_area.yview)
```

Смуга прокрутки (Scrollbar) синхронізується з текстовим віджетом за допомогою параметра yscrollcommand. Це забезпечує можливість прокручування вмісту текстового поля. У нижній частині вікна розміщуються кнопки, які дозволяють змінювати розмір тексту. Це реалізується через функції increase_font та decrease_font, які змінюють стиль тексту.

```
button_frame = tk.Frame(root, bg="#f5f5f5")
button_frame.pack(pady=10)
increase_button = tk.Button(button_frame, text="A+",
font=("Helvetica", 14), command=increase_font)
increase_button.pack(side=tk.LEFT, padx=5)
decrease_button = tk.Button(button_frame, text="A-",
font=("Helvetica", 14), command=decrease_font)
decrease_button.pack(side=tk.LEFT, padx=5)
```

Кнопки розміщуються горизонтально, використовуючи метод pack() із параметром side=tk.LEFT. Відступи між кнопками додаються через padx=5. Функції increase_font та decrease_font змінюють розмір шрифту текстового поля. Вони отримують поточний шрифт і збільшують чи зменшують його розмір.

```
def increase_font():
    current_font = text_area["font"].split()
    new_size = int(current_font[1]) + 2
    text_area.config(font=(current_font[0], new_size))
def decrease_font():
    current_font = text_area["font"].split()
    new_size = int(current_font[1]) - 2
    text_area.config(font=(current_font[0], new_size))
```

Ці функції забезпечують гнучкість у налаштуванні відображення тексту. Ця програма призначена для стабільної роботи в інтерактивному режимі, надаючи користувачам можливість генерувати текст на основі введених

ключових слів. Вона побудована з урахуванням зручності користувача, забезпечуючи простоту введення даних та зручне відображення результатів у графічному вікні. Завдяки використанню надійного інструментарію `Tkinter` та перевірених алгоритмів обробки тексту, програма повинна працювати стабільно навіть при великих обсягах даних. Крім того, функції масштабування тексту та прокрутки забезпечують гнучкість у роботі, що робить її придатною для різноманітних завдань, від навчання до професійного використання. На рисунку 3.3.1 відображається зовнішній вигляд робочої програми, яка буде генерувати текст. Програмний продукт на даний момент на початковій стадії розробки відповідно дизайн буде максимально мінімалістичний.

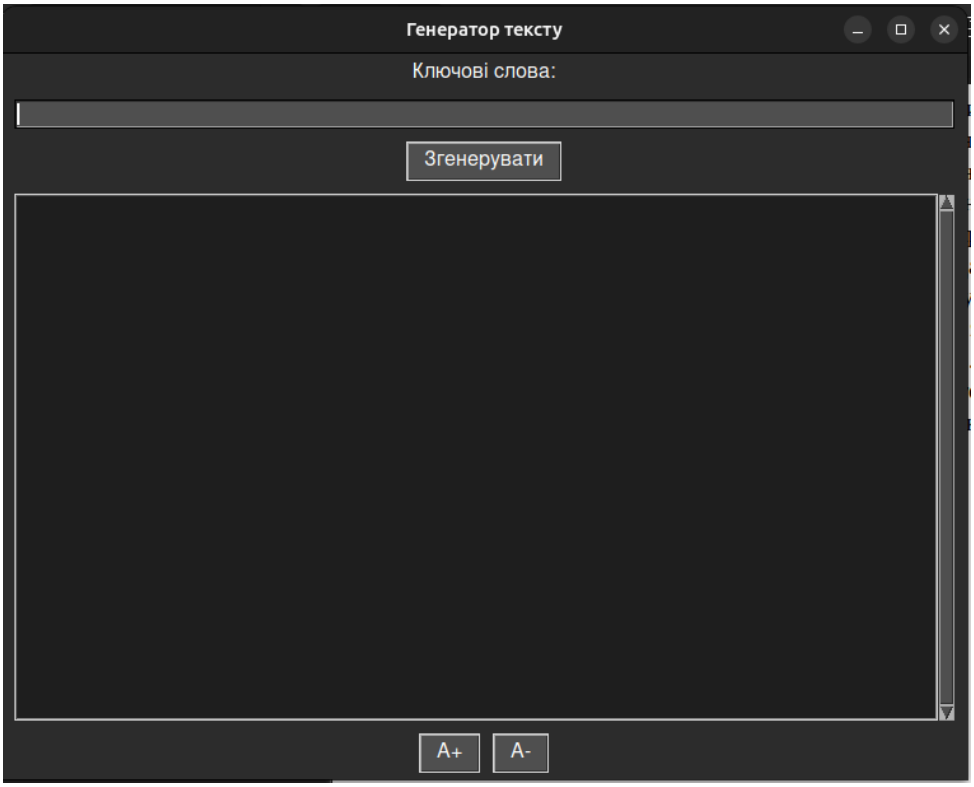


Рисунок 3.13 – Зовнішній вигляд програми для генерації тексту

Як можна спостерігати у вікні є невеличкий набір інструментів, який створений для того щоб користувач міг із легкістю вводити текст і одразу його виводити на екран. Також можна спробувати побачити як буде відображатись текст і в цьому вікні. На рисунку 3.3.2 і 3.3.3 відображається генерація тексту із топографічним ефектом. Щоб текст з'являвся по букві (так званий ефект друку),

можна використовувати метод для поступового додавання кожного символу до текстового поля. У бібліотеці tkinter для цього можна використовувати метод after(), який дозволяє додавати затримку між відображеннями кожного символу.

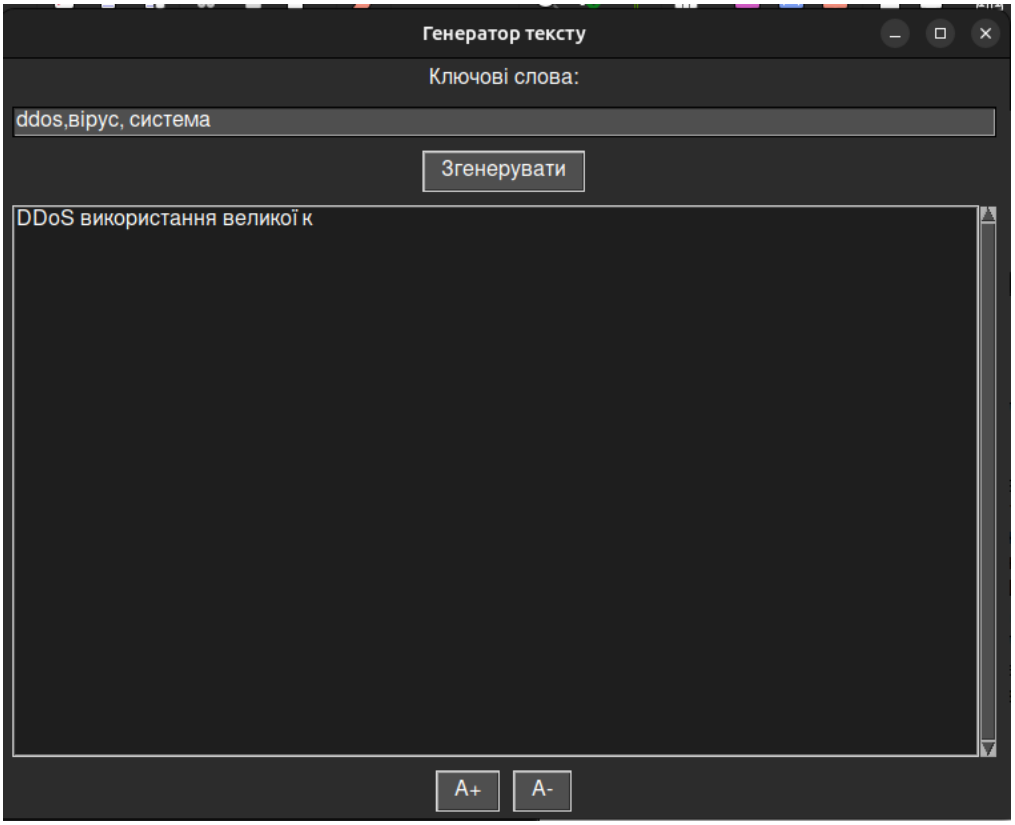


Рисунок 3.14 – Початок генерації тексту

У представленому програмному інтерфейсі можна спостерігати ефект, при якому текст генерується по букві, починаючи з перших літер, що відповідають заданим ключовим словам. Це відображає процес, коли кожен символ статті поступово з'являється на екрані, створюючи відчуття динамічного і "живого" відображення результату. Важливим моментом є те, що спочатку генерується саме та частина тексту, яка безпосередньо пов'язана з введеними ключовими словами, що підвищує точність і відповідність очікувань користувача. При використанні цього методу, програма спершу починає обробляти текстові рядки, вибираючи ті, що містять відповідні слова, і тільки потім поступово їх виводить, додаючи один символ за раз. Це створює ефект, схожий на написання тексту в реальному часі, що може бути корисно для демонстраційних або навчальних

цілей. Такий підхід надає користувачеві відчуття, що процес генерації не є статичним, а поступово адаптується до введених даних. Ключові слова впливають на форму і структуру згенерованого тексту, тому перші літери завжди будуть прямо пов'язані з запитом користувача. Завдяки цьому користувач має можливість спостерігати за процесом і переконатися, що результати відповідають їхнім вимогам. Цей ефект друку дозволяє зацікавити користувача, підвищуючи інтерактивність програми і надаючи їй більш сучасний вигляд. Крім того, додавання таких функцій, як збільшення і зменшення розміру шрифту, а також прокручування тексту, дозволяє створити більш зручний та адаптивний інтерфейс для користувача. Всі ці елементи разом створюють атмосферу активного взаємодії, де користувач може не тільки генерувати контент, а й безпосередньо впливати на спосіб його відображення.

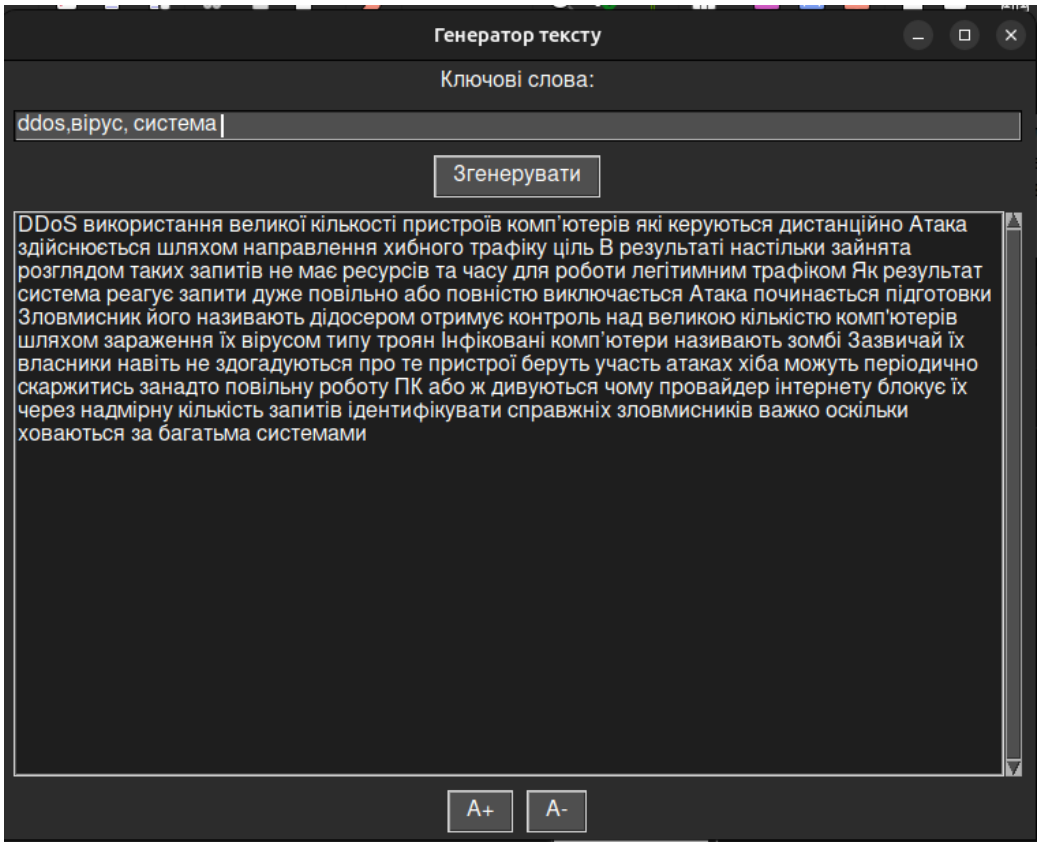


Рисунок 3.15 – Повністю згенерований текст. Являється повністю читабельним

Також як можна спостерігати текст є невеликого розміру тобто людям із обмеженими можливостями важко буде прочитати вміст і зрозуміти його.

Відповідно за допомогою навігаційних клавіш можна буде збільшувати текст то зручних розмірів. На рисунку 3.16 відображається згенерований текст у більшому розмірі.

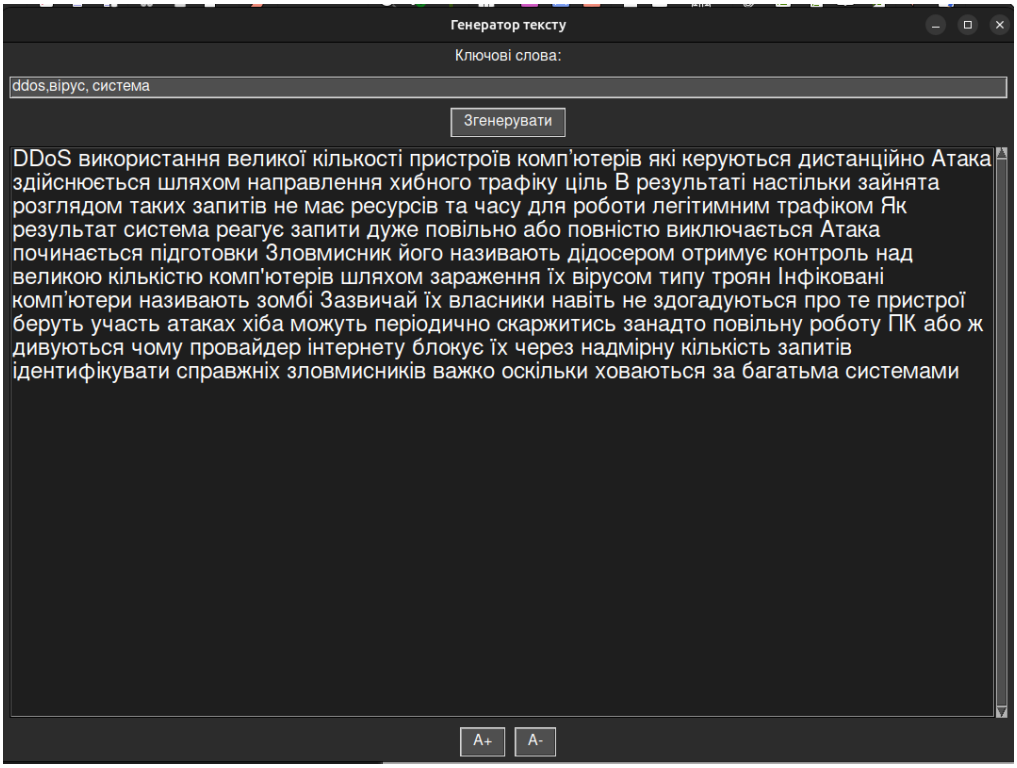


Рисунок 3.16 – Збільшений згенерований текст

Згідно рисунка можна спостерігати що програма працює коректно і надає інформацію зрозумілу та складову між собою. Звичайно порівнятись із аналогами дуже важко оскільки база інформації, яка знаходиться в таких продуктах як open ai або microsoft є дуже великою але як локальна програма, яка може допомогти у формуванні невеликих текстів є дуже актуальною та дуже зручною, оскільки усі ключові аспекти програма взяла на себе, як обробка тексту, його фільтрація та генерація.

## 3.4 Висновок до розділу 3

Робота шутчного інтелекту завжди за собою несла великі зусилля і роки праці провідних людей. Для того щоб генерація тексту була правильно потрібно розуміти логіку роботи реального інтелекту, щоб на базі цього алгоритму відтворити процес генерації тексту.

Система генерації наукового інтелекту допоможе покращити якість роботи наукових статей, в умовах відсутності мережі.

# ВИСНОВКИ

1. Проведено аналіз штучних інтелектів які базуються на генерації науково технічних текстів, це було досягнуто методом порівняння та аналізу вихідних данних популярних програмних продуктів. Систематизовано інформацію для ефективного налаштування алгоритму, що забезпечує швидку генерацію даних при великій кількості інформації, яку потрібно відфільтрувати.

2. Для оптимізації та покращенню швидкодії систему було розроблено алгоритм з використанням віртуального середовища, що включає в собі роботу системи не впливаючи на основні процеси будь якого пристрою на якому буде відбуватись генерація даних. Це було досягнуто шляхом подачі інформації із різних джерел, її порівнянні та фільтрування непотрібної та зайвої інформації .

3. У процесі розробки програмного забезпечення було реалізовано швидку роботу системи, згідно ключових слів користувача, також фільтруванню непотрібних символів та попередній підготовці тексту до належної та швидкої генерації необхідної інформації. Це досягнуто через використання nltk бібліотек, та віртуального середовища myenv.

4. Етап тестування продемонстрував ефективність розробленої програми завдяки попередній обробці тексту це сприяє зменшенню помилкових та невірних відповідей на поставлені запити. Це стало можливим завдякий швидкій та гнучкі роботи мови програмування Python та бібліотек, які створенні для роботи із штучним інтелектом, що дозволяє навчати систему і в результаті отримувати кращі результати.

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ


1. Березький О.М., Лящинський П.Б., Лящинський П.Б., Сухович А.Р., Долинюк Т.М. Синтез біомедичних зображень на основі генеративно-змагальних мереж. Український журнал інформаційних технологій. 2019, т. 1, № 1. С. 35-40.

2. Berezsky, O.M., Liashchynskyi, P.B. 2021. Comparison of generative adversarial networks architectures for biomedical images synthesis. Applied Aspects of Information Technology. Vol. 4, N 3, 250–260.

3. Berezsky O. M., Liashchynskyi P. B. Method of generative-adversarial networks searching architectures for biomedical images synthesis. Radio Electronics, Computer Science, Control. 2024. No 1, 104-117.

4. Березький О. М., Піцун О.Й., Лящинський П.Б., Лящинський П.Б., Мельник Г.М. Інтелектуальна система автоматизованої мікроскопії аналізу гістологічних та цитологічних зображень. Штучний інтелект, Київ, 2017. №2 (76). С. 128-140.

5. Березький О. М., Піцун О.Й., Вербовий С.О., Дацко Т.В. Розроблення реляційної бази даних інтелектуальної системи автоматизованої мікроскопії. Науковий вісник національного лісотехнічного університету України: Збірник науково-технічних праць. Львів: РВВ НЛТУ України. 2017. Т. 27, № 5. С.125-129.

6. Березький О. М., Мельник Г.М., Березька К. М., Дацко Т.В. Інтелектуальна система аналізу зображень ауто- та ксеногенних тканин. Науковий вісник НЛТУ України: зб. наук.-техн. праць. Львів: РВВ НЛТУ України. 2014. Вип. 24.11. С. 323-330.

7. Березький О. М., Білоус Н. В. Гібридна інтелектуальна інформаційна технологія опрацювання біомедичних зображень. Матеріали Міжнародної наукової конференції «Інтелектуальні системи прийняття рішень та проблеми обчислювального інтелекту» (ISDMCI’2015), м. Залізний Порт, 25–28 травня 2015 р. Херсон: ХНТУ, 2015. С. 30–32.

8. Березький О. М., Батько Ю.М., Мельник Г.М. Комп’ютерна система аналізу біомедичних зображень. Вісник Національного університету «Львівська політехніка». Комп’ютерні науки та інформаційні технології. 2009. № 570. С. 84-89.

9. Berezsky O., Berezska K., BatkoYu., Melnyk G. Vision-based medical expert system. 6th International Scientific and Technical Conference «Computer Sciences and Information Technologies» (CSIT'2011), Lviv, November 16-19, 2011. Lviv, 2011. P. 288-289.

10. Березький О.М., Мельник Г.М., Батько Ю.М. Текстурна сегментація біомедичних зображень на основі просторових моментів. Матеріали 4-ї міжнародної науково-технічної конференції "Комп’ютерні науки та інформаційні технології". 15-17 жовтня, 2009, Львів, Україна. Львів: ПП "Вежа і Ко", 2009 С.42–45.

11. Сич Т.А. Алгоритими та програмний засіб формування науково-технічної статті / Т.А. Сич // Всеукраїнська науково-практична конференція студентів, аспірантів та молодих вчених «Інтелектуальні комп’ютерні системи та мережі», 5 листопада 2024, м. Тернопіль. – С. 113.

12. Сич Т.А. Аналіз програмних засобів для генерації тексту в текст / Т.А. Сич // IX науково-практична конференція молодих вчених та студентів «Інтелектуальні комп’ютерні системи та мережі», 21 травня 2024 року. – С. 18.

13. Березький О. М., Дубчак Л. О., Мельник Г. М. Методичні рекомендації до виконання кваліфікаційної роботи з освітнього ступеня “Магістр”. Спеціальність: 123 – Комп’ютерна інженерія. Магістерська програма – «Комп’ютерна інженерія». Тернопіль: ЗУНУ, 2021. 32 с.

14. Learning PHP, MySQL & JavaScript: A Step-by-Step Guide to Creating Dynamic Websites (6th Edition)

15. spaCy-and-NLTK-tutorial: веб сайт. URL: https://github.com/galenwilkerson/spaCy-and-NLTK-tutorial (дата звернення 20.10.2024).

16. Natural Language Toolkit: веб сайт. URL: https://www.nltk.org/ (дата звернення 21.10.2024)

17. Python 3.13.0 documentation: веб сайт. URL: https://docs.python.org/3/ (дата звернення 21.10.2024)

18. Real Python Tutorials: веб сайт. URL: https://realpython.com/ (дата звернення 21.10.2024)

19. Гудзь, О. В., Сич, Ю. Л. Використання алгоритмів кластеризації у наукових дослідженнях. Наукові записки НаУКМА. Комп'ютерні науки, 2020. – С. 15-28. (дата звернення 25.10.2024)

20. Василенко, Р. І., Кравець, Л. О. Автоматизоване формування текстів: сучасний стан і перспективи розвитку. Вісник Житомирського державного університету імені Івана Франка. Серія: Комп'ютерні науки та інформаційні технології, 2019. – С. 55-70.(дата звернення 25.10.2024)

21. Васильєв, Д. П. Структурування та семантичний аналіз наукових текстів із використанням NLP. Вісник Національного технічного університету України "КПІ". Серія: Інформатика та кібернетика, 2022. – С. 43-58. (дата звернення 25.10.2024)

22. Василенко, О. І., Борисенко, П. П. Обробка природної мови: сучасні технології та перспективи. Збірник наукових праць НТУУ "КПІ". Серія: Комп’ютерні науки, 2022. – С. 89-105.

23. Devlin, J., Chang, M. W., Lee, K., Toutanova, K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. arXiv preprint arXiv:1810.04805. 2019. – 14 с.

24. Марченко, В. В. Машинне навчання та штучний інтелект: впровадження у наукові дослідження. Вісник Київського національного університету ім. Т. Шевченка. Серія: Інформатика, 2021. – С. 34-47.

25. Радченко, М. А., Гаврилюк, Т. В. Машинне навчання та обробка природної мови для автоматизації наукових досліджень. Вісник Харківського національного університету ім. В. Н. Каразіна. Серія: Комп'ютерні науки, 2021. – С. 76-89.

26. Rajpurkar, P., Zhang, J., Lopyrev, K., Liang, P. SQuAD: 100,000+ Questions for Machine Comprehension of Text. arXiv preprint arXiv:1606.05250. 2016. – 12 с.

27. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., Polosukhin, I. Attention is All You Need. Advances in Neural Information Processing Systems, 2017. – С. 5998-6008.

28. Алгоритми машинного навчання: від теорії до практики / за ред. М. О. Кравченка. Київ: Вид-во КНУ, 2022. – 408 с.

29. Косенко, Л. С., Поліщук, А. В. Моделі обробки текстової інформації. Вісник Чернівецького університету. Серія: Комп'ютерні науки, 2020. – С. 56-70.

30. OpenAI API Documentation: веб сайт. URL: https://platform.openai.com/docs/ (дата звернення 21.11.2024)

31. TensorFlow Documentation: веб сайт. URL: https://www.tensorflow.org/ (дата звернення 21.11.2024)

32. spaCy Models & Pipelines: веб сайт. URL: https://spacy.io/models (дата звернення 21.11.2024)

33. Rao, D., McMahan, B. Natural Language Processing with PyTorch. O'Reilly Media, 2019. – 256 с.

34. Kaur, P., Sharma, M., Sunkaria, R. Machine Learning Approaches for Text and Image Processing. Springer, 2022. – 340 с.

35. Воробей, І. М., Шевченко, Л. М. Програмні засоби для роботи з текстовою інформацією. Науковий журнал "Інформатика та кібернетика", 2022. – С. 28-40.

36. NLTK Documentation: веб сайт. URL: https://www.nltk.org/book/ (дата звернення 21.11.2024)

37. SciPy Documentation: веб сайт. URL: https://scipy.org/docs.html (дата звернення 21.11.2024)

38. GitHub NLP Projects: веб сайт. URL: https://github.com/topics/nlp (дата звернення 21.11.2024)

39. Аналітичні підходи до виявлення мовних патернів: теорія та практика / за ред. С. В. Попова. Львів: Вид-во ЛНУ, 2021. – 324 с.

40. Baroni, M., Dinu, G., Kruszewski, G. Don't count, predict! A systematic comparison of context-counting vs. context-predicting semantic vectors. ACL, 2014. – С. 238–247.

41. Manning, C. D., Schütze, H. Foundations of Statistical Natural Language Processing. MIT Press, 1999. – 720 с.

42. Jurafsky, D., Martin, J. H. Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition. 3rd edition. Pearson, 2023. – 1034 с

43. TensorFlow Documentation: веб сайт. URL: https://www.te nsorflow.org/ (дата звернення 21.11.2024)

44. Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., Dean, J. Distributed Representations of Words and Phrases and Their Compositionality. Advances in Neural Information Processing Systems, 2013. – С. 3111–3119.

45. Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P. Language Models are Few-Shot Learners. arXiv preprint arXiv:2005.14165. 2020. – 76 с.

46. Zhang, Y., Zhao, J., LeCun, Y. Text Understanding from Scratch. arXiv preprint arXiv:1502.01710. 2015. – 21 с.

47. Соколов, М. Ю. Нейронні мережі у задачах класифікації текстів. Журнал "Наукові дослідження в галузі інформаційних технологій", 2021. С. 41-58.

48. Goldberg, Y. Neural Network Methods for Natural Language Processing. Morgan & Claypool, 2017. – 309 с.

49. Elman, J. L. Finding Structure in Time. Cognitive Science, 1990. – С. 179–211.

50. Hirons, R., Sharpe, M. Practical Natural Language Processing: A Comprehensive Guide to Building NLP Solutions. O'Reilly Media, 2020. – 312 с.

51. Коваленко, В. І., Поляков, О. С. Технології автоматизації аналізу текстів: виклики та перспективи. Вісник Національного університету "Львівська політехніка". Серія: Інформаційні системи та мережі, 2022. – С. 120–134.

52. Pennington, J., Socher, R., Manning, C. D. GloVe: Global Vectors for Word Representation. Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, 2014. – С. 1532–1543.

53. Transformers and Sequence Modeling: веб сайт. URL: https://huggingface.co/transformers/ (дата звернення 21.11.2024)

54. Власенко, А. І., Карпенко, Н. Ю. Використання алгоритмів штучного інтелекту в аналізі текстових даних. Журнал "Інформатика та системний аналіз", 2023. – С. 55–70.

55. Grover, A., Leskovec, J. Node2Vec: Scalable Feature Learning for Networks. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2016. – С. 855–864.

56. Vaswani, A., et al. The Transformer Model for NLP Tasks. Advances in Neural Networks, 2018. – С. 249–255.

57. Bengio, Y., Ducharme, R., Vincent, P., Janvin, C. A Neural Probabilistic Language Model. Journal of Machine Learning Research, 2003. – С. 1137–1155.

58. Жуков, С. А., Дяченко, І. С. Машинне навчання в обробці природної мови: поточний стан і тенденції. Журнал "Інформаційні системи та технології", 2022. – С. 70–85.

59. Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., Zettlemoyer, L. Deep Contextualized Word Representations. arXiv preprint arXiv:1802.05365. 2018. – 14 с.

60. Word Embeddings and Natural Language Processing: веб сайт. URL: https://www.analyticsvidhya.com/ (дата звернення 21.11.2024)

61. Embedding Projector by TensorFlow: веб сайт. URL: https://projector.tensorflow.org/ (дата звернення 21.11.2024)

62. Cutting-Edge NLP Techniques in Practice: веб сайт. URL: https://github.com/topics/cutting-edge-nlp (дата звернення 21.11.2024)

63. Chollet, F. Deep Learning with Python. 2nd edition. Manning Publications, 2021. – 504 с.