Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Факультет комп’ютерної інженерії та управління
(повна назва)

Кафедра електронних обчислювальних машин
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА

Пояснювальна записка

Рівень вищої освіти другий (магістерський)

Аналіз впливу використання контекстуальних ембедингів на точність класифікації тексту

(тема)

Виконав:

студент II курсу, групи КСМм-22-2

Воропаєва К.А.
(прізвище, ініціали)

Спеціальність

123 «Комп’ютерна інженерія»
(код і повна назва спеціальності)

Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма

Комп’ютерні системи та мережі
(повна назва освітньої програми)

Керівник: доц. Барковська О.Ю.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри ЕОМ _____ Коваленко А.А.
(підпис) (прізвище, ініціали)

2024 р.

Харківський національний університет радіоелектроніки

Факультет комп'ютерної інженерії та управління

Кафедра електронних обчислювальних машин

Рівень вищої освіти другий (магістерський)

Спеціальність 123 «Комп'ютерна інженерія»
(код і повна назва)

Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма Комп'ютерні системи та мережі
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
"____" _____ 20__ р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

студенту Воропаєвої Ксенії Андріївни
(прізвище, ім'я, по батькові)

1. Тема роботи Аналіз впливу використання контекстуальних ембедингів на точність класифікації тексту

затверджена наказом по університету від "06" листопада 2023 р. № 1298Ст

2. Термін подання студентом роботи до екзаменаційної комісії 15 січня 2024 р.

3. Вхідні дані до роботи Текстові дані IMDB, контекстуальні ембединги, алгоритми Word2Vec та GloVe, трансформерні моделі BERT і GPT, класифікація емоційного забарвлення тексту, машинне навчання, програмування на Python, бібліотеки TensorFlow та Keras, візуалізація даних за допомогою Matplotlib та Seaborn, документація з обробки природної мови та глибокого навчання, дослідження впливу векторних представлень на точність класифікації.

4. Перелік питань, що потрібно опрацювати у роботі

1. Аналіз предметної області
2. Дослідження існуючих методів Word Embedding та Contextual Embedding.
3. Вибір технологій та інструментальних засобів для реалізації експерименту.
4. Програмна реалізація моделей з використанням різних видів ембедингів.
5. Аналіз даних і методів їх попередньої обробки для навчання моделей.
6. Оцінка впливу різних видів ембедингів на точність класифікації тексту.
7. Висновки.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (слайдів) Слайд-презентація - 18 слайдів

6. Консультанти розділів роботи (заповнюється за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Вибір та обґрунтування методики дослідження	07.11.2023 – 26.11.2023	
2	Вибір інструментальних засобів	27.11.2023 – 4.12.2023	
3	Проведення експериментів	05.12.2023 – 11.12.2023	
4	Оформлення матеріалів кваліфікаційної роботи	12.12.2023 – 20.12.2023	
5	Подання кваліфікаційної роботи керівникові та	21.12.2023 – 14.12.2023	
	її попередній захист		
6	Подання кваліфікаційної роботи на рецензу-	14.12.2023 –15.01.2024	
	вання		

Дата видачі завдання 06 листопада 2023 р.

Студент _____
(підпис)

Керівник роботи _____ доц. Барковська О.Ю.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка кваліфікаційної роботи: 87 с., 10 рис., 11 табл., 1 дод., 34 джерел.

КЛАСИФІКАЦІЯ ТЕКСТУ, WORD EMBEDDING, CONTEXTUAL EMBEDDING, LSTM, WORD2VEC, GLOVE, BERT, GPT, ТОЧНІСТЬ ПРОГНОЗУВАННЯ, ОБРОБКА ПРИРОДНОЇ МОВИ.

Метою даного дослідження є аналіз впливу використання Contextual та Word ембедінгів на точність класифікації текстових масивів. Contextual ембединги є сучасним методом представлення слів, який враховує семантичний контекст слів у тексті, що може впливати на точність аналізу текстового контенту. На відміну від Contextual ембедингів, Word ембединги захоплюють лише статичну інформацію про слова і не враховують їх семантичний зв'язок у конкретному контексті.

У ході виконання кваліфікаційної роботи були проведені дослідження, в ході яких було порівняно різні стратегії та підходи до використання ембедингів у задачах класифікації тексту.

За результатами дослідження було встановлено, що найкращою моделлю Word Embedding є GloVe, яка продемонструвала кінцеву точність на рівні 87.72%. Це було вище, ніж у моделі Word2Vec з точністю 86.87%. У контексті Contextual Embedding, BERT виявився ефективнішим порівняно з GPT, з кінцевою точністю 91.23% проти 89.45% в GPT. Ці результати свідчать про перевагу Contextual Embedding у завданнях обробки природної мови і підтверджують їхню перспективність для сучасних додатків та систем текстового аналізу.

ABSTRACT

Master’s thesis: 87 pages, 10 figures, 11 tables, 1 appendix, 34 sources.

TEXT CLASSIFICATION, WORD EMBEDDING, CONTEXTUAL EMBEDDING, LSTM, WORD2VEC, GLOVE, BERT, GPT, PREDICTION ACCURACY, NATURAL LANGUAGE PROCESSING (NLP).

The major goal of this thesis is to analyze the impact of using Contextual and Word embeddings on the accuracy of text array classification. Contextual embeddings are a modern method of word representation that takes into account the semantic context of words in the text, which can influence the accuracy of text content analysis. In contrast to Contextual embeddings, Word embeddings capture only static information about words and do not consider their semantic relationship in a specific context.

In order to accomplish this thesis goal, various strategies and approaches to using embeddings in text classification tasks were investigated.

According to the research results, the best Word Embedding model was found to be GloVe, which demonstrated a final accuracy of 87.72%. This was higher than the Word2Vec model with an accuracy of 86.87%. In the context of Contextual Embedding, BERT proved to be more effective compared to GPT, with a final accuracy of 91.23% compared to 89.45% in GPT. These results indicate the advantage of Contextual Embedding in natural language processing tasks and confirm their potential for modern text analysis applications and systems.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

API – інтерфейс програмування додатків (англ., Application Programming Interface)

BERT – двонаправлене кодування репрезентацій з трансформерів (англ., Bidirectional Encoder Representations from Transformers)

CPU – центральний процесор (англ., Central Processing Unit)

GloVe – глобальні вектори для представлення слів (англ., Global Vectors for Word Representation)

GPU – графічний процесор (англ., Graphics Processing Unit)

GPT – генеративний натренований трансформер (англ., Generative Pretrained Transformer)

HFT – бібліотека трансформерів від Hugging Face (англ., Hugging Face Transformers)

Keras – високорівневий API для нейронних мереж

LSTM – довгострокова короткочасна пам'ять (англ., Long Short-Term Memory)

NLP – обробка природної мови (англ., Natural Language Processing)

NLTK – інструментарій для обробки природної мови (англ., Natural Language Toolkit)

RAM – оперативна пам'ять (англ., Random Access Memory)

ROC-AUC – характеристика роботи приймача – площа під кривою (англ., Receiver Operating Characteristic – Area Under Curve)

TF – бібліотека для глибинного навчання (англ., TensorFlow)

Word2Vec – метод векторного представлення слів

ВСТУП

У світі надзвичайно швидких технологічних змін і невпинного розвитку інтернет-простору аналіз текстового контенту стає надзвичайно важливим завданням. Від розпізнавання сутностей в текстах до автоматичної класифікації новинних статей, текстовий аналіз став невід'ємною частиною нашого цифрового життя. Один із ключових аспектів аналізу тексту – це класифікація текстів за категоріями або визначення їхнього сенсу. При цьому, точність класифікації є справжнім викликом для дослідників і розробників програмного забезпечення. З погляду бізнесу, правоохоронних органів, медіа, та інших галузей, здатність точно класифікувати текстовий контент є ключовою для роботи з великими обсягами інформації та для прийняття рішень.

З огляду на цей контекст, ми проведемо аналіз впливу використання контекстуальних ембедингів на точність класифікації тексту. Контекстуальні ембединги – це сучасний метод векторного представлення слів, який дозволяє краще захоплювати семантичний контекст слів у тексті. Ці ембединги були вперше запропоновані в останні десятиліття і швидко здобули популярність у глибокому навчанні та обробці природної мови.

Дослідницькі питання відображаються у наступних аспектах. Для початку, ми детально розглянемо сучасний стан аналізу тексту, подаючи обзор технологій і методів, які зараз використовуються в цій галузі. Далі, ми вивчимо практичні аспекти використання контекстуальних ембедингів у задачах класифікації тексту, розглядаючи різні стратегії та підходи. Ми проведемо порівняльний аналіз результатів, щоб з'ясувати, наскільки використання контекстуальних ембедингів може покращити точність класифікації порівняно з іншими методами представлення тексту.

Наше дослідження має на меті внести вагому допомогу в розуміння ролі контекстуальних ембедингів [2] у сфері аналізу тексту та визначити їхні

можливості та обмеження в різних сценаріях. Для досягнення цієї мети, ми дослідимо різні підходи та робочі методи, що дозволить нам зрозуміти їхню ефективність і вплив на точність класифікації тексту.

У наступних розділах ми докладно розглянемо сучасний стан аналізу тексту, представимо методологію дослідження, подамо результати та обговоримо їхнє значення для практичних застосувань. В кінці ми підсумуємо основні висновки та окреслимо перспективи подальших досліджень у цій галузі.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Обґрунтування актуальності обраної теми

Сучасні вимоги до обробки мови ставлять перед нами завдання розробки більш ефективних систем, які можуть аналізувати та генерувати природну мову. Це стосується не лише текстової інформації, але і голосових комунікацій. Розуміння та обробка мови [3] є важливими складовими для подальшого розвитку інтелектуальних систем, які можуть взаємодіяти з людьми на природній мові. У цьому контексті, важливо досліджувати та розробляти методи та технології, які дозволять покращити якість розпізнавання та розуміння природної мови. Однією з таких технологій є використання контекстуальних ембедингів, які дозволяють краще захоплювати семантичний контекст слів у тексті. Вивчення та аналіз впливу цих ембедингів на точність класифікації тексту є актуальним завданням у галузі обробки мови та машинного навчання.

1.1.1 Важливість точності класифікації тексту

Класифікація тексту відіграє важливу роль у сфері обробки природної мови (NLP) і займає центральне місце серед інших завдань NLP. Завдяки класифікації текстів можна визначити приналежність текстів до певних категорій чи тем або вирішити інші проблеми, пов’язані з розподілом текстової інформації [4]. Аналіз і класифікація текстової інформації в сучасному світі стали надзвичайно актуальними завдяки великому обсягу даних, які генеруються щоденно в Інтернеті та інших джерелах. Текст виступає як важливий носій інформації, і його класифікація дозволяє вирішувати різноманітні завдання, починаючи від автоматичної фільтрації спаму і закінчуючи покращенням результатів пошукових систем та аналізом

великих обсягів документів. Правильна та точна класифікація тексту важлива з багатьох причин.

По-перше, це допомагає в покращенні пошукових систем, що дозволяє користувачам знаходити необхідну інформацію швидше та ефективніше. Відповідно, підвищення точності класифікації тексту сприяє поліпшенню результатів пошуку та якості інформації, доступної в мережі.

По-друге, точна класифікація тексту важлива в сферах, де безпека та фільтрація грають вирішальну роль. Наприклад, в області електронної пошти інструменти класифікації допомагають ідентифікувати спам та фішингові повідомлення, захищаючи користувачів від шахраїв та кібератак.

По-третє, точність класифікації тексту має велике значення в галузях, пов'язаних з аналізом суспільних думок та настроїв. Наприклад, у соціальних мережах і медіа-компаніях, точна класифікація тексту дозволяє аналізувати реакції громадськості на події, вирішувати питання репутації та прогнозувати тенденції.

Для деяких галузей, таких як медицина та юриспруденція, точність класифікації тексту має критичне значення. Наприклад, у медичній діагностиці автоматична класифікація тексту допомагає швидко знаходити та аналізувати наукові статті та медичну інформацію для покращення діагностики та лікування.

Точності класифікації тексту відчутна в різних аспектах життя та різних галузях, і її значення постійно зростає з розвитком інформаційних технологій та збільшенням обсягів текстових даних. Точність є ключовим показником ефективності систем класифікації тексту і впливає на якість прийнятих рішень та користувацьке задоволення.

1.1.2 Сфери застосування класифікації тексту

Класифікація тексту за допомогою контекстуальних ембедінгів відкриває широкі перспективи в різних сферах застосування (рисунок 1.1).

Рисунок 1.1 – Сфери застосування класифікації текстів

Ці технології дозволяють отримувати більш точні та зрозумілі результати в порівнянні з традиційними методами класифікації тексту [5]. Ось деякі з сфер, де вони мають суттєве значення:

- класифікація тексту з контекстуальними ембедінгами може використовуватися для аналізу медичних документів, статей та пацієнтських записів. Це допомагає в автоматичній діагностиці захворювань та виявленні медичних аномалій;

- в фінансовому секторі контекстуальні ембедінги можуть застосовуватися для аналізу новин, фінансових звітів та статей для прогнозування рухів на фондовом ринке. Вони також корисні для виявлення шахраївства та обману в фінансових операціях;

- класифікація тексту з контекстуальними ембедінгами допомагає аналізувати відгуки, коментарі та публічні висловлювання на соціальних мережах. Це важливо для бізнесу та маркетингу, оскільки дозволяє зрозуміти громадську думку та реакцію на продукти та послуги;

- використання контекстуальних ембедінгів дозволяє більш точно персоналізувати рекламні повідомлення та аналізувати реакцію споживачів на рекламні кампанії;

- класифікація тексту на основі контекстуальних ембедінгів допомагає виділяти спамові та шахраївські повідомлення в електронній пошті, що забезпечує безпеку та вибірковість користувачів;

- в освіті класифікація тексту на основі контекстуальних ембедінгів використовується для оцінки навчальних текстів, автоматичного оцінювання завдань та навіть розробки індивідуальних навчальних програм;

- в політичних дослідженнях класифікація тексту з контекстуальними ембедінгами застосовується для аналізу політичних програм, виборчих обіцянок та публічних висловлювань політиків.

Ці приклади лише невеликий відсоток можливих сфер застосування контекстуальних ембедінгів у класифікації тексту. Розвиток цих технологій відкриває нові можливості для автоматизації та покращення обробки текстової інформації в різних галузях. Загалом, класифікація тексту має широкі застосування в різних галузях і допомагає покращити робочі процеси, забезпечити безпеку та вдосконалити прийняття рішень. Вона відкриває можливості для автоматизації та оптимізації роботи в багатьох сферах і продовжує зростати у важливості з розвитком інформаційних технологій.

1.1.3 Особливості роботи з текстом

Робота з текстом має свої особливості, які важливо враховувати при класифікації текстових даних [6]. Деякі з основних особливостей включають:

- неоднорідність тексту. Текст може бути неоднорідним, і це включає в себе різні стилі письма, лексичну різноманітність, використання синонімів і т. д. Це може створювати складнощі при аналізі та класифікації;

- полісемія та омонімія. Слова можуть мати кілька значень (полісемія) або бути однаковими за написанням, але мати різні значення (омонімія);

Розуміння контексту важливо для правильної класифікації.

- залежність від мови. Різні мови мають свою синтаксичну та семантичну особливість. Моделі класифікації тексту повинні бути адаптовані до конкретної мови;

- дисбаланс класів. У деяких завданнях класифікації тексту може бути нерівний розподіл класів, що потребує уваги до методів балансування даних;

- закономірності тексту. Текст може містити різні закономірності, такі як ключові слова, структуровані дані (наприклад, таблиці або списки), або особливості, пов'язані з граматикою мови;

Розуміння цих особливостей є важливим для успішної класифікації текстових даних і вибору найкращого методу для конкретного завдання.

1.1.4 Варіанти та особливості оцінювання результатів

Оцінка результатів є важливою частиною дослідження, оскільки він дозволяє визначити, наскільки ефективні були застосовані методи класифікації тексту [7]. Оцінка результатів включає в себе наступні ключові аспекти:

- метрики класифікації. Перш за все, необхідно визначити метрики, які будуть використовуватися для оцінки результатів. Зазвичай для задач класифікації тексту використовуються метрики, такі як точність (accuracy), відновлення (recall), точність (precision), F1-показник і ROC-AUC;

- крос-валідація. Для надійних результатів важливо використовувати крос-валідацію, щоб перевірити, як модель працює на різних підвибірках даних. Це допомагає уникнути перенавчання та забезпечити об'єктивну оцінку продуктивності моделі;

- матриця помилок. Матриця помилок дозволяє визначити, скільки елементів було класифіковано правильно і скільки помилково для кожного класу. Це допомагає визначити, які класи мають кращу або гіршу продуктивність;

- аналіз помилок. Детальний аналіз помилок може виявити, в яких випадках модель найбільше ризикує невірно класифікувати текст. Це може призвести до подальшої покращення моделі або до удосконалення набору даних;

- порівняння методів. Іноді проводять порівняння різних методів класифікації тексту для визначення, який з них найкраще справляється з конкретною задачею. Це може включати порівняння глибокого навчання, класичних методів і використання статичних чи контекстуальних ембедінгів;

- визначення досягнутого результату. Остаточно важливо визначити досягнутий результат та порівняти його з попередніми дослідженнями або з базовими моделями, які можуть використовуватися для цієї задачі;

- узагальнення результатів. Оцінка результатів дозволяє зробити висновки про те, наскільки ефективно використані методи можуть бути застосовані в практичних завданнях класифікації тексту. Важливо визначити обмеження та можливості розробленої моделі та надати рекомендації для подальших досліджень.

1.2 Огляд існуючих аналогів. Аналіз переваг і недоліків наявних рішень

У сучасному світі обробка тексту та аналіз природної мови (Natural Language Processing, NLP) є однією з найактуальніших та найбільш динамічно розвиваючихся галузей штучного інтелекту. Однією з ключових складових в цій галузі є поняття ембедінгів. Ембедінги є числовими представленнями слів або фраз, які дозволяють комп'ютерам розуміти та обробляти природну мову [8].

Вираз "ембедінги" походить від ідеї "вбудовування" слів у числовий простір, де кожне слово або фраза представлені у вигляді векторів чисел. Ці вектори можуть відображати семантику слів, їхнє значення та взаємозв'язки з іншими словами в тексті.

1.2.1 Порівняння Word Embedding та Contextual Embedding

Зважаючи на роль, яку відіграють векторні представлення слів (embeddings) у роботі з текстом, важливо розуміти різницю між Word Embedding та Contextual Embedding.

Word Embedding (словний вбудовування) - це підхід, де кожному слову надається однаковий вектор, який відображає його значення у вигляді статичного представлення [9]. Однак цей вектор залишається незмінним незалежно від контексту, в якому слово вживається. Це означає, що одне й те ж слово матиме однаковий вектор, незалежно від контексту, в якому воно вживається. Такий підхід, наприклад, в Word2Vec або GloVe, дозволяє представити слова у векторному просторі [10-12], але не враховує їхнього різноманітного значення в залежності від контексту. Наприклад, слово "банк" буде мати однаковий вектор незалежно від того, чи йдеться про фінансовий банк чи берег річки.

Насупроти цього, Contextual Embedding (контекстуальне вбудовування) - це підхід, де для кожного слова створюється набір векторів, де кожен вектор враховує контекст вживання слова [13]. Завдяки цьому кожне слово може мати різні вектори в залежності від контексту, в якому воно зустрічається. Це дозволяє краще розуміти значення слова в різних ситуаціях і ефективніше враховувати синонімію та амбігвалентність слів.

Word Embedding застосовується в різних завданнях обробки природної мови, таких як кластеризація, семантичний пошук, аналіз настроїв та багато інших [14]. Однак він не завжди здатний ефективно вирішувати завдання, де важливий контекст. У таких випадках, наприклад, у виправленні помилок в розумінні тексту, машинному перекладі чи аналізі сентименту, Contextual Embedding [15] виявляється більш ефективним, оскільки дозволяє враховувати контекст вживання слів. Крім того, важливо враховувати, що Contextual Embedding вимагає більше обчислювальних ресурсів і даних для навчання порівняно з Word Embedding через більшу кількість параметрів в моделі.

Таблиця 1.1 – Технічна порівняльна характеристика Word Embedding та Contextual Embedding

Параметр / Особливість	Word Embedding	Contextual Embedding
Основний підхід	Фіксовані вектори слів, незалежні від контексту.	Вектори слів залежать від контексту та речення.
Розмір словнику	Великий словник із заздалегідь обчисленими векторами для кожного слова.	Маленький словник або нульовий словник, оскільки вектори генеруються під час навчання моделі.
Смислова інтерпретація	Вектори слів представляють слова незалежно від контексту та семантики.	Вектори слів враховують контекст та семантику в конкретному реченні.
Застосування	Зазвичай використовуються для базових завдань NLP, таких як машинний переклад та класифікація.	Ефективні для складних завдань, де важливий контекст, таких як розуміння тексту, заповнення пропусків та генерація тексту.
Варіативність слова	Не враховує змінюваність значень слів в різних контекстах.	Ураховує змінюваність значень слів в залежності від контексту, що робить їх більш гнучкими.
Попереднє навчання	Вимагає попереднього навчання на великих корпусах тексту.	Може бути навчана без попереднього навчання на специфічних завданнях.

Продовження таблиці 1.1

Ресурси	Вимагає великих ресурсів для попереднього навчання та зберігання великих словників.	Може бути більш ефективними для завдань з обмеженими ресурсами, оскільки словник генерується динамічно.

1.2.2 Порівняння видів Word Embedding (Word2Vec, GloVe, FastText)

У цьому розділі ми проведемо детальний аналіз та порівняння трьох популярних методів векторизації слів: Word2Vec, GloVe та FastText. Кожен з цих методів відомий своєю унікальною архітектурою та підходами до навчання, і вони широко використовуються в обробці природної мови (Natural Language Processing, NLP).

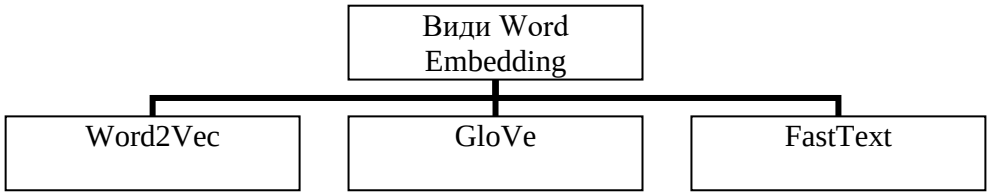

Рисунок 1.1 – Види Word Embedding

Word2Vec є однією з найпоширеніших та важливих технологій у галузі обробки природної мови (NLP) та векторного представлення слів. Ця техніка була розроблена та представлена компанією Google в 2013 році і відразу ж стала переворотом у способі роботи з текстовими даними. Основна ідея Word2Vec полягає в тому, щоб навчити модель генерувати вектори, які представляють слова, на основі контексту, у якому вони з'являються.[16] Це означає, що слова, які з'являються в схожих контекстах, отримують схожі вектори. Вектори, отримані за допомогою Word2Vec, дозволяють

враховувати семантичну схожість слів, враховуючи контекст, у якому вони зустрічаються.

Word2Vec використовує два основних підходи: Continuous Bag of Words (CBOW) і Skip-gram. У CBOW модель намагається передбачити слово на основі контексту, тобто слов, які йдуть перед і після даного слова. У варіанті Skip-gram, навпаки, модель намагається передбачити контекст на основі даного слова. Обидва підходи мають свої переваги та недоліки і можуть бути використані в залежності від конкретних завдань.

Однією з головних переваг Word2Vec є те, що вектори слів можуть бути використані для вирішення різноманітних завдань у галузі обробки природної мови. Вони допомагають покращити результати у семантичному пошуку, кластеризації тексту, сентимент-аналізі, машинному перекладі та інших завданнях. Векторні представлення слів важливі для багатьох NLP-застосувань та дозволяють комп'ютерам краще розуміти та обробляти природну мову. Деякі з основних викликів, пов'язаних із використанням Word2Vec, включають необхідність великих корпусів тексту для навчання якісних моделей, а також неспроможність враховувати слова, які не зустрічаються у навчальних даних. Також важливо враховувати, що Word2Vec може створювати вектори для слів, але не може розуміти слова в контексті абстрактних понять або концепцій.

Global Vectors for Word Representation (GloVe) – це інноваційна технологія векторного представлення слів у галузі обробки природної мови. [17] Розроблена в 2014 році на стику мовознавства та математики, GloVe стала важливим інструментом для розуміння семантичних зв'язків між словами в тексті. Основна ідея GloVe полягає в тому, щоб створити вектори, які відображають статистичні зв'язки між словами на основі їхньої спільної появи в тексті. У відміну від інших методів, які спираються на синтаксичну структуру речень або логіку слів, GloVe використовує статистичний підхід до аналізу тексту. Однією з головних переваг GloVe є його здатність до точного відображення семантичних відносин між словами. Вектори,

отримані за допомогою цієї техніки, дозволяють комп'ютерам краще розуміти, які слова синонімічні, а які антонімічні, а також визначати аналогії між словами. GloVe добре працює на великих корпусах даних та може бути використаний для вирішення різноманітних завдань в обробці природної мови, таких як семантичний пошук, кластеризація тексту та сентимент-аналіз. Він також допомагає зменшити розмірність даних, що робить його корисним для роботи з великими обсягами інформації.

FastText – це потужний інструмент в галузі обробки природної мови, розроблений на базі Facebook AI Research (FAIR).[18] Він привніс новий підхід до векторного представлення слів, який відрізняється від традиційних методів, таких як Word2Vec та GloVe. Основна інновація FastText полягає в тому, що він розглядає слова як сукупності підслів або символів. Таким чином, кожне слово розбивається на багато менших частин, і для кожного підслова обчислюються вектори. Це дозволяє FastText краще враховувати морфологічні та семантичні особливості мови, а також ефективно впоратися з мовами, які мають складну морфологію. FastText також славиться високою швидкістю навчання і роботи. Він може обробляти великі обсяги даних швидко та ефективно, що робить його ідеальним інструментом для задач, пов'язаних із обробкою тексту в реальному часі. Крім того, FastText підтримує багатомовність та може працювати з даними на різних мовах.[19] Однією з основних відмінностей FastText є можливість працювати з незнайомими словами або словами, які відсутні в словнику. Він може генерувати вектори для слів, які були поза корпусом навчальних даних, шляхом комбінування векторів їхніх підслів. Ця технологія широко використовується в різних сферах, таких як машинний переклад, аналіз тексту, сентимент-аналіз, класифікація тексту та багато інших завдань в обробці природної мови. FastText дозволяє покращити якість та швидкість обробки текстових даних, робить їх більш доступними для розробників та дослідників і грає важливу роль у розвитку сучасних систем обробки природної мови.

Таблиця 1.2 – Аналіз методів навчання векторизації слів: Word2Vec, GloVe, FastText

Параметр	Word2Vec	GloVe	FastText
Метод навчання	Нейронна мережа (CBOW або Skip-gram)	Факторизація матриці спільної входженості	Нейронна мережа з розбиттям на підслова
Розмірність векторів	Зазвичай фіксована, наприклад, 100, 300 тощо	Зазвичай фіксована, наприклад, 100, 300 тощо	Зазвичай менші завдяки використанню підслів
Підтримка мов	Відомий основний корпус для англійської та інших мов, якщо є відповідні дані	Відомий основний корпус для англійської та інших мов, якщо є відповідні дані	Розроблена для роботи з багатьма мовами, включаючи мови зі складною морфологією
Особливості	Швидка і проста для реалізації, добрі результати для багатьох завдань	Ефективна відображає статистичні зв'язки між словами, має здатність до репрезентації семантичних відношень	Добре справляється з мовами зі складною морфологією та дозволяє використовувати вектори для підслів

Продовження таблиці 1.2

Використання	Різні завдання обробки природної мови, включаючи кластеризацію, семантичний пошук, сентимент-аналіз тощо	Різні завдання обробки природної мови, здатна до репрезентації статистичних взаємозв'язків	Особливо ефективна для мов зі складною морфологією, машинного перекладу, аналізу мови тощо

1.2.3 Порівняння видів Contextual Embedding (BERT, GPT, XLNet)

У цьому розділі ми розглянемо та порівняємо три передових методи векторизації слів, які базуються на контексту та мають значний вплив на сучасні методи обробки природної мови: BERT (Bidirectional Encoder Representations from Transformers), GPT (Generative Pre-trained Transformer) та XLNet. Кожен з цих методів відзначається своєю спроможністю аналізувати контекст слів та генерувати змістовний текст.

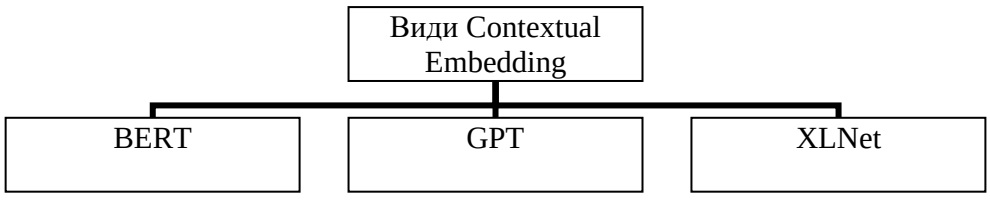

Рисунок 1.2 – Види Contextual Embedding

BERT (Bidirectional Encoder Representations from Transformers) - це інноваційна та потужна модель для векторного представлення слів, яка змінила гру в обробці природної мови. Розроблений в 2018 році фахівцями з

Google, BERT представляє собою модель глибокого навчання на основі архітектури Transformer.

Однією з ключових особливостей, які ми використовуємо в нашій LSTM моделі, є здатність BERT розуміти контекстуальну семантику слів, тобто він дозволяє враховувати слова, що оточують дане слово у тексті, під час процесу векторизації. [20]. Це робить BERT надзвичайно ефективним для різноманітних завдань обробки природної мови, таких як класифікація тексту, сентимент-аналіз, машинний переклад, відповіді на питання та багато інших. BERT навчається на великих корпусах тексту та піддається двосторонньому навчанню, що дозволяє йому розуміти слова та фрази в контексті. Модель включає велику кількість параметрів та зазвичай має декілька варіацій, такі як BERT-Base та BERT-Large, в залежності від розміру та складності завдань. BERT став важливим кроком у розвитку обробки природної мови, оскільки він значно покращив результати для багатьох NLP-завдань. Ця модель використовується в різних сферах, включаючи пошукові системи, маркетинг, медицину, фінанси та інші галузі. BERT відкриває нові можливості для розуміння та аналізу природної мови, і він залишається однією з найважливіших моделей в сфері NLP.

GPT (Generative Pre-trained Transformer) – це інноваційна модель штучного інтелекту, яка відкриває нові горизонти у розумінні та генерації природної мови. Розроблена компанією OpenAI, GPT використовує архітектуру Transformer і навчається на великих корпусах тексту [21]. Основна ідея GPT полягає в передбаченні наступного слова в тексті на основі контексту, що йому передує. Модель навчається на масиві тексту, розуміючи лексичний та семантичний зв'язок між словами та фразами. Ця передбачувальна здатність дозволяє GPT генерувати тексти, які мають логічну послідовність та природний стиль. GPT став першою моделлю, яка досягла значних успіхів у генерації тексту, що надало йому великий вплив у багатьох сферах. Він використовується для автоматичної генерації статей, пошуку відповідей на питання, підтримки в розмовних ботах, автоматичного

перекладу та багатьох інших завдань обробки природної мови. Однією з найважливіших особливостей GPT є його здатність до творчого написання та генерації тексту. Велика кількість параметрів моделі, яка зазвичай складається з мільйонів нейронів, дозволяє створювати тексти високої якості з різними стилями та темами. Завдяки своїй потужності та гнучкості, GPT відкриває широкий спектр можливостей для розробки нових застосувань в обробці природної мови, і він залишається однією з найбільш впливових моделей в цій сфері.

XLNet – це інноваційна модель обробки природної мови, розроблена для досягнення високої продуктивності у завданнях, пов'язаних з текстовою аналітикою [22]. Ця модель використовує архітектуру Transformer та спеціальну методологію навчання, яка відрізняє її від інших моделей. Однією з основних особливостей XLNet є те, що вона використовує багатопрохідний підхід до обробки тексту. Це означає, що модель не обмежена послідовним розгляданням слів у тексті, як це роблять багато інших моделей. Замість цього, вона може враховувати контекст у будь-якому порядку, враховуючи всі можливі зв'язки між словами та фразами. Ця здатність робить XLNet особливо потужною для завдань, пов'язаних із залежностями між словами в тексті, семантичними відношеннями та аналізом синтаксису. Вона дозволяє моделі легко розуміти та враховувати різні варіанти контексту та сенсу речень, що покращує якість аналізу тексту. XLNet також славиться своєю здатністю до передбачення слів та фраз в тексті, враховуючи всі можливі комбінації контексту. Це робить модель дуже ефективною в завданнях, де необхідно передбачити наступні слова на основі поточного контексту.

Ці методи Contextual Embedding – BERT, GPT і XLNet – є важливими досягненнями у сфері обробки природної мови. Вони вирізняються своєю здатністю до розуміння контексту в тексті та глибоким аналізом семантики. Їх використання дозволяє вирішувати складні завдання, пов'язані з розумінням мови, з максимальною точністю та ефективністю. Таблиця 1.3 порівнює ці три методи за різними характеристиками.

Таблиця 1.3 – Порівняння характеристик BERT, GPT та XLNet

Характеристика	BERT	GPT	XLNet
Архітектура	Bidirectional Transformer	Unidirectional Transformer	Bidirectional Transformer
Кількість параметрів	Зазвичай більше мільярда	Зазвичай від декількох мільйонів до декількох мільярдів	Зазвичай більше мільярда
Велика кількість завдань	Так, включаючи Masked Language Modeling, Next Sentence Prediction та інші	Ні	Так, включаючи Permutation Language Modeling, Masked Language Modeling та інші
Розмірність векторів	Зазвичай велика, наприклад, 768 або 1024	Зазвичай велика, наприклад, 768 або 1408	Зазвичай велика, наприклад, 768 або 1024
Підтримка мов	Може бути адаптований для багатьох мов	Може бути адаптований для багатьох мов	Може бути адаптований для багатьох мов
Використання	Класифікація тексту, сентимент-аналіз, машинний переклад та інші завдання обробки природної мови	Генерація тексту, завдання з розумінням мови та інші текстові завдання	Класифікація тексту, машинний переклад, сентимент-аналіз та інші завдання обробки природної мови

Продовження таблиці 1.3

Застосування	Широкий спектр завдань обробки природної мови та різних NLP-завдань	Генерація тексту, чат-боти та інші текстові завдання	Класифікація тексту, сентимент-аналіз, машинний переклад та інші завдання обробки природної мови

1.3 Обґрунтування доцільності вдосконалення існуючих рішень

Обґрунтування доцільності вдосконалення існуючих рішень в галузі аналізу впливу використання контекстуальних ембедінгів на точність класифікації тексту полягає у необхідності врахування ефективності інноваційних підходів у сучасних методах обробки природної мови (NLP). При цьому слід враховувати кілька ключових аспектів, які свідчать про доцільність такого вдосконалення.

Контекстуальні ембедінги представляють собою важливий крок вперед у векторному представленні тексту. Вони дозволяють враховувати семантичний контекст слів в реченні чи документі, що веде до кращого розуміння тексту. У звичайних word2vec-ембедінгах кожне слово представлене статичним вектором, тоді як контекстуальні ембедінги здатні враховувати, як слово використовується в конкретному контексті. Це допомагає поліпшити точність класифікації тексту та розпізнавання семантики.

Контекстуальні ембедінги вже широко використовуються в NLP завданнях, таких як класифікація тексту, аналіз тональності, розпізнавання іменованих сутностей, та багатьох інших. Вони дозволяють моделям краще розуміти текст та враховувати семантичний контекст при прийнятті рішень.

Це може призвести до підвищення точності класифікації та зниження кількості помилкових позитивних та негативних результатів.

Сучасні NLP-моделі, такі як BERT, GPT і ELMo [23], відкривають доступ до потужних інструментів для отримання контекстуальних ембедінгів, які можна успішно використовувати в рамках LSTM моделі. Вони стали стандартами в галузі і мають відкритий доступ до передньо-навчених моделей. Використання цих моделей дозволяє значно підвищити точність класифікації тексту, оскільки вони враховують більше інформації з контексту. Для вдосконалення класифікації тексту з використанням контекстуальних ембедінгів існують багато відкритих даних і ресурсів. Це робить впровадження таких рішень більш доступним та обґрунтованим для дослідників і розробників.

У висновку, використання контекстуальних ембедінгів для аналізу впливу на точність класифікації тексту обґрунтоване через їхню здатність краще враховувати семантичний контекст слів. Це може призвести до покращення рішень в галузі NLP та розширення областей застосування, де точність класифікації є критично важливою.

1.4 Постановка задачі

Метою даного дослідження є аналіз впливу використання Contextual та Word ембедінгів на точність класифікації текстових масивів.

Для досягнення поставленої мети мають бути вирішені наступні задачі:

- порівняльний аналіз методів векторизації текстових документів;
- огляд існуючих конвеєрів класифікації тексту, включаючи препроцесінг та безпосередній аналіз;
- дослідження впливу методів векторизації тексту (Word2Vec та GloVe) на точність класифікації текстових масивів в рамках підходу Word Embedding на основі нейромережевої моделі LSTM;

- дослідження впливу методів векторизації тексту (подібні до векторизації в моделях BERT та GPT) на точність класифікації текстових масивів в рамках підходу Contextual Embedding на основі нейромережевої моделі LSTM;
- аналіз точності отриманих результатів.

Для подальшого розвитку цього дослідження і підвищення його практичної цінності можна розглянути кілька перспективних напрямків. Експерименти з альтернативними моделями контекстуальних ембедінгів. У цьому дослідженні ми розглядаємо модель LSTM з використанням Word Embedding та Contextual Embedding. Проте існують інші контекстуальні моделі, такі як BERT і GPT, які також заслуговують на увагу. Порівняння цих моделей може розширити наше розуміння їхнього впливу на точність класифікації тексту. Для підвищення точності класифікації можна розглянути можливість використання додаткових навчальних даних. Додавання більш різноманітних та об'ємних корпусів тексту може позитивно вплинути на результати. Подальший розвиток дослідження може включати оптимізацію параметрів моделі LSTM та регуляризацію для зменшення перенавчання та підвищення точності класифікації. Результати цього дослідження можуть бути розширені на інші завдання NLP, такі як аналіз тональності, визначення іменованих сутностей та машинний переклад. Аналіз впливу контекстуальних ембедінгів на ці завдання може розкрити їхню універсальність.

2 АНАЛІЗ ТЕХНОЛОГІЙ ДЛЯ РОЗВ'ЯЗАННЯ ЗАДАЧІ

2.1 Визначення апаратної бази для виконання експериментальної частини проєкту

Апаратна база виконання експериментів є одним із критичних аспектів, що впливають на успішне проведення класифікації тексту з використанням контекстуальних ембедінгів. У даному розділі ми ретельно розглянемо архітектуру обчислювачів та серверів, які будуть використовуватися для наших експериментів, а також визначимо ключові параметри, такі як центральний процесор (CPU), графічний процесор (GPU) та оперативна пам'ять (RAM), які суттєво впливають на продуктивність обробки тексту.

2.1.1 Детальний огляд архітектури обчислювачів та серверів, придатних для виконання обчислень.

Апаратна база виконання експериментів з класифікацією тексту з використанням контекстуальних ембедінгів визначається у великій мірі архітектурою обчислювачів та серверів. Огляд цих архітектур є важливим кроком у підготовці до ефективних експериментів. Сервери та обчислювачі, які придатні для обробки тексту, мають різну архітектуру, що може суттєво вплинути на продуктивність і результати наших досліджень. Один із ключових аспектів – це кількість процесорів. Багатоядерні процесори дозволяють паралельно обробляти багато текстових документів, що підвищує продуктивність. Також важливо враховувати можливість розширення обчислювального потенціалу шляхом додавання нових процесорів.

Графічний процесор (GPU) важливий для виконання операцій над великими об'ємами даних. Особливо це стосується моделей машинного навчання, які використовують глибоке навчання [24]. GPU може значно

прискорити обчислення та тренування моделей завдяки великій кількості паралельних обчислень.

Обсяг оперативної пам'яті (RAM) грає ключову роль при обробці великих текстових корпусів та при тренуванні моделей з великою кількістю параметрів. Недостатній обсяг RAM може призвести до обмежень у розмірах корпусів та моделей, які можна використовувати [25].

Окрім цього, сервери повинні мати високу надійність та можливість автоматичного відновлення в разі відмови. Непередбачувані перерви у роботі обчислювача можуть негативно вплинути на результати експериментів та призвести до втрати даних. Важливим аспектом є також відповідність апаратної бази програмним засобам, які ми плануємо використовувати для класифікації тексту. Деякі програмні бібліотеки та фреймворки можуть бути оптимізовані для роботи з конкретними типами обчислювачів та GPU.

У підсумку, вибір апаратної бази має бути обґрунтованим і враховувати вимоги до продуктивності та обсягу обробки тексту в рамках нашого дослідження. Неправильний вибір апаратної бази може призвести до невдалого проведення експериментів та незадовільних результатів.

2.1.2 Вибір корпусів

Існує багато різних корпусів для навчальних моделей класифікації текстів, які стали важливим ресурсом для наукових досліджень і прикладних застосувань у сфері обробки природної мови [27, 28]. Ці корпуси представляють різні типи текстових даних з різних джерел і охоплюють широкий спектр тем. Щоб забезпечити успішне порівняння моделей нейронних мереж для класифікації тексту, важливо ретельно вибрати відповідний корпус, який відповідає цілям дослідження та характеристикам завдання. До найпопулярніших випадків відносяться, зазначені в таблиці 2: Корпус IMDB, корпус новин Reuters, корпус наукових статей PubMed, корпус аналізу настроїв у Twitter.

Таблиця 2.2 – Корпуси текстових даних для класифікації тексту у завданнях НЛП

№	Корпус	Опис	Задача класифікації
1	Корпус IMDB	Містить рецензії на фільми з сайту IMDB, розділені на позитивні та негативні категорії.	Бінарна класифікація текстів (позитивні/негативні відгуки)
2	Корпус новин Reuters	Містить статті новин, упорядковані за темами.	Категоріальна класифікація текстів
3	Корпус наукових статей PubMed	Містить наукові статті з медичних джерел для класифікації за темою чи важливістю.	Категоріальна класифікація текстів
4	Корпус аналізу настроїв у Twitter	Містить повідомлення з Twitter, класифіковані за настроями (позитивні, негативні чи нейтральні).	Класифікація настроїв у коротких текстах

Враховуючи складність завдання класифікації тексту та бажання виконати якісне порівняння моделей нейронних мереж, було важливо вибрати відповідний корпус для навчання моделей. Найбільш зручним і відповідним для наших дослідницьких цілей виявився корпус IMDB.

Причинами вибору корпусу IMDB є його різноманітність, доступність і репрезентативність. Завдяки великій кількості даних IMDB може надати достатню кількість прикладів для навчання та тестування моделей, що важливо для надійного порівняння їх продуктивності. Крім того, IMDB містить текстові рецензії з емоційним забарвленням, що є важливою функцією для вирішення задачі бінарної класифікації. Ще однією важливою перевагою IMDB є його доступність, що дозволяє дослідникам з усього світу використовувати цей корпус для своїх досліджень. Це сприяє широкій

застосовності результатів роботи та можливості порівняння з іншими науковими дослідженнями. Крім того, різноманітність довжин тексту в корпусі IMDB дозволяє перевірити здатність моделей працювати з послідовностями різної складності та довжини, що є вирішальним для реалістичного аналізу продуктивності нейронної мережі на різноманітних вхідних даних.

Результати нашого дослідження, що порівнює моделі нейронних мереж для класифікації тексту, демонструють багато перспективних областей і завдань, у яких можна застосувати ці ідеї та методи. Наш вибірковий збірник рецензій на фільми IMDB дозволяє нам визначити потенціал у таких сферах і проблемах:

- аналіз настроїв відгуків про продукти та послуги - це допоможе компаніям зрозуміти задоволеність клієнтів, виявити проблемні області та покращити свої продукти;

- класифікація вмісту у веб-службах – це допоможе організаціям і платформам автоматично фільтрувати вміст, щоб забезпечити безпеку та позитивну взаємодію з користувачем;

- аналіз емоцій у соціальних мережах – це допоможе зрозуміти реакцію на новини, події та публікації, що важливо для рекламодавців і маркетологів;

- моніторинг брендів та компанії для відстежування й аналізу громадську думку, допомагаючи менеджерам реагувати на зміни у сприйнятті товарів і послуг.

Аналіз точності та ефективності різних підходів є важливим кроком для покращення розуміння потенційних переваг і обмежень контекстного вбудовування порівняно з традиційними методами. Ці дані можуть служити важливим орієнтиром у виборі оптимальної моделі для конкретних завдань класифікації тексту. Метою наших досліджень є покращення якості та точності результатів класифікації та сприяння розвитку та впровадженню новітніх методів у цій галузі.

2.2 Аналіз технологій для розв'язання поставленої задачі

В сучасному світі обробка та аналіз текстової інформації стали ключовими компонентами багатьох діяльностей, від прийняття рішень в бізнесі до автоматизованого класифікації контенту в соціальних мережах. Однак, завдання класифікації тексту виявляються складними через різноманітність і структурованість мовлення, що вимагає вдосконалених методів та технологій для досягнення високої точності та ефективності. В даному розділі ми проведемо аналіз технологій, які застосовуються для розв'язання завдань класифікації тексту, і зосередимося на порівнянні двох ключових підходів: використання моделей з довільними векторами слів (Word Embedding) та використання контекстуальних ембедінгів.

Цей розділ розглядає інструменти та бібліотеки, необхідні для виконання дослідження та практичних експериментів у галузі класифікації тексту, з особливим акцентом на мові програмування Python та наступних бібліотеках: TensorFlow, Keras, Gensim, Hugging Face Transformers та Natural Language Toolkit (NLTK).

Ми розглянемо різні аспекти вибору технологій та бібліотек для побудови та навчання моделей, дослідження ефективності методологій та підходів, а також обрання ефективних методів векторизації тексту, що сприятиме досягненню найкращих результатів. Мета цього розділу полягає у розкритті потенціалу та обмежень різних технологічних рішень, що стосуються класифікації тексту, та у визначенні оптимального підходу для вирішення поставленої задачі через подальший аналіз та порівняння.

2.2.1 Мова програмування Python

Python – це високорівнева, інтерпретована, загальнопризначена мова програмування, яка стала однією з найпопулярніших та найбільш використовуваних мов у світі програмування. Його велика популярність

пояснюється численними перевагами та різнобічністю використання. Python був розроблений Гвідо ван Россумом і перша версія була випущена в 1991 році. З тих пір Python зазнав значних змін і розвинувся у потужний інструмент для різноманітних завдань.

Однією з ключових особливостей Python є його читабельний та зрозумілий синтаксис. Python став "зенітним" мовою, що означає, що розробники намагаються писати код, який легко читати та зрозуміти. Це робить мову дуже доступною, навіть для початківців.

Python також підтримує об'єктно-орієнтоване програмування (ООП), функціональне програмування, та інші парадигми, що дозволяють розробникам використовувати різноманітні стилі програмування залежно від потреб проекту. Інтерпретована природа Python робить його дуже інтерактивним та дозволяє розробникам виконувати код без необхідності компіляції. Це робить розробку та тестування швидшим та більш ефективним процесом.

Python має велику кількість стандартних бібліотек, які розширюють його можливості у різних галузях, включаючи роботу з мережами, обробку даних, роботу з базами даних, графіку та багато іншого. Це робить Python універсальним інструментом для різних завдань. Для наукових досліджень та аналізу даних Python став дуже популярним завдяки бібліотекам, таким як NumPy, Pandas, Matplotlib, та SciPy. Велика наукова спільнота використовує Python для статистичного аналізу, машинного навчання, обробки зображень та багатьох інших завдань. Особливою популярністю користується використання Python у сфері штучного інтелекту та машинного навчання. Інструменти, такі як TensorFlow, Keras, PyTorch та бібліотеки для обробки природної мови, дозволяють розробникам створювати і навчати складні моделі штучного інтелекту та нейронних мереж для різних завдань.

Один з важливих аспектів Python – це активна спільнота розробників, яка надає велику кількість допомоги, документації, та багато інших ресурсів для початківців та досвідчених розробників. Python зберігається на GitHub, і

розробники по всьому світу активно вносять вклад у розвиток мови та її інфраструктури.

2.2.2 Бібліотека TensorFlow

TensorFlow – це відкритий інструмент для розробки та навчання моделей штучного інтелекту та нейронних мереж, розроблений компанією Google. Він став однією з найпопулярніших бібліотек для машинного навчання та глибокого навчання, завдяки своїй потужності та розширюваності.

Однією з головних переваг TensorFlow є підтримка обчислень на графах, що робить його ідеальним для створення складних моделей та обчислень, які вимагають великої обчислювальної потужності. Він дозволяє визначити обчислювальний граф, в якому визначаються операції та їх зв'язки, і потім виконувати обчислення на цьому графі. Це особливо корисно при роботі з глибокими нейронними мережами, де може бути сотні тисяч операцій.

TensorFlow підтримує багато різних типів штучних нейронних мереж, включаючи звичайні нейронні мережі, згорткові нейронні мережі (CNN), рекурентні нейронні мережі (RNN) та багато інших. Він також надає підтримку для роботи з послідовністю даних, великими наборами даних та обробки зображень. TensorFlow має велику кількість користувацьких інтерфейсів, включаючи Keras, що робить його дружнім для початківців та досвідчених розробників. Keras - це вищестояча бібліотека, що спрощує процес створення та навчання моделей штучного інтелекту, і вона інтегрована в TensorFlow.

Однією з важливих особливостей TensorFlow є можливість використання обчислювальних ресурсів на різних платформах, включаючи локальні сервери, хмарні рішення та мобільні пристрої. Це робить його універсальним для використання в різних сценаріях.

2.2.3 Бібліотека Keras

Keras – це високорівнева бібліотека для машинного навчання та глибокого навчання, яка призначена для створення та навчання нейронних мереж швидко та ефективно. Однією з головних переваг Keras є його простий та інтуїтивний інтерфейс, який робить його ідеальним для початківців та досвідчених розробників. Keras був розроблений як вищестоячий інтерфейс для роботи з бібліотеками глибокого навчання, такими як TensorFlow, Theano, та CNTK. Це дозволяє розробникам легко створювати та навчати нейронні мережі без необхідності написання великої кількості коду.

Однією з ключових особливостей Keras є його модульність. Ви можете легко створювати та комбінувати різні шари нейронних мереж, створюючи складні моделі за допомогою простого API. Це дозволяє розробникам швидко створювати різноманітні архітектури нейронних мереж для різних завдань.

Keras підтримує багато різних типів штучних нейронних мереж, включаючи звичайні нейронні мережі, згорткові нейронні мережі (CNN), рекурентні нейронні мережі (RNN) та багато інших. Це дозволяє розробникам працювати з різноманітними типами даних та завдань. Keras також має велику кількість вбудованих функцій для обробки даних, включаючи завантаження та попередню обробку даних, розробку функцій втрат та оптимізаторів для навчання моделей. Це спрощує роботу з даними та навчанням моделей.

Однією з істотних переваг Keras є його переносимість. Оскільки він працює на базових бібліотеках глибокого навчання, ви можете легко переносити ваші моделі між різними фреймворками, такими як TensorFlow, що робить його ідеальним для проектів, де може знадобитися зміна фреймворку.

Keras також активно підтримується та розвивається, з новими версіями, які надають нові можливості та поліпшення. Розробники по всьому світу

активно внесли свій вклад у цей проект, що робить його однією з найпопулярніших бібліотек для роботи з нейронними мережами.

Загалом, Keras – це потужна та зручна бібліотека для розробки нейронних мереж та глибокого навчання, яка робить роботу з нейронними мережами доступною для всіх. Вона надає інструменти для створення та навчання моделей швидко та ефективно, що робить її незамінною для багатьох розробників та дослідників у галузі машинного навчання та глибокого навчання.

2.2.4 Бібліотека Gensim

Gensim – це бібліотека для обробки та аналізу текстових даних, спеціалізована на роботі з векторними представленнями слів та текстів, що стала дуже популярною в галузі обробки природної мови та тематичного моделювання. Розроблена Радімом Рехуреком та Лукашем Русінським, Gensim дозволяє виконувати операції, які допомагають зрозуміти зміст текстів та виділити важливі теми. Однією з головних функцій Gensim є побудова векторних представлень слів та текстів, відомих як "word embeddings" та "document embeddings". Це дозволяє перетворити слова та текст на числові вектори, що можна використовувати для подальшої аналізу та порівняння. Векторні представлення дозволяють вимірювати подібність між словами, знаходити семантичні асоціації та виконувати багато інших завдань в галузі обробки природної мови. Gensim також надає інструменти для тематичного моделювання, що допомагає виокремити теми та концепції в текстах. Тематичне моделювання корисне для аналізу колекцій текстів, виявлення ключових слів та тем, а також для рекомендаційних систем та інших завдань.

Однією з ключових особливостей Gensim є його підтримка для роботи з великими обсягами даних та потужність обробки тексту. Він може ефективно опрацьовувати великі тексти та колекції документів, що робить його

корисним інструментом для дослідників та компаній, які працюють з великими обсягами текстової інформації. Gensim також підтримує багато різних алгоритмів, включаючи Word2Vec, Doc2Vec, та Latent Dirichlet Allocation (LDA). Це дозволяє розробникам вибирати найкращий підхід для своїх завдань та даних.

2.2.5 Бібліотека Hugging Face Transformers

Hugging Face Transformers – це бібліотека та платформа для навчання, розробки та використання моделей обробки природної мови (NLP) та перекладу мови. Вона набула надзвичайної популярності серед дослідників, розробників та компаній завдяки своїм потужним та готовим до використання моделям, які допомагають вирішувати різноманітні завдання NLP.

Однією з ключових особливостей Hugging Face Transformers є її велика бібліотека готових моделей для NLP та перекладу мови. Вона включає в себе різноманітні моделі, включаючи BERT, GPT-2, RoBERTa та багато інших. Ці моделі навчені на великих обсягах тексту та можуть вирішувати різноманітні завдання, включаючи класифікацію текстів, генерацію тексту, питання-відповідь та багато інших. Hugging Face Transformers також надає інструменти для навчання власних моделей. Ви можете використовувати існуючі моделі як основу та налаштовувати їх для своїх конкретних завдань та даних. Це робить платформу дуже гнучкою та придатною для різних завдань.

Іншою важливою особливістю є можливість використання Hugging Face Transformers як сервісу через хмарні рішення. Це означає, що ви можете використовувати готові моделі та ресурси без необхідності великих обчислювальних витрат та інфраструктури. Це особливо корисно для стартапів та проектів, які мають обмежений бюджет.

Загалом, Hugging Face Transformers – це потужна та готова до використання платформа для роботи з NLP та перекладом мови, яка дозволяє легко вирішувати складні завдання завдяки готовим моделям та інструментам для навчання власних. Вона відкриває нові можливості для розробників та дослідників у галузі обробки природної мови та робить її доступною для всіх, хто має інтерес до цього захопливого поля.

2.2.6 Бібліотека Natural Language Toolkit (NLTK)

Natural Language Toolkit (NLTK) – це популярна бібліотека для обробки природної мови (NLP) в середовищі програмування Python.[29] NLTK розроблений для спрощення завдань, пов'язаних з аналізом та обробкою текстової інформації, і став однією з ключових бібліотек для дослідників, розробників та фахівців у галузі NLP. Однією з головних переваг NLTK є його широкий спектр функцій та можливостей. Він надає інструменти для різноманітних завдань, включаючи токенізацію (розбиття тексту на окремі слова чи фрази), аналіз частин мови (визначення частин мови для кожного слова), стеммінг (скорочення слів до їх базової форми), лематизацію (перетворення слів до їх словникової форми) та багато інших.

NLTK також включає в себе корпуси текстових даних та словники для різних мов, що дозволяє розробникам та дослідникам працювати з реальними даними та текстами. Це особливо корисно для навчання та тестування моделей в галузі NLP. Однією з ключових особливостей NLTK є його підтримка для освіти та навчання. Він містить велику кількість навчальних матеріалів та прикладів, що допомагають новачкам ознайомитися з основами NLP та розвинути свої навички. NLTK використовується у навчальних програмах та курсах з обробки природної мови в університетах та онлайн-курсах.

Бібліотека NLTK також дозволяє розробникам створювати власні алгоритми та оброблювати тексти під свої потреби. Вона має активну

спільноту розробників, яка публікує нові розширення та функції для бібліотеки, що робить її більш потужною та розширює її можливості. Загалом, NLTK - це потужний та різнобічний інструмент для обробки тексту та аналізу природної мови в середовищі Python. Він надає інструменти для різноманітних завдань та завдяки своєму активному спільнотою користувачів та розробників залишається однією з найпопулярніших бібліотек у галузі NLP.

2.3 Аналіз методологічного підґрунтя для розв'язання поставленої задачі

Для класифікації текстів використовуються різні інструменти, які дозволяють досягти високого рівня точності в цьому завданні, часто комбінуючи вже існуючі методи, які добре себе зарекомендували [30]. Перш за все, використовуються правила та евристичні методи, які базуються на заданих правилах та експертних знаннях. Ці методи особливо ефективні, коли структура даних проста і зв’язки між категоріями вже відомі. Для класифікації також використовуються методи машинного навчання. За допомогою машинного навчання ви можете створювати моделі, які автоматично розпізнають шаблони в текстових даних і виконують класифікацію на основі набору навчальних даних. У цій галузі широко використовуються такі методи, як Наївний Байєс, Метод опорних векторів (SVM) [31], Дерева рішень та інші.

Нейронні мережі є особливим видом методів машинного навчання, натхненний структурою нейронної мережі мозку. Нейромережеві моделі показують вражаючі результати та переваги порівняно з традиційними методами, зображеними на рисунку 2, такими як наївний Байєс, метод опорних векторів (SVM), дерева рішень та інші, оскільки вони дозволяють автоматично виявляти внутрішні шаблони в текстових даних і класифікувати з високою точністю, що показано в роботі [32].

Вибір правильної архітектури нейронної мережі є важливим кроком у вирішенні проблеми класифікації тексту. На рисунку нижче (рисунок 2.1) наведено діаграму, яка ілюструє базові етапи класифікації текстових масивів, від їх ініціального введення до отримання кінцевих результатів. Кожен блок на діаграмі відображає окремий процес - від препроцесингу та нормалізації даних до їх векторизації та подальшого аналізу за допомогою нейромережевих алгоритмів. Діаграма також визначає вхідні та вихідні потоки, а також ресурси та інструменти, які задіяні на кожному з цих етапів.

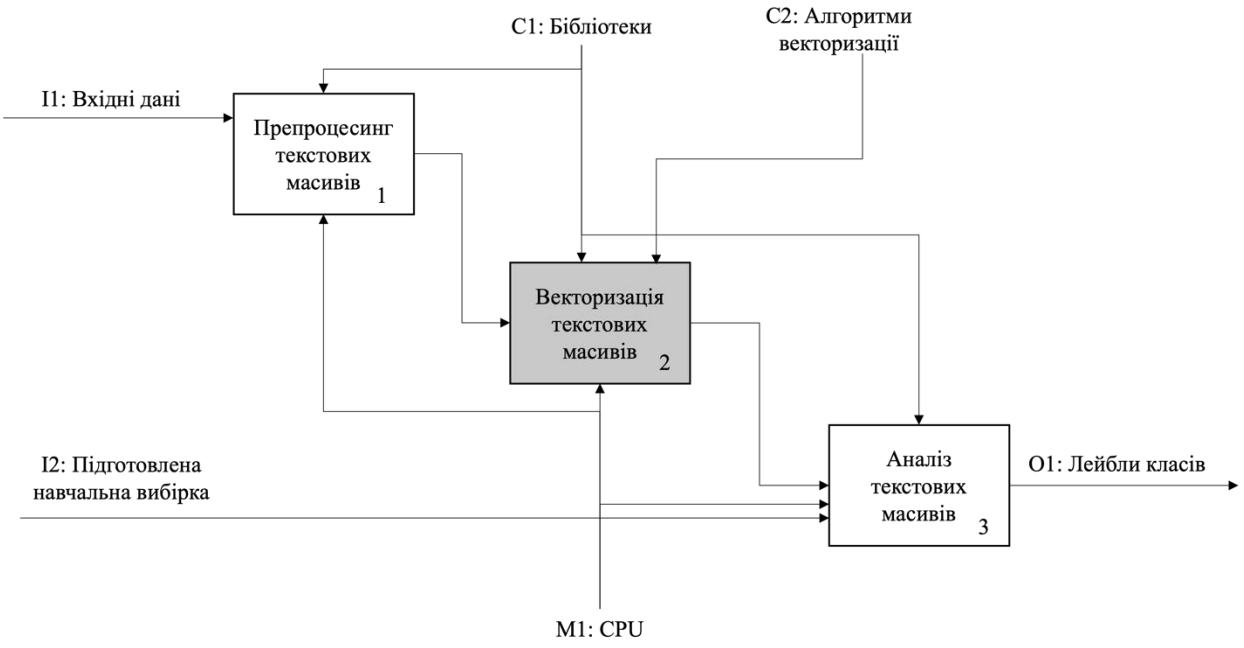

Рисунок 2.1 – Базові етапи класифікації текстових масивів

Відповідна архітектура може вплинути на ефективність і точність моделі в задачі класифікації тексту з різних точок зору. Архітектура нейронної мережі визначає, як вона вирішує проблему аналізу текстових даних. Різні архітектури можуть мати різні підходи до розпізнавання шаблонів, виявлення залежностей слів та інтерпретації текстової інформації. Правильно вибрана архітектура може допомогти вирішити проблеми в задачі класифікації тексту та забезпечити більш точні та надійні результати. Далі ми

порівняємо дві популярні моделі Convolutional Neural Network (CNN) та Long Short-Term Memory (LSTM), тому що ці мережі представляють два різних підходи до аналізу текстових даних і мають свої особливості, які можуть бути корисними для різних типів текстів і задач класифікації.

Таблиця 2.3 – Порівняльний аналіз моделей LSTM та CNN для класифікації тексту

Характеристика	LSTM Model	CNN Model
Основна архітектура	Рекурентна нейронна мережа зі шаром LSTM	Згорткова нейронна мережа зі шарами Conv1D
Типові завдання	Аналіз послідовних даних, текстові дані	Аналіз послідовних даних, зображення
Особливості	Враховує контекст залежностей слів	Визначає локальні шаблони в даних
Алгоритми навчання	Зворотнє поширення помилки, оптимізатор Adam	Зворотнє поширення помилки, оптимізатор Adam
Функції активації	Tanh, Sigmoid	ReLU
Використання пам'яті	Використовує короткотермінову та довготермінову пам'ять	Не використовує пам'ять
Застосування	Послідовний аналіз тексту, мовний переклад	Зображення, відеоаналіз
Бібліотеки реалізації	TensorFlow, Keras	TensorFlow, Keras

Модель CNN є унікальною та спеціалізованою для розпізнавання шаблонів у зображеннях, але її також можна успішно використовувати для обробки текстових даних, де вона розпізнає локальні залежності та важливі особливості текстового контексту. З іншого боку, модель LSTM є частиною

рекурентних нейронних мереж і має здатність зберігати та використовувати інформацію з попередніх кроків, що дозволяє ефективно працювати з послідовними даними, особливо текстовими послідовностями, і враховувати контекст у текстах.

Конволюційні нейронні мережі (CNN, Convolutional Neural Networks)[33,34] – це клас нейронних мереж, спеціально призначених для обробки даних, організованих у вигляді сітки або масиву, як отображається, наприклад, у зображеннях. CNN відзначаються високою ефективністю у завданнях комп'ютерного бачення, включаючи класифікацію зображень, виявлення об'єктів, сегментацію та багато інших. Основною ідеєю CNN є застосування фільтрів (які також називають ядрами) до вхідних даних з метою виділення корисних ознак. Ці фільтри рухаються по вхідному зображенню, виконуючи операцію згортки (convolution). Згортка полягає в обчисленні скалярного добутку між фільтром та частиною вхідних даних, що знаходиться під фільтром. Ця операція відбувається в різних місцях зображення, створюючи карту ознак (feature map) для кожного фільтру. Після застосування фільтрів до вхідних даних отримуємо карту ознак, яка відображає важливі особливості зображення. Потім до цих карт ознак застосовуються подальші шари, які виконують операції пулінгу (pooling) та активації. Пулінг дозволяє зменшити розмір карт ознак, а активація надає моделі нелінійність. CNN також можуть мати кілька шарів (зазвичай від 3 до 20 та більше) і містити багато фільтрів на кожному шарі. Це дозволяє виявляти більш складні ознаки та шаблони на різних рівнях абстракції. Після обробки всіма шарами, дані переводяться у вектор та подаються на повністю з'єднаний (fully connected) шар, де виконується класифікація або інші завдання.

Важливою характеристикою CNN є їх здатність до автоматичного вивчення корисних ознак і масштабовуваності на різних завданнях. Вони використовуються в багатьох важливих застосуваннях, включаючи розпізнавання зображень, обробку природних мов, рекомендаційні системи

та багато інших. При згортці CNN використовує фільтри для пошуку ознак на всьому зображенні, а не обмежується абсолютними координатами. Оскільки фільтри переміщуються по всьому зображенню, мережа може розпізнавати одні й ті ж ознаки, незалежно від їх розташування.

CNN в основному асоціюються з обробкою зображень, але вони також можуть бути використані для аналізу тексту. У такому контексті CNN використовуються для виділення ознак у текстових даних, подібно до того, як вони виділяють ознаки в зображеннях. Для використання CNN в аналізі тексту текстові дані зазвичай спершу конвертуються у векторну форму, наприклад, використовуючи вектори слів (word embeddings), такі як Word2Vec або GloVe. Текст подається у вигляді послідовності векторів слів, і CNN використовуються для виділення локальних ознак або шаблонів у цій послідовності.

Основні компоненти використання CNN для тексту включають:

1. Згортковий шар (Convolutional Layer): Згортковий шар використовує фільтри (ядра), що зсовуються по тексту для виділення різних ознак. Результатом цього шару є набір карт ознак (feature maps). Формула для згорткового шару:

$$f_i(x) = \sigma(W_i * x + b_i),$$

де $f_i(x)$ – і-та карта ознак;

σ – активаційна функція;

W_i – ваги для і-того фільтра;

X – вхідні дані;

b_i – зсув для і-того фільтра.

2. Пулінг (Pooling). Шари пулінгу дозволяють зменшити розмір вихідних даних, зберігаючи найважливіші ознаки. Це робить модель менш чутливою до малих зсувів в тексті. Формула для максимального пулінгу:

$$y_i = \max(x_i : k),$$

де y_i – вихідний обраний максимум;

$x_i : k$ – підмножина вхідних даних, яку ми розглядаємо (зазвичай з не перекриваються навполовину вікна).

3. Повністю з'єднаний шар (Fully Connected Layer). Після використання згортки та пулінгу, векторні ознаки подаються на повністю з'єднаний шар для класифікації або інших завдань. Формула для повнозв'язаного шару:

$$y = \sigma(W_x + b),$$

де y – вихідні дані класифікації;

σ – активаційна функція, зазвичай softmax для класифікації;

W – ваги шару;

x – вхідні дані, які можуть бути результатом роботи попередніх шарів;

b – зсув шару.

Це загальна структура для класифікації тексту за допомогою CNN. В реальних застосуваннях параметри та архітектура можуть відрізнятися в залежності від конкретного завдання.

Важливою перевагою використання CNN для тексту є їх здатність виявляти локальні ознаки та шаблони в тексті, а також їх здатність до автоматичного вивчення цих ознак з даних. Це особливо корисно для завдань, де текст має важливий контекст або де локальні ознаки грають важливу роль. Однак, слід зазначити, що для більш складних завдань обробки тексту, таких як розпізнавання сенсу, використання рекурентних

нейронних мереж (RNN) або трансформерів (Transformer) може бути більш ефективними, оскільки вони враховують послідовний контекст та взаємозв'язки між словами у тексті. Втім, CNN залишаються корисним інструментом для обробки тексту у багатьох випадках, особливо для завдань, де локальні ознаки грають ключову роль.

Лінійна рекурентна нейронна мережа (LSTM, Long Short-Term Memory) – це вид рекурентної нейронної мережі, розроблений для розв'язання проблеми зниклої (вицілпеної) градієнта в класичних рекурентних нейронних мережах. LSTM були запропоновані Юргеном Шмідгубером та Фредеріком Гагерстером в 1997 році і стали важливим кроком у розвитку рекурентних нейронних мереж. Основна ідея LSTM полягає в тому, щоб дати можливість мережі визначати, яку інформацію вона повинна зберегти та яку вона може забути, при роботі з послідовними даними. LSTM мають спеціальні внутрішні структури, що називаються "ворота" (gates), які регулюють потік інформації через мережу.

Основні компоненти LSTM включають наступне:

1. Ворота забуття (Forget Gate). Це ворота дозволяють мережі вирішувати, яку інформацію потрібно забути з минулого стану. Вони використовують сигмоїдальну функцію активації, яка вирішує, які значення мають бути забуті (близько 0) та які збережені (близько 1).

$$f_t = \sigma(W_f[h_t - 1, x_t] + b_f,$$

де σ – сигмоїдальна функція активації;

W_f – матриця ваг;

$h_t - 1$ – попередній вихід LSTM на кроці t-1;

x_t – вхід на кроці t;

b_f – зсув воріт забуття.

2. Ворота оновлення (Update Gate). Це ворота визначають, яку нову інформацію маємо додати до стану пам'яті. Вони використовують сигмоїдальну функцію для визначення, які значення оновити.

$$i_t = \sigma(W_i * [h_t - 1, x_t] + b_i),$$

де σ – сигмоїдальна функція активації;

W_i – матриця ваг;

$h_t - 1$ – попередній вихід LSTM на кроці t-1;

x_t – вхід на кроці t;

b_i – зсув воріт оновлення.

3. Стан пам'яті (Cell State). Це внутрішнє подання інформації, яке оновлюється та підтримується впродовж часу. Ворота забуття та оновлення регулюють, які значення додаються та віднімаються від стану пам'яті.

$$c_t = f_t * c_{t-1} + i_t * \tanh(W_c * [h_{t-1}, x_t] + b_c),$$

де f_t – вихід воріг забуття;

c_{t-1} – попередній стан пам'яті на кроці;

i_t – вихід воріт оновлення;

tanh – гіперболічний тангенс для створення нового вмісту.

4. Вихід (Output Gate). Це ворота вирішують, який вміст стану пам'яті буде виведений з LSTM. Вони використовують сигмоїдальну функцію для визначення виходу та гіперболічний тангенс для створення виходу.

Важливою перевагою LSTM є їх здатність зберігати та використовувати інформацію на довгий строк, що робить їх особливо корисними для обробки послідовних даних, таких як мова, музика або часові ряди. LSTM здатні враховувати контекст на довгий строк та автоматично

вивчати корисні залежності у даних. Однією з менш очевидних, але важливих характеристик LSTM є їхнє використання для вирішення проблеми зниклої (вибухової) градієнта, яка є однією з основних проблем у класичних рекурентних нейронних мережах (RNN). В LSTM використовується спеціальна структура з воротами, що дозволяє тримати та передавати градієнти на більшу кількість кроків часу назад, і це робить їх особливо ефективними для навчання на довгих послідовностях. Лінійні рекурентні нейронні мережі (RNN) мають тенденцію до втрати градієнта під час навчання на великих послідовностях, що робить їхнє навчання важким. Однак в LSTM ворота дозволяють контролювати, яка частина градієнту зберігається та передається далі, що дозволяє моделі навчатися на довгих послідовностях і ефективно передавати інформацію на велику кількість кроків назад. Ця властивість LSTM робить їх особливо корисними для завдань, де важлива здатність моделі аналізувати контекст на довгий строк та розуміти довгострокові залежності в даних. Наприклад, в мовному моделюванні, LSTM можуть допомагати моделі розуміти довгострокові залежності між словами в тексті, що дозволяє покращити якість генерації тексту та розпізнавання мови.

Отже, важливість вирішення проблеми зниклої градієнта та збереження градієнтів на велику кількість кроків назад робить LSTM потужним інструментом у глибокому навчанні для аналізу послідовних даних.

Використання моделі LSTM для обробки тексту має свої особливості:

- робота з послідовностями. LSTM ідеально підходять для обробки послідовних даних, таких як тексти, де важливий контекст та залежності між словами. Вони можуть ефективно моделювати послідовності та розуміти довгострокові зв'язки в тексті;

- здатність до рекурентності. LSTM мають рекурентну структуру, що дозволяє їм пам'ятати попередні стани та враховувати їх при аналізі нових даних. Це особливо корисно для аналізу тексту, де слова в контексті можуть мати різний сенс;

- аналіз довгострокових залежностей. LSTM здатні розуміти довгострокові залежності в тексті, що робить їх ефективними для завдань, де важлива історія та контекст. Наприклад, в мовному моделюванні важливо аналізувати, як попередні слова впливають на вибір наступного слова;

- класифікація тексту. LSTM можуть використовуватися для класифікації тексту, де потрібно визначити до якої категорії або класу належить текст. Наприклад, визначення сентименту (позитивний, негативний, нейтральний) у відгуках користувачів;

- генерація тексту. LSTM можуть бути використані для генерації тексту, де модель створює нові тексти, які мають подібний стиль або контекст до вхідних даних. Це може бути корисним для створення текстів, наприклад, генерації новинних статей;

- аналіз часових рядів у тексті. LSTM можуть бути використані для аналізу часових рядів у тексті, таких як прогнозування та розпізнавання подій у великих текстах або новинах;

- робота з нейронними векторами слів. LSTM можуть використовувати векторні представлення слів, такі як Word2Vec або GloVe, для кращого розуміння семантики слів у тексті.

Загалом, використання LSTM для обробки тексту дає можливість ефективно аналізувати послідовності слів, враховувати контекст та розуміти складні залежності, що робить їх потужним інструментом для багатьох завдань у сфері обробки тексту та аналізу послідовностей.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ

3.1 Визначення моделі та попередня обробка даних

3.1.1 Функціональна модель класифікації текстових масивів

У цьому розділі ми фокусуємося на розробці функціональної моделі, яка є основою нашого дослідження в області класифікації текстів з використанням контекстуальних векторних представлень. Застосування IDEF0 дозволяє нам структуровано представити процеси та відношення між різними елементами системи. Цей підхід сприяє чіткому визначенню функцій системи, а також подальшому аналізу та вдосконаленню процесу класифікації. Враховуючи складність сучасних NLP задач, ефективна функціональна модель є ключовим елементом для досягнення високої точності та надійності результатів. Модель представлена за допомогою IDEF0 діаграми, що візуалізує основні функціональні процеси та їх взаємозв'язки (рисунок 3.1).

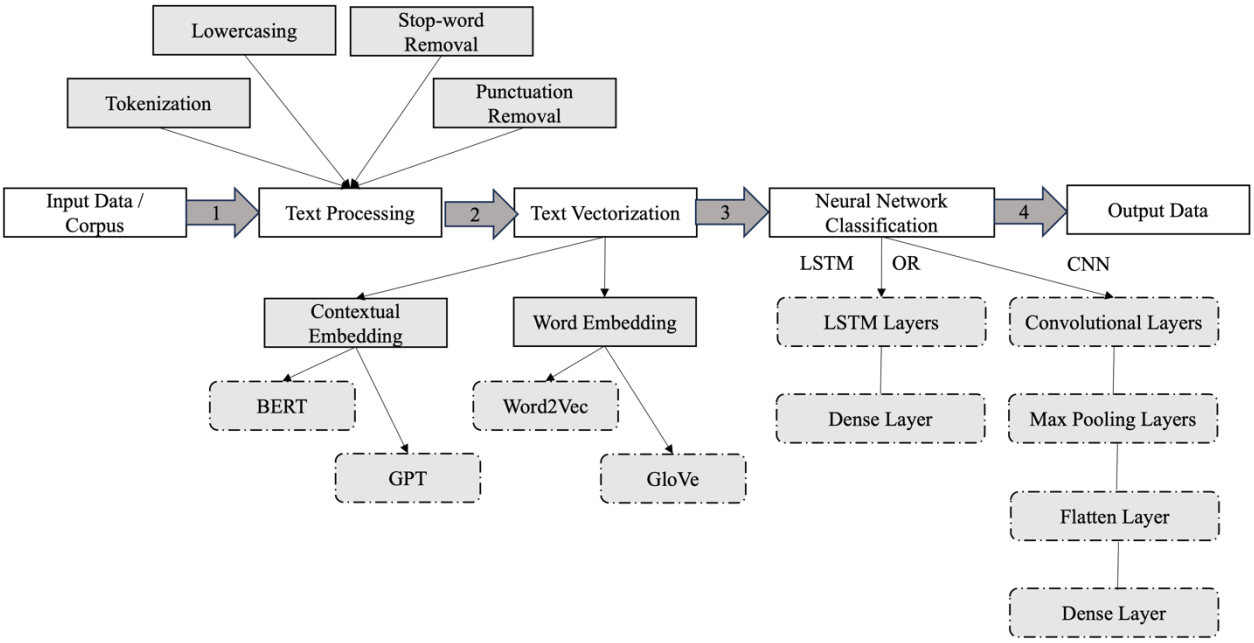

Рисунок 3.1 – Визначення факторів дослідження

Функціональна модель складається з:

- масив текстових документів, який слугує первинним джерелом даних для обробки;
- використання ембедингів для трансформації нормалізованих текстових даних у векторний простір;
- передпроцесинг, що включає нормалізацію та токенізацію, для забезпечення ефективної векторизації;
- класифікація (аналіз за допомогою LSTM-мережі). Застосування моделі глибокого навчання для визначення категорій текстових даних;
- матриця експериментів (Виходи). Систематизація результатів класифікації для оцінювання та порівняння різних моделей та підходів;
- застосування спеціалізованого програмного забезпечення та обладнання для ефективної роботи моделі (CPU, бібліотеки) ;
- оцінка результатів.

3.1.2 Опис компонентів функціональної моделі класифікації текстових масивів

Вхідні дані. У цій частині ми зосередимося на первинних даних, які служать вхідними даними для процесу класифікації тексту в нашому дослідженні. Використаний набір даних IMDB є добре відомим і загальнодоступним ресурсом, який зазвичай використовується для обробки природної мови та завдань аналізу настроїв. Цей набір даних містить 50 000 відгуків, рівномірно розподілених між позитивними та негативними категоріями, що робить його ідеальним для навчання та оцінювання моделей класифікації. Позитивні відгуки оцінюються 7 з 10 або вище, а негативні – 4 з 10 або нижче. Така розмітка забезпечує помітну різницю в настроях, що сприяє точності алгоритмів класифікації.

Крім того, важливо зазначити, що набір даних містить лише текст рецензій без додаткової метаінформації, такої як рейтинги фільмів або дані

рецензій, що дозволяє нам зосередитися безпосередньо на аналізі тексту. Відгуки можуть відрізнятися за розміром і надавати широкий спектр даних для аналізу, від коротких коментарів до детальних оглядів.

Вибір саме цього набору даних обумовлений його збалансованістю, великою кількістю даних і широким використанням у сфері обробки природної мови, що дозволяє співвіднести отримані результати з існуючими дослідженнями в цій галузі.

Передпроцесинг текстових масивів відіграє життєво важливу роль у підготовці до аналізу та класифікації. У моєму дослідженні було застосовано ретельно спланований процес передпроцесингу, який охоплював нормалізацію та токенізацію відгуків з IMDB датасету. Нормалізація включала перетворення всього тексту в нижній регістр та видалення зайвих символів та знаків пунктуації, що не несуть важливої інформативної навантаження. Цей крок забезпечував однорідність даних та спрощував подальше використання текстів для машинного навчання.

Після нормалізації наступним етапом була токенізація – процес розділення тексту на індивідуальні елементи, або токени. Токени, зазвичай слова, були виділені з тексту для побудови вокабуляру, який потім використовувався для векторизації. Ключовим моментом у процесі токенізації було забезпечення того, щоб кожне слово або фраза були акуратно розділені, що дозволило створити чітку мапу для векторизації.

Завершальним кроком передпроцесингу було видалення стоп-слів – найбільш часто вживаних слів, які не несуть значного семантичного навантаження, таких як 'і', 'але', та 'якщо'. Видалення цих слів дозволило зменшити шум у даних та зосередити увагу на важливіших словах для аналізу сентименту.

Цей підхід до передпроцесингу забезпечив чистий та структурований набір даних, готовий для наступних етапів аналізу та класифікації у рамках моєї кваліфікафійної роботи.

Векторизація та ембедінги. Важливою складовою мого дослідження є процес перетворення текстових даних у формат, придатний для машинного навчання, що здійснюється через векторизацію. У моїй роботі основний акцент було зроблено на використанні моделей Word Embedding, зокрема Word2Vec та GloVe, а також на застосуванні контекстуальних ембедінгів за допомогою таких технологій, як BERT і GPT.

Word2Vec та GloVe є інструментами для створення ембедінгів, які вловлюють семантичні відношення між словами, генеруючи їх векторні представлення на основі структурних відношень у великих текстових корпусах. В той час як Word2Vec використовує локальний контекст навколишніх слів, GloVe заснований на глобальних статистиках співвживання слова у всьому корпусі.

BERT та GPT представляють собою розвиток концепції ембедінгів, вони здатні генерувати контекстуальні вектори для слів, враховуючи всі взаємозв'язки в рамках цілого речення або навіть більших текстових фрагментів. Це забезпечує більш точне та глибоке розуміння мовних структур, що, в свою чергу, підвищує точність моделей класифікації текстів.

Програмна реалізація цих методів, їх застосування та оптимізація будуть детально описані в наступному розділі, де я надам код та обговорення використаних бібліотек і інструментів. Це дозволить нам відобразити всю глибину використаних підходів та підкреслити їх значимість для досягнення мети дослідження.

Генеральна LSTM модель. Для задачі класифікації тексту була обрана модель LSTM (Long Short-Term Memory). LSTM є різновидом рекурентних нейронних мереж, які ефективно використовуються для аналізу послідовностей даних, таких як текст. У практичному огляді архітектури моделі LSTM для класифікації тексту, ми виходимо з принципу, що ефективність моделі визначається не тільки вибором архітектури, але і її реалізацією.

Для початку, використовуючи методи ембедингів, кожне слово у корпусі тексту перетворюється на вектор фіксованої довжини. Отримана матриця ембедингів передається як вхідні дані для LSTM мережі. Структура моделі включає наступні шари:

- перший шар у архітектурі – LSTM, який має 100 одиниць. Це дозволяє ефективно зловлювати залежності в текстових послідовностях з достатньою деталізацією. Використання гіперболічної тангенсальної активаційної функції (tanh) сприяє підтримці нелінійності у внутрішніх структурах даних. Цей шар включає вентилі входу, забуття та виходу, а також внутрішній стан (cell state), що забезпечує моделі здатність запам'ятовувати інформацію на довші періоди та ефективніше працювати з послідовностями;

- після LSTM шару використовується повнозв'язний шар (Dense Layer) з однією одиницею та сигмоїдною активаційною функцією для виведення остаточного прогнозу. Сигмоїдна функція ідеально підходить для бінарної класифікації, оскільки вона виводить значення між 0 та 1, що інтерпретується як ймовірність належності до певного класу;

- модель компілюється з використанням оптимізатора adam, який ефективно працює для широкого спектру задач, та функції втрат binary_crossentropy, яка є стандартним вибором для бінарної класифікації. Як метрику визначено accuracy, яка оцінює точність класифікації.

Архітектура LSTM-моделі була реалізована у Python з використанням бібліотеки Keras (лістинг 3.1).

Лістинг 3.1 – Реалізація LSTM-моделі з використанням Keras

```
model = Sequential()
model.add(LSTM(units=100, activation='tanh',
input_shape=(max_length, embedding_dim)))
model.add(Dense(units=1, activation='sigmoid'))
model.compile(optimizer='adam', loss='binary_crossentropy',
metrics=['accuracy'])
history = model.fit(X_train_vec, y_train, epochs=10,
validation_data=(X_test_vec, y_test), batch_size=64)
```

Структура LSTM моделі була обрана з огляду на специфіку датасету IMDB. Embedding шар з незмінними вагами дозволяє зберігати багатство контекстуальної інформації, яку надають ембединги, і ефективно втілює важливість слів у векторному просторі. LSTM шари із 100 вузлами забезпечують глибоке розуміння послідовних залежностей в тексті, що є критично важливим для точного аналізу відгуків. Dense шар з активацією 'sigmoid' дозволяє досягнути чіткої класифікації. Всі ці елементи спільно формують модель, оптимально підходящу для задачі аналізу сентименту великих текстових масивів (рисунок 3.2), в нашому випадку – для задачі бінарної класифікації (позитивний, негативний) відгуків на фільми.

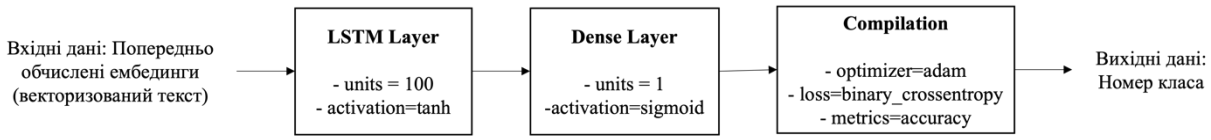

Рисунок 3.2 – Архітектура LSTM моделі

3.1.3 Порядок декомпозиції датасета

У моєму дослідженні, важливим аспектом була правильна організація та розподіл IMDB датасету для різних етапів процесу машинного навчання. Датасет був поділений на три основні частини: тренувальну, валідаційну, та тестову:

- тренувальний набір даних (70%). Більша частина датасету, яка використовувалася для тренування LSTM моделі. Застосування значної частини даних для тренування забезпечує моделі достатньо прикладів для ефективного вивчення та адаптації до різноманітності текстових відгуків;

- валідаційний набір даних (15%). Використовувався для періодичної оцінки моделі під час тренування. Це дозволяло моніторити і попереджати перенавчання моделі, а також допомагало в тонкій настройці гіперпараметрів;

- тестовий набір даних (15%). Використовувався для оцінки кінцевої точності моделі після тренування. Такий розподіл допомагає забезпечити об'єктивність оцінки ефективності моделі, оскільки тестування проводиться на даних, які не використовувались під час тренування;

Розподіл датасету на тренувальний, валідаційний, та тестовий набори даних має вирішальне значення для точності та загальної ефективності моделі класифікації текстів. Великий обсяг даних для тренування дає моделі можливість "вивчити" та адаптуватися до широкого спектру мовних особливостей та варіацій, що знаходяться у текстах. Це сприяє формуванню більш точної та гнучкої моделі, здатної розпізнавати суттєві відмінності в сентиментах. Використання окремого валідаційного набору дозволяє періодично оцінювати ефективність моделі без ризику її перенавчання. Це допомагає уникнути "запам'ятовування" даних замість "вивчення" їх, що є важливим для забезпечення загальної точності моделі. Використання тестового набору, який не брав участі у тренуванні моделі, є вирішальним для об'єктивної оцінки її точності. Такий набір даних надає чітке уявлення про те, як модель буде працювати з новими, невідомими даними, мімікуючи реальні сценарії використання.

3.1.4 Формування матриці експериментів

У цьому розділі ми зосередилися на розробці систематичного підходу до порівняння чотирьох типів векторних представлень слів: Word2Vec та GloVe для Word Embedding, а також BERT та GPT для Contextual Embedding. Метою було створити умови для об'єктивного аналізу та порівняння цих методів щодо їх ефективності у класифікації тексту.

Для цього ми сформували матрицю експериментів, в якій кожен метод ембедингу був випробуваний із заздалегідь визначеною LSTM-архітектурою, що дозволяло нам зіставляти результати в однакових умовах (таблиця 3.1).

Таблиця 3.1 – Матриця планування багатофакторного експерименту для визначення впливу кількості епох навчання для обраних типів векторизації текстових масивів на точність класифікації вхідних даних в межах Word Embedding або Contextual Embedding.

Номер експерименту	Фактори впливу			Цілі, для яких оцінюються залежності			
	Варіативна кількість епох навчання	Метод векторизації текстових масивів № 1	Метод векторизації текстових масивів № 2	Train accuracy	Validation accuracy	Validation loss	Training time
1	1	+	-	+	+	+	+
2	1	-	+	+	+	+	+
3	2	+	-	+	+	+	+
4	2	-	+	+	+	+	+
…	…	…	…	…	…	…	…
2N-1	N	+	-	+	+	+	+
2N	N	-	+	+	+	+	+

Метою виконання експериментів є визначення методу векторизації текстових масивів та кількості епох навчання нейромережевої моделі класифікації тексту, для яких буде досягнуто найбільші показники з:

- train accuracy (показує, наскільки ефективно модель класифікує дані під час навчання.);
- validation accuracy (показує, наскільки ефективно модель класифікує нові дані не бачені під час тренування.);
- validation loss (показує середній рівень помилки моделі на валідаційному наборі даних.);
- training time (показує загальний час, витрачений на навчання моделі.).

Кількість експериментів не є визначеною завчасно та залежить від кількості епох N, починаючи з якої валідаційна точність класифікації почне зменшуватися.

У рамках формування матриці ми спочатку розглянули порівняння Word2Vec і GloVe. Ми визначили, які параметри будуть варіюватися та які

залишаться незмінними, а також які метрики будуть використовуватися для оцінки кожного ембедингу. Потім ми аналогічно підійшли до порівняння BERT і GPT.

Кожен етап порівняння був ретельно планований, щоб забезпечити рівні умови для всіх ембедингів. Ми приділили особливу увагу контролю над варіабельністю експериментальних умов, забезпечуючи що різниця в результатах відображатиме лише властивості методів ембедингів, а не зовнішні фактори.

Заключний етап полягав у розробці процедури порівняння найкращих представників кожної групи, що дозволило б нам визначити найефективніший метод векторного представлення для нашої задачі. Цей підхід забезпечує систематичний та науково обґрунтований процес порівняння, що є ключовим для досягнення валідних висновків у дослідженні

3.2 Програмне підтвердження виконання дослідження

Програмна реалізація векторного представлення слів є фундаментом для побудови ефективних моделей машинного навчання в галузі обробки природної мови. Вона дозволяє перетворити текст у числовий формат, оптимальний для алгоритмів глибокого навчання. Векторизація забезпечує не лише поверхневий аналіз тексту, але й уможливлює зрозуміння мови на більш глибокому рівні, враховуючи контекст та семантику.

В нашій роботі ми застосували методики Word2Vec і GloVe, які вимагають ретельного підходу до тренування моделей на обширних текстових корпусах. Процес включає в себе підготовку даних, їхню токенізацію та очищення, що готує платформу для ефективного тренування. Векторизація з використанням цих методів створює представлення слів, які відображають їхні семантичні значення. Для інтеграції Word2Vec ми використали бібліотеку Gensim та налаштували параметри, щоб оптимізувати модель (лістинг 3.2). Після тренування моделі, отримані вектори були

використані для аналізу тексту і наступно інтегровані у LSTM архітектуру для класифікації тексту.

Лістинг 3.2 – Реалізація Word2Vec

```
from gensim.models import Word2Vec
tokenized_text = [review.split() for review in
train_data['review']]
word2vec_model = Word2Vec(sentences=tokenized_text,
vector_size=100, window=5, min_count=1, workers=4)
```

Також ми інтегрували GloVe, використовуючи ваги з предтренованих векторів, щоб створити матрицю ембедингів, яка потім була використана як перший шар у моделі нейронної мережі (лістинг 3.3).

Лістинг 3.3 – Реалізація GloVe

```
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import
pad_sequences
tokenizer = Tokenizer()
tokenizer.fit_on_texts(train_data['review'])
X_train = tokenizer.texts_to_sequences(train_data['review'])
X_train = pad_sequences(X_train, maxlen=max_length)
```

Контекстуальні ембединги, такі як BERT та GPT, забезпечують ще глибший аналіз тексту, враховуючи двонаправлений контекст. Вони відображають складні відносини та нюанси мови, що не досягається з традиційними статичними ембедингами.

BERT був застосований за допомогою бібліотеки transformers, де BertTokenizer перетворив вхідні дані на формат, придатний для моделі, та BertModel, який повернув високорівневі векторні представлення (лістинг 3.4).

Лістинг 3.4 – Реалізація BERT

```
from transformers import BertTokenizer, BertModel
tokenizer = BertTokenizer.from_pretrained("bert-base-uncased")
model = BertModel.from_pretrained("bert-base-uncased")
inputs = tokenizer(review, return_tensors="tf", truncation=True,
padding=True)
outputs = model(**inputs)
```

Аналогічно, GPT-2 був інтегрований для отримання високорівневих векторних представлень, які покращують точність класифікації тексту (лістинг 3.5).

Лістинг 3.5 – Реалізація GPT-2

```
from transformers import GPT2Tokenizer, GPT2Model
tokenizer = GPT2Tokenizer.from_pretrained("gpt2")
model = GPT2Model.from_pretrained("gpt2")
inputs = tokenizer.encode_plus(review, return_tensors="tf",
add_special_tokens=True)
outputs = model(inputs["input_ids"])
```

Ці методики та програмні реалізації були важливими для формування нашої експериментальної матриці, яка дозволила нам систематично порівнювати ефективність різних видів ембедингів у задачі класифікації тексту.

4 АНАЛІЗ РЕЗУЛЬТАТІВ ДОСЛІДЖЕНЬ

У цьому розділі ми зосереджуємося на аналізі результатів експериментів, проведених з метою оцінки ефективності різних методів векторизації слів – Word2Vec, GloVe, BERT та GPT - у контексті класифікації тексту. Основна увага приділяється вивченню впливу кількості епох навчання на точність цих моделей. Цей підхід дозволить нам не тільки оцінити загальну продуктивність кожного методу, але й розуміння їхньої оптимальної конфігурації для досягнення найкращих результатів.

Методологія дослідження базується на сформованій матриці експериментів, що включає систематичне порівняння ефективності кожного виду ембедингу. Кожна модель була оцінена на основі кількох ключових метрик, включаючи точність (accuracy), матрицю помилок (confusion matrix), F1-міру та ROC-AUC Score. Зокрема, ми аналізували як статичні, так і контекстуальні ембединги, щоб зрозуміти їхні переваги та обмеження у контексті задачі класифікації тексту. Такий підхід дозволяє нам отримати цілісне уявлення про вплив різних типів векторизації на результати моделей.

4.1 Аналіз впливу кількості епох на точність моделей та інші метрики

В рамках дослідження використовувалися різні критерії для оцінки ефективності моделей, що включають точність (accuracy), втрати (loss), валідаційні втрати (val_loss), валідаційну точність (val_accuracy), а також розширені метрики, такі як ROC-AUC Score, матриця помилок, precision, recall і F1-score. Ці метрики разом дозволяють зробити всеохоплюючий аналіз моделей, оцінюючи не тільки їхню здатність правильно класифікувати дані, але й їхню стабільність та надійність у різних умовах.

Дослідження впливу кількості епох на моделі включало систематичне збільшення числа епох тренування та аналіз зміни вищезгаданих метрик. Це

дозволило оцінити, як тривалість навчання моделі впливає на її здатність загальної класифікації та специфічних характеристик, таких як здатність правильно ідентифікувати кожен клас (precision та recall) та баланс між ними (F1-score).

Очікувалося, що з підвищенням кількості епох загальна точність моделі покращиться, але лише до певної межі, після чого можливе перенавчання. Також передбачалося, що взаємодія між різними типами ембедингів та довжиною тренування може мати різний вплив на інші метрики, такі як precision, recall, та F1-score. Було важливо виявити оптимальне співвідношення між тривалістю тренування та якістю моделі, враховуючи всі важливі метрики.

4.2 Порівняльний аналіз Word Embedding моделей

Для виконання аналізу впливу таких Word Embedding моделей, як Word2Vec та GloVe, на точність класифікації тек тових масивів, було виконано ряд екпериментів. Для кожного методу була розроблена окрема нейронна мережа з використанням LSTM архітектури.

Основними параметрами, що налаштовувалися, були розмір вектора, кількість епох та структура мережі. Метою було визначити, як впливає кількість епох навчання нейронної мережі на точність, швидкість навчання, валідаційні втрати та валідаційну точність для різних методів векторизації (Word2Vec та GloVe) в рамках підходу Word Embedding.

Таблиця надає детальний огляд змін точності, валідаційної точності та валідаційних втрат для кожного методу Word Embedding протягом 10 епох тренування. Вона дозволяє порівняти, як кожен метод розвивається з плином часу та які особливості має кожен з них у процесі тренування (таблиця 4.1).

Таблиця 4.1 – Таблиця метрик по кожній з 10 епох для Word2Vec та GloVe

Epoch	Word2Vec Accuracy, %	Word2Vec Val Accuracy, %	Word2Vec Val Loss	GloVe Accuracy, %	GloVe Val Accuracy, %	GloVe Val Loss
1	84,94	88,77	0.3037	76,97	82,33	0.4019
2	94,10	88,19	0.2963	83,79	85,41	0.3406
3	96,82	86,80	0.3543	86,53	86,43	0.3227
4	97,02	87,40	0.4230	87,76	87,06	0.3077
5	98,94	87,13	0.5181	88,77	86,61	0.3308
6	99,22	86,47	0.6262	89,54	87,30	0.3056
7	99,26	83,90	0.7275	90,63	87,98	0.3073
8	98,74	85,93	0.5562	91,23	87,73	0.3033
9	99,37	86,55	0.5966	92,12	86,86	0.3273
10	**99,75**	**88,77**	**0.7497**	**93,17**	**87,72**	**0.3166**
15	99,75	86,03	0.8173	93,00	87,21	0.3392

У даному варіанті тренування моделей ми спостерігаємо спад точності на валідаційному наборі та збільшення втрат, це може свідчити про перенавчання моделі після 10-ої епохи тренування. Перенавчання виникає, коли модель стає надто адаптованою до навчальних даних і втрачає здатність до загальноїлізації на нових даних. У цьому контексті, результати для 15-ої епохи показують, що подальше тренування не призводить до поліпшення якості моделей, а навпаки, може погіршити їхню здатність до узагальнення на нові дані.

Отже, було зупинено тренування моделей після 10-ої епохи, оскільки ця точка вже вказує на досягнення гарної ефективності. Додаткові епохи можуть вести до перенавчання і збільшення обчислювальних витрат без покращення результатів.

Після визначення необхідної кількості епох навчання, на яких досягається найвища валідаційна точність, була виконана оцінка наступних показників: тестова точність, ROC-AUC Score, precision, recall та F1-score для кожного методу векторизації тексту (таблиця 4.2).

Таблиця 4.2 – Таблиця результатів експериментів

Embedding Type	Word2Vec	GloVe
Number of Epochs	10	10
Test Accuracy, %	86,87	87,72
Test Loss	0.7497	0.3166
ROC-AUC Score	0.9333	0.9457
Precision (avg)	0.87	0.88
Recall (avg)	0.87	0.88
F1-Score (avg)	0.87	0.88
Training Time, sec	6340	3830

Для візуалізації динаміки навчання моделей з використанням методів Word2Vec та GloVe було сконструйовано графіки зміни втрат протягом усіх епох тренування. На додаток до табличних даних, графік забезпечує наглядне порівняння між ефективністю обох методів ембедингу. Графік показує втрати на тренувальному наборі даних (Train Loss) та на валідаційному наборі (Test Loss), що дозволяє оцінити перенавчання моделі та її загальну стабільність. З цього графіку можна зробити висновок про тенденції зміни втрат моделей, що є важливим показником для аналізу та оптимізації процесу навчання (рисунок 4.1 та рисунок 4.2).

Рисунок 4.1 – Тенденції втрати при навчанні та тестуванні моделі Word2Vec

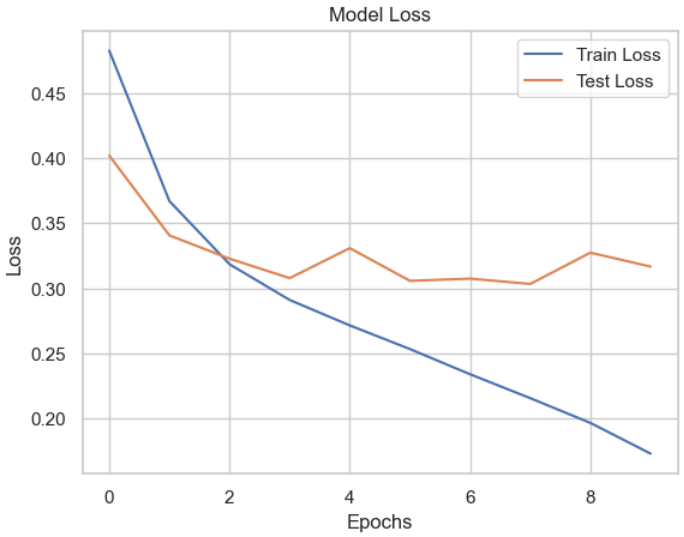

Рисунок 4.2 – Тенденції втрати при навчанні та тестуванні моделі GloVe

Аналіз результатів показав, що кожен метод має свої переваги та недоліки. Word2Vec продемонстрував високу точність, але з певним рівнем втрат, особливо у пізніших епохах, що може свідчити про перенавчання. GloVe, з іншого боку, показав більш стабільні результати на валідаційному наборі даних, хоча його загальна точність була дещо нижчою порівняно з Word2Vec. Ці відмінності підкреслюють важливість вибору відповідного методу векторизації з урахуванням конкретної задачі та набору даних.

В цілому, експерименти підтвердили, що обидва методи є ефективними для задачі класифікації тексту, але їх вибір має базуватися на конкретних потребах та характеристиках проекту.

4.3 Порівняльний Аналіз Contextual Embedding Моделей

Для виконання аналізу впливу таких Contextual Embedding моделей, як BERT та GPT, на точність класифікації текстових масивів, було виконано ряд екпериментів. Для кожного методу була розроблена окрема нейронна мережа з використанням LSTM архітектури.

Основними параметрами, що налаштовувалися, були розмір вектора, кількість епох та структура мережі. Метою було визначити, як впливає кількість епох навчання нейронної мережі на точність, швидкість навчання, валідаційні втрати та валідаційну точність для різних методів векторизації (BERT та GPT) в рамках підходу Contextual Embedding.

Таблиця надає детальний огляд змін точності, валідаційної точності та валідаційних втрат для кожного методу Contextual Embedding протягом 10 епох тренування. Вона дозволяє порівняти, як кожен метод розвивається з плином часу та які особливості має кожен з них у процесі тренування (таблиця 4.3).

Таблиця 4.3 – Таблиця метрик по кожній з 10 епох для BERT та GPT

Epoch	BERT Accuracy, %	BERT Val Accuracy, %	BERT Val Loss	GPT Accuracy, %	GPT Val Accuracy, %	GPT Val Loss
1	87,62	90,14	0.2601	85,50	88,92	0.2987
2	91,53	90,98	0.2427	88,04	89,27	0.2789
3	93,27	91,45	0.2285	89,31	89,84	0.2623

Продовження таблиці 4.3

4	94,52	91,82	0.2204	90,28	90,29	0.2507
5	95,58	92,07	0.2153	91,12	90,65	0.2421
6	96,41	92,31	0.2135	91,76	90,97	0.2365
7	97,12	92,56	0.2141	92,39	91,20	0.2320
8	97,65	92,79	0.2168	92,87	91,43	0.2295
9	**98,14**	**92,97**	**0.2212**	**93,34**	**91,65**	**0.2281**
10	98,91	91,23	0.2919	97,79	89,45	0.3851

У даному варіанті тренування моделей ми спостерігаємо спад точності на валідаційному наборі та збільшення втрат, це може свідчити про перенавчання моделі після 9-ої епохи тренування. Результати для 10-ої епохи показують, що подальше тренування не призводить до поліпшення якості моделей, а навпаки, може погіршити їхню здатність до узагальнення на нові дані.

Отже, було зупинено тренування моделей після 9-ої епохи, оскільки ця точка вже вказує на досягнення гарної ефективності. Додаткові епохи можуть вести до перенавчання і збільшення обчислювальних витрат без покращення результатів.

Після визначення необхідної кількості епох навчання, на яких досягається найвища валідаційна точність, була виконана оцінка наступних показників: тестова точність, ROC-AUC Score, precision, recall та F1-score для кожного методу векторизації тексту (таблиця 4.4).

Таблиця 4.4 – Таблиця результатів експериментів

Embedding Type	BERT	GPT
Epochs	9	9
Test Accuracy, %	91,23	89,45
Test Loss	0.2919	0.3851

Продовження таблиці 4.3

ROC-AUC Score	0.951	0.939
Precision (avg)	0.91	0.89
Recall (avg)	0.91	0.90
F1-Score (avg)	0.91	0.89
Training Time, sec	11340	14030

Для ілюстрації прогресу навчання моделей BERT та GPT були створені графіки, які відображають зміни рівня втрат на протязі всіх епох. Ці графіки доповнюють дані таблиць, забезпечуючи більш наочний зіставлення між ефективністю цих передових методів контекстуалізації. Вони візуалізують втрати як на етапі тренування, так і на етапі валідації, дозволяючи виявити можливості перенавчання моделей та оцінити їхню надійність протягом усього процесу навчання. Завдяки цим графікам можна візуально оцінити та аналізувати тенденції у змінах втрат, що є ключовим аспектом для покращення стратегій навчання (рисунок 4.3 та рисунок 4.4).

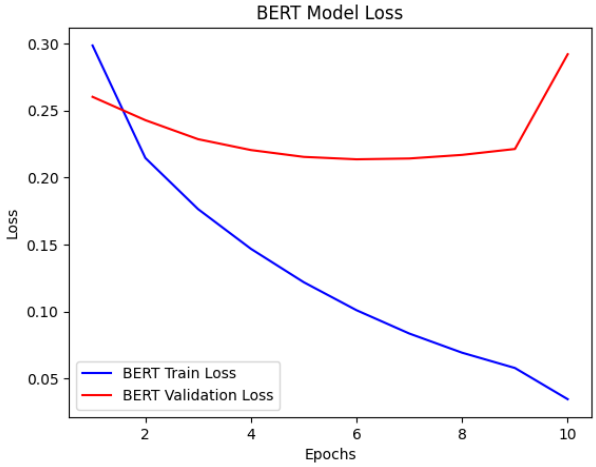

Рисунок 4.3 – Тенденції втрати при навчанні та тестуванні моделі BERT

Аналіз результатів для BERT та GPT демонструє різні аспекти ефективності цих моделей у класифікації тексту. BERT, із кінцевою точністю 0.9123 та відносно низькими втратами 0.2919, показав високу здатність до

аналізу та розуміння контексту тексту. Його високий ROC-AUC Score 0.951 свідчить про вдосконалене розуміння нюансів мови. З іншого боку, GPT, з кінцевою точністю 0.8945 та втратами 0.3851, виявився трохи менш точним, але все ще ефективним, особливо в контексті генерації тексту та обробки більш творчих задач.

Рисунок 4.4 – Тенденції втрати при навчанні та тестуванні моделі GPT

В цілому, обидві моделі демонструють значну ефективність у класифікації тексту, проте вибір між ними залежить від специфіки задачі. BERT є ідеальним для задач, де потрібно глибоке розуміння мови, тоді як GPT може бути більш вигідним у ситуаціях, що вимагають генеративного підходу. Важливо розуміти ці ключові відмінності, щоб правильно застосувати кожну модель у відповідності до конкретних потреб проекту.

4.4 Зіставлення найкращих Word та Contextual Embedding моделей

У рамках моєї кваліфікаційної роботи для визначення найкращих моделей у категоріях Word та Contextual Embedding, основними критеріями були точність, втрати, ROC-AUC Score та стабільність результатів. Точність моделі, як на тренувальному, так і на валідаційному наборі даних, вважалася одним із вирішальних факторів, оскільки висока точність вказує на

ефективність моделі у класифікації тексту. Втрати моделі під час тренувань та валідації також враховувались, оскільки низькі втрати свідчать про кращу здатність моделі адаптуватися та узагальнювати інформацію. ROC-AUC Score, який оцінює здатність моделі розрізняти між класами, також враховувався як ключовий показник продуктивності. Останнім, але не менш важливим, була стабільність моделі, яка вимірювалась здатністю зберігати постійні результати на різних наборах даних, що є критично важливим для оцінки її надійності та практичності у реальних умовах. Ці критерії дозволили оцінити та порівняти моделі, виходячи з їх функціональності та ефективності в задачах класифікації тексту (таблиця 4.5).

Таблиця 4.5 – Таблиця результатів експериментів

Embedding Type	Word Embedding (GloVe)	Contextual Embedding (BERT)
Test Accuracy, %	87.72	91.23

Цікавим аспектом у порівнянні цих моделей є те, що, хоча GloVe має нижчу точність порівняно з BERT, вона все ж показала вражаючі результати, особливо з огляду на те, що вона є менш складною та вимагає менших обчислювальних ресурсів. З іншого боку, BERT, з його високою точністю та низькими втратами, підтвердив свою ефективність у складніших задачах NLP, де необхідне глибше розуміння контексту та мовних нюансів.

В цілому, порівняння цих моделей вказує на те, що вибір між ними залежить від специфіки задачі та доступних ресурсів. GloVe відмінно підходить для задач, де потрібна баланс між точністю та обчислювальною ефективністю, тоді як BERT є кращим вибором для складніших задач обробки мови, де важливіше глибоке розуміння тексту.

В цілому, результати демонструють, що немає універсальної моделі, яка підійде для всіх типів задач NLP. Важливо вибирати метод ембедингу, орієнтуючись на специфіку задачі, доступні ресурси та конкретні цілі дослідження.

ВИСНОВКИ

У цій кваліфікаційній роботі була проведена всебічна робота щодо аналізу впливу використання контекстуальних та словесних ембедингів на точність класифікації текстових масивів. В рамках дослідження вдалося досягти встановлених цілей та вирішити поставлені задачі.

Було проведено детальне дослідження та порівняння Word2Vec та GloVe, а також BERT та GPT методів векторизації. Це дозволило визначити, які техніки найкраще підходять для різних типів текстових даних.

Було вивчено та описано різні підходи до препроцесингу та аналізу текстів, що підкреслює важливість кожного етапу в побудові ефективної системи класифікації.

Застосування Word2Vec та GloVe в контексті нейромережевої моделі LSTM показало, що обидва методи ефективно векторизують текст для класифікації. Хоча Word2Vec та GloVe мають різні підходи до векторизації, обидва демонструють здатність вловлювати семантичні відносини між словами. Порівняння точності між цими двома методами показало, що GloVe має деяку перевагу у класифікації текстових масивів, що може бути пов'язано з його здатністю інтегрувати глобальну статистику співвживання слова у всьому корпусі.

Застосування BERT та GPT в контексті нейромережевої моделі LSTM значно покращило результати класифікації порівняно з методами Word Embedding. Це підтверджує, що вони ефективніше вловлюють складні мовні відносини та контекст. Контекстуальні ембединги продемонстрували здатність до глибшого розуміння мовних нюансів, що сприяло поліпшенню точності класифікації, особливо в складних випадках, де контекст слова має вирішальне значення. Результати підкреслюють важливість використання контекстуальних ембедингів для складних завдань обробки тексту, зокрема у задачах класифікації.

Було проведено зіставлення результатів між найкращими моделями Word Embedding та Contextual Embedding. Найкращою моделлю Word Embedding, на основі наших критеріїв оцінювання, виявилась GloVe, яка продемонструвала кінцеву точність 87.72%, що було вище, ніж у моделі Word2Vec з точністю 86.87%. У контексті Contextual Embedding, BERT виявився ефективнішим порівняно з GPT, з кінцевою точністю 91.23% проти 89.45% в GPT

Враховуючи отримані результати, можна рекомендувати BERT та GPT як найефективніші інструменти для завдань класифікації текстів, особливо коли доступні відповідні обчислювальні ресурси. Для ситуацій з обмеженими ресурсами або при вимогах до швидкості обробки, Word2Vec та GloVe залишаються надійними варіантами.

Наукове значення роботи полягає у поглибленні розуміння механізмів, що лежать в основі векторного представлення слів та їх впливу на процеси класифікації. Викладені висновки можуть слугувати основою для подальших досліджень у галузі машинного навчання та розробки інтелектуальних систем, що сприятиме прогресу в області штучного інтелекту.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. O. BARKOVSKA, K. VOROPAIEVA, O. RUSKIKH Justifying the selection of a neural network linguistic classifier // Innovative technologies and scientific solutions for industries. - 2023. - №. 3 (25). - pp. 5-14.

2. Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., & Zettlemoyer, L. (2018). Deep contextualized word representations.

3. Howard, J., & Ruder, S. (2018). Universal Language Model Fine-tuning for Text Classification.

4. Barkovska, O., Pyvovarova, D., Kholiev, V., Ivashchenko, H., & Rosinskiy, D. (2021, April). Information Object Storage Model with Accelerated Text Processing Methods. In COLINS (pp. 286-299).

5. Serdechnyi, V., Barkovska, O., Rosinskiy, D., Axak, N., & Korablyov, M. (2020). Model of the internet traffic filtering system to ensure safe web surfing. In Lecture Notes in Computational Intelligence and Decision Making: Proceedings of the XV International Scientific Conference “Intellectual Systems of Decision Making and Problems of Computational Intelligence”(ISDMCI'2019), Ukraine, May 21-25, 2019 15 (pp. 133-147). Springer International Publishing. https://doi.org/10.1007/978-3-030-26474-1_10

6. Jurafsky, D., & Martin, J. H. (2019). Speech and Language Processing (3rd ed. draft).

7. Vujović, Ž. Đ. (2021). Classification Model Evaluation Metrics

8. Jazayeri, M., & Ostojic, S. (2021, October). Interpreting Neural Computations by Examining Intrinsic and Embedding Dimensionality of Neural Activity. Current Opinion in Neurobiology, 70, 113-120.

9. Wang, Congcong & Nulty, Paul & Lillis, David. (2020). A Comparative Study on Word Embeddings in Deep Learning for Text Classification. 37-46. https://doi.org/10.1145/3443279.3443304

10. Jang B. et al. Bi-LSTM model to increase accuracy in text classification:

Combining Word2vec CNN and attention mechanism //Applied Sciences. - 2020. - Т. 10. - №. 17. - С. 5841. https://doi.org/10.3390/app10175841

11. Gao M., Li T., Huang P. Text classification research based on improved Word2vec and CNN //Service-Oriented Computing-ICSOC 2018 Workshops: ADMS, ASOCA, ISYyCC, CloTS, DDBS, and NLS4IoT, Hangzhou, China, November 12-15, 2018, Revised Selected Papers 16. - Springer International Publishing, 2019. - С. 126-135. https://doi.org/10.1007/978-3-030-17642-6_11

12. Zhou H. Research of text classification based on TF-IDF and CNN-LSTM //Journal of Physics: Conference Series. - IOP Publishing, 2022. - Т. 2171. - №. 1. - С. 012021. https://doi.org/10.1088/1742-6596/2171/1/012021

13. Mars M. From word embeddings to pre-trained language models: A state-of-the-art walkthrough //Applied Sciences. - 2022. - Т. 12. - №. 17. - С. 8805. https://doi.org/10.3390/app12178805

14. Selva Birunda S., Kanniga Devi R. A review on word embedding techniques for text classification //Innovative Data Communication Technologies and Application: Proceedings of ICIDCA 2020. - 2021. - С. 267-281. https://doi.org/10.1007/978-981-15-9651-3_23

15. Patil, Rajvardhan & Boit, Sorio & Gudivada, Venkat & Nandigam, Jagadeesh. (2023). A Survey of Text Representation and Embedding Techniques in NLP. IEEE Access. PP. 1-1. https://doi.org/10.1109/ACCESS.2023.3266377.]

16. Mikolov, Tomas & Chen, Kai & Corrado, G.s & Dean, Jeffrey. (2013). Efficient Estimation of Word Representations in Vector Space. Proceedings of Workshop at ICLR. 2013.

17. Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global Vectors for Word Representation. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1532-1543, Doha, Qatar. Association for Computational Linguistics. https://doi.org/10.3115/v1/D14-1162.

18. Joulin, A., Grave, E., Bojanowski, P., Mikolov, T., Bagheri, M., Lample, G., ... & Mikolov, T. (2017). FastText: Enriching Word Vectors with Subword

Information.

19. Bojanowski, P., Grave, E., Joulin, A., Mikolov, T., & Mikolov, T. (2016). Enriching Word Vectors with Subword Information.

20. Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). BERT: Bidirectional Encoder Representations from Transformers.

21. Radford, A., Narasimhan, K., Salimans, T., & Sutskever, I. (2018). Improving Language Understanding by Generative Pre-training.

22. Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Le, Q. V., & Salakhutdinov, R. (2019). XLNet: Generalized Autoregressive Pretraining for Language Understanding.

23. Alshemali, B., & Kalita, J. (2020, March). Improving the Reliability of Deep Neural Networks in NLP: A Review. Knowledge-Based Systems, 191, 105210.

24. Gao, Y., Liu, Y., Zhang, H., Li, Z., Zhu, Y., Lin, H., & Yang, M. (2020, November). Estimating GPU Memory Consumption of Deep Learning Models. In Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE 2020) (pp. 1342-1352). DOI: 10.1145/3368089.3417050.

25. Luo, M., Yu, L., & Yao, Y. (2021, February). Analysis on RAMS Information for Metro Vehicles Using Natural Language Processing Algorithm: Evidence From China. Paper No: IMECE2020-24363, V014T14A003; 7 pages.

26. Dice, D., & Kogan, A. (2021, February). Optimizing Inference Performance of Transformers on CPUs.

27. Panchenko D. et al. Ukrainian news corpus as text classification benchmark //International Conference on Information and Communication Technologies in Education, Research, and Industrial Applications. - Cham : Springer International Publishing, 2021. - -°. 550-559. https://doi.org/10.1007/978-3-319-76168-8

28. Schwenk, H., & Li, X. (2018). A Corpus for Multilingual Document Classification in Eight Languages. ArXiv, abs/1805.09821.

29. Bird, S., & Loper, E. (2019). NLTK: The Natural Language Toolkit.

30. Zhang J. et al. LSTM-CNN hybrid model for text classification //2018 IEEE 3rd Advanced Information Technology, Electronic and Automation Control Conference (IAEAC). - IEEE, 2018. - C. 1675-1680. https://doi.org/10.1109/IAEAC.2018.8577620

31. WANG Haitao, HE Jie, ZHANG Xiaohong, et al., “A Short Text Classification Method Based on N-Gram and CNN,” Chinese Journal of Electronics, vol. 29, no. 2, pp. 248-254, 2020, https://doi.org/10.1049/cje.2020.01.001

32. Mohammad, A.H., Alwada’n, T., Al-Momani, O., "Arabic Text Categorization Using Support Vector Machine, Naïve Bayes and Neural Network." https://doi.org/10.5176/2251-3043_4.4.360

33. Krizhevsky, Alex & Sutskever, Ilya & Hinton, Geoffrey. (2012). ImageNet Classification with Deep Convolutional Neural Networks. Neural Information Processing Systems. p.25. https://doi.org/10.1145/3065386.

34. Wang S. et al. Densely connected CNN with multi-scale feature attention for text classification //IJCAI. - 2018. - T. 18. - C. 4468-4474. https://doi.org/10.24963/ijcai.2018/621