Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»
Кафедра мультимедійних систем

Automated Coursework Evaluation Using NLP and DL

Текстова частина до курсової роботи
за спеціальністю «Прикладна математика»

Керівник курсової роботи
ас. Смиш О. Р.

_____
(підпис)
“____” _____ 2023 р.
Виконав студент 3 р. н.
Кузьменко Ю.А.
“____” _____ 2023 р.

Київ – 2023

Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»

Кафедра мультимедійних систем

ЗАТВЕРДЖУЮ

Зав. кафедри мультимедійних систем

доцент, к.ф.-м.н.

О. П. Жежерун

_____2023 р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ

на курсову роботу

студенту Кузьменку Юрію Андрійовичу 3-го курсу факультету інформатики

ТЕМА Automated Coursework Evaluation Using NLP and DL

Зміст ТЧ до курсової роботи:

Вступ

1. Аналіз предметної галузі
2. Отримання та попередня обробка тексту та даних
3. Вилучення змістовних ознак
4. Моделі глибокого навчання для класифікації тексту

Висновки

Список літератури

Дата видачі „___” _____ 2022 р. Керівник _____

Завдання отримав _____

Тема: Автоматизоване оцінювання курсових робіт з використанням NLP та глибокого навчання

Календарний план виконання роботи:

№ п/п	Назва етапу	Термін виконання	Примітка
1.	Отримання завдання на курсову роботу.	25.09.2022	
2.	Обговорення деталей теми та розроблення плану виконання	3.10.2022	
3.	Огляд технічної літератури за темою роботи та опрацювання матеріалів	15.11.2022 — 09.03.2023	
3.	Написання практичної частини роботи	Січень 2023 — Квітень 2023	
4.	Аналіз практичної частини роботи	Січень 2023 — 03.05.2023	
5.	Написання текстової частини роботи.	04.05.2023 — 12.05.2023	
6.	Здача роботи на перевірку на плагіат	15.05.2023	

Студент Кузьменко Ю.А.

Керівник Смиш О.Р.

“_____”_____

Зміст

Анотація 5
Вступ 6
1 Аналіз предметної сфери.....7
1.1 Огляд NLP 7
1.2 Використання NLP для класифікації тексту 7
2 Отримання та попередня обробка тексту та даних 8
2.1 Отримання даних..... 8
2.2 Попередня обробка тексту для класифікації тексту 9
2.3 Попередня обробка даних 10
3 Вилучення змістовних ознак 13
4 Модель глибинного навчання для класифікування текстів.....14
4.1 Пояснення логіки тренування та роботи моделі глибинного навчання.... 14
4.2 Огляд архітектури та тренування моделі глибокого навчання для класифікації тексту 17
4.3 Оцінка точності моделі для класифікації тексту 22
Висновки 29
Список літератури.....30

Анотація

У курсовій роботі розглянуто створення та аналіз моделі глибокого навчання для класифікації текстів із використанням сучасних технік обробки природної мови. У роботі описано детальний аналіз текстів природної мови та аналіз методів для вилучення змістовних ознак з текстів. Кінцевий програмний продукт дає змогу оцінювати тексти за шкалою від 1 до 3.

Ключові слова: Python, PyTorch, Pycharm, Neural Network, NLP, text scoring, imbalanced learning

Вступ

З розвитком сучасних технологій та інформаційного суспільства виникає потреба в автоматизованому оцінюванні курсових робіт. Автоматизація процесу оцінювання значно зменшує час, витрачений на оцінювання робіт, а також допомагає уникнути суб'єктивного оцінювання викладачів. Найпоширенішими методами оцінювання є перевірка робіт на граматичні та стилістичні помилки, оцінювання логіки викладу.

Метою курсової роботи є дослідження можливості застосування методів обробки живої мови (NLP) та глибокого навчання для автоматизованого оцінювання курсових робіт. Ця робота пропонує новий підхід до оцінювання курсових робіт, який базується на стилі написання та граматиці з використанням NLP технологій та глибинного навчання.

Отже, ця робота може бути корисною для викладачів, які займаються оцінюванням курсових робіт, а також для студентів, які прагнуть здобути якісну оцінювання своєї роботи. Результати дослідження можуть бути використані для подальшого покращення та розроблення систем оцінювання на основі NLP та глибокого навчання.

1 Аналіз предметної сфери

1.1 Огляд NLP

Обробка природної мови (NLP) - це галузь науки, яка вивчає взаємодію між комп'ютерами та людьми, що використовують мову. Основна мета NLP полягає в тому, щоб розробити алгоритми та програми, які можуть розуміти, аналізувати та генерувати людську мову.

У NLP використовуються методи та техніки зі статистики, машинного навчання, лінгвістики та комп'ютерної науки, щоб вивчати та розуміти мовні дані. Наприклад, ці методи охоплюють аналіз тексту, синтаксичний та семантичний аналіз, машинний переклад.

Одним з головних завдань NLP є розуміння контексту людської мови. Це завдання містить аналіз тональності, виявлення емоцій, вивчення граматики та інші властивості мови, які вказують на наміри людей.

NLP має широкий спектр застосувань, ця галузь застосовується для токенізації, систематизації, лематизації, очистки та аналізу тексту.

Загалом, NLP є важливою галуззю науки, яка дає змогу комп'ютерам розуміти та використовувати людську мову, що актуально в наш час, коли зростає кількість інформації, яку потрібно обробити та зрозуміти.

1.2 Використання NLP для класифікації тексту

Одним з важливих завдань, яке вирішується з допомогою NLP, є класифікація тексту. Класифікація тексту є процесом призначення певних категорій текстам із використанням автоматичних алгоритмів та методів машинного навчання.

Класифікація тексту використовують для автоматичного оцінювання текстів. Процес класифікації тексту полягає у визначенні того, до якої категорії текст належить на основі його змісту.

Для класифікації тексту використовуються різноманітні підходи та методи, які базуються на обробці та аналізі текстової інформації. Один з найпоширеніших методів класифікації тексту — це метод «мішок слів» (Bag of Words, BoW). У цьому методі кожен текст розбивається на відокремлені слова, які називаються «токенами». Текст представляється у як множина токенів, без урахування порядку слів у тексті. Далі, для кожного тексту будується вектор, що відображає кількість входжень кожного токену в тексті. Ці вектори потім використовуються для тренування моделі класифікації тексту.

Методи, які використовуються для класифікації тексту, охоплюють статистичні моделі та штучні нейронні мережі.

2 Отримання та попередня обробка тексту та даних

2.1 Отримання даних

Отримання точних та якісних даних для класифікації тексту є вагомим етапом у створенні ефективних систем аналізу текстових даних. З огляду на те, що тексти можуть відрізнятися за стилем, тематикою, мовою та іншими параметрами, процес отримання даних вимагає систематичного та методичного підходу.

У цій роботі використано різні методи отримання даних, зокрема, збір даних з мережі Інтернет, збір текстів курсових робіт, а також метод генерації дубльованих даних.

Для побудови першої нейронної мережі використано датасет «The Hewlett Foundation: Automated Essay Scoring». Цей датасет є частиною змагання «ASAP-AES» по оцінці текстів на веб-сайті kaggle.com.[1]. Змагання відбулось 11 років тому, найкраще значення метрики F1 яке змогли досягти на той час - 0.81407. У цій роботі запропоновано більш сучасний метод оцінки есе, при якому досягається більше значення метрики F1.

2.2 Попередня обробка тексту для класифікації тексту

У NLP попередня обробка тексту це набір процедур, які виконуються над необробленими текстовими даними, щоб підготувати тексти до аналізу та моделей машинного навчання. Ця процедура передбачає перетворення сирих текстових даних у структурований формат, який комп'ютер може обробити.

У роботі використано такі методи попередньої обробки тексту: видалення розділових знаків, токенізація, видалення чисел, видалення стоп-слів та лематизація.

Видалення розділових знаків впливає на подальше опрацювання, оскільки знаки пунктуації іноді не впливають на семантику, а залишення пунктуації в текстових даних призводить до появи шуму, знижуючи точність аналізу або моделі. Наприклад, якщо речення закінчується крапкою, вона не надає додаткової інформації про зміст речення, хоча може вплинути на те, як алгоритми машинного навчання оцінюють текст.

Токенізація — процес розбиття тексту на відокремлені одиниці («токени»), які можуть бути легше оброблені комп'ютером. Токени — це слова, числа, пунктуаційні знаки та інші символи, які використовуються в тексті. Процес токенізації є значущим етапом підготовки тексту для подальшої обробки в NLP-задачах.

Видалення стоп-слів — техніка обробки природної мови, що спрямована на підвищення ефективності аналізу тексту через видалення слів, які не несуть значення в контексті поставленого завдання. Стоп-слова — це часто вживані слова, як-от «і», «в», «на», «з», «що», «та», «де», «не», «його» тощо, які часто трапляються в текстах різних жанрів, але не мають значущого впливу на зміст тексту. Видалення цих слів із документів або корпусів може допомогти зменшити розмір даних та знизити кількість шуму, що відображається на точності та ефективності обробки.

Лематизація — це процес автоматичної обробки природної мови, що полягає у визначенні базової форми (леми) слова на основі його граматичних форм та контексту використання. У порівнянні зі стемінгом, який відтинає закінчення слова, лематизація дає більш точний результат, оскільки враховує контекст та морфологічні ознаки слова.[2]

2.3 Попередня обробка даних

Дані розділено на 3 частини – тренувальні, тестові та дані для перевірки.

Тренувальна частина має розмір 7432 елементи, тестова – 1858, а дані для перевірки – 1032.

Розділення даних на тренувальні та тестові набори є важливим кроком для оцінення ефективності моделі. Тренувальний набір використовується для навчання моделі, тобто використання даних для знаходження залежностей між вхідними та вихідними даними. Тестовий набір використовується для оцінювання точності моделі на незалежних даних, які не використано для навчання. Це допомагає визначити, наскільки добре модель може узагальнити знання на нових даних.

Додатково до тренувального та тестового наборів, набір перевірки використано для покращення точності моделі. Набір даних для перевірки використано для налаштування параметрів моделі, вибору кращого алгоритму навчання та налаштування гіперпараметрів.

2.4 Балансування категорій у цільовому класі в тренувальних даних

Дисбаланс даних є однією з найпоширеніших проблем у класифікації тексту. Це означає, що кількість даних у кожному класі може значно відрізнятися, що призводить до низької точності класифікації.

Під час аналізу даних, виявлено що цільовий клас в тренувальних даних не є збалансованим(*Рис 2.1*).

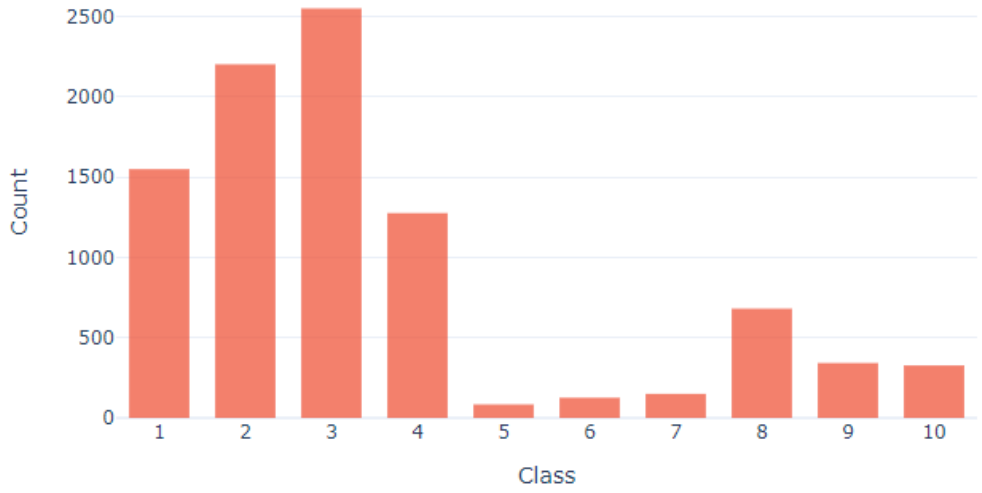

Рис. 2.1 Розподіл оцінок у тренувальних даних

Для вирішення цієї проблеми використано клас RandomOverSampler з пакету imblearn.

RandomOverSampler - це алгоритм передискретизації даних, який використовується для балансування наборів даних з незбалансованими класами через збільшення кількості зразків в меншому класі. Алгоритм

RandomOverSampler обирає випадкові зразки з меншим класом та дублює ці зразки, щоб зрівняти розмір класів.

RandomOverSampler є досить простим і швидким алгоритмом з часовою складністю O(N), де N - кількість зразків у датасеті. Проте, він може створювати дублювання зразків, що може призвести до перенавчання моделі. Також, у разі, якщо менший клас має дуже малу кількість зразків, RandomOverSampler може не бути ефективним у збалансуванні даних.

У роботі використано стратегію вибірки «не більшість», яка передбачає перевибірку всіх класів, окрім класу більшості. Ця стратегія гарантує, що мажоритарний клас не буде переоброблений, що може призвести до надмірного перенавчання моделі.[3]

Після балансування даних (*Рис 2.2*) розподіл класів став рівномірним. Як наслідок, у майбутньому модель буде мати кращі показники точності та відтворюваності у разі класифікації тексту.[4]

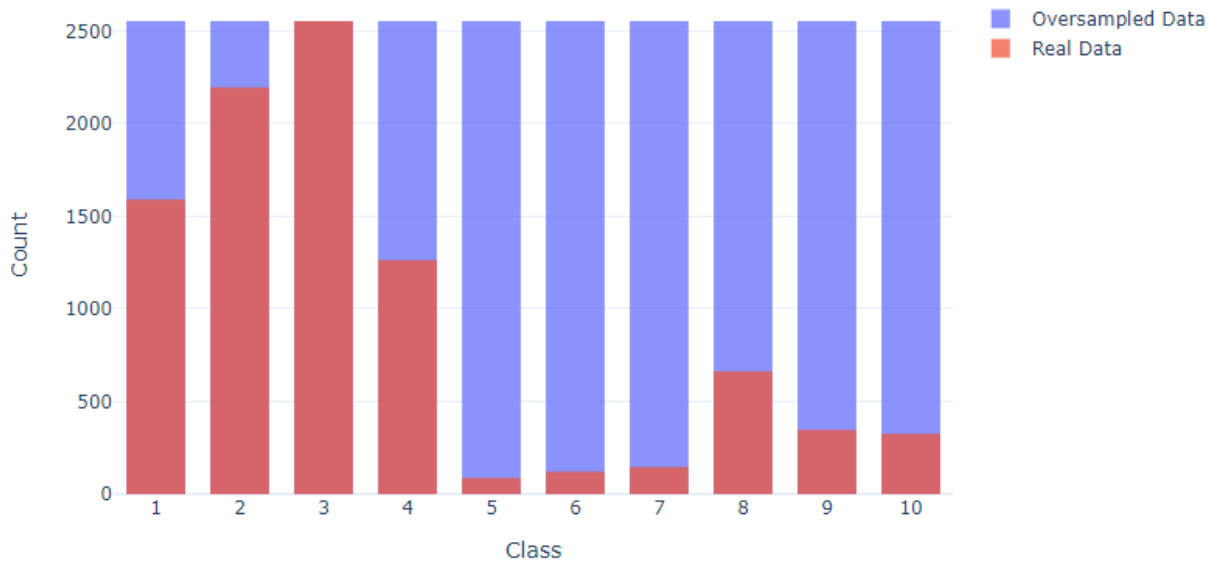

Рис. 2.2 Порівняння розподілів збалансованих та незбалансованих даних

3 Вилучення змістовних ознак

Процес виявлення та вибору найбільш релевантних та інформативних характеристик з необроблених текстових даних називається вилученням ознак. Цими елементами можуть бути слова, речення або інші лінгвістичні аспекти, які допомагають відобразити зміст і контекст тексту. Виокремлюючи ці елементи, машини можуть краще зрозуміти зміст і структуру тексту, що дає змогу їм виконувати такі завдання, як класифікація текстів, аналіз настроїв і машинний переклад.

Модель «мішок слів» (bag-of-words, BoW) є популярною технікою для вилучення ознак. Цей метод інтерпретує текст як «мішок» слів, ігноруючи синтаксис і порядок слів, водночас відстежуючи частоту кожного слова.[5]

У цій роботі використано нестандартний метод BoW, його реалізовано функцією «vectorize» який перетворює «токени» кожного тексту на вектори з їхніми обмеженнями за кількістю «токенів» у кожному тексті:

```
def vectorize_batch(batch):
    Y, X = list(zip(*batch))
    X = [vocab(tokenize(text)) for text in X]
    X = [tokens + ([0] * (max_words - len(tokens))) if len(tokens) <
max_words else tokens[:max_words] for tokens in X]

    return torch.tensor(X, dtype=torch.int32), torch.tensor(Y) - 1
```

Цей підхід є більш ефективним ніж звичайний метод BoW, оскільки в ньому є функціонал обмеження довжини тексту, що сприяє регулюванню довжини вектора, який надходить до нейронної мережі.

4 Модель глибинного навчання для класифікування текстів

4.1 Пояснення логіки тренування та роботи моделі глибинного навчання

Нейронна мережа або алгоритм навчання штучної нейронної мережі — це обчислювальна система навчання, що використовує мережу функцій для розуміння і генерування бажаного результату, як правило, в іншій формі, на основі вхідних даних одного типу. Біологія людини й те, як нейрони працюють разом у людському мозку для розуміння інформації, що надходить від органів чуття, є основами для концепції штучної нейронної мережі.

Алгоритми машинного навчання використовують різноманітні інструменти та методи, зокрема нейронні мережі. У багатьох різних методах машинного навчання нейронна мережа може використовуватися як компонент для перетворення складних вхідних даних на мову, зрозумілу комп'ютерам.[6]

У цьому розділі наведено огляд штучних нейронних мереж прямого поширення.

На рисунку 4.1 розглянуто топології штучної нейронної мережі. Мережа складається з взаємопов'язаних вузлів (так званих нейронів), які організовані у три основні шари: вхідний, прихований і вихідний (існують різні варіації цієї топології, але вони виходять за рамки цієї роботи). Вхідні вузли на цьому зображенні не виконують жодних обчислень і використовуються лише для розподілу входів у мережу. Цей тип мережі відомий як мережа прямого поширення, тому що інформація проходить обчислення в одному напрямку від вхідного шару до прихованого і, нарешті, до вихідного шару.

Кількість вхідних вузлів, N, та число вихідних вузлів, M, у штучній нейронній мережі залежать від задачі, до якої застосовують мережу. Не існує чітких рекомендацій щодо того, скільки вузлів має бути розташовано у прихованому шарі. Якщо у прихованому шарі занадто мало вузлів, мережа може мати труднощі з узагальненням задачі, яку вона ніколи раніше не вирішувала. З іншого боку, якщо у прихованому шарі занадто багато вузлів, мережа може витратити неприйнятно багато часу, щоб навчитися чомусь цінному.

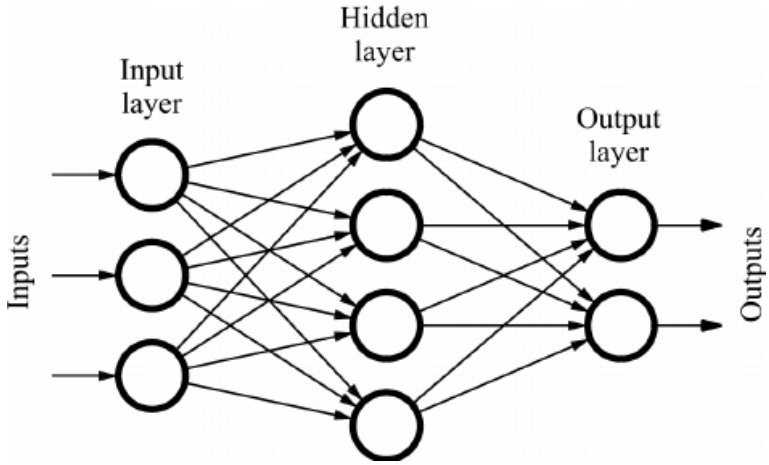

Рис. 4.1 Топологія штучної нейронної мережі прямого поширення

На рисунку 4.2 розглянуто штучний нейрон мережі (у прихованому та вихідному шарах). Кожен нейрон має набір вхідних елементів від x_1 до x_n. Кожен вхідний елемент пов'язаний з його вагою, w_i, яка являє собою коефіцієнтом, на який множаться всі значення, що надходять на нейрон. Нейрон підсумує значення всіх пар $x_i w_i, i = 1,2 \dots . n$ за принципом (1):

$$S = \sum_{i=1}^{n} x_i w_i + w_0$$

У рівнянні (1) додатковий елемент w_0 називається зміщенням. Функція активації - до кожного значення S_j, де j – номер нейрону, щоб забезпечити кінцевий вихід нейрону. Функція активації може бути лінійною, дискретною або неперервною функцією розподілу. Однак для того, щоб використовувати процес зворотного поширення для навчання мережі, ця функція повинна мати властивість бути диференційованою скрізь.[7]

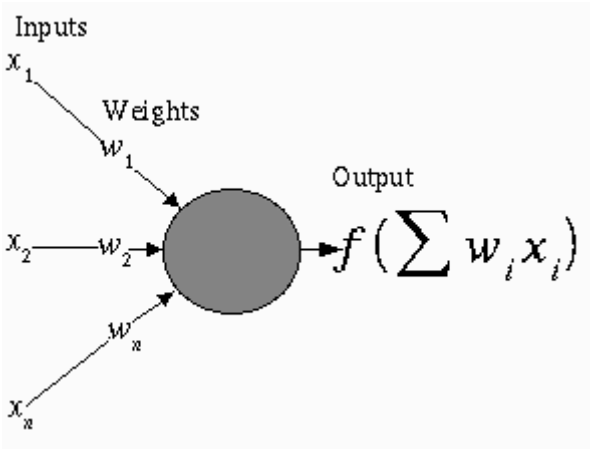

Рис. 4.2 Штучний нейрон

Навчання нейронної мережі відбувається під час зворотного поширення, процесу пошуку найкращих ваг, w_{ij}, для кожного нейрону. Отримавши вхідний шаблон з навчального набору пар вхідних/вихідних шаблонів, мережа сформує вихідний шаблон навчального набору, враховуючи вхідні дані. Після вивчення цього відображення між входом і виходом для навчального набору, застосовують новий, раніше не оброблений вхід і приймають вихід як «висновок» нейромережі, заснований на вивченні фундаментальних кореляцій між входом і виходом з навчального набору.

Якщо t_i цільові вихідні значення для вхідних елементів x_i, то мережа тренується мінімізуючи e:

$$E = \sum_{x} \sum_{i} \left(t_i^{(x)} - u_i^{(x)} \right)^2$$

де t_i^x цільові вихідні значення (взяті з тренувального набору даних), а u_i^x - це вихід нейронної мережі, коли на вхідному шарі представлено x-й вхідний шаблон навчальної вибірки.[8]

4.2 Огляд архітектури та тренування моделі глибокого навчання для класифікації тексту

Підходи, засновані на шаблонах, використовують набір заздалегідь визначених правил для класифікації тексту на кілька категорій, а отже, потребують глибоких предметних знань. З іншого боку, методи, засновані на машинному навчанні, вчаться класифікувати текст на основі спостережень за даними. Алгоритм машинного навчання вивчає внутрішні кореляції між текстами та їхніми мітками, використовуючи попередньо позначені екземпляри, як навчальні дані.[9]

Для побудови нейронної мережі використано мову програмування Python та пакет PyTorch. PyTorch - це відкритий програмний засіб для машинного навчання, що базується на бібліотеці тензорних обчислень Torch. Вона надає різноманітні функції для побудови та навчання нейронних мереж з використанням автоматичного диференціювання та підтримує роботу на графічних процесорах. PyTorch також надає простоту в розробці та зручний інтерфейс для створення складних моделей.[10]

Для цієї задачі використано вбудовування (Embeddings) нейронної мережі, блоки двосторонньої довгої короткочасної пам'яті (Bidirectional LSTM) та лінійні нейрони.

Відображення дискретної або категоріальної змінної у вектор неперервних цілих чисел називається вбудовуванням. Вбудовування - це низьковимірні, неперервні векторні представлення дискретних змінних у

налаштуваннях нейронних мереж. Нейромережеві вбудовування цінні тим, що вони можуть зменшити розмірність категоріальних змінних, але при цьому змістовно представляти категорії у зміненому просторі.[11]

LSTM - це блок нейронної мережі, який створено для вирішення проблеми зникання градієнту в рекурентних нейронних мережах при опрацюванні текстів великого розміру.

Коли градієнт функції помилки нейронної мережі поширюється назад через блок нейронної мережі, він масштабується на певний коефіцієнт. Цей коефіцієнт майже в усіх практично важливих випадках є або більшим, або меншим за одиницю. В результаті градієнт у рекурентній нейронній мережі зростає або спадає експоненціально з плином часу. (У моделюванні мови, часові кроки відповідають позиціям слів у фразі). У результаті градієнт або домінує на наступному кроці вагової адаптації, або фактично втрачається.[12]

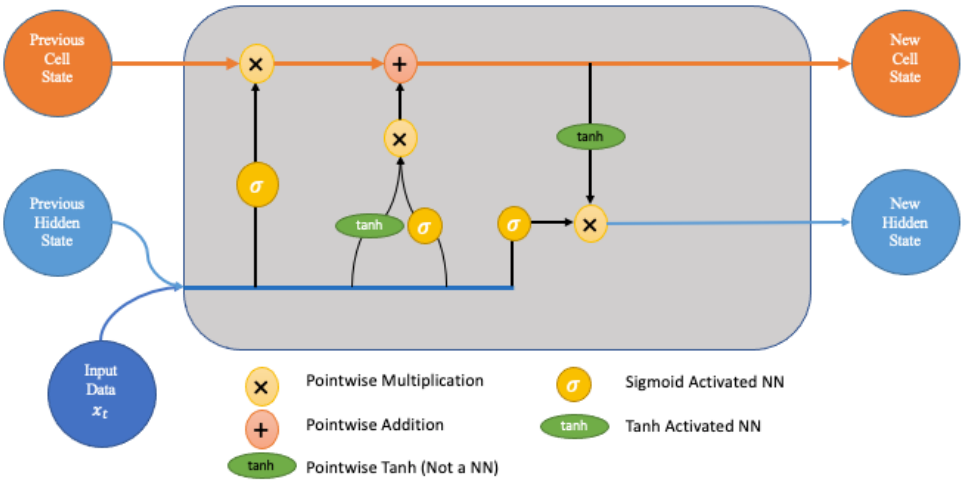

Рис. 4.3 Архітектура блоку довгої короткочасної пам'яті

Комірка пам'яті, відома як «комірка стану», яка зберігає свій стан протягом певного часу, відіграє фундаментальну роль у блоці LSTM. Горизонтальна лінія, яка проходить через верхню частину діаграми вище, являє собою стан комірки. Її можна уявити як конвеєр, через який інформація проходить без змін. У LSTM блоці інформацію може бути додано або вилучено зі стану комірки, який контролюється за допомогою воріт. Ці ворота регулюють інформації яка надходить в комірку і виходить з неї. Механізм містить операцію поточкового множення, а також сигмоїдний шар нейронної мережі, який допомагає цьому механізму. Сигмоїдний шар виводить числа від нуля до одиниці, де нуль означає «нічого не пропускати», а одиниця – «все пропускати».

Кожна навчальна послідовність у двонаправлених LSTM подається вперед і назад до незалежних рекурентних мереж. Дві послідовності пов'язані з одним вихідним шаром. Двонаправлені LSTM знають все про кожну точку в заданій послідовності, як до, так і після неї.[13]

На рисунку 4.4 представлено умовне накреслення нейронної мережі використаної у цій роботі.

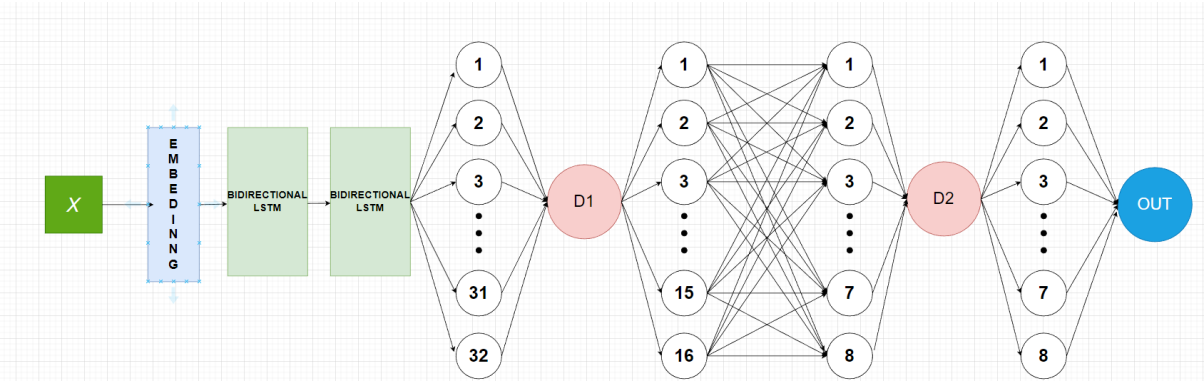

Рис 4.4 Діаграма архітектури нейронної мережі

Блакитний шар Embedding має довжину 150 умовних одиниць, та кількість вкладань слів дорівнює довжині словника.

Двоє зелених шарів двосторонньої довгої короткочасної пам'яті мають 128 прихованих нейронів. Також в кожному з них є шанс на вимкнення (dropout) нейронів, вірогідність якого дорівнює 0.3.

Вимкнення (Dropout) – це штучний нейрон у мережі, який використовується для її регуляризації. Цей нейрон «вимикає», тобто зводить w_{i_j} до 0, ваги, які через нього проходять, під час тренування з заданою ймовірністю.

Білі нейрони - це лінійні нейрони з функцією активації ReLU (зрізаний лінійний вузол).

ReLu - це нелінійна функція активації, яка використовується в багатошарових нейронних мережах або глибоких нейронних мережах. Ця функція описується так:

$$g(x) = max(0, x)$$

Таку архітектуру обрано методом тестування різних моделей глибокого навчання та змінення їхніх параметрів для отримання необхідного результату.

4. 3 Тренування нейронної мережі

Для тренування нейронної мережі використано архітектуру CUDA. NVIDIA CUDA - це універсальна платформа паралельних обчислень і модель програмування, яка прискорює глибоке навчання та інші обчислювальні програми, використовуючи переваги паралельної обчислювальної потужності графічних процесорів.[14]

Функція втрат, яку обрано для тренування нейронної мережі – це перехресна ентропія. Ця функція використовується для задач класифікації, також вона підтримує мультикласифікацію.

Перехресна ентропія між двома дискретними розподілами ймовірностей пов'язана з розходженням Кульбака - Лейблера, метрикою, яка відображає, наскільки схожі ці два розподіли в контексті теорії інформації.

Перехресна ентропія між істинним розподілом t і прогнозованим розподілом p описується рівнянням:

$$H(t,p) = -\sum_{s\epsilon S} t(s) * log\big(p(s)\big)$$

І t, і p розподілені на множині S, але можуть приймати потенційно різні значення.[15]

Для оптимізації нейронної мережі було обрано оптимізатор Adam з швидкістю навчання 0.3. Adam — це адаптивний алгоритм оптимізації швидкості навчання, який використовує як імпульс, так і масштабування, поєднуючи переваги RMSProp та SGD з імпульсом. Оптимізатор розроблено для нестаціонарних задач і задач з зашумленими та/або розрідженими градієнтами.

Оновлення ваги виконується таким чином:

$$w_t = w_{t-1} - \eta \frac{\widehat{m_t}}{\sqrt{\widehat{v_t} + \epsilon}}$$

,де

$$\widehat{m_t} = \frac{m_t}{1-\beta_1^t};$$

$$\widehat{v_t} = \frac{v_t}{1-\beta_2^t};$$

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t;$$

$$v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2.$$

Змінна η - це розмір кроку/швидкість навчання. ϵ - невелике число, зазвичай 1e-8 або 1e-10, щоб запобігти діленню на нуль. β_1 та β_2 є параметрами забування, з типовими значеннями 0.9 та 0.999 відповідно.[16]

Нейронна мережа тренувалась 350 епох, це число обрано методом експерименту, при ньому модель передбачає дані з найбільшою точністю на тестовій та тренувальній вибірках.

4.4 Оцінка точності моделі для класифікації тексту

Оцінку точності моделі проведено на обмеженому наборі даних, оскільки на нього не застосовувався алгоритм балансування даних та він не брав участі у навчанні моделі.

Для оцінки точності використано метрики F1 та матриця помилок. Оцінка $F1$ є стандартною метрикою для визначення успішності класифікатора. Це середнє гармонійне значення двох додаткових метрик - точності та запам'ятовування. Формула для задачі бінарної класифікації має вигляд:

$$F1\,Score = \frac{2 \times Precision \times Recall}{Precision + Recall}$$

Для багатокласової задачі класифікації ми не обчислюємо загальну оцінку F-1. Замість цього ми обчислюємо оцінку F-1 для кожного класу за принципом «один проти решти». У цьому підході успішність кожного класу оцінюється незалежно, як якщо б кожен клас мав свій власний класифікатор.[17]

Матриці помилки являють собою підрахунки прогнозованих і фактичних значень. Для бінарної класифікації, матриця має вигляд:

		Predicted	
		Negative	**Positive**
Actual	**Negative**	TN	FP
	Positive	FN	TP

Таблиця 1. Матриця помилок

Вихід «TN» означає «істинно негативний» і представляє кількість правильно ідентифікованих негативних випадків. Аналогічно, «TP» розшифровується як «істинно позитивний» і означає кількість правильно ідентифікованих позитивних випадків. Абревіатура «FP» – це кількість фактично негативних випадків, класифікованих як позитивні, тоді як «FN» означає кількість фактично позитивних прикладів, класифікованих як негативні.[18]

Для цієї задачі метрика F1 дорівнює:

Клас класифікації	**F1**	**Зважений F1**
1	0.91	0.90 (на 1033 записах)
2	0.90	
3	0.88	
4	0.88	
5	0.94	
6	0.97	
7	0.84	
8	0.93	
9	0.91	
10	0.91	

Таблиця 2. Результати метрики F1 для задачі класифікації есе

Та матриця помилок має вигляд:

	Передбачені значення										
Реальні значення		1	2	3	4	5	6	7	8	9	10
	1	155	11	6	0	0	0	0	0	0	0
	2	11	209	15	1	0	0	0	0	0	0
	3	1	11	239	23	0	0	0	0	0	0
	4	0	0	11	135	0	0	0	1	0	0
	5	0	0	0	1	15	0	1	0	0	0
	6	0	0	0	0	0	16	0	0	0	0
	7	0	0	0	0	0	1	16	1	0	0
	8	0	0	0	0	0	0	3	79	4	0
	9	0	0	0	0	0	0	0	1	43	0
	10	0	0	0	0	0	0	0	2	2	27

Таблиця 3. Матриця помилок для задачі класифікації есе

За цими двома метриками видно, що модель чітко виконує класифікацію з похибкою на сусідні класи.

4.5 Оцінка курсових робіт моделлю глибокого навчання

Дані зібрано вручну. Усі курсові роботи було анонімізовано та перекладено з української на англійську мову. Загалом отримано та оброблено 174 курсові роботи з третього, четвертого курсу бакалаврської програми та першого, другого курсу магістерської програми. Більшість курсових робіт написано в 2022 році. Переклад курсових робіт зроблено ітеративно за допомогою Google Translation API.[19] Попередня обробка тексту та вилучення змістовних ознак описано в попередніх розділах.

Для розуміння датасету проведено його аналіз. Проаналізовано довжину тексту та його кореляцію з оцінкою курсової роботи. Середня довжина тексту для цього набору даних — 42600 символів. Розподіл

довжини курсових робіт можна описати Log-logistic розподілом (*Рис. 4.5*).Для аналізу кореляції довжини тексту та оцінки за курсову роботу дані було розділено на 11 частин та в кожній частині прораховано середню довжину тексту та середню оцінку за роботу. На рисунку 4. зображено залежність оцінки від довжини тексту, кореляція між довжиною тексту та оцінкою очевидна, тому в майбутньому довжину тексту можна додати як додаткову ознаку для нейронної мережі.

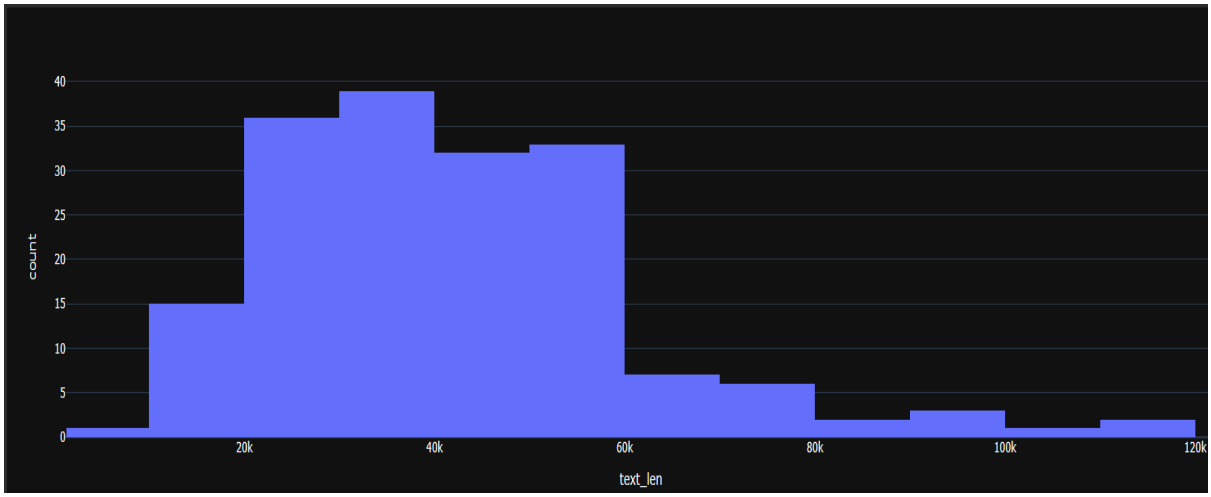

Рис. 4.5 Розподіл довжин курсових робіт

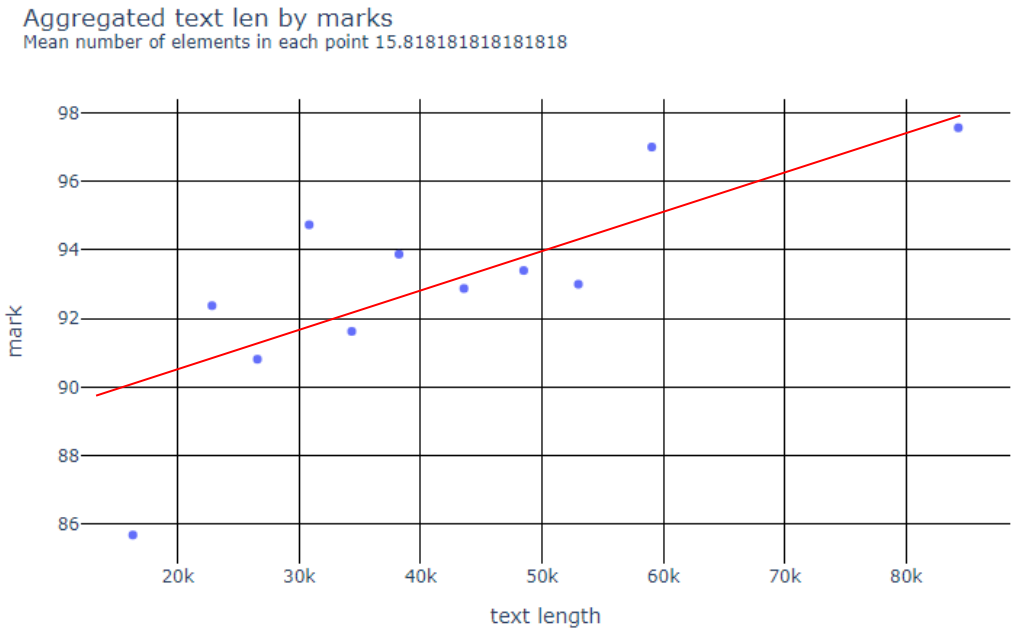

Рис. 4.6 Залежність оцінки від довжини тексту

Дані розділено аналогічно до розділу 2.3. Тренувальна частина має 125 елементів, тестова 31, а дані для перевірки – 18 елементів.

Цільова змінна, оцінка курсової роботи, має нерівномірний розподіл (*Рис 4.7*), оскільки датасет містить малу кількість даних. Для вирішення цієї проблеми змінено тип цільової змінної з дискретного на категоріальний.

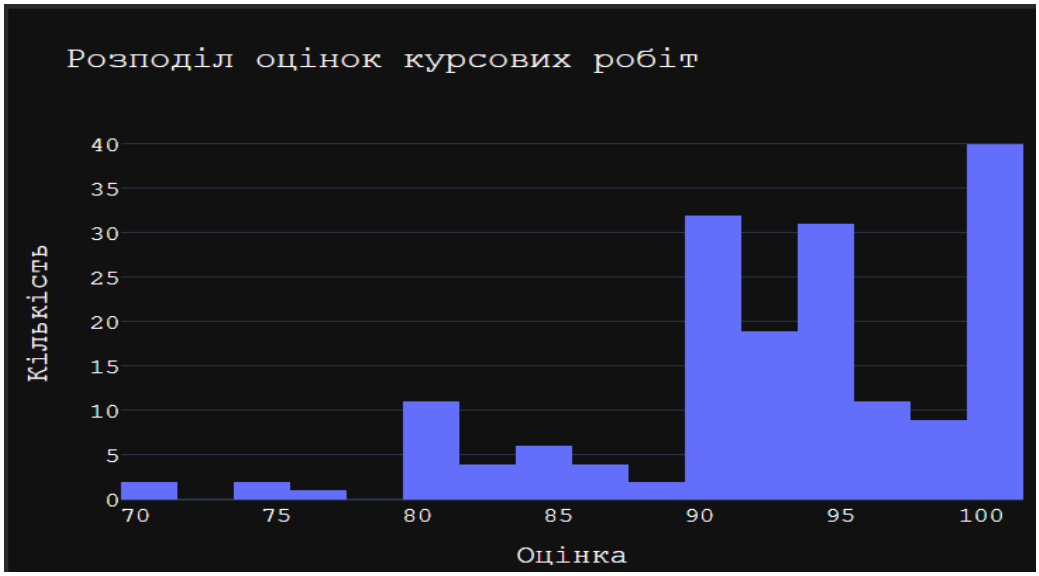

Рис. 4.7 Розподіл оцінок курсових робіт

Серед категорій, на які переформатовано зміну оцінок: (69.97, 80.0], (80.0, 90.0], (90.0, 100.0], яким присвоєно числа 1, 2, 3 відповідно. Оскільки розподіл сформованої змінної є незбалансованим (*Рис 4.8*), до нього застосовано алгоритм балансування даних аналогічний до розділу 2.2.

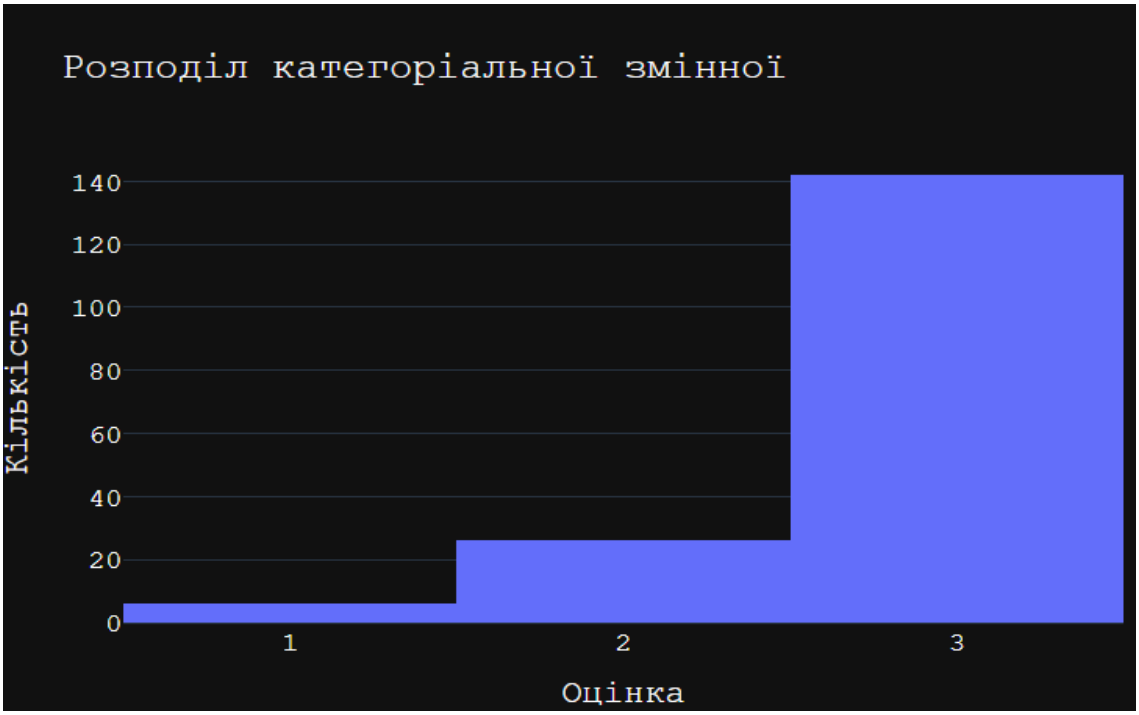

Рис. 4.8 Розподіл категоріальної змінної

Оскільки в датасеті мала кількість даних, то тренування моделі глибокого навчання тільки на цих даних є невиправданим. Модель не знайде загальні залежності та не розрізнятиме класи. Одним із методів вирішення цієї проблеми є використання мережі «донора» з розділу 4.2. Цей метод називається трансферним навчанням.

Трансферне навчання — це метод машинного навчання, в якому модель, створена для одного завдання, використовується як основа для моделі іншого завдання. Однією з основних переваг методу трансферного навчання є можливість зменшення кількості даних, необхідних для тренування мережі. Це досягається через використання вивчених знань з попередньої мережі, яка натреновано на подібних даних. Таким чином, вивчення нових задач стає більш ефективним та економним, оскільки не потрібно створювати великі обсяги нових даних.[20]

Архітектура мережі трансферного навчання майже аналогічна до моделі з розділу 4.2, але в неї видалено останній шар та замінено на лінійний шар з 32 нейронами (*Рис 4.9*).

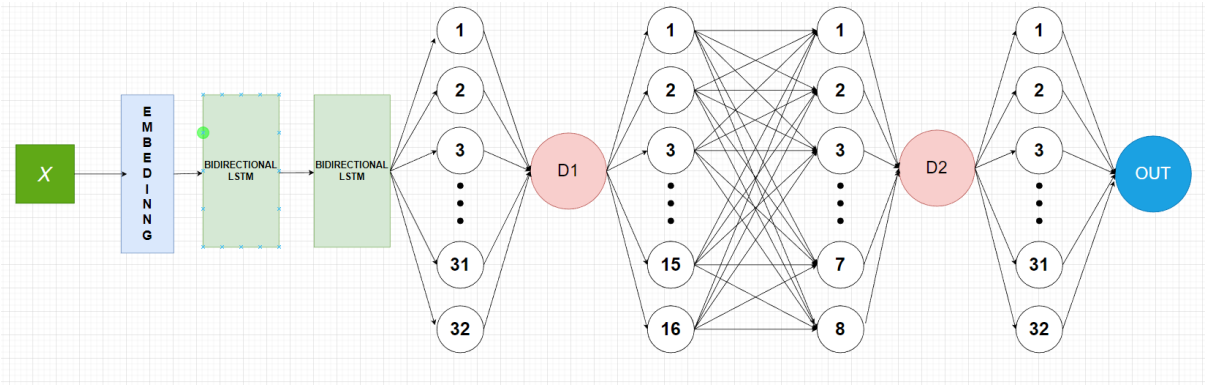

Рис. 4.9 Архітектура нейронної мережі трансферного навчання

Тренування моделі відбувалось 200 епох та з оптимізатором Adam при швидкості навчання 0.01.

Точність моделі оцінено на наборі даних для перевірки. Результати метрики F1 наведено в таблиці: ПІДПИСАТИ ТАБЛИЦІ

Клас класифікації	**F1**	**Зважений F1**
1	0.90	0.81 (на 18 записах)
2	0.0	
3	0.67	

Таблиця 4. Результати метрики F1 для задачі класифікації курсових робіт

Результати не є показовими, оскільки експеримент проведено на малому наборі даних. Для досягнення вагомих результатів потрібно використати більшу кількість даних.

Висновки

У цій курсовій роботі розглянуто тему автоматизованого оцінювання курсових робіт із використанням методів обробки природної мови та глибокого навчання. У ході роботи досліджено різні методи та алгоритми, що використано для аналізу текстової інформації, як-от токенізація, лематизація, метод BoW, балансування цільового класу та методи глибокого навчання.

На основі проведеного дослідження є спроба створення моделі для автоматичної оцінки курсових робіт, яка використовує методи глибокого навчання та обробки природної мови. За допомогою експериментів показано, що створена модель не дає високу точність оцінки курсових робіт, оскільки модель навчено на малому наборі даних. Цей результат можна покращити, збільшивши вибірку даних.

Отже, як висновок, можна зазначити, що використання методів обробки природної мови та глибокого навчання є ефективним підходом до автоматизованої оцінки курсових робіт.

Список літератури

1. The Hewlett Foundation: Automated Essay Scoring Develop an automated scoring algorithm for student-written essays. [Електронний ресурс] – Режим доступу до ресурсу: https://www.kaggle.com/competitions/asap-aes/overview.
2. Harshith. Text Preprocessing in Natural Language Processing [Електронний ресурс] / Harshith. – 2019. – Режим доступу до ресурсу: https://towardsdatascience.com/text-preprocessing-in-natural-language-processing-using-python-6113ff5decd8.
3. Документація imblearn [Електронний ресурс] – Режим доступу до ресурсу: https://imbalanced-learn.org/stable/references/generated/imblearn.over_sampling.RandomOverSampler.html
4. Кузьменко Ю.А Балансування категорій у цільовому класі в тренувальних даних для задач класифікації тексту / Смиш О.Р
5. Siddharth. Feature Extraction and Embeddings in NLP: A Beginners guide to understand Natural Language Processing [Електронний ресурс] / Siddharth. – 2021. – Режим доступу до ресурсу: https://www.analyticsvidhya.com/blog/2021/07/feature-extraction-and-embeddings-in-nlp-a-beginners-guide-to-understand-natural-language-processing/
6. What is a Neural Network? [Електронний ресурс] – Режим доступу до ресурсу: https://deepai.org/machine-learning-glossary-and-terms/neural-network.
7. DAWSON C. W. An artificial neural network approach to rainfall-runoff modelling [Електронний ресурс] / C. W. DAWSON, R. WILBY // Hydrological Sciences Journal. – 2009. – Режим доступу до ресурсу: https://www.tandfonline.com/doi/epdf/10.1080/02626669809492102?needAccess=true&role=button
8. Lapedes A. How Neural Nets Work [Електронний ресурс] / A. Lapedes, R. Farber. – 1987. – Режим доступу до ресурсу: https://proceedings.neurips.cc/paper_files/paper/1987/file/093f65e080a295f8076b1c5722a46aa2-Paper.pdf.
9. Minaee S. Deep Learning--based TText Classification: A Comprehensive Review [Електронний ресурс] / S. Minaee, N. Kalchbrenner, E. Cambria // ACM Computing Surveys. – 2021. – Режим доступу до ресурсу: https://arxiv.org/pdf/2004.03705.pdf
10. Офіційний сайт PyTorch [Електронний ресурс] – Режим доступу до ресурсу: https://pytorch.org/

11. Koehrsen W. Neural Network Embeddings Explained [Електронний ресурс] / Will Koehrsen – Режим доступу до ресурсу: https://towardsdatascience.com/neural-network-embeddings-explained-4d028e6f0526
12. Sundermeyer M. LSTM neural networks for language modeling [Електронний ресурс] / M. Sundermeyer, R. Schl¨uter, H. Ney – Режим доступу до ресурсу: https://www.isca-speech.org/archive_v0/archive_papers/interspeech_2012/i12_0194.pdf
13. What is LSTM? Introduction to Long Short Term Memory [Електронний ресурс]. – 2023. – Режим доступу до ресурсу: https://intellipaat.com/blog/what-is-lstm/
14. Heller M. What is CUDA? Parallel programming for GPUs [Електронний ресурс] / Martin Heller // InfoWorld. – 2022. – Режим доступу до ресурсу: https://www.infoworld.com/article/3299703/what-is-cuda-parallel-programming-for-gpus.html
15. Bala P. Cross-Entropy Loss: Make Predictions with Confidence [Електронний ресурс] / Priya Bala – Режим доступу до ресурсу: https://www.pinecone.io/learn/cross-entropy-loss/
16. Kingma D. Adam: A Method for Stochastic Optimization [Електронний ресурс] / D. Kingma, J. Ba. – 2014. – Режим доступу до ресурсу: https://arxiv.org/abs/1412.6980v9
17. F-1 Score for Multi-Class Classification [Електронний ресурс]. – 2022. – Режим доступу до ресурсу: https://www.baeldung.com/cs/multi-class-f1-score.
18. Batarseh F. Data Democracy At the Nexus of Artificial Intelligence, Software Development, and Knowledge Engineering / F. Batarseh, Y. Ruixin., 2020
19. Google Translation API [Електронний ресурс] – Режим доступу до ресурсу: https://cloud.google.com/translate
20. Brownlee J. A Gentle Introduction to Transfer Learning for Deep Learning [Електронний ресурс] / Jason Brownlee – Режим доступу до ресурсу: https://machinelearningmastery.com/transfer-learning-for-deep-learning/