ДВНЗ «ДОНЕЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ»

Факультет комп'ютерно-інтегрованих технологій та автоматизації
Кафедра прикладної математики і інформатики

«До захисту допущено»
Завідувач кафедри ПМІ
_____ **О.А. Дмитрієва**
(підпис) (ініціали, прізвище)
«_____ » _____ 2021 р.

Випускна кваліфікаційна робота
магістра
(освітній ступінь)

на тему
«Застосування обчислювального інтелекту для розпізнавання тексту »

Виконав (-ла): студент (-ка) _2__ курсу, групи ІПЗм-20
(шифр групи)
напряму підготовки (спеціальності) спеціальності 121 «Інженерія програмного
(шифр і назва напряму підготовки, спеціальності)
забезпечення»

Бабенко Є. О.
(прізвище та ініціали) (підпис)

Керівник зав. каф. ПМІ, проф., д.т.н. Дмитрієва О. А.
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

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

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

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

Засвідчую, що у цій випускній кваліфікаційній роботі немає запозичень з праць інших авторів без відповідних посилань.

Студент _____
(підпис)

Покровськ - 2021

ДВНЗ «Донецький національний технічний університет»
Факультет комп'ютерно-інтегрованих технологій та автоматизації
Кафедра прикладної математики та інформатики
Освітньо-кваліфікаційний рівень (освітній ступінь) магістр
Напрям підготовки (спеціальність) 121 Інженерія програмного забезпечення
(шифр і назва)

ЗАТВЕРДЖУЮ:
Завідувач кафедри

Дмитрієва О. А.
/_____/
"____" _____2021 р.

ЗАВДАННЯ
НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Бабенко Євгенія Олександрівна
(прізвище, ім'я, по батькові)

1. Тема проекту (роботи) Застосування обчислювального інтелекту для розпізнавання тексту

Керівник проекту (роботи) Дмитрієва Ольга Анатоліївна, д.т.н, проф., зав. кафедри ПМІ
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом від "_22__" вересня_____ 2021 року №____570_____

2. Строк подання студентом проекту (роботи)_____

3. Вихідні дані до проекту (роботи) результати з науково-дослідних робіт, метеріали наукових видань, матеріали переддипломної практики, відомості з літератури, довідників, інтернет ресурси.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) дослідження сутності й актуальності розпізнавання тексту; постановка завдання та визначення критеріїв оцінки; проектування архітектури моделі і програмного додатку; розробка моделі розпізнавання тексту; навчання моделі розпізнавання тексту; розробка програмного засобу; оформлення правил користування програмним забезпеченням; тестування програмного додатку; оцінка результатів моделі;

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень

1) блок-схеми розроблених алгоритмів кластерного аналізу;
2) діаграма варіантів використання, поведінки;
3) екранні форми;
4) алгоритми навчання моделі;
5) архітектура моделі класифікації текстів.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Підпис, дата	Підпис, дата

7. Дата видачі завдання 22 вересня 2021

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломної роботи	Строк виконання етапів роботи	Примітка
1.	Дослідження сутності й актуальності розпізнавання тексту	12.10.2021	
2.	Постановка завдання та визначення критеріїв оцінки	15.10.2021	
3.	Проектування архітектури моделі і програмного додатку	28.10.2021	
4.	Розробка моделі розпізнавання тексту	04.11.2021	
5.	Навчання моделі розпізнавання тексту	11.11.2021	
6.	Розробка програмного засобу	16.11.2021	
7.	Тестування програмного додатку	19.11.2021	
8	Оцінка результатів моделі	21.11.2021	
9.	Оформлення пояснювальної записки	01.12.2021	
10.	Підготовка слайдів презентації	05.12.2021	
11.	Підготовка демонстраційної версії розробленого програмного продукту	6.12.2021	
12.	Написання доповіді	7.12.2021	

Студент _____ Бабенко Є. О.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Дмитрієва О. А.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Бабенко Є. О. Застосування обчислювального інтелекту для розпізнавання тексту / Випускна кваліфікаційна робота на здобуття освітнього ступеня «магістр» за спеціальністю 121 «Інженерія програмного забезпечення». – ДВНЗ ДонНТУ, Покровськ, 2021.

Об’єктом дослідження є процеси машинного навчання з застосуванням штучних нейронних мереж для розпізнавання текстових даних.

Предмет дослідження нейромережеві методи розпізнавання текстів.

Мета роботи полягає у розробці, обґрунтуванні та програмній реалізації універсальної моделі розпізнавання текстових даних, що дозволить отримати нові більш детальні знання за допомогою поєднання різних класів і методів аналізу текстових даних.

В ході написання кваліфікаційної роботи використані наступні методи: методи наукового узагальнення, аналізу й синтезу, індукції й дедукції (при дослідженні існуючих в сучасній літературі методів класифікації); порівняльний аналіз показників моделей розпізнавання тексту, засоби обчислювального інтелекту для розпізнавання текстових даних.

Наукова новизна полягає у розробці та застосуванні моделі нейронної мережі, яка поєднує згорткові нейронні мережі та двонаправлені вентильні рекурентні вузли для багатомірної класифікації з розпізнаванням тематики та емоційного забарвлення текстових даних.

Практична цінність роботи полягає в багатомірній класифікації, яка дозволить проаналізувати різноманітні типи текстових документів і отримати сполучені класи за критеріями тональності і тематикою із забезпеченням високої якості розпізнавання тексту.

Ключові слова: інтелектуальний аналіз, класифікація, згорткові нейронні мережі, вентильні рекурентні вузли, модель розпізнавання тексту

Список публікацій:

1. Дмитрієва О.А. Розробка комп’ютерної моделі системи діагностики технічного стану електрообладнання на основі кластерного аналізу/ О.А. Дмитрієва, Т.В. Алтухова, Є.О. Бабенко// Наукові праці Донецького національного технічного університету. Серія: Інформатика, кібернетика та обчислювальна техніка. – 2021. – №1 (32). – С. 24-31.

2-4 [1-3]

ABSTRACT

Babenko E. A. Application of computational intelligence for text recognition / Final qualification work for the degree of "master" in specialty 121 "Software Engineering". - SHEI DonNTU, Pokrovsk, 2021.

The object of research is machine learning processes with the use of artificial neural networks for text data recognition.

The subject of research is neural network methods of text recognition.

The aim of the work is to develop, substantiate and program implementation of a universal model of text data recognition, which will provide new more detailed knowledge through a combination of different classes and methods of text data analysis.

The following methods were used in writing the qualification work: methods of scientific generalization, analysis and synthesis, induction and deduction (in the study of existing classification methods in modern literature); comparative analysis of indicators of text recognition models, means of computational intelligence for text data recognition.

The scientific novelty is the development and application of a neural network model that combines convolutional neural networks and bidirectional valve recurrent nodes for multidimensional classification with subject recognition and emotional coloring of text data.

The practical value of the work lies in the multidimensional classification, which will analyze different types of text documents and obtain combined classes on the criteria of tonality and subject matter, ensuring high quality text recognition.

Keywords: Intelligence Analysis, Classification, Convolutional Neural Networks, Valve Recurrent Nodes, Text Recognition Model

List of publications:

1. Дмитрієва О.А. Розробка комп’ютерної моделі системи діагностики технічного стану електрообладнання на основі кластерного аналізу/ О.А. Дмитрієва, Т.В. Алтухова, Є.О. Бабенко// Наукові праці Донецького національного технічного університету. Серія: Інформатика, кібернетика та обчислювальна техніка. – 2021. – №1 (32). – С. 24-31.
2-4 [1-3]

ЗМІСТ

стр.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, СКОРОЧЕНЬ І ТЕРМІНІВ

CNN Convolutional neural network

GRU Gated recurrent units

LSTM Long short-term memory

Conv Bi LSTM Convolutional bidirectional long short-term memory

Conv Bi GRU Convolutional bidirectional gated recurrent units

UML Unified modeling language

MVP Model-View-Presenter

RNN Recurrent neural network

ВСТУП

У сучасному світі присутня тенденція збільшення інформації у великих темпах. Постає питання у фільтрації та розподіленні інформаційних потоків з метою швидкого, найбільш стислого та змістовного вилучення потрібних та корисних знань з масиву текстових колекцій.

Як дослідити величезну кількість текстових документів є основним питанням в галузі інформаційного пошуку та обробки тексту. При цьому одним із найважливіших методів інтелектуального аналізу текстових даних стає класифікація, спрямована на ефективну організацію, узагальнення та впорядковування текстових документів. Організація великої кількості даних в ряд значущих класів може бути використана для перегляду колекції документів, або для організації результатів, повернутих пошуковою системою в відповідь на запит користувача. Це може значно покращити точність і відкликання в системах пошуку інформації, а також є ефективним способом знайти найближчих сусідів документа [1].

Об’єктом дослідження є процеси машинного навчання з застосуванням штучних нейронних мереж для розпізнавання текстових даних.

Предмет дослідження нейромережеві методи розпізнавання текстів.

Мета роботи полягає у розробці, обґрунтуванні та програмній реалізації універсальної моделі розпізнавання текстових даних, що дозволить отримати нові більш детальні знання за допомогою поєднання різних класів і методів аналізу текстових даних.

Досягнення поставленої мети реалізується виконанням наступні завдань:

- дослідження сутності й актуальності розпізнавання тексту;
- постановка завдання та визначення критеріїв оцінки;
- проектування архітектури моделі і програмного додатку;
- розробка моделі розпізнавання тексту;

- навчання моделі розпізнавання тексту;
- розробка програмного засобу;
- оформлення правил користування програмним забезпеченням;
- тестування програмного додатку;
- оцінка результатів класифікації моделі.

Наукова новизна полягає у розробці та застосуванні моделі нейронної мережі, яка поєднує згорткові нейронні мережі та двонаправлені вентильні рекурентні вузли для багатомірної класифікації з розпізнаванням тематики та емоційного забарвлення текстових даних.

Практична цінність роботи полягає в багатомірній класифікації, яка дозволить проаналізувати різноманітні типи текстових документів і отримати сполучені класи за критеріями тональності і тематикою із забезпеченням високої якості розпізнавання тексту.

В ході написання кваліфікаційної роботи використані наступні методи: методи наукового узагальнення, аналізу й синтезу, індукції й дедукції (при дослідженні існуючих в сучасній літературі методів класифікації); порівняльний аналіз показників моделей розпізнавання тексту, засоби обчислювального інтелекту для розпізнавання текстових даних.

РОЗДІЛ 1
РОЗКРИТТЯ СУТНОСТІ ТА АКТУАЛЬНОСТІ РОЗПІЗНАВАННЯ ТЕКСТУ

1.1 Сутність інтелектуального аналізу текстових даних

Термін інтелектуальний аналіз даних охоплює поєднання широкого математичного інструментарію і останніх досягнень в сфері інформаційні технології, за допомогою яких може бути вилучена точна і раніше невідома інформація з великих обсягів даних у формі зрозумілим для діяльності, і використовуватись для вдосконалення процесів прийняття рішення. За аналогією можна визначити поняття інтелектуального аналізу текстових даних, як процес придбання дійсних, потенційно корисних знань з великого обсягу текстових колекцій [4].

Області застосування інтелектуального аналізу текстових даних:

- витяг інформації – процес автоматичного вилучення структурованої інформації з неструктурованих текстових документів (ідентифікація сутностей);

- інформаційний пошук (вилучення певної інформації за запитом користувача);

- обробка природньої мови (розпізнавання, генерація тексту або мови граматично вірно і найшвидшим шляхом) [5];

- узагальнення інформації – процес створення стислого представлення великої кількості даних;

- класифікація інформації – процес знаходження класів у частково структурованих даних за визначними параметрами;

- кластеризація – угрупування неструктуровані текстові данні у кластери [6].

Класифікація тексту у більш широкому визначенні можна описати, як одну із головних задач обробки природньої мови, яка полягає у знаходженні певних ознак класу (поміток) серед частково структурованих текстових даних.

Для більшої деталізації і візуалізації розпізнавання тексту наведено схему процесу класифікації, де p - вірогідність відношення тексту до заданих класів (рис. 1.1).

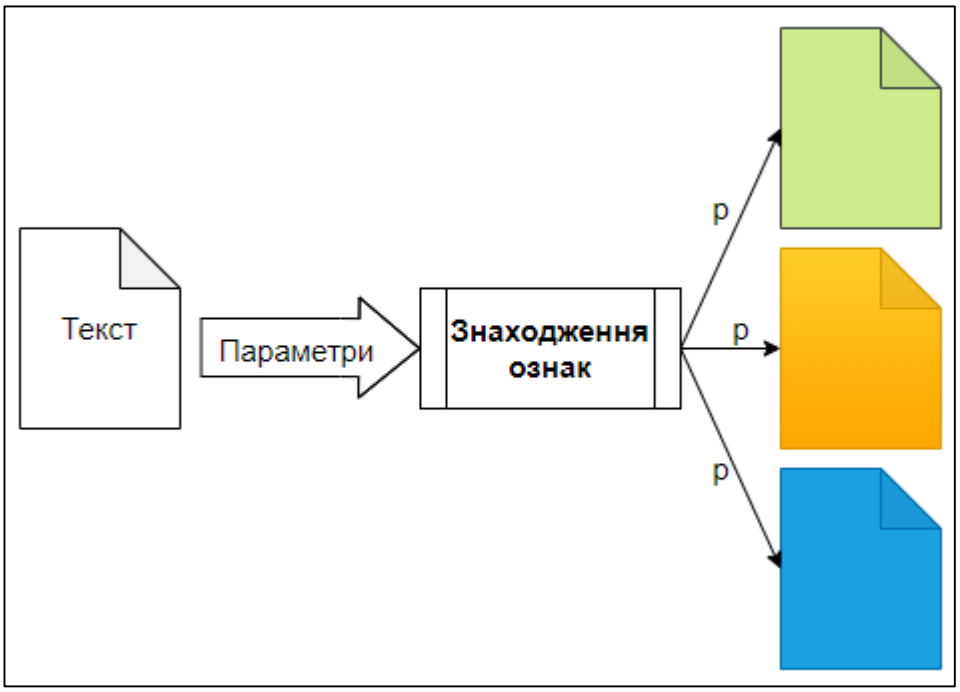

Рисунок 1.1 – Класифікація текстових документів

Математично задача класифікації описується так: мається множина документів $D = \{d_i\}$, кожному елементу d ставиться у відповідність набір ознак .Далі застосовується алгоритм класифікації для виділення документів найбільш відповідних заданому класу з непересічній підмножини класів [7]:

$$C = \{C_i\}, \bigcup_{d \in C_i}^{\forall_i} d = D, C_i \cap C_j = 0 \ (i| \neq j).$$

Під класом розуміють безліч документів, які мають певну загальну ознаку, що відрізняє цю сукупність від інших об'єктів [8].

Кластерний аналіз є більш автоматизованим підвидом класифікації. Відмінність класифікації та кластеризації: при класифікації у вас є набір зумовлених класів, ви навчаєте машину на наборі прикладів і потім хочете знати, до якого класу належить новий об'єкт. При кластеризації ви використовуєте алгоритм, який намагається згрупувати набір об'єктів і

визначити, чи існує якийсь взаємозв'язок між об'єктами, машина навчається сама.

Кластеризація – процес розбиття завдань вибірки об'єктів (спостережень) на підмножини, що не перехрещуються, які називаються кластерами, так, щоб кожен кластер складався з схожих об'єктів, а об'єкти різних кластерів істотно відрізнялися [9].

В основі математичної моделі кластерного аналізу лежить відстань між точками (1.1) на координатних осях та центрами тяжкості (1.2), де x_i – координата но осі елементарної маси Δm_i, m – маса тіла [10].

$$d = \sqrt{\sum_i^n (x_i - x_i')^2}, \tag{1.1}$$

де x та x' – точки, між якими розраховується відстань.

$$x_c = \frac{\sum_i \Delta m_i x_i}{m}, \tag{1.2}$$

де x_i – координата но осі елементарної маси, Δm_i, m – маса тіла.

Візуальне представлення віднесення елементу з набору даних до відповідного кластеру зображено на рисунку 1.2.

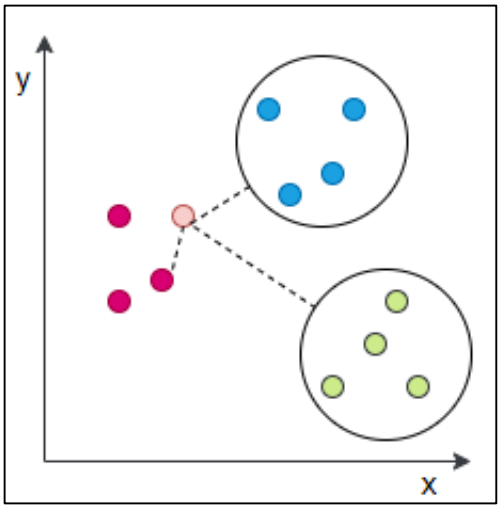

Рисунок 1.2 – Приклад кластерного аналізу

Кластеризацію застосовують для аналізу та пошуку ознак за якими можна об'єднати об'єкти, стиснення даних та пошуку новизни (що не входить до жодного кластеру) [11].

У наступних підрозділах розглянуто більш детальніше класифікацію текстових даних.

1.2 Етапи розпізнавання текстових даних

Розпізнавання текстових даних (рис. 1.3) полягають в послідовному розв'язанні наступних завдань [1]:

- вилучення інформації;
- подання інформації у вигляді текстового документу;
- попередня підготовка текстових даних;
- аналіз текстових даних.

Рисунок 1.3 – Етапи розпізнавання текстових даних

Перший етап розпізнавання інформації також включає у себе попередню підготовку даних і кластерний аналіз. Так попередня обробка зображень для витягу текстової інформації включає у себе такі аспекти: видалення шуму; видалення складних фонів; обробка освітленості; виявлення тексту у вигляді обмежувальної рамки, методами ковзного вікна, YOLO, або EAST [8]. А для попередній обробки звукових сигналів використовують вейвлет-аналіз для усунення шуму та параметризації сигналу [9,10].

Наступним етапом кластерного аналізу текстових даних є формування документів автоматично (розпізнавання символів чи сигналів), або за допомогою вводу інформації людиною через пристрої вводу.

До класифікації підлягають наступні типи даних: текстові документи, таблиці, веб-сторінкки (html файли) та інші відомості.

Останні етапи аналізу текстових даних представлено докладніше далі у наступних підрозділах.

1.3 Попередня підготовка текстових масивів

Попередню підготовку тексту можна поділити на такі етапи [12]:

- токенізація;
- фільтрація текстових даних;
- видалення стоп-слів;
- лематизація або стемінг;
- векторне представлення слів.

Токенізація це вид лексичного аналізу, який досягається розбиттям тексту на елементарні одиниці – токени. Даний аналіз є початковим етапом обробки текстів, адже дає змогу працювати зі словом як з окремою одиницею, при цьому знаючи його контекст [13].

Токенізацію тексту поділяють на такі підходи:

- розбиття на основі пробілів;
- розбиття на основі пробілів та розділових знаків;
- розбиття на основі набору регулярних виразів, які будуть застосовуватись до вхідного тексту ітеративно.

До фільтрації текстових даних відноситься видалення знаків, іншомовних слів, пропусків, обрання потрібного об’єму з текстового масиву.

Стоп-слова – слова, що не несуть в собі сенсу і не впливають на його тематику. Такими словами найчастіше є прийменники, сполучники і інші частини мови, призначені для зв'язку слів у реченнях. Приблизний список стоп-слів можна знайти за посиланням [13].

Процес видалення стоп-слів може відбуватися одразу після або одночасно при фільтрації тексту, токенізації, лематизації та стемінгу, або зовсім не відбуватися, це залежить від подальшого методу класифікації тексту.

Лема – основна або словникова форма слова. Наприклад, в українській мові це називний відмінок однини для іменника, інфінітив для дієслова і таке інше.

Лематизації - це процес, який використовує морфологічний аналіз для приведення слова до леми (нормальній формі). До методів лематизації відносять: алгоритм усічення закінчень, Stanford Core NLP, Tree Tagger [12].

Стемінг (англ. stemming) – це процес знаходження основи слова для заданого вихідного слова, тобто відсікання від слова закінчень і суфіксів. Результати стемінгу можуть бути схожі на визначення кореня слова, але його алгоритми базуються на інших принципах. Слова після стемінгу можуть відрізнятися від морфологічного кореня слова. Найкращім алгоритмом стемінгу вважається алгоритм Мартіна Портера.

Текст не можна подати безпосередньо на вхід моделі машинного навчання, його треба спочатку перевести в цифровий формат – векторне подання слова. До методів векторизації слів відносять [14]:

- CountVectorizer – цей спосіб видає значення відсутності або присутності слова в існуючому словнику;

- TfidfVectorizer – розраховує кількість входжень кожного слова у тексті, та понижує вагу з найбільшою кількістю повторень;

- Word2vec – вектор слова розраховується у контексті слів, тобто спосіб дивиться на слова, які зустрічаються разом;

- FastText – даний метод розбиває слова на кілька n-грам (частинки слів по 3 букви) і розраховує відношення слів у контексті [15];

- GloVe – основна його ідея полягає в тому, щоб отримати семантичні відносини між словами використовуючи матрицю спільного використання [16].

Таким чином, у попередній обробці, текст проходить достатню кількість перетворень, після яких складається вектор значень і підраховується статистика кожного слова і з кінцевим результатом можна переходити до класифікації.

1.4 Основні підходи та методи автоматичної класифікації текстів

У класифікації та кластерному аналізі можна виділити наступні підходи (рис 1.4). Даний поділ підходів не є чітким, адже алгоритмів класифікації існує дуже багато, і вони можуть бути комбіновані, тому слід виділити основні, найбільш використовувані груп [17-19].

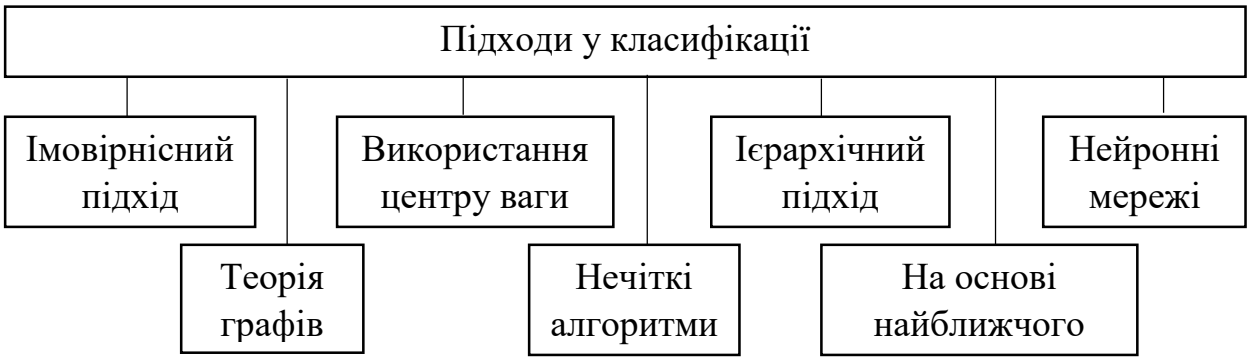

Рисунок 1.4 – Основні підходи у класифікації та кластерному аналізі

У імовірнісному підході для кожного кластеру необхідно визначити параметри розподілу, шляхом перевірки статистичну гіпотезу на закон розподілу для кожного класу, або алгоритмами, заснованих на непараметричних оцінках щільності (ЕМ-алгоритм). У даному підході данні обираються з генеральної сукупності випадково.

У підході основаному на використанні центру ваги, для кожної групи визначається вектор середніх значень показників, що інтерпретується як «центр ваги» групи. Прикладом даного підходу є алгоритми К-середніх та FOREL.

Підхід, заснований на теорії графів. Прикладом цього підходу є алгоритм найкоротшого незамкнутого шляху. Попередньо будується мінімальне остовне дерево графа, в якому вершини відповідають об'єктам, а ребра мають довжину, рівну відстані між відповідними об'єктами. Для навчання кластерів з побудованого дерева видаляються ребра максимальної довжини.

Ієрархічний підхід. Результати угруповання – дерева угруповання (дендрограмми). Алгоритми, засновані на цьому підході, поділяються на агломеративні (поетапно об'єднують найближчі групи або об'єкти) і дівізімние (в яких поетапно здійснюється поділ вихідної групи на найбільш віддалені підгрупи; ті в свою чергу також поділяються на підгрупи і т.д.). Згруповані рішення являють собою вкладену ієрархію підгруп.

У підході найближчого сусіда, класифікація здійснюється послідовно шляхом приписування об'єкту класу, в якому знаходиться найближчий об'єкт, за умови, що відстань до об'єкта не перевищує заданий поріг. Існують різні варіанти визначення відстані; при визначенні міри близькості може враховуватися і розташування інших сусідніх точок.

Нечіткі алгоритми кластерного аналізу. При використанні даного підходу передбачається, що кожен кластер являє собою нечітку множину об'єктів. До найбільш популярним алгоритмам цього сімейства можна віднести алгоритм нечітких С-середніх.

Нейронні мережі представляють собою математичну модель, а також її програмні або апаратні реалізації, побудовані за подібністю мереж нервових клітин живого організму.[19]. У процесі навчання мережі відбувається ітеративна зміна передавальних ваг між вхідними та вихідними вузлами мережі; тим самим здійснюється пошук оптимального значення критерію угруповання.

Машинне навчання також поділяються за способом їх навчання: з вчителем, без вчителя, з підкріпленням.

Але не всі методи класифікації показують хороший результат саме у аналізі текстів. Порівняння найбільш використовуваних методів класифікації текстів знаходиться у таблиці 1.1.

Таблиця 1.1 – Переваги та недоліки алгоритмів класифікації текстів

Метод	Переваги	Недоліки
Дерево рішень [20]	- висока швидкість роботи; - проста програмна реалізація алгоритму; - легка інтерпретованість результатів роботи алгоритму.	- нестійкість алгоритму щодо шуму у вихідних даних; - нестійкість алгоритму щодо великого обсягу даних.
k-середніх [21]	- можливість оновлення навчальної вибірки без перенавчання класифікатора; - відносно проста програмна реалізація алгоритму; - легка інтерпретованість результатів роботи алгоритму; - хороша класифікація у випадку з лінійно нероздільними вибірками.	- чутливість до шуму; -результат роботи залежить від критеріїв, обраних фахівцем; - велика тривалість роботи через необхідність повного перебору навчальної вибірки; - неможливість вирішення завдань великої розмірності за кількістю класів та документів.
Спосіб опорних векторів [22]	- можливість роботи з невеликим набором даних для навчання; - зведення до завдання опуклої оптимізації, має єдине рішення.	- невисока швидкість; - вимагає великого обсягу пам’яті і значних витрат машинного часу на навчання.
CNN [23]	- швидкість навчання; - висока точність класифікації; - дуже добре підходить до пошуку слів та враз.	- не враховує контекст у реченнях; - не враховує послідовність слів у тексті.

Продовження таблиці 1.1

Метод	Переваги	Недоліки
RNN [24]	- швидкість навчання; - простота реалізації.	- низькі показники точності за рахунок проблеми затухання .
Нейронні мережі Кохонена [25]	- самоорганізація мережі; - простота реалізації; - гарантоване отримання відповіді після проходження даних по нейронній мережі.	- не сама висока точність класифікації тексту.
LSTM [26]	- враховує контекст; - висока точність результату; - має шари довгостроковій та короткочасній пам'яті.	- велика кількість затрат пам’яті та часу розрахунку.
GRU[27]	- враховує контекст; - висока точність результату; - має шари оновлення та забування, подібні LSTM; - менше параметрів ніж LSTM.	- достатня кількість затрат пам’яті та часу розрахунку.

Отже, ідеального методу поки що не існує. Обрання методу класифікації тесту залежить від потреб, на приклад, якщо необхідно визначити нові класи або кластери то підходить метод k-середніх, якщо немає навчальної та тестової вибірок, то дерева рішень, або спосіб опорних векторів. Нейронні мережі мають більш чіткі результати, але й займають більше ресурсів та часу на навчання.

У таблиці 1.1 представлені тільки найвідоміші і більш використовувані нейронні мережі, але також існують інші й більш нові версії з наведених, таких як GRU, LSTM, CNN. Також існують комбінації даних методів, які показують високі результати.

1.5 Дослідження сучасних програмних засобів класифікації текстів

Для постановки критеріїв до розробки моделі класифікації тестових документів необхідно дослідити сучасні програмні засоби.

До сучасних існуючих програмних засобів класифікації тексту відносяться такі програмні додатки:

- MegaIndex;
- Amazon Comprehend;
- Mediatoolkit;
- ABBYY FlexiCapture.

MegaIndex є комплексною системою автоматизації просування та аналітики. Сервіс дає користувачам можливість відстежувати динаміку зміни позицій та оцінювати ефективність просування. Держава виробник – Росія.

Визначення тематики тексту програмою MegaIndex дозволяє аналізувати якість контенту з точки зору робота, що допоможе покращити його видимість наповнюючи сайт, або його категорій, які містять одну тематику. Для визначення рубрики текстів система використовує власний алгоритм та базу даних, зібрані роботом, що індексує мільйони сторінок сайтів. Вони дозволяють запропонувати три максимально відповідні тематики щодо збігу вмісту [28].

Amazon – американська компанія, найбільша у світі на ринках платформ електронної комерції та публічно-хмарних обчислень щодо виручки та ринкової капіталізації. Штаб-квартира – у Сіетлі.

Amazon Comprehend – це сервіс обробки природної мови (NLP), в якому для пошуку цінної інформації та взаємозв'язків у тексті застосовуються технології машинного навчання. Amazon Comprehend вирішує такі завдання [29]:

- аналіз роботи кол-центру;
- індексування оглядів продуктів та пошук по них;
- автоматична класифікація документів;
- сортування документів.

Mediatoolkit – сервіс сентимент аналізу у коментарях та повідомлень соціальних мереж, таких як Twitter, Instagram, YouTube та Facebook. Держава виробник – Словенія [30].

ABBYY – міжнародна компанія-розробник рішень у галузі інтелектуальної обробки інформації та аналізу бізнес-процесів, розпізнавання текстів (OCR) та лінгвістики. Головний офіс знаходиться у Москві.

ABBYY FlexiCapture – універсальна платформа для інтелектуальної обробки інформації. Рішення дозволяє вилучати дані з будь-яких типів вхідних документів: наприклад, відсканованих паперів, фотографій, електронних листів або вкладень. Рішення класифікує, розпізнає документи, витягує дані, верифікує та передає їх у корпоративні інформаційні системи. Для класифікації текстів використовуються згорткові нейронні мережі.

ABBYY FlexiCapture може виступати єдиною корпоративною платформою для багатьох бізнес-процесів, таких як управління взаємовідносинами з клієнтами, виробнича, закупівельна та юридична діяльність компанії. [31].

Для узагальнення результатів дослідження програмних засобів класифікації текстових даних проведена порівняння характеристика багатокритеріальним методом (табл. 1.2).

Таблиця 1.2 – Порівняльна характеристика програмних засобів

Параметр	MegaIndex	Amazon Comprehend	Mediatoolkit	ABBYY FlexiCapture
Вид додатку	Web-сервіс	Web-сервіс	IOS-сервіс	Програмне забезпечення
Підтримка мови	Російська, англійська	Більшість мов	Більшість мов	Більшість мов
Підтримка форматів даних	URL	URL, текстові документи,.	URL	URL, текстові документи, зображення , аудіо-файли
Ціна	0,2 USD /стр.	0,0001 USD за 100 слів	-	-
Сентимент аналіз	-	+	+	-
Аналіз тематики	+	+	-	+

Продовження таблиці 1.2

Параметр	MegaIndex	Amazon Comprehend	Mediatoolkit	ABBYY FlexiCapture
Добавлення нового класу	-	+	-	+
Сортування документів	-	+	-	+
Статистика класів	+	+	+	+

Отже, найбільш універсальними додатками є Amazon Comprehend та ABBYY FlexiCapture. Вони працюють з різними форматами файлів, можуть сортувати їх. Amazon Comprehend також може проводити як аналіз тематики так і сентимент аналіз окремо. Але на даний момент не існує додатку, який зможе провести класифікацію за емоційною окраскою та тематикою одночасно, тобто мультикласифікацію.

1.6 Актуальність застосування розпізнавання текстових документів

Класифікація займає одне з центральних місць серед методів інтелектуального аналізу даних і має достатню широку область застосування:

- виявлення сутностей;
- виділення тематики з тексту;
- сентимент аналіз;
- знаходження найближчого сусіду документа;
- пошук документа за параметрами або ключовим словом.

Розпізнавання текстових даних може застосовуватися майже в усіх сферах суспільної діяльності: юриспруденції, медицині, науковій діяльності, а особливо у бізнесі та інтернет-мереж.

Класифікація документів у бізнесі та інтернет-мережі автоматизує такі задачі:

- розділення веб-сторінок та сайтів за тематичними каталогами;
- боротьба зі спамом у повідомленнях;

- боротьба зі некультурною лексикою у коментарях;

- аналіз якості товарів чи послуг за відгуками;

- визначення мови тексту;

- розподіл документів, та пошук відповідного;

- показ більш релевантної реклами.

Однією з головних задач розпізнавання тесту є оцінка тональності текстів, яка призначена для отримання зворотного зв'язку між споживачами послуги та виробниками послуги наприклад [32]:

- оцінка діяльності держустанови в ЗМІ і в соціальних мережах;

- можливість передбачення котирувань (в якості додаткового індексу) на фондових ринках;

- моніторинг екстремістських висловлювань у ЗМІ, інтернеті.

Таким чином, розпізнавання тексту є достатньо актуальною темою, а особливо, коли весь документообіг переходить до електронного типу зберігання даних.

1.7 Висновки до розділу

У першому розділі були розкриті поняття інтелектуального аналізу даних, класифікації та кластеризації текстових даних. Також було визначено основні етапи класифікації, більш детально розглянуто попередню підготовку текстових масивів, та досліджено основні підходи та методи класифікації документів. Було досліджено сучасні програмні засоби класифікації текстових документів, а також актуальність застосування класифікації тестів у сучасному світі.

Таким чином, у якості підходу розпізнавання тексту буде обрано нейронні мережі тому, що вони дають більш точні результати та використовуються у сучасних існуючих програмних засобів. У якості методу класифікації краще обрати гібридну модель, но основі поєднання CNN та GRU, так як вони дають більш чіткі та найшвидші результати.

РОЗДІЛ 2
ОБҐРУНТУВАННЯ МАТЕМАТИЧНОЇ МОДЕЛІ РОЗПІЗНАВАННЯ ТЕКСТУ

2.1 Постановка завдання

Формальна постановка завдання класифікації описується таким чином.

Є безліч документів $D = \{d_1, \dots, d_{|D|}\}$ та безліч можливих класів $C = \{c_1, \dots, c_{|C|}\}$. Невідома цільова функція $\Phi: D \times C \to \{0, 1\}$ задається формулою (2.1).

$$\Phi(D_j, C_i) = \begin{cases} 0, \text{якщо } d_j \notin c_i \\ 1, \text{якщо } d_j \in c_i \end{cases}. \quad (2.1)$$

Потрібно побудувати класифікатор Φ', максимально близький до Φ.

При однозначній класифікації кожен документ $d \in D$ може відповідати лише одному класу $c \in C$. При багатозначній класифікації кожен документ d ∈ D може відповідати декільком класах одночасно $c_i \in C$ [33].

Класифікація називається точною, якщо класифікатор видає точну відповідь:

$$\Phi': D \times C \to \{0, 1\}. \quad (2.2)$$

Класифікація називається пороговою, якщо класифікатор визначає ступінь подібності (Categorization Status Value) документа:

$$CSV: D \to [0, 1]. \quad (2.3)$$

У якості метода класифікації було обрано нейронні мережі, навчання з вчителем, сутність якого полягає у наступному. Обирається набор прикладів навчальна вибірка L, де відомо відношення до класу, але з не відомими параметрами. Навчальну вибірку використовують для навчання класифікатора та визначення значення його параметрів, при яких класифікатор видає найкращий результат. Після навчання класифікатора перевіряється його якість тестовим набором T. При цьому необхідно, щоб виконувались такі умови:

$$L \cap T = \emptyset, \quad (2.4)$$

$$\Omega = L \cup T \subset C \times D. \quad (2.5)$$

Отже, необхідно вибрати такі класифікатор і навчальну вибірку, які дадуть найбільш чіткі результати.

2.2 Критерії оцінки якості нейронних мереж

Підготувавши модель необхідно адекватно оцінити її якість. Для цього визначмо такі поняття:

- TP (True Positive) – істино позитивний, виданий результат класифікатора відповідає дійсному заданому класу;

- FP (False Positive) - хибно позитивний, класифікатор видав не вірний клас з заданих, помилка першого роду;

- FN (False Negative) - хибно негативний, класифікатор не видав заданий клас, помилка другого роду;

- TN (True Negative) - істино негативний, класифікатор не видав невірний результат [34].

Найпростіша метрика – це метрика достовірності (англ. Accuracy) (2.6).

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN}. \quad (2.6)$$

Але ця метрика не має бути єдиною метрикою моделі, особливо у тих випадках, коли існує перекіс у вибірці, тобто представники різних класів зустрічаються з різною імовірністю.

Для оцінки якості роботи алгоритму кожному з класів окремо введемо метрики точність і повнота [35].

Точність (англ. precision) – відношення правильно вгаданих об'єктів класу до всіх об'єктам, які ми визначили як об'єкти класу (2.7).

$$\mathrm{P}recision = \frac{TP}{TP + FP}. \quad (2.7)$$

Повнота (англ. recall) – відношення правильно вгаданих об'єктів класу до всіх представникам цього класу (2.8).

$$Recall = \frac{TP}{TP + FN}. \quad (2.8)$$

Для узагальнення критеріїв точності і повноти існує F-мера (англ. F score) - середня гармонійна точність і повнота (2.9).

$$F = (1 + \beta^2)\frac{\mathrm{P}recision \cdot Recall}{\beta^2 \cdot \mathrm{P}recision + Recall}, \quad (2.9)$$

де β - точність метрики, зазвичай дорівнює одиниці.

Критерієм оцінки якості навчання нейронної мережі є функція втрат, яка вимірює наскільки нейронна мережа відповідає щодо даної навчальної вибірки та очікуваних відповідей. Вона також може залежати від таких змінних, як ваги та зміщення.

Деякі відомі функції втрат:

- квадратична (середньоквадратичне відхилення);

- крос-ентропія;

- експонентна (AdaBoost);

- відстань Кульбака — Лейблера або збільшення інформації.

2.3. Очікувані наукові та практичні результати роботи

Дослідження наукових публікацій [36-38], за останні два роки, показують, що поєднання різних нейронних мереж дають високі показники якості класифікації текстових документів, а особливо поєднання CNN та bi LSTM.

Архітектура CNN та Bi LSTM зображена на рисунку 2.1. Дана схема показу наступну послідовність шарів моделі [38]:

- шар представлення слів у векторній формі Embedding;

- шар одномірної згорткової нейронної мережі; який зменшує розміри вхідних даних

- шар Bi LSTM, який прогнозує;

- щільний шар для розподілу класів.

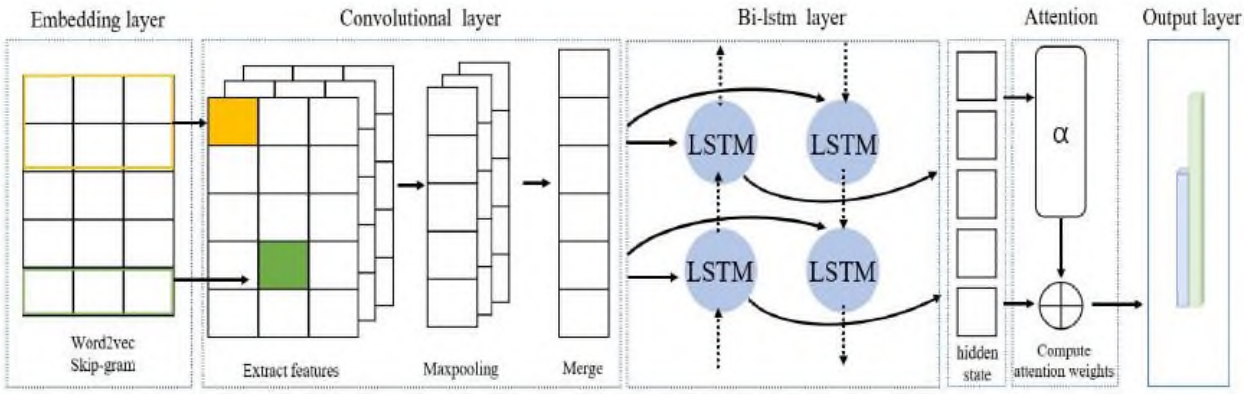

Рисунок 2.1 – Архітектура Conv Bi LSTM [36]

Результати експерименту за джерелом [38] представлені на рисунках 2.2 та 2.3, де Proposed – запропонована модель Conv Bi LSTM.

Таким чином, дана модель показує найвищі показники достовірності класифікації текстових даних. Але нейронна мережа LSTM потребує достатньо багато обчислювальних затрат. Рішенням цієї проблеми є більш нова подібна модель GRU (вентильні рекурентні вузли), у якої менше параметрів та обчислювальних процесів. За дослідженнями [39, 40] GRU не тільки має високу швидкість, а й більшу достовірність при вирішенні задачі класифікації текстів (рис 2.4, рис. 2.5).

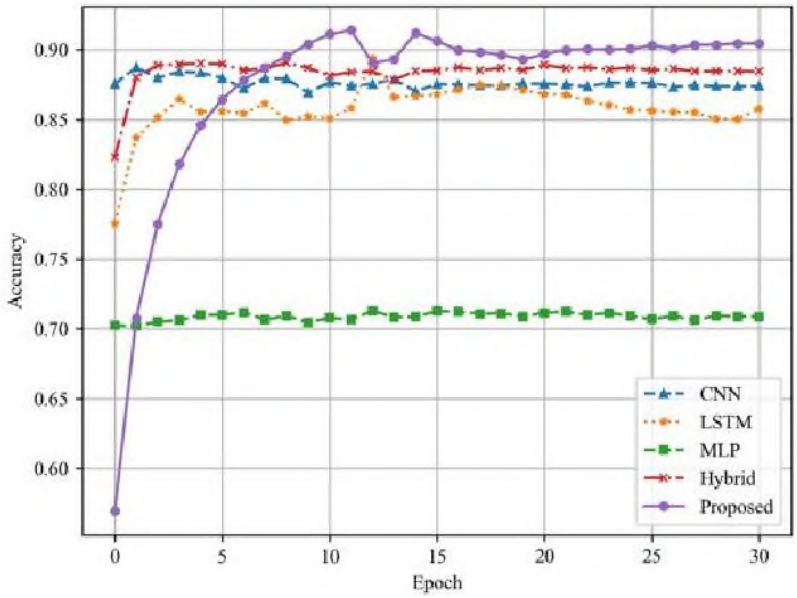

Рисунок 2.2 – Достовірність моделі по епохам навчання [36]

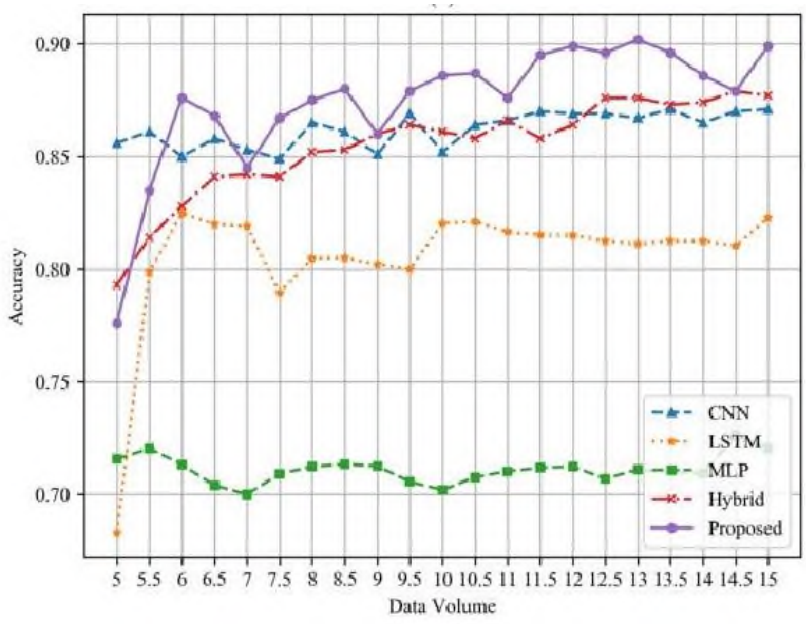

Рисунок 2.3 – Достовірність моделі по об'єму тестових даних [38]

Neural Network	Character Input			Word Input		
	P	R	F1	P	R	F1
Bi-GRU	**97.81%**	97.69%	97.75%	**99.22%**	99.10%	**99.16%**
Bi-LSTM	97.69%	**97.88%**	**97.78%**	99.14%	**99.11%**	99.12%
GRU	91.15%	85.81%	88.40%	94.62%	91.87%	93.22%
LSTM	90.65%	86.56%	88.56%	93.99%	91.92%	92.94%

Рисунок 2.4 – Порівняльний аналіз нейронних мереж за показниками якості [40]

Neural Network	Character Input	Word Input
Bi-GRU	**517 s/epoch**	**521 s/epoch**
Bi-LSTM	562 s/epoch	567 s/epoch
GRU	**269 s/epoch**	**270 s/epoch**
LSTM	292 s/epoch	293 s/epoch

Рисунок 2.5 – Порівняльний аналіз нейронних мереж за часом навчання [40]

Отже, в ході подальшого дослідження буде використана гібридна модель згорткові бі-направлені вентильні рекурентні вузли (Conv Bi GRU). Передбачається, що при дослідженні, дана модель буде кращою за параметром

швидкості та давати не меншу достовірність результатів у порівнянні з Conv Bi LSTM.

Практична цінність полягає у розробці програмного додатку багатозначної класифікації текстових даних на основі моделі Conv Bi GRU. У якості результату класифікації буде поєднано два підкласи: аналіз за темою та сентимент аналіз. Запропонована багатозначна класифікація скоротить час у порівнянні з поодинокій класифікації, дасть можливість більшої автоматизації, та дасть більш ємке вилучення знань з даних, адже у результаті буде класифікація по кожному виду продукту та за відліком споживача. Програмний додаток призначений для аналізу коментарів у інтернет магазинах, але й ще можливо пристосувати модель до угрупування новин, інтернет каналів.

Наукова новизна полягає у розробці, оцінки якості та застосування для багатозначної класифікації тексту моделі Conv Bi GRU, яка ще немає наукових результатів дослідження.

2.4 Висновки до розділу

У другому розділі було поставлене завдання магістерської дисертації. Формальна постановка завдання класифікації. Були описані основні параметри якості розпізнавання тексту нейронними мережами. А також були визначені практична та наукова цінність магістерської дисертації та наукова новизна.

Отже, головною метою є розробка програмного додатку багатозначної класифікації текстових даних на основі розробки нейронної мережі Conv Bi GRU.

РОЗДІЛ 3
ПРОЄКТУВАННЯ АРХІТЕКТУРИ МОДЕЛІ РОЗПІЗНАВАННЯ ТЕКСТУ

3.1 Загальний опис програмної моделі розпізнавання тексту

Назва програмного додатку – Text Classification .

Основна мета програмного застосунку – двомірна класифікація за параметрами аналізу тематики та сентимент аналізу коментарів споживачів у інтернет-магазинах.

Двомірна класифікація буде основана на використанні розробленої нейронної мережі Conv Bi GRU.

Текстові данні можна бути завантажити по URL адресі веб-сторінки, з текстових документів форматів docx, txt, а також з таблиці Excel.

Результати класифікації будуть відображатися у табличному та графічному форматах, які можливо зберегти у таблиці Excel.

Також за допомогою програмного додатку можна бути створити власну класифікацію і навчити її.

Сам комп’ютерний додаток буде у форматі exe, що робить програму незалежної від інтерпретатору. Але даний формат запускається лише у Windows.

Типи користувачів у програмній системі: власники інтернет-магазинів, або їх працівники, а також усі інші заінтересовані особи.

Управління програмою буде виконуватися такими подіями:

- завантаження текстових даних;
- проведення класифікації;
- збереження результатів класифікації;
- створення власної класифікації текстів;
- навчання власної класифікації текстів.

3.2 Архітектура моделі класифікації текстів

Архітектура моделі класифікації текстів Conv Bi GRU (рис. 3.1) розроблена на основі моделі Conv Bi LSTM, яка розглянута у другому розділі.

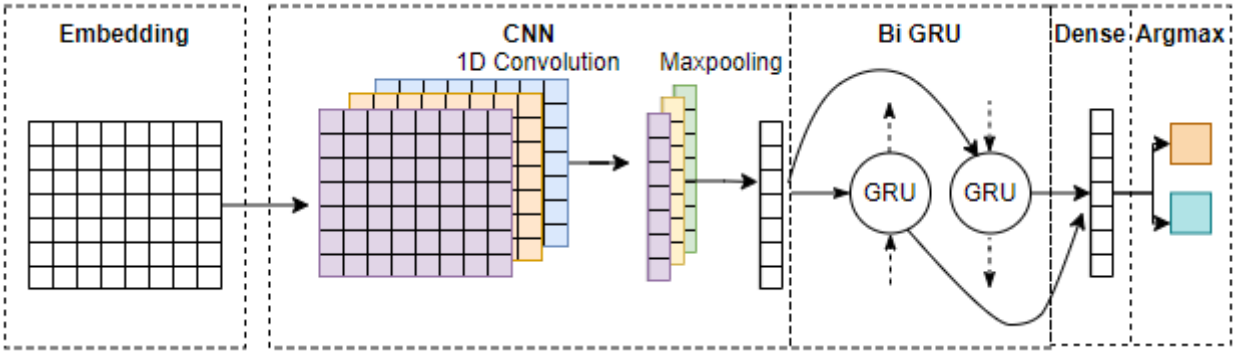

Рисунок 3.1 – Архітектура моделі класифікації текстів Conv Bi GRU

Розглянемо більш детальніше кожен з етапів моделі Conv Bi GRU.

На першому етапі Embedding відбувається наступна послідовність дій:

- токенізація слів;
- очищення від стоп-слів;
- лематезація слів;
- кодування слів за методикою one-hot-encoding;
- формування embedding матриці;
- множення вектору слова one-hot-encoding на матрицю embedding, у результаті чого формується вектор слова [41, 42].

На другому етапі застосовується шар CNN. Роль процесу згортки полягає у вилученні ознак із введеного тексту. Згорткові шари використовуються для виділення низькорівневих семантичних ознак із оригінального тексту та зменшення кількості вимірів на відміну від Bi-GRU, який обробляє текст як послідовні дані [43].

Так само, як і в типовій згортковій нейронній мережі, одного згорткового ядра недостатньо для виявлення всіх різних видів функцій, які будуть корисні для завдання класифікації. Щоб налаштувати мережу так, щоб вона могла вивчати різноманітні відносини між словами, знадобиться багато

фільтрів різної висоти. У науковій статті «Згорткові нейронні мережі для класифікації речень» [43] використовується всього 300 ядер; 100 ядер для кожної висоти: 3, 4 і 5. Ці висоти ефективно фіксують шаблони в послідовних групах по 3, 4 і 5 слів. Було обрано обмеження з 5 слів, оскільки слова, які були далі, як правило, були менш релевантними або корисними щодо визначення шаблонів у фразі. Складені вихідні вектори ознак, які виникають у результаті кількох з цих згорткових операцій, називаються згортковим шаром [44].

Процес згортання описується за формулою 3.1 [45].

$$h_{d,t} = \tan h(W_d \cdot x_{t:t+d-1} + b_d), \quad (3.1)$$

де $x_{t:t+d-1}$ – вектори вбудовування слів у вікні, W_d – матриця ваг, b_d – зміщення, t-ї згортки, вікна з d слів.

Для кожного ядра згортки застосовуємо max-overtime pooling до карт об’єктів, що відповідає конкретному фільтру (3.2) і об’єднуємо p_d для кожного розміру фільтра d (3.3).

$$p_d = \max\{h_{d,t}\}. \quad (3.2)$$

$$h_d = [p_1, p_2, p_3]. \quad (3.3)$$

Далі параметри h_d передаються на вхід нейронної мережі GRU, алгоритм якої реалізується у наступній послідовності вирішення таких операцій (рис 3.2):

$$z_t = \sigma(w_{zx}x_t + u_{zh}h_{t-1} + b_t); \quad (3.4)$$

$$r_t = \sigma(w_{rx}x_t + u_{rh}h_{t-1} + b_t); \quad (3.5)$$

$$\tilde{h}_t = \tan h\,(w_{hx}x_t + r_t \odot u_{hh}h_{t-1} + b_t); \quad (3.6)$$

$$h_t = (1 - z_t)\odot\tilde{h}_t + z_t\odot h_{t-1}, \quad (3.7)$$

де σ – сигмовидна функція активації, яка коливається від 0 до 1, $\odot$ – це добуток Адамара матриць, w і u – вагові матриці, b – зсув матриці, z – вектор вузла уточнення, r – вектор вузла скидання, h – вектор виходу [46].

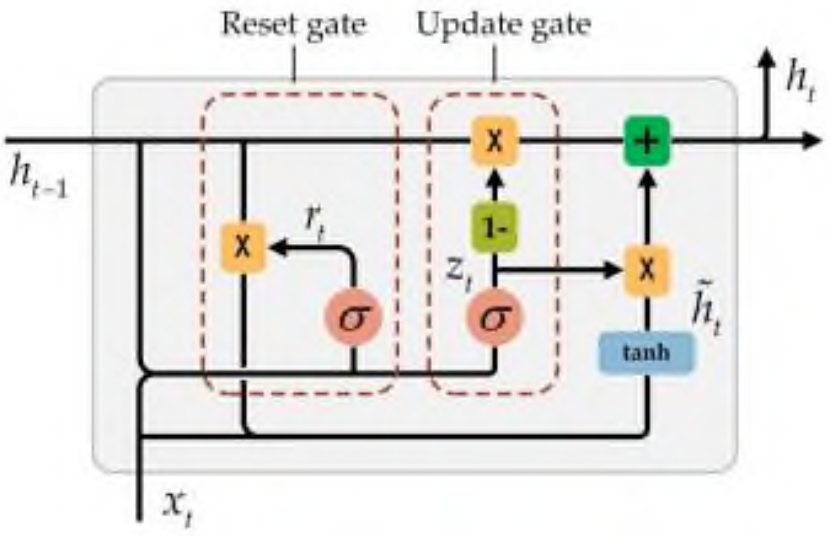

Рисунок 3.2 – Архітектура GRU [46]

У двонаправлених вентильних рекурентних вузлах на виході формуються два вектору виходу (рис. 3.3). Для отримання остаточного вихідного вектору необхідно об’єднати два попередніх (3.8).

$$score(\vec{h}, \overleftarrow{h}) = \tanh(V^T [\vec{h} + \overleftarrow{h}]), \quad (3.8)$$

де V^T – ваговий вектор.

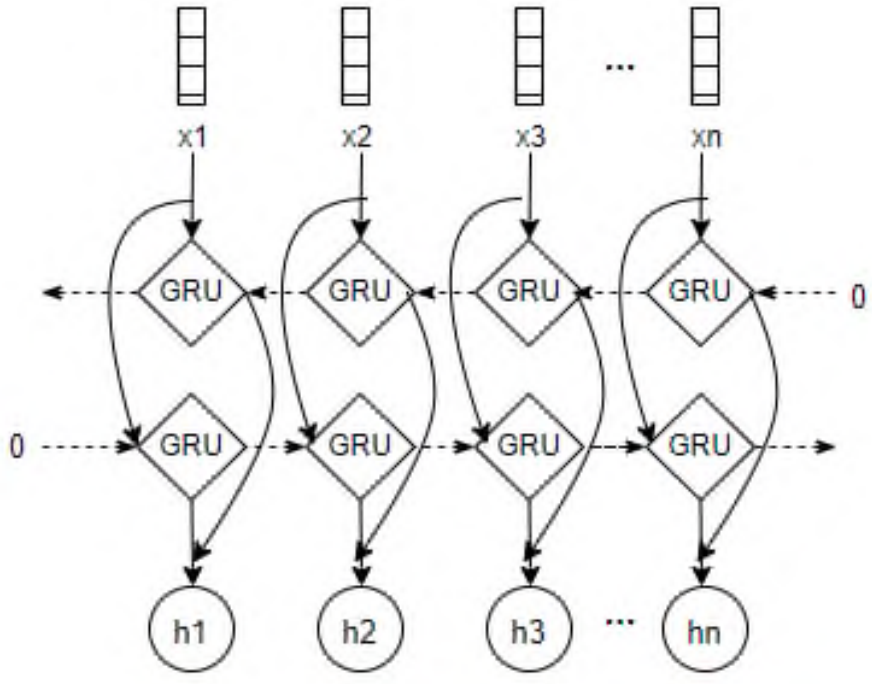

Рисунок 3.3 – Структура Bi-GRU

Два шари GRU можна розраховувати паралельно. Тоді швидкість Bi-GRU майже буде дорівнювати звичайної GRU. Тоді різниця буде тільки у останньому кроці Bi-GRU.

Наступним кроком у моделі є повно зв’язковий багатошаровий перцептрон, який формує вірогідність відношення вхідного вектору до кожного класу.

Найбільш поширений метод навчання багатошарового перцептрону – метод зворотного розповсюдження помилки. Суть даного методу полягає в тому, що сигнал помилки кожного вихідного значення, розрахований на поточному такті навчання, поширюється по верствах у зворотному напрямку (від вихідного до першого) з урахуванням тих же вагових коефіцієнтів, які використовувалися при прямому проходженні вхідних сигналів по нейронній мережі розрахунку вихідних значень.

Алгоритм методу зворотного поширення помилки включає такі етапи.

Перший етап. Вагові коефіцієнти багатошарового перцептрону обраної структури ініціалізуються невеликими за абсолютною величиною (не більше $M^{-1}(L\ +\ 1)^{-1}$) випадковими значеннями, де L - кількість прихованих шарів, M – кількість входів.

Другий етап. На входи нейронної мережі подається вхідний вектор одного з прикладів навчальної вибірки. Проводиться пряме поширення сигналів по мережі з розрахунком значень вихідних змінних $\tilde{y}_j^p$.

Третій етап. Для кожного розрахованого значення вихідний змінної по співвідношенню $\Delta_j =\ y_j^n - y_j^p$ обчислюється похибка порівняно зі значеннями елементів вихідного вектора взятого навчального прикладу y_j^n.

Четвертий етап. За отриманими похибками розраховуються нев'язки нейронів вихідного шару з урахуванням похідних відповідних активаційних функцій:

$$\delta_j =\ \Delta_j \cdot f'(s_j), \tag{3.9}$$

$$s =\ w_0 +\ \sum_{i=1}^{m} w_i x_i\ , \tag{3.10}$$

$$f(s) = \frac{1}{e^{-\alpha \cdot s}+1}, \quad (3.11)$$

$$f'(s) = \alpha \cdot f(s) \cdot (1 - f(s)). \quad (3.12)$$

Якщо активаційна функція має вигляд сигмоїди чи функції Гауса, слід скористатися співвідношеннями для аналітичного розрахунку її похідної.

П’ятий етап. У зворотній послідовності (від останнього прихованого шару до першого) розраховуються нев'язки нейронів інших шарів з урахуванням сполучних шарів синаптичних зв'язків:

$$\delta_i = \left(\sum_{j=1}^{m_i} w_{ij}\delta_j\right) \cdot f'(s). \quad . (3.13)$$

Шостий етап. Для всіх шарів нейронів, крім першого, перераховуються значення вагових коефіцієнтів за такими співвідношеннями:

$$w_{0j}^{(q+1)} = w_{0j}^{(q)} + v\delta_i\,, \quad (3.14)$$

$$w_{ij}^{(q+1)} = w_{ij}^{(q)} + v\delta_i\tilde{y}_i^p, \quad (3.15)$$

де v - коефіцієнт швидкості навчання, який задається позитивною константою або змінною величиною $(0 < v <= 1)$, що поступово зменшується в процесі навчання нейронної мережі.

Для першого прихованого шару нейронів корекція вагових коефіцієнтів відбувається по нормалізованим значенням вхідних змінних нейронної мережі:

$$w_{ij}^{(q+1)} = w_{ij}^{(q)} + v\delta_i\tilde{x}_i. \quad (3.16)$$

Для розрахунку коефіцієнтів усунення нейронів першого прихованого шару, як і всіх інших шарів, використовується співвідношення (3.14).

Сьомий етап. Цикл повторюється з кроку два до виконання однієї чи кількох умов закінчення:

- вичерпано задану граничну кількість епох навчання;

- досягнуто задовільний рівень помилки по всій навчальній вибірці;

- не відбувається зменшення помилки навчальної вибірки протягом заданої граничної кількості епох навчання;

- вичерпано заданий граничний фізичний час навчання [47].

На останньому кроці Argmax алгоритму моделі Conv Bi GRU обираються класи переможці за найвищими показниками повно зв’язкового багатошарового перцептрону.

3.3 Архітектура програмного застосунку

Архітектура програмного забезпечення описується UML діаграмами варіантів користування та станів.

Користувач програмного застосунку зможе виконувати дві основні дії: проведення класифікації тексту та створити власну класифікацію (рис. 3.4).

Рисунок 3.4 – Діаграма варіантів користування

За рисунком 3.4 бачимо, що до проведення класифікації текстів включається завантаження даних та сама класифікація. Завантажити текстові дані можливо різними способами, а також різними форматами документів. Та якщо у програмі не існує вірного роздільника текстів, можливо буде ввести його у текстове поле додатку. Усі результати класифікації можливо буде зберегти у файлі excel.

Також користувач зможе створити власну класифікацію, що включає в себе введення параметрів класифікації та навчання новоствореної моделі. До параметрів класифікації включається наступне: кількість класів, котрі будуть на виході класифікації, назва класифікатору.

Більш детальніше розглянемо послідовність дій користувача за діаграмами станів (рис. 3.5, 3.6).

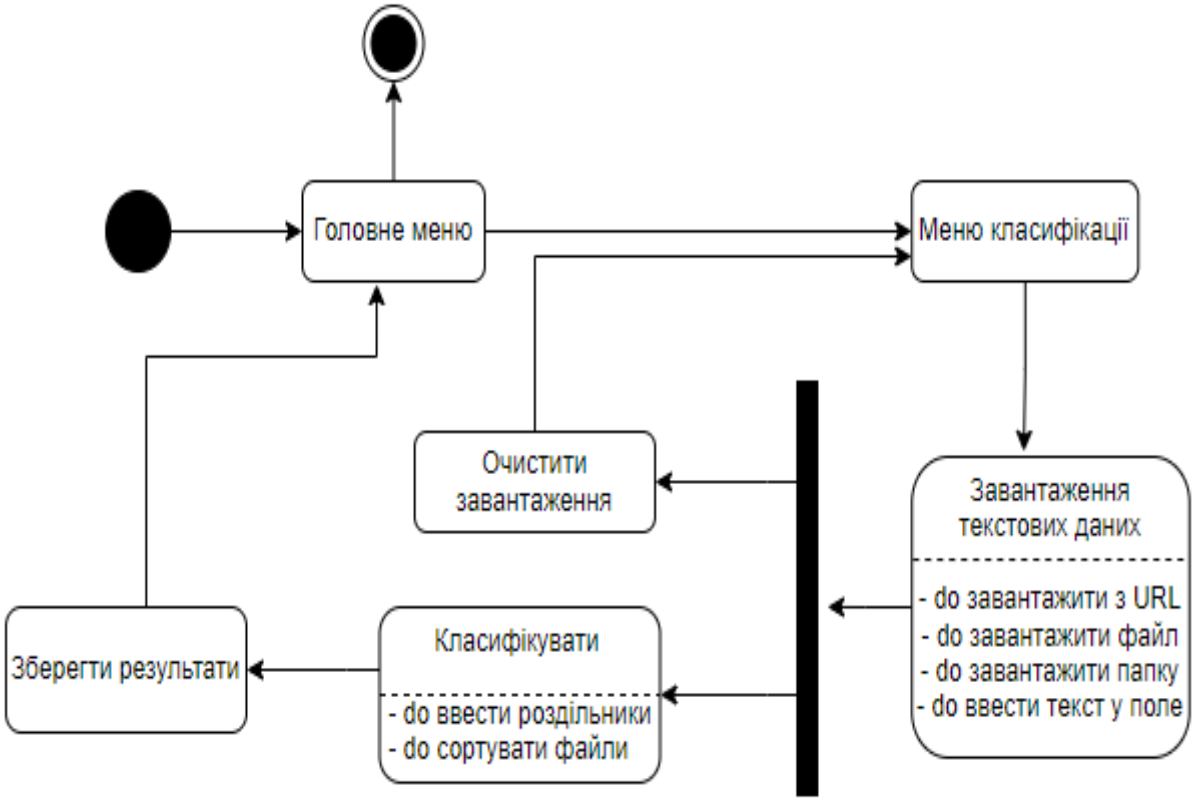

Рисунок 3.5 – Діаграма станів класифікації тестів

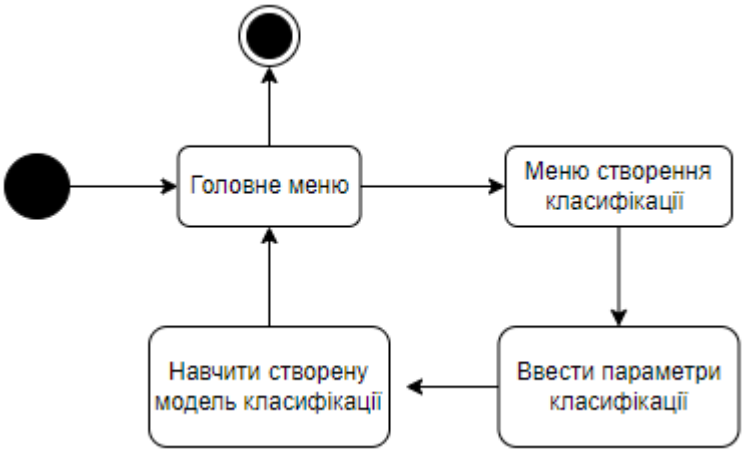

Рисунок 3.6 – Діаграма станів створення нового класифікатору

Отже, програма починається з головного меню і їм же завершується. Керування програмою та перехід між станів відбувається натисканням відповідним керуючим інтерфейсом програмного додатку.

3.4 Висновки до третього розділу

Отже, у третьому розділі було описано загальні відомості про програмну модель розпізнавання тексту. Визначено, що користувачами програмного забезпечення можуть бути власники або працівники інтернет магазинів на яких є можливість написати відгук про товар чи послугу. а також користувачами можуть бути усі інші заінтересовані особи, яким необхідно провести розпізнавання тексту за класами.

Була розроблена архітектура моделі розпізнавання текстів і детально описано кожний крок алгоритму роботи моделі класифікації текстів Conv Bi GRU, а саме крок Embedding, крок згорткової нейронної мережі, основний шар Bi GRU, крок повно зв’язного багатошарового перцептрона та крок вибору класів.

А також була розроблена архітектура програмного додатку і було з’ясовано, що користувач зможе виконувати у програмі дві основні дії: провести класифікацію та створити власний класифікатор.

РОЗДІЛ 4
РОЗРОБКА ТА РЕАЛІЗАЦІЯ МОДЕЛІ РОЗПІЗНАВАННЯ ТЕКСТУ

4.1 Характеристика технічної бази розробки програмного додатку

4.1.1 Обґрунтування вибору мови програмування

Для розробки штучного інтелекту використовують п’ять мов програмування [48]: Python, R, C++

Python зарекомендував себе в якості основної мови програмування для розробки штучного інтелекту. Однією з причин, чому вона краще для штучного інтелекту, є її простота. Синтаксис програмування на Python може бути легко вивчений будь-яким, хто займається програмуванням. Те ж саме відноситься і до реалізації алгоритмів цієї мови. Python - це універсальна мова, яка підтримує різні стилі програмування. Вона включає в себе об'єктно-орієнтоване, функціональне і процедурне програмування. Мова має безліч бібліотек, які підтримують штучний інтелект, таких як Keras, Tensorflow, Pytorch, Numpy та інші.

Мова R широко відома завдяки аналізу та обробці даних. В основному використовується в області статистики. Програміст може використовувати R для створення математичних символів, графіків і формул, коли це необхідно. Вона включає в себе моделі G, RODBC, Tm і Class, що спрощує процес реалізації алгоритмів машинного навчання.

C++ складна, але дуже швидка мова програмування. На C++ написані бібліотеки, пакети для інших мов програмування, а також написані інші мови такі як Python, Java. У C++ також з’являються бібліотеки для штучного інтелекту, наприклад OpenCV

З трьох вище перерахованих мов програмування у розробці програми буде використана Python тому, що ця мова має динамічний розвивиток і має безліч безкоштовних відкритих бібліотек та широкий спектр можливостей.

4.1.2 Обґрунтування вибору програмного середовища

IDE (або інтегроване середовище розробки) - це програма, призначена для розробки програмного забезпечення. Програмне середовище включає у себе: редактор, призначений для роботи з кодом (наприклад, підсвічування синтаксису), інструменти збірки, виконання та налагодження, і певну форму системи управління версіями.

У 2020 році виділяють найкращими наступні IDE і редактори коду [49]:

- Jupyter;
- Pycharm;
- Spyder;
- PyDev;
- Idle;
- Visual Studio;
- Атом;
- Wing.

За причиною того що, програмний додаток потрібно буде запускати з самого середовища краще використовувати середовище Spyder. Також IDE Spyder має наступні переваги [50]:

- це вільна і кроссплатформена інтерактивна IDE для наукових розрахунків;
- інтеграція з науковими бібліотеками Python - NumPy, SciPy, Matplotlib, Pandas;
- підтримка одночасного використання безлічі консолей Python;
- гнучке налаштування інтерфейсу;
- постійне розширення функціональності.

Для успішного запуску Spyder краще установити платформу Anaconda. Anaconda - платформа мов програмування Python і R, що включає набір популярних вільних бібліотек і деякі середовища програмування, а також консольні вікна для встановлення інших бібліотек.

4.1.3 Технічні характеристики персонального комп’ютера

У процесі проектування та розробки використовується ноутбук Dell Inspiron 15 Series 5000 з наступними характеристиками:

- операційна система: Windows 10 корпоративна версія;
- процесор: Intel® Core™ i5-7200U CPU @ 2.50GHz 2.70GHz;
- кількість ядр: 2;
- тип системи; 64-розрядна;
- відеокарта: AMD Radeon R7 M445;
- оперативна пам’ять: 8.00 ГБ;
- жорсткий диск пам’ять: 1 ТБ.

4.2 Програмна розробка моделі розпізнавання тексту

Модель розпізнавання тексту реалізується у двох класах: Bi GRU та conv Bi GRU (рис. 4.1). Як видно з рисунком 4.1, клас conv Bi GRU наслідує клас Bi GRU. Розглянемо більш детально кожен клас.

Клас conv Bi GRU має такі функції:

- init – ініціалізація параметрів;
- get_parameters – функція отримання параметрів;
- get_test_data – функція завантаження тестових даних;
- get_train_data – функція завантаження тренувальних даних;
- normalize – функція нормалізації текстових даних з використанням пакету Tokenizer;
- creat_x_y – зчитування текстових даних з файлу для навчання моделі;
- creat_model – створення моделі conv Bi GRU, за допомогою пакетів Conv1D, MaxPooling1D, Input, concatenate, Dense, Embedding та створеного класу Bi GRU;
- learn_model – функція навчання моделі;
- test_model – функція тестування моделі та оцінки показників якості;
- save_model – збереження моделі.

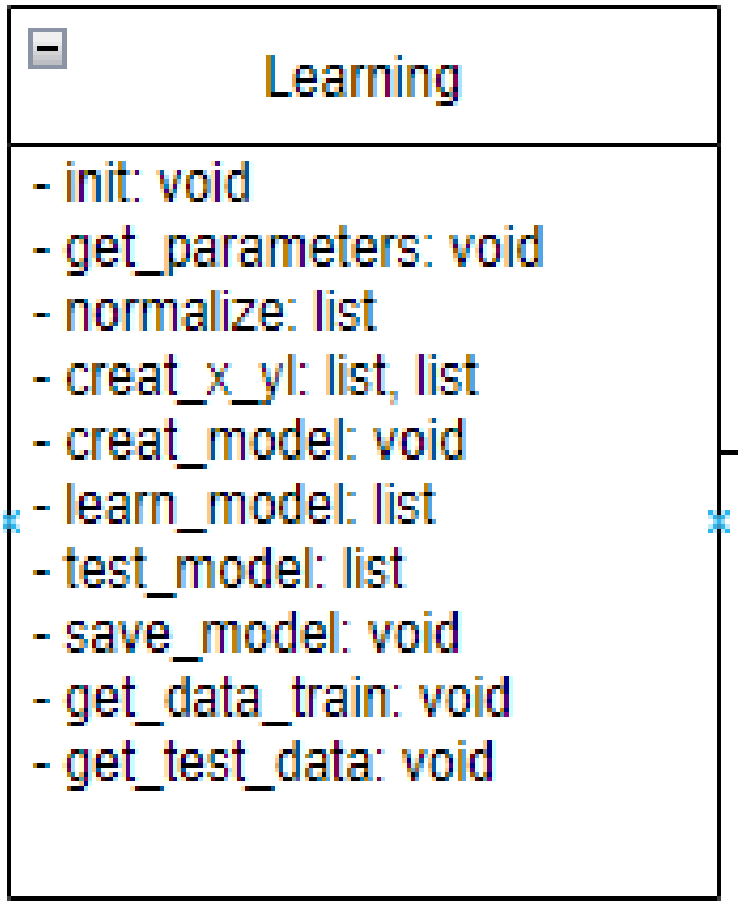

Рисунок 4.1 – UML діаграма класів моделі розпізнавання тексту

Отже, шари Embedding та CNN моделі були реалізовані за допомогою пакетів мови Python, адже потребують заздалегідь задані параметри великої кількості. Шар Bi GRU реалізовано теж за допомогою пакетів, так як модель потребує певного зв’язку між елементами.

4.3 Навчання моделі розпізнавання тексту

Алгоритм навчання моделі розпізнавання тексту зображений на рисунку 4.2.

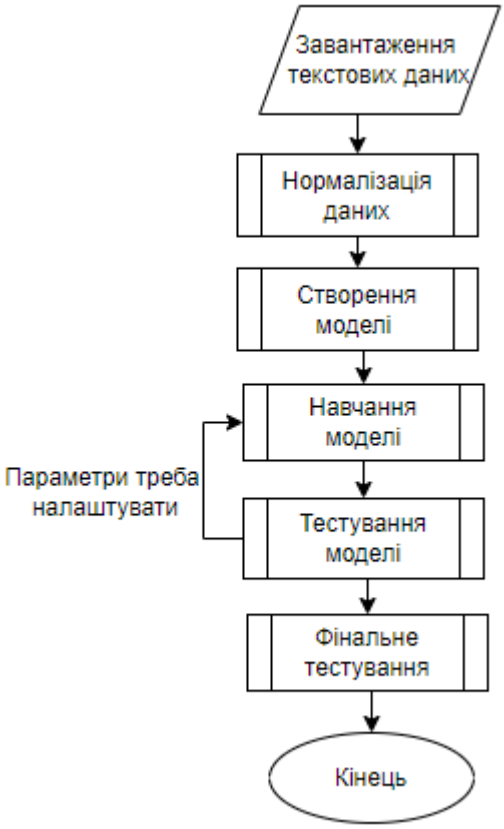

Рисунок 4.2 – Алгоритм навчання моделі

З початку слід завантажити текстові дані для навчання моделі першого тестування та кінцевого тестування. Усі дані повинні бути з різними текстами. Навчальних даних повинно бути більше всього. Так для навчання було взято 204 відгука про телефони та ноутбуки з інтернет-магазину «Розетка»[51], для першого тесту 27 відгука, для фінального тесту 24 відгука. Структура навчальних даних зображена на рисунку 4.3.

Коментар	Телефон	Ноутбук	Позитив	Негатив
Телефоном користуюся пару днів, доки враження позит	1	0	1	0
Добрий день. Взяв за порадою друга айтішника на замі	1	0	1	0
Дуже задоволений покупкою. Дивився аналоги від імен	1	0	1	0
Класний смартфон. Перейшов на нього з Redmi note	1	0	1	0
Дуже задоволений телефоном. Замовив версію 8/25	1	0	1	0
Телефон чудовий!	1	0	1	0

Рисунок 4.3 – Структура навчальних даних

На етапах навчання та тестування відбувається налаштування моделі. Обираються кількість епох навчання та їх вмістимось, при зміні яких слід враховувати якість результатів на процесі навчання та тестування, адже при збільшенні кількості епох з’являється процес перенавчання моделі.

Приклад навчання моделі зображено на рисунку 4.4.

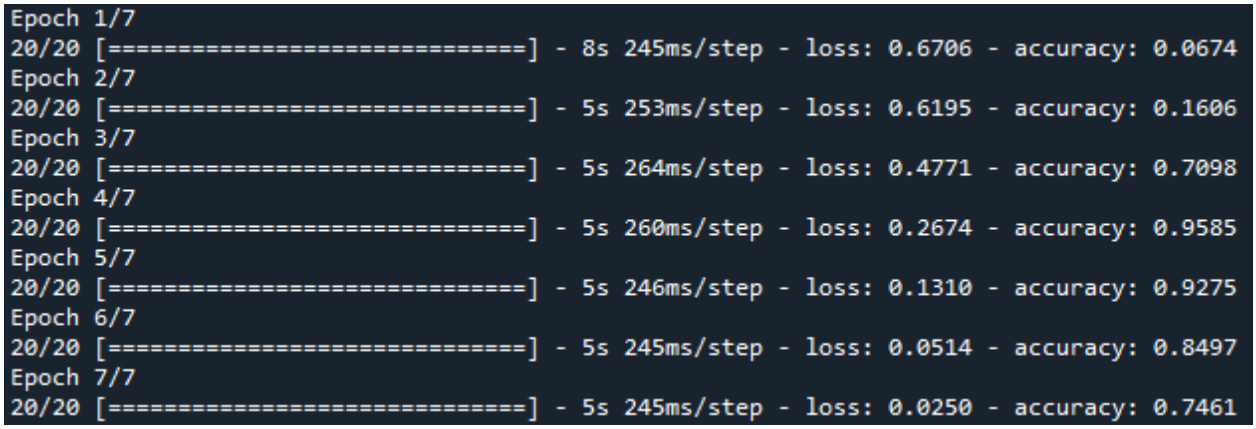

Рисунок 4.4 – Навчання моделі

Отже, за рисунком 4.3 видно, що на п’ятій епосі відбувається процес перенавчання моделі – зменшується показник точності класифікації, та слід зменшити кількість епох до п’яти.

4.4 Розробка програмного застосунку

У розробленій моделі розпізнавання тексту був використаний MVP-патерн (рисунок 4.5).

На рисунку можна побачити п’ятнь класів та два типи взаємозв’язку між ними: асоціація та наслідування. Взаємозв’язок асоціація поєднує класи моделей і клас виду з представником.

У методах класу представника Presenter main міститься виклик методів моделей та передача їх результатів до елементів класу вид Ui Main Window. Методи представників викликаються при натисканні на відповідний елемент виду додатку.

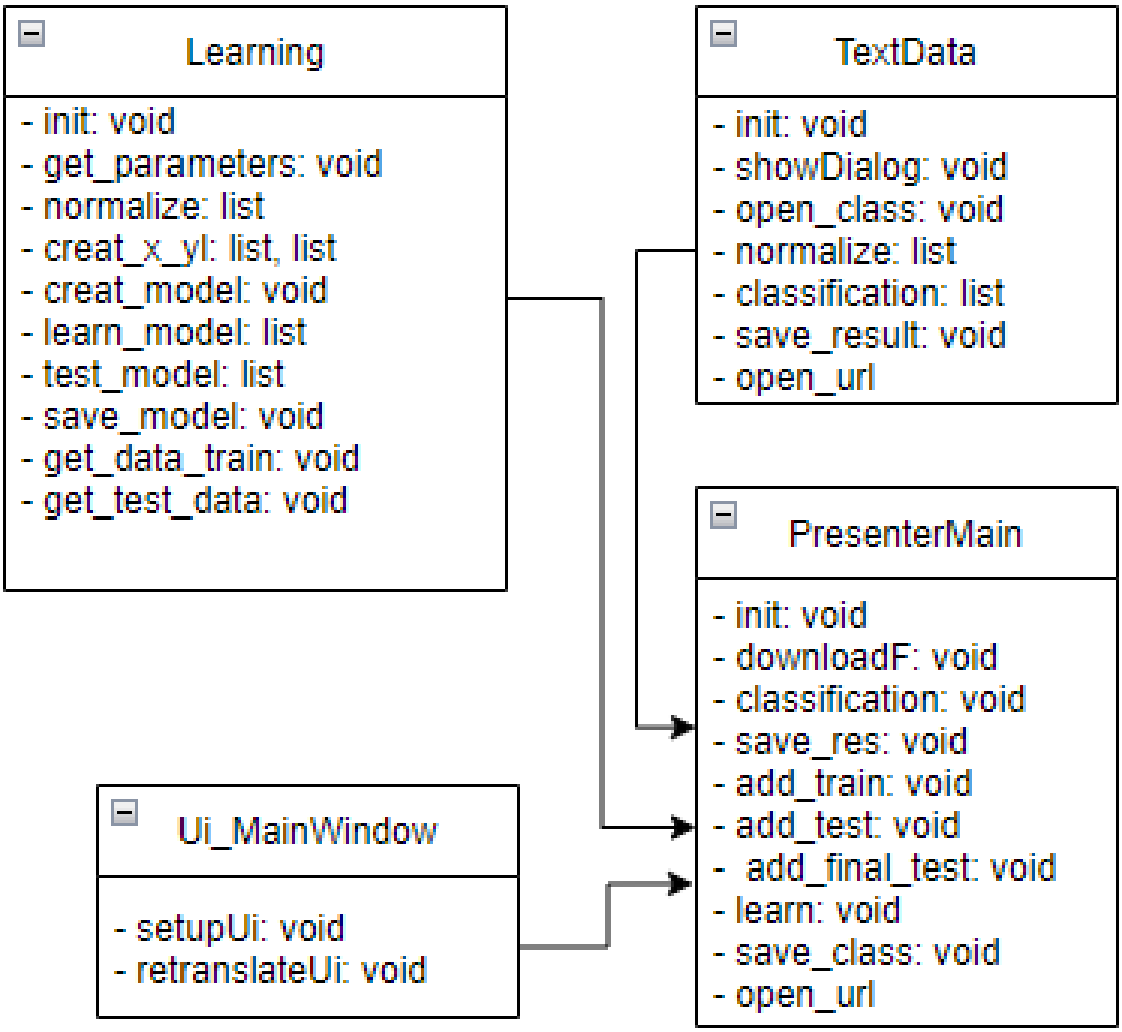

Рисунок 4.5 – UML діаграма класів програмного додатку

Для побудови класу вид Ui Main Window використовувались такі бібліотеки мови Python: PyQT5 та matplotlib для відображення графіків.

До складу моделей шаблону MVP входитять клас ConvBiGRU, який унаслідується від класу BiGRU та TextData. У моделях відбувається логіка програмного додатку.

Клас TextData має такі функції:

- init – ініціалізація параметрів;

- showDialog – функція показу діалогове вікна обрання й загрузки текстових файлів;

- open_class – функція завантаження класифікаторів;

- normalize – функція нормалізації текстових даних;

- classification – функція завантаження моделі та класифікація текстів;

- save_result – функція збереження результатів класифікації;

- open_url – функція завантаження тексту за URL адресою.

Весь код програми знаходиться у додатку Б.

4.5 Інтерфейс програмної реалізації моделі

Інтерфейс програмної реалізації моделі розпізнавання тексту включає в себе наступні елементи:

- вікно класифікації;

- вікно створення класифікатора;

- вікно з правилами користування програмним додатком.

Наведемо екранні знімки усіх елементів програмного засобу (рис. 4.6 – рис. 4.9).

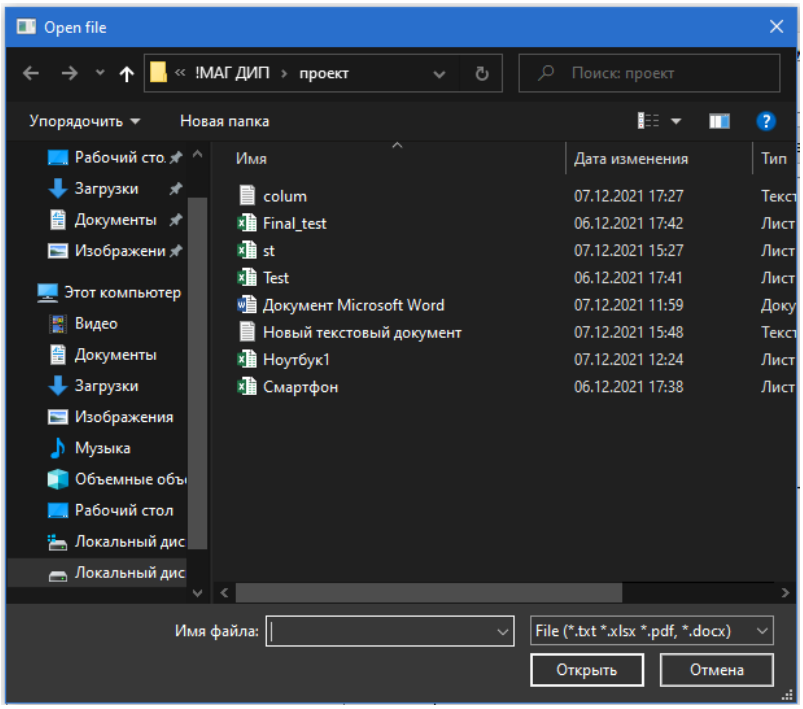

Рисунок 4.6 – Діалогове вікно вибору файлу

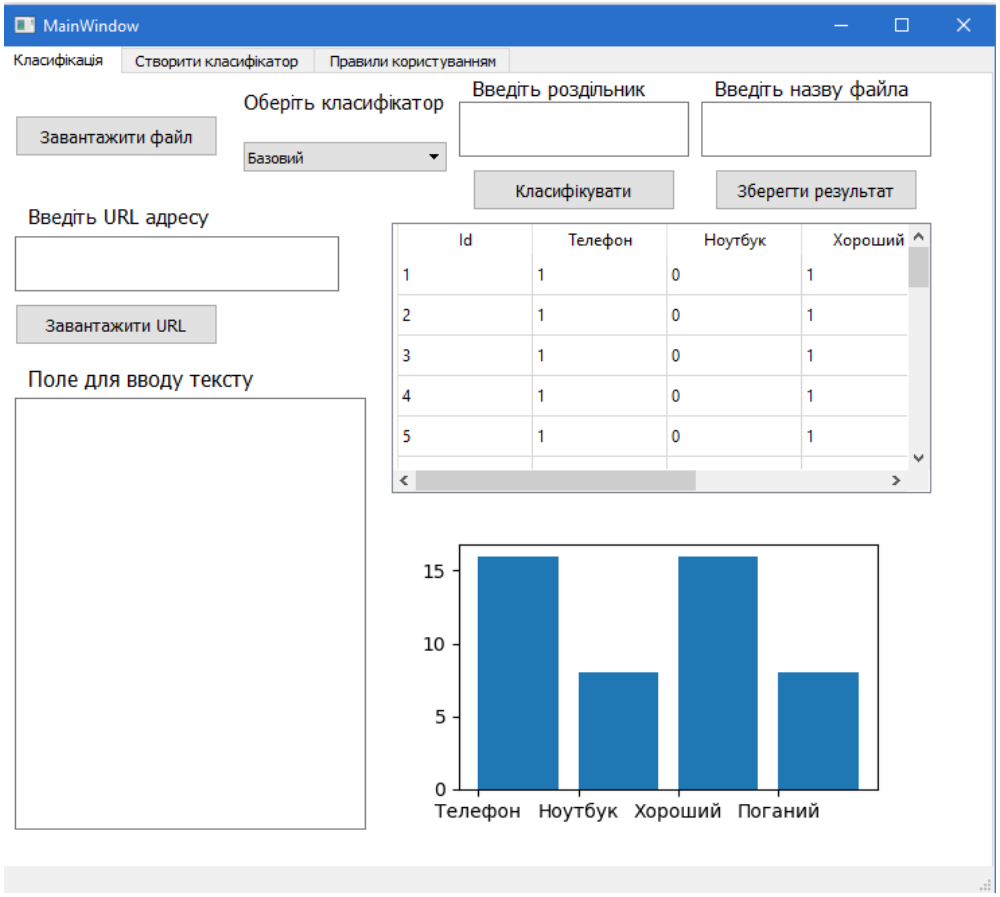

Рисунок 4.7 – Вікно класифікації тексту

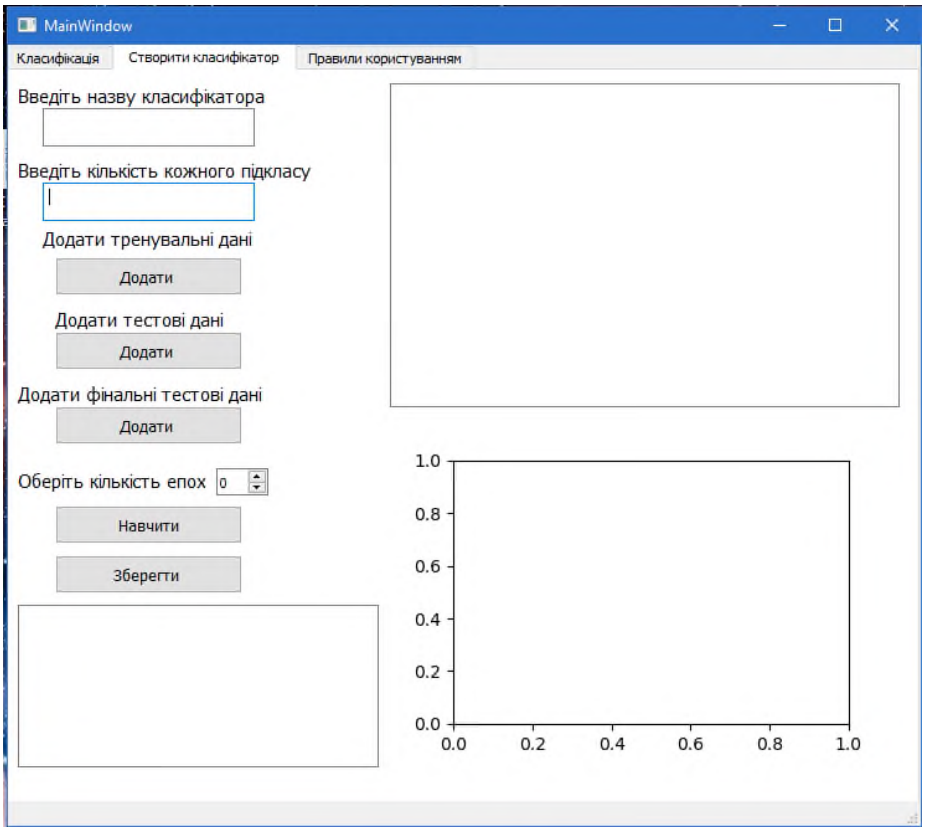

Рисунок 4.8 – Вікно створення класифікатору

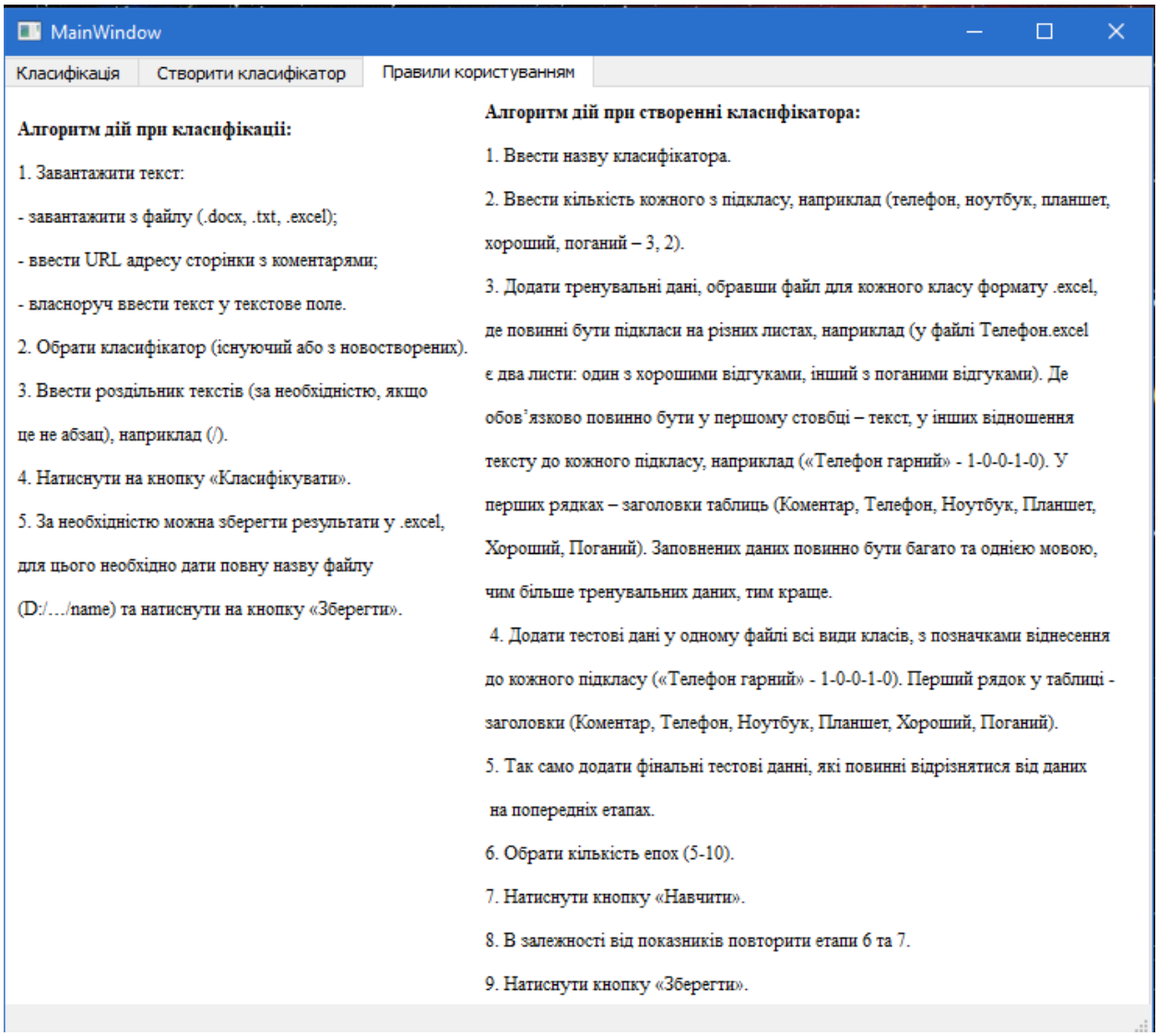

Рисунок 4.9 – Вікно з правилами користування

Отже, програма має усі необхідні елементи керування для розпізнавання тексту.

4.6 Правила користуванням програмним засобом

Для запуску програмного додатку необхідно встановити платформу Anaconda і наступні пакети мови Python у консолі Powershell за допомогою команди pip install:

- PyQt5;
- tensorflow.

Далі необхідно зайти до середовища Spyder та обрати відповідний файл програмного коду і запустити його.

Користувач програмного застосунку може виконувати дві основні дії: проведення класифікації тексту та створити власний класифікатор.

Алгоритм дій при класифікації:

а) завантажити текст:

- завантажити з файлу (.docx, .txt, .excel);

- ввести URL адресу сторінки з коментарями;

- власноруч ввести текст у текстове поле;

б) обрати класифікатор (існуючий або з новостворених);

в) ввести роздільник текстів (за необхідністю, якщо це не абзац), наприклад (/);

г) натиснути на кнопку «Класифікувати».

д) за необхідністю можна зберегти результати у .excel, для цього необхідно дати повну назву файлу (D:/…/name) та натиснути на кнопку «Зберегти».

Алгоритм дій при створенні класифікатора:

а) ввести назву класифікатора;

б) Ввести кількість кожного з підкласу, наприклад (телефон, ноутбук, планшет, хороший, поганий – 3, 2);

в) додати тренувальні дані, обравши файл для кожного класу формату .excel, де повинні бути підкласи на різних листах, наприклад (у файлі Телефон.excel є два листи: один з хорошими відгуками, інший з поганими відгуками), де обов’язково повинно бути у першому стовбці – текст, у інших відношення тексту до кожного підкласу, наприклад («Телефон гарний» - 1-0-0-1-0), у перших рядках – заголовки таблиць (Коментар, Телефон, Ноутбук, Планшет, Хороший, Поганий);

г) додати тестові дані у одному файлі всі види класів, з позначками віднесення до кожного підкласу («Телефон гарний» - 1-0-0-1-0), перший рядок

у таблиці – заголовки Коментар, Телефон, Ноутбук, Планшет, Хороший, Поганий);

д) так само додати фінальні тестові данні, які повинні відрізнятися від даних на попередніх етапах;

е) Обрати кількість епох (5-10);

є) натиснути кнопку «Навчити»;

ж) в залежності від показників повторити етапи шість та сім;

з) натиснути кнопку «Зберегти».

4.7 Висновки до розділу

У четвертому розділі були обрані мова програмування Python та програмне середовище Spyder. Була розроблена та навчена модель розпізнавання тексту. При розробці програмного додатку використовувався патерн MVP. Також наведені екранні знімки головних елементів програми, а саме класифікація та створення класифікатору, та описано правила користування їми.

РОЗДІЛ 5
ТЕСТУВАННЯ МОДЕЛІ ТА ОЦІНКА РЕЗУЛЬТАТІВ

5.1 Тестування програмного додатку та оцінка його якості

При розробці програмного застосунку використовувались наступні види тестування:

- функціональне (рис. 5.1);
- автоматичні тести (рис. 5.2, рис. 5.3);
- тестування білого ящику;
- модульне тестування;
- ручне тестування.

```
In [1]: import re

In [2]: text = 'Базова 2, 2/'
   ...: text = re.sub(r"[-()\"#/@;:<>{}=~|.?,]","",text)

In [3]: text
Out[3]: 'Базова 2 2'
```

Рисунок 5.1 – Приклад функціонального тестування очищення тексту

```
fname2 = 'Ноутбук1.xlsx'
fname1 = 'Смартфон.xlsx'
fnames = [fname1, fname2]
cl = Learning()
X, Y = cl.creat_x_y(fnames)
cl.creat_model()
history = cl.learn_model(X, Y)
fname3 = ['test.xlsx']
x, y = cl.creat_x_y(fname3)
cl.test_model(x, y)
fname4 = ['Final_test.xlsx']
x1, y1 =  cl.creat_x_y(fname4)
cl.test_model(x1, y1)
```

Рисунок 5.2 – Приклад автоматичного тесту

```
Epoch 1/4
20/20 [==============================] - 1s 17ms/step - loss: 0.6748 - accuracy: 0.1320 - f1_m: 0.5615 -
precision_m: 0.6349 - recall_m: 0.5193
Epoch 2/4
20/20 [==============================] - 0s 15ms/step - loss: 0.6061 - accuracy: 0.4365 - f1_m: 0.6368 -
precision_m: 0.6368 - recall_m: 0.6368
Epoch 3/4
20/20 [==============================] - 0s 16ms/step - loss: 0.5148 - accuracy: 0.5685 - f1_m: 0.6648 -
precision_m: 0.6742 - recall_m: 0.6561
Epoch 4/4
20/20 [==============================] - 0s 16ms/step - loss: 0.4096 - accuracy: 0.6193 - f1_m: 0.8060 -
precision_m: 0.8190 - recall_m: 0.7939
TP =  38 , FP =  10 , FN =  10 , TN =  38
1/1 [==============================] - 1s 929ms/step - loss: 0.5806 - accuracy: 0.5833 - auc: 0.8570
accuracy =  0.7916666666666666 f1_score =  0.7916666666666666 precision =  0.7916666666666666 recall =
0.7916666666666666
TP =  34 , FP =  12 , FN =  12 , TN =  34
1/1 [==============================] - 1s 907ms/step - loss: 0.7533 - accuracy: 0.5217 - auc: 0.7968
```

Рисунок 5.3 – Результат автоматичного тесту

У ході тестування були виявлені наступні помилки:

- неможливість розпізнавання тексту при відсутності даних;

- пустий файл;

- таблиці з навчальними даними не відповідають нормам;

- файл не завантажився.

Виникненні проблеми вирішалися методами виключення блоком try, умовами та виводом повідомлень про помилку (рис. 5.4).

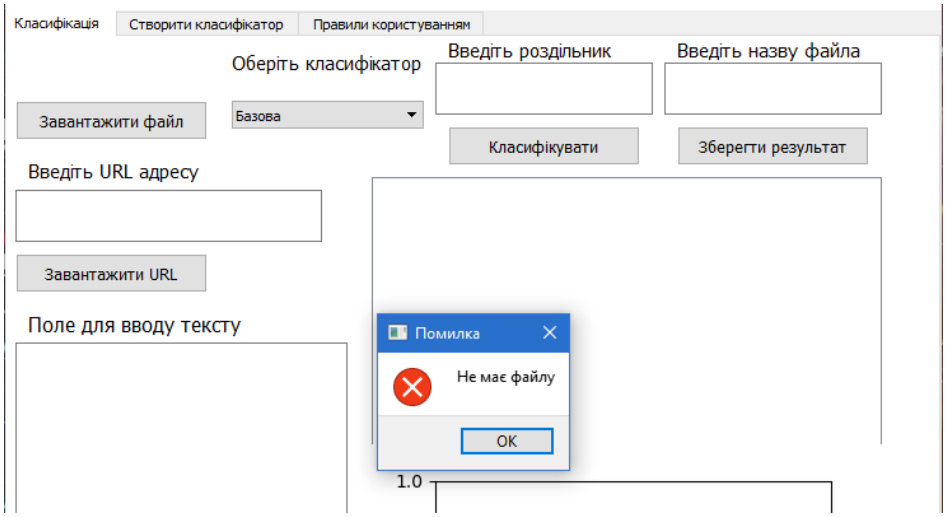

Рисунок 5.4 – Вивід повідомлення про помилку

Після проходження усіх тестів наступним кроком є користувацька оцінка програмного застосунку та проведення порівняння з іншими програмними забезпеченнями (табл. 5.1)

Таблиця 5.1 – Порівняльна характеристика програмних засобів

Параметр	MegaIndex	Amazon Comprehend	Mediatoolkit	ABBYY FlexiCapture	Власна модель
Вид додатку	Web-сервіс	Web-сервіс	IOS-сервіс	Програмне забезпечення	Програмне забезпечення
Підтримка мови	Російська, англійська	Більшість мов	Більшість мов	Більшість мов	Українська й інші
Підтримка форматів даних	URL	URL, текстові документи,.	URL	URL, текстові документи, зображення , аудіо-файли	URL, текстові документи та таблиці,.
Сентимент аналіз	-	+	+	-	+
Аналіз тематики	+	+	-	+	+
Добавлення нового класифікатору	-	+	-	+	+
Сортування документів	-	+	-	+	-
Статистика класів	+	+	+	+	+

Отже, розроблена програмна модель має достатню конкурентоспроможність. До недоліків розробленої програми можна віднести: відсутність сортування документів, недостатня кількість класів при

розпізнаванні тексту, навчена класифікація тільки на українській мові. Переваги розробленої програми: одночасний сентимент аналіз та класифікація за тематикою, добавлення нового класифікатора на різних мовах, можливість багатомірної класифікації.

5.2 Оцінка якості класифікації текстових даних

Оцінка якості навчання моделі зображена на рисунку 5.2.

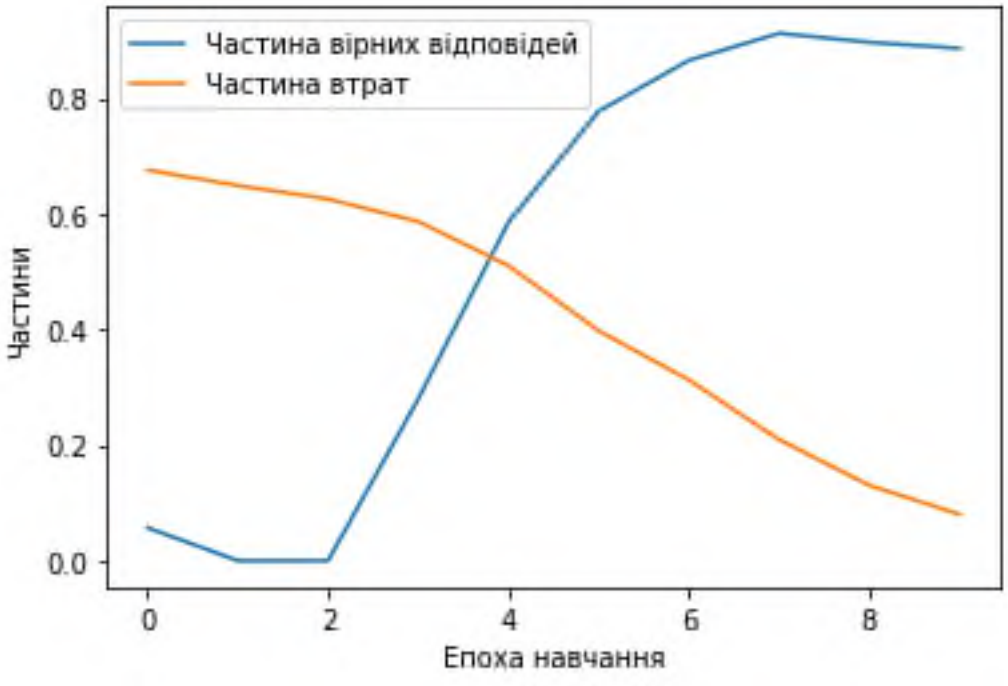

Рисунок 5.2 – Якість навчання моделі

За рисунком 5.2 можна сказати, що точність моделі росте до 7 епохи, а втрати мають постійну тенденцію зменшення. Отже слід обрати 7 епох для навчання.

Далі порівняємо модель ConvBiGRU з іншими нейронними мережами оцінивши якість класифікації кожного з них за фінальним тестом з однаковими параметрами (табл. 5.2).

Таблиця 5.2 – Порівняння показників якості моделей

Показник	ConvBiGRU	ConvBiLSTM	CNN	GRU	LTSM
Accuracy	1.0000	0.9565	0.7957	0.8261	0.8298
Precision	0.8667	0.8043	0.7872	0.8000	0.8125
Recall	0.8478	0.8043	0.8043	0.7826	0.8043
F1	0.8571	0.8043	0.7957	0.7912	0.8043
Time	86ms/step	104ms/step	26ms/step	45ms/step	54ms/step

Отже, модель ConvBiGRU має найвищі показники якості, подібні моделі ConvBiLSTM, але швидкість виконання класифікації більша. Але швидкість комбінованих моделей може бути кращою у порівнянні з простими моделями, якщо обирати великі масиви текстових даних.

5.3 Висновки до розділу

У п’ятому розділі було здійснено тестування програмного застосунку такими видами: - функціональне, автоматичні тести, тестування білого ящику, модульне тестування. Проведено порівняльну характеристику розробленого додатку з іншими існуючими програмними забезпеченнями розпізнавання текстів, при цьому було вияснено переваги та недоліки програми ц те, що розроблена програмна модель має достатню конкурентоспроможність.

Також були оцінені показники якості моделі ConvBiGRU і порівняно з іншими моделями класифікації.

Таким чином, очікувані наукові та практичні результати роботи були підтверджені.

ВИСНОВКИ

Отже мета роботи була досягнута. Було розроблено, обґрунтувано та програмно реалізовано універсальну модель розпізнавання текстових даних, що дозволить отримати нові більш детальні знання за допомогою поєднання різних класів і методів аналізу текстових даних.

Для досягнення мети були вирішні такі задачі:

- досліджено сутність й актуальність розпізнавання тексту;
- формально постановлене завдання та визначено критерії оцінки;
- спроектовані архітектури моделі і програмного додатку;
- розроблені моделі розпізнавання тексту;
- навчанні моделі розпізнавання тексту;
- розроблено програмний засіб;
- оформлені правила користування програмним забезпеченням;
- протестований програмний додаток;
- оцінено результати класифікації моделі.

Тематика дипломної роботи обговорювалась на всеукраїнській студентській науково-практичній конференції «Математика та математичне моделювання у сучасному технічному університеті.» 2021 року у Донецькому національному технічному університеті, м. Покровськ.

Бабенко Є. О. Дмитрієва О.А. Аналіз основних підходів до розпізнавання тексту на основі кластерного аналізу // Математика та математичне моделювання у сучасному технічному університеті.[Електронний ресурс] Збірник тез доповідей II Всеукраїнської науково-практичної конференції студентів та молодих вчених, 2021 р. – Покровськ ДонНТУ – 154 с. 2021.

Дмитрієва О.А. Розробка комп’ютерної моделі системи діагностики технічного стану електрообладнання на основі кластерного аналізу/ О.А. Дмитрієва, Т.В. Алтухова, Є.О. Бабенко// Наукові праці Донецького

національного технічного університету. Серія: Інформатика, кібернетика та обчислювальна техніка. – 2021. – №1 (32). – С. 24-31.

Дмитрієва О. Програмна система кластерного аналізу даних для діагностики стану електрообладнання/ О.А. Дмитрієва , Є. Бабенко, Т. Алтухова//XXIX Міжнародна науково-практична конференція «Інформаційні технології: Наука, Техніка, Технологія, Освіта, Здоров'я» (MicroCAD-2021), 18-20 травня 2021 р.: у 5 ч., Ч. IV. Харків: НТУ «ХПІ». - 2021.- С. 92

Бабенко Є. О. Дмитрієва О.А. Аналіз основних підходів до розпізнавання тексту на основі кластерного аналізу // Математика та математичне моделювання у сучасному технічному університеті.[Електронний ресурс] Збірник тез доповідей II Всеукраїнської науково-практичної конференції студентів та молодих вчених, 2021 р. – Покровськ ДонНТУ – 154 с. 2021.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Бабенко Є. О. Дмитрієва О.А. Аналіз основних підходів до розпізнавання тексту на основі кластерного аналізу // Математика та математичне моделювання у сучасному технічному університеті.[Електронний ресурс] Збірник тез доповідей II Всеукраїнської науково-практичної конференції студентів та молодих вчених, 2021 р. – Покровськ ДонНТУ – 154 с. 2021.

2. Бабенко Є.О. Моделі кластерного аналізу даних для діагностики стану електрообладнання/ Є.О. Бабенко, О.А. Дмитрієва// Матеріали Всеукраїнської студентської науково-практичної конференції «Математика та математичне моделювання у сучасному технічному університеті», 27-28 травня 2020 року. – Покровськ: ДВНЗ "ДонНТУ". – 2020. – С. 9-11.

3. Дмитрієва О. Програмна система кластерного аналізу даних для діагностики стану електрообладнання/ О.А. Дмитрієва , Є. Бабенко, Т. Алтухова//XXIX Міжнародна науково-практична конференція «Інформаційні технології: Наука, Техніка, Технологія, Освіта, Здоров'я» (MicroCAD-2021), 18-20 травня 2021 р.: у 5 ч., Ч. IV. Харків: НТУ «ХПІ». - 2021.- С. 92

4. Оксанич І.Г., Піскунов Д.М., Черниш Д.П. Інтелектуальний аналіз масиву текстових документів на основі технології Text Mining // Обробка інформації в складних організаційних системах. Кременчуцький національний університет імені Михайла Остроградського. с 139–143.

5. Карпов І.А., Антоненко С.В. Огляд методів інтелектуального аналізу тексту // Актуальні проблеми автоматизації та інформаційних технологій . Том 24. Дніпровський національний університет імені Олеся Гончара. 2020. Р 40–46.

6. Констянтинів В.Р. Огляд методів інтелектуального аналізу текстових даних // Матеріали VI Міжнародної науково-технічної конференції молодих

учених та студентів. Актуальні задачі сучасних технологій – Тернопіль 16-17 листопада, 2017. с 93–94.

7. Волосюк Ю. В. Методи класифікації текстових документів в задачах text mining// Наукові записки Українського науково-дослідного інституту зв'язку, 2014. – №6(34)

8. Sholom M.W. Text minig. Predictive methods of analyzing unstructured information. M. W. Sholom, N.Indurkhya, T.Zhang, F.J.Damarau. — 2004. — 236с.

9. Черезов Д.С., Тюкачев Н.А. Обзор Основных Методов Классификации И Кластеризации Данных // Вестник Вгу. Серия Системный Анализ И Информационыне Технологии, 2009. № 2. Р 25–29.

10. Відстань між об'єктами (кластерами) і міри близькості груп об'єктів [Електронний ресурс]/ Мхітерян В. С. Аналіз даних // Підручник – 2017. – Режим доступу: https://stud.com.ua/93356/statistika/vidstan_obyektami_klasterami_miri_blizkosti_grup_obyektiv

11. Классическое машинное обучение: задачи классификации, обобщения, кластеризации данных [Електронний ресурс]. – Режим доступу: https://evergreens.com.ua/ru/articles/classical-machine-learning.html

12. Лавров Ю.А. Применение кластерного анализа методом k-средних для классификации текстов научной направленности // Математические структуры и моделирование. Омский государственный университет им. Ф.М. Достоевского, 2017. № 43. с 108–121.

13. Глибовець А. М. Алгоритм токенізації та стемінгу для текстів українською мовою // Наукові записки УКМА. Том 198. Комп'ютерні науки, 2017. с 4–8.

14. Классификация документов: 7 практических подходов для небольших наборов данных [Електронний ресурс]. – Режим доступу: https://habr.com/ru/post/504744/

15. Kavita Ganesan. FastText vs. Word2vec: A Quick Comparison [Електронний ресурс]. – Режим доступу: https://kavita-ganesan.com/fasttext-vs-word2vec/#.YZu7W6JBzIU

16. GloVe: Global Vectors for Word Representation/ Jeffrey Pennington, Richard Socher, Christopher D. Manning [Електронний ресурс]. – Режим доступу: https://nlp.stanford.edu/projects/glove/

17. Перелыгин А. А. Кластеризация многомерных данных: методы, алгоритмы, программы//Вестник Алтайского государственного педагогического университета: естественные и точные науки, 2015. с 24–31.

18. Аналіз систем розпізнавання образів структури композитів : монографія / Добротвор І.Г., Стухляк П.Д., Микитишин А.Г., Митник М.М. – Тернопіль : Тернопільський національний технічний університет імені Івана Пулюя, 2018. – 224 с. ISBN. Тернопіль.

19. Шпаргалка по разновидностям нейронных сетей. Часть первая. Элементарные конфигурации [Електронний ресурс]. – Режим доступу:https://tproger.ru/translations/neural-network-zoo-1/

20. Kubat M. An Introduction to Machine Learning // An Introduction to Machine Learning. 2017. 1–348 p.

21. Dinov I.D. Data science and predictive analytics: Biomedical and health applications using R // Data Science and Predictive Analytics: Biomedical and Health Applications using R. 2018. 1–832 p.

22. Волосюк Ю. В. Методи класифікації текстових документів в задачах text mining// Наукові записки Українського науково-дослідного інституту зв’язку. – 2014. – №6(34)

23. Awet Fesseha. Text Classification Based on Convolutional Neural Networks and Word Embedding for Low-Resource Languages: Tigrinya/ Awet Fesseha, Shengwu Xiong, Eshete Derb Emiru, Moussa Diallo, Abdelghani Dahou// Information. 2021, 12, 52 [Електронний ресурс]. – Режим доступу:. https://doi.org/10.3390/info12020052

24. Ebubekir Buber, Banu Diri// Web Page Classification using RNN. 2019. [Електронний ресурс]. – Режим доступу:https://www.researchgate.net/publication/334474452_Web_Page_Classification_Using_RNN

25. Нейронные сети Кохонена [Електронний ресурс]. – Режим доступу: neuronus.com/theory/nn/955-nejronnye-seti-kokhonena.html.

26. GRU и LTSM сети [Електронний ресурс]. – Режим доступу: file:///E:/Work/!%D0%9C%D0%90%D0%93%20%D0%94%D0%98%D0%9F/information-12-00052-v2.pdf

27. Michael Phi. Illustrated Guide to LSTM’s and GRU’s: A step by step explanation. 2018 [Електронний ресурс]. – Режим доступу: https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21

28. MegaIndex. Определение тематики текста [Електронний ресурс]. – Режим доступу: https://ru.megaindex.com/a/tcategories

29. AWS. Amazon Comprehend [Електронний ресурс]. – Режим доступу: https://aws.amazon.com/ru/comprehend/

30. Mediatoolkit. Analyze social media sentiment about your brand. Get real-time, actionable insights [Електронний ресурс]. – Режим доступу: https://www.mediatoolkit.com/benefits/social-media-sentiment-analysis?campaign=Sentiment_SKAG_ROAS_to_CPA&keyword=sentiment%20analysis&matchtype=b&device=c&gclid=Cj0KCQjw5oiMBhDtARIsAJi0qk0BHj3UmrpSW_Ot14PIemF3TvpGNQEpvm2NI63qEp_s1T_sixCNk5gaAjpIEALw_wcB

31. ABBYY FlexiCapture [Електронний ресурс]. – Режим доступу: https://www.abbyy.com/ru/flexicapture/how-it-works/

32. Orăsan C. Aggressive Language Identification Using Word Embeddings and Sentiment Features // Proc. First Work. Trolling, Aggress. Cyberbullying. 2018р. – ст.113–119.

33. Батура Т.В. Методы автоматической классификации текстов// Международный журнал «Программные продукты и системы». 2017 р. [Електронний ресурс]. – Режим доступу: https://www.researchgate.net/publication/315328102

34. Краткий курс машинного обучения или как создать нейронную сеть для решения скоринг задачи// Habr [Електронний ресурс]. – Режим доступу: https://habr.com/ru/post/340792/

35. Метрики в задачах машинного обучения// Habr [Електронний ресурс]. – Режим доступу: https://habr.com/ru/company/ods/blog/328372/

36. Alexandre Xavier. An introduction to ConvLSTM// Medium. 2019. [Електронний ресурс]. – Режим доступу: https://medium.com/neuronio/an-introduction-to-convlstm-55c9025563a7

37. Maryem Rhanoui. A CNN-BiLSTM Model for Document-Level Sentiment Analysis/ Maryem Rhanoui, Mounia Mikram, Siham Yousfi, Soukaina Barzali// Machine Learning and Knowledge Extraction. 2019. [Електронний ресурс]. – Режим доступу: https://www.mdpi.com/journal/make

38. Beakcheol Jang. Bi-LSTM Model to Increase Accuracy in Text Classification: Combining Word2vec CNN and Attention Mechanism/ Beakcheol Jang, Myeonghwi Kim, Gaspard Harerimana, Sang-ug Kang , Jong Wook Kim// Applied Sciences. 2020. . [Електронний ресурс]. – Режим доступу: https://www.mdpi.com/journal/applsci

39. Text Classification with GloVe, LSTM/GRU(99% acc)// Kaggel 2020. [Електронний ресурс]. – Режим доступу: https://www.kaggle.com/sepidafs/text-classification-with-glove-lstm-gru-99-acc

40. Pengpeng Li. Bidirectional Gated Recurrent Unit Neural Network for Chinese Address Element Segmentation/ Pengpeng Li, An Luo, Jiping Liu, Yong Wang, Jun Zhu,Yue Deng, Junjie Zhang// ISPRS Int. J. Geo-Inf. 2020, 9, 635. Електронний ресурс]. – Режим доступу: https://www.mdpi.com/journal/ijgi

41. Matyas Amrouche. Word Embedding (Part I). 2019. [Електронний ресурс]. – Режим доступу: https://towardsdatascience.com/word-embeddings-

intuition-and-some-maths-to-understand-end-to-end-skip-gram-model-cab57760c745

42. Word2vec - это просто// ICHI.PRO [Електронний ресурс]. – Режим доступу: https://ichi.pro/ru/word2vec-eto-prosto-21552978770094

43. Yoon Kim. Convolutional Neural Networks for Sentence Classification//Cornell University. 2014. [Електронний ресурс]. – Режим доступу: https://arxiv.org/abs/1408.5882

44. Cezanne Camacho. CNNs for Text Classification [Електронний ресурс]. – Режим доступу: https://cezannec.github.io/CNN_Text_Classification/

45. Awet Fesseha. Text Classification Based on Convolutional Neural Networks and Word Embedding for Low-Resource Languages: Tigrinya/ Awet Fesseha, Shengwu Xiong, Eshete Derb Emiru, Moussa Diallo, Abdelghani Dahou// Information 2021, 12, 52 [Електронний ресурс]. – Режим доступу: https://doi.org/10.3390/info12020052

46. Pengpeng Li. Bidirectional Gated Recurrent Unit Neural Network for Chinese Address Element Segmentation/ Pengpeng Li, An Luo, Jiping Liu, Yong Wang, Jun Zhu, Yue Deng, Junjie Zhang// ISPRS Int. J. Geo-Inf. 2020, 9, 635 [Електронний ресурс]. – Режим доступу: https://www.mdpi.com/journal/ijgi

47. Многослойные перцептроны/ Neuronus [Електронний ресурс]. – Режим доступу: https://neuronus.com/theory/nn/953-mnogoslojnye-pertseptrony.html

48. Топ 5 языков программирования для разработки искусственного интеллекта [Електронний ресурс]. – Режим доступу: https://senior.ua/articles/top-5-yazykov-programmirovaniya-dlya-razrabotki-iskusstvennogo-intellekta

49. 10 лучших Python IDE и редакторов кода [2020] [Електронний ресурс]. – Режим доступу: https://vc.ru/dev/104257-10-luchshih-python-ide-i-redaktorov-koda-2020

50. Spyder Documention – Features Overview [Електронний ресурс]. – Режим доступу: https://docs.spyder-ide.org/overview.html

51. Інтернет-магазин Розетка [Електронний ресурс]. – Режим доступу: https://rozetka.com.ua/

ДОДАТКИ

ДОДАТОК А
Зауваження нормоконтролера

ДОДАТОК Б

Лістинг програми

```
import sys
from PyQt5 import QtCore, QtGui, QtWidgets
from PyQt5.QtWidgets import (QTableWidgetItem, QApplication, QMainWindow, QFileDialog, QMessageBox)
import pandas as pd
from PyQt5.QtCore import QAbstractTableModel, Qt
import os
import numpy as np
import matplotlib
matplotlib.use('Qt5Agg')
from matplotlib.backends.backend_qt5agg import FigureCanvasQTAgg
from matplotlib.figure import Figure
import re
from tensorflow.keras.preprocessing.text import Tokenizer, text_to_word_sequence
import docx
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.layers import Dense, Embedding, SpatialDropout1D, Bidirectional, GRU, Conv1D, MaxPooling1D, GlobalAveragePooling1D, GlobalMaxPooling1D, Input, concatenate
from keras.models import Model
from keras.models import load_model
import requests
from bs4 import BeautifulSoup
import matplotlib.pyplot as plt
from keras import backend as K



def recall_m(y_true, y_pred):
    true_positives = K.sum(K.round(K.clip(y_true * y_pred, 0, 1)))
    possible_positives = K.sum(K.round(K.clip(y_true, 0, 1)))
    recall = true_positives / (possible_positives + K.epsilon())
    return recall

def precision_m(y_true, y_pred):
    true_positives = K.sum(K.round(K.clip(y_true * y_pred, 0, 1)))
    predicted_positives = K.sum(K.round(K.clip(y_pred, 0, 1)))
    precision = true_positives / (predicted_positives + K.epsilon())
    return precision

def f1_m(y_true, y_pred):
    precision = precision_m(y_true, y_pred)
    recall = recall_m(y_true, y_pred)
    return 2*((precision*recall)/(precision+recall+K.epsilon()))



class Learning(QMainWindow):
```

```
        def __init__(self):
            self.name = ''
            self.maxWordsCount = 1000
            self.max_text_len = 300
            self.tokenizer = Tokenizer(num_words=self.maxWordsCount, filters='!–"—#$%&amp;()*+,-./:;<=>?@[\\]^_`{|}~\t\n\r«»', lower=True, split=' ', char_level=False)
            self.k = []
            self.fns = []
            self.epochs = 1

        def get_test_data(self):
            fn = QFileDialog.getOpenFileName(self, 'Open file', None, "File (*.xlsx)")
            fname = "".join(fn[0])
            return fname


        def get_data_train(self):
            fn = QFileDialog.getOpenFileName(self, 'Open file', None, "File (*.xlsx)")
            fname = "".join(fn[0])

            if fn[0]:
                self.fns.append(fname)
            else:
                QMessageBox.critical(self, "Помилка ", "Не має файлу", QMessageBox.Ok)


        def get_param(self, name, number, n_w, n_ep):
            self.name = name
            number = re.sub(r"[-()\"#/@;:<>{}=~|.?,]","",number)
            self.k = number.split(' ')
            self.epochs = n_ep
            self.max_text_len = n_w
            self.column = []


        def normalize(self, texts):
            self.tokenizer.fit_on_texts(texts)
            data = self.tokenizer.texts_to_sequences(texts)
            return pad_sequences(data, maxlen=self.max_text_len)


        def creat_x_y(self):
            x = []
            y = []
            k = 0
            l = 0
            for fn in self.fns:
                xl = pd.ExcelFile(fn)
                sh = xl.sheet_names
                for el in sh:
                    df = xl.parse(el)
                    n = df.shape[0]
                    k = k+n
```

```
            l = df.shape[1]
            for i in range(n):
                x.append(df.values[i, 0])
                for j in range(1,l):
                    y.append(df.values[i, j])
    y = np.array(y)
    Y = y.reshape(k, l-1)
    X = self.normalize(x)
    return X, Y

def creat_model(self):
    inp = Input(shape=(self.max_text_len,))
    x = Embedding(self.maxWordsCount, 128, input_length = self.max_text_len)(inp)
    x = Conv1D(100, 3, activation='relu')(x)
    x = GlobalMaxPooling1D()(x)
    out = Dense(4, activation='sigmoid')(x)
    self.model = Model(inp, out)
    self.model.compile(loss='binary_crossentropy', metrics=['accuracy', f1_m,precision_m, recall_m], optimizer='adam')

def learn_model(self, X, Y):
    indeces = np.random.choice(X.shape[0], size=X.shape[0], replace=False)
    X = X[indeces]
    Y = Y[indeces]
    history = self.model.fit(X, Y, batch_size=10, epochs=4)
    plt.plot(history.history['accuracy'],label='Частина вірних відповідей')
    plt.plot(history.history['loss'], label='Частина втрат')
    plt.xlabel('Епоха навчання')
    plt.ylabel('Частини')
    plt.legend()
    plt.show()
    return history



def test_model(self, X, Y):
    self.k = [2, 2]
    res = self.model.predict(X)
    n = len(res)
    m = len(self.k)
    k = sum(self.k)
    v = 0
    matr = []
    for i in range(n):
        arr = []
        for j in range(m):
            maxi = res[i][0+v]
            j = 0+v
            for f in range(1,self.k[j]):
                if maxi< res[i][f+v]:
                    maxi = res[i][f+v]
                    j = f+v
            for f in range(self.k[j]):
                if f+v == j:
```

```
                    arr.append(1)
                else:
                    arr.append(0)
            matr.append(arr)
        TP = 0
        FP = 0
        FN = 0
        TN = 0
        for i in range(n):
            for j in range(k):
                if matr[i][j] == 1 and Y[i][j] == 1:
                    TP = TP+1
                elif matr[i][j] == 0 and Y[i][j] == 0:
                    TN = TN+1
                elif matr[i][j]== 1 and Y[i][j] == 0:
                    FP = FP+1
                elif matr[i][j] == 0 and Y[i][j] == 1:
                    FN = FN+1
        print('TP = ', TP, ', FP = ', FP, ', FN = ', FN, ', TN = ', TN)
        accuracy = (TP+TN)/(TP+FP+FN+TN)
        precision = TP/(FP+TP)
        recall = TP/(TP+FN)
        f1_score = 2*precision*recall/(recall+precision)
        self.model.evaluate(X, Y, verbose=1)
        print ('accuracy = ', accuracy, 'f1_score = ', f1_score, 'precision = ', precision, 'recall = ', recall)

    def save_model(self):
        model_path = self.name +'.h5'
        self.model.save(model_path)



class MplCanvas(FigureCanvasQTAgg):

    def __init__(self, parent=None, width=5, height=4, dpi=100):
        fig = Figure(figsize=(width, height), dpi=dpi)
        self.axes = fig.add_subplot(111)
        super(MplCanvas, self).__init__(fig)


class pandasModel(QAbstractTableModel):

    def __init__(self, data):
        QAbstractTableModel.__init__(self)
        Id = [i for i in range(1, data.shape[0] + 1)]
        self._data = pd.DataFrame()
        self._data['Id'] = Id
        liss = list(data)
        for i in range (data.shape[1]):
            self._data[liss[i]] = data.values[:,i]


    def rowCount(self, parent=None):
```

```
        return self._data.shape[0]

    def columnCount(self, parnet=None):
        return self._data.shape[1]

    def data(self, index, role=Qt.DisplayRole):
        if index.isValid():
            if role == Qt.DisplayRole:
                return str(self._data.iloc[index.row(), index.column()])
        return None

    def headerData(self, col, orientation, role):
        if orientation == Qt.Horizontal and role == Qt.DisplayRole:
            return self._data.columns[col]
        return None



class TextData (QMainWindow):

    def __init__(self):
        super().__init__()
        self.max_text_len = 300
        self.maxWordsCount = 1000
        self._data = []
        self.tokenizer = Tokenizer(num_words=self.maxWordsCount, filters='!–"—#$%&amp;amp;()*+,-./:;<=>?@[\\]^_`{|}~\t\n\r«»', lower=True, split=' ', char_level=False)
        self.k = []
        self.list = []
        self.colum = []
        self.spl = '\n'


    def open_class(self):
        text = ''
        try:
            with open('classes.txt', 'r', encoding='utf-8') as f:
                text = f.read()
        except:
            QMessageBox.critical(self, "Помилка ", "Файлу з класифікаторами не має", QMessageBox.Ok)

        if text != '':
            text = re.sub(r"[-()\"#/@;:<>{}=~|.?,]","",text)
            classes = text.split('\n')
            for el in classes:
                clas = el.split(' ')
                self.list.append(clas[0])
                n = len(clas)
                for i in range(1, n):
                    if clas[i].isdigit():
                        self.k.append(int(clas[i]))
                    else:
                        self.colum.append(clas[i])
```

```
    return self.list


def showDialog(self):
    fn = QFileDialog.getOpenFileName(self, 'Open file', None, "File (*.txt *.xlsx *.pdf, *.docx)")
    fname = "".join(fn[0])

    if fn[0]:
        try:
            if fname.endswith(".txt") == True:
                with open(fname, 'r', encoding='utf-8') as f:
                    self._data = f.readlines()

            elif fname.endswith(".xlsx") == True:
                xl = pd.ExcelFile(fname)
                sh = xl.sheet_names
                for el in sh:
                    df = xl.parse(el)
                    n = df.shape[0]
                    for i in range(n):
                        self._data.append(df.values[i, 0])


            elif fname.endswith(".docx") == True:
                data = docx.Document(fname)
                all_paras = data.paragraphs
                for para in all_paras:
                    self._data.append(para.text)


            else:
                QMessageBox.critical(self, "Помилка ", "Неможливо відкрити файл", QMessageBox.Ok)
        except:
            QMessageBox.critical(self, "Помилка ", "Файл пустий", QMessageBox.Ok)
    else:
        QMessageBox.critical(self, "Помилка ", "Не має файлу", QMessageBox.Ok)

    if  len(self._data) <= 0:
        QMessageBox.critical(self, "Помилка ", "Немає таблиці даних у файлу", QMessageBox.Ok)


def normalize(self, texts):
    self.tokenizer.fit_on_texts(texts)
    data = self.tokenizer.texts_to_sequences(texts)
    return pad_sequences(data, maxlen=self.max_text_len)


def classification(self, index, text):
    if self.spl != '\n':
        self._data = self._data[0].split(self.spl)
    if len(text) > 1:
        self._data = text.split(self.spl)
    r = pd.DataFrame()
```

```
    vec = []
    if self._data != []:
        x = self.normalize(self._data)
        f_model = self.list[index] + '.h5'
        model = load_model(f_model)
        res = model.predict(x)
        n = len(res)
        m = len(self.k)
        k = sum(self.k)
        v = 0
        matr = []
        for i in range(n):
            arr = []
            for j in range(m):
                maxi = res[i][0+v]
                j = 0+v
                for f in range(1,self.k[j]):
                    if maxi< res[i][f+v]:
                        maxi = res[i][f+v]
                        j = f+v
                for f in range(self.k[j]):
                    if f+v == j:
                        arr.append(1)
                    else:
                        arr.append(0)
            matr.append(arr)
        for j in range(k):
            arr = []
            for i in range(n):
                arr.append(matr[i][j])
            r[str(self.colum[j])] = arr
        for j in range(k):
            a = 0
            for i in range(n):
                if r.values[i, j] == 1:
                    a = a+1
            vec.append(a)
    else:
        QMessageBox.critical(self, "Помилка ", "Треба завантажити данні", QMessageBox.Ok)
    return r, vec, self.colum


def open_url(self, url):
    rs = requests.get(url)
    root = BeautifulSoup(rs.content, 'html.parser')
    com = root.find_all("dd")
    com = re.sub(r"<dd><br/>","",com)
    self._data = com.split('</dd>')


def save_result(self, r, vec, fn):
    if fn != '' and len(r)>0 and len(vec)>0:
        fn = fn+'.xlsx'
```

```
            r.to_excel(fn, sheet_name='Sheet_1')
            fig, ax = plt.subplots()
            ax.bar(self.colum, vec)
            name_p = 'im.png'
            plt.savefig(name_p)
            writer = pd.ExcelWriter(fn, engine = 'xlsxwriter')
            r.to_excel(writer, sheet_name='Sheet_2')
            worksheet = writer.sheets['Sheet_2']
            worksheet.insert_image('C2',name_p)
            writer.save()
            os.remove(name_p)
        else:
            QMessageBox.critical(self, "Помилка ", "Треба ввести назву файла або провести класифікацію", QMessageBox.Ok)


class Ui_MainWindow(object):
    def setupUi(self, MainWindow):
        MainWindow.setObjectName("MainWindow")
        MainWindow.resize(736, 629)
        self.centralwidget = QtWidgets.QWidget(MainWindow)
        self.centralwidget.setObjectName("centralwidget")
        self.tabWidget = QtWidgets.QTabWidget(self.centralwidget)
        self.tabWidget.setGeometry(QtCore.QRect(-4, -1, 741, 621))
        self.tabWidget.setObjectName("tabWidget")
        self.tab_3 = QtWidgets.QWidget()
        self.tab_3.setObjectName("tab_3")
        self.readFileButton = QtWidgets.QPushButton(self.tab_3)
        self.readFileButton.setGeometry(QtCore.QRect(10, 30, 151, 31))
        font = QtGui.QFont()
        font.setPointSize(10)
        self.readFileButton.setFont(font)
        self.readFileButton.setObjectName("readFileButton")
        self.clasButton = QtWidgets.QPushButton(self.tab_3)
        self.clasButton.setGeometry(QtCore.QRect(350, 70, 151, 31))
        font = QtGui.QFont()
        font.setPointSize(10)
        self.clasButton.setFont(font)
        self.clasButton.setObjectName("clasButton")
        self.saveButton = QtWidgets.QPushButton(self.tab_3)
        self.saveButton.setGeometry(QtCore.QRect(530, 70, 151, 31))
        font = QtGui.QFont()
        font.setPointSize(10)
        self.saveButton.setFont(font)
        self.saveButton.setObjectName("saveButton")
        self.comboBox = QtWidgets.QComboBox(self.tab_3)
        self.comboBox.setGeometry(QtCore.QRect(180, 50, 151, 22))
        self.comboBox.setObjectName("comboBox")
        self.label = QtWidgets.QLabel(self.tab_3)
        self.label.setGeometry(QtCore.QRect(180, 10, 161, 20))
        self.label.setObjectName("label")
        self.label_2 = QtWidgets.QLabel(self.tab_3)
        self.label_2.setGeometry(QtCore.QRect(530, 0, 161, 20))
        self.label_2.setObjectName("label_2")
```

```
self.File_textEdit = QtWidgets.QTextEdit(self.tab_3)
self.File_textEdit.setGeometry(QtCore.QRect(520, 20, 171, 41))
self.File_textEdit.setObjectName("File_textEdit")
self.textEdit = QtWidgets.QTextEdit(self.tab_3)
self.textEdit.setGeometry(QtCore.QRect(10, 240, 261, 321))
self.textEdit.setObjectName("textEdit")
self.label_3 = QtWidgets.QLabel(self.tab_3)
self.label_3.setGeometry(QtCore.QRect(20, 210, 201, 31))
self.label_3.setObjectName("label_3")
self.URL_textEdit = QtWidgets.QTextEdit(self.tab_3)
self.URL_textEdit.setGeometry(QtCore.QRect(10, 120, 241, 41))
self.URL_textEdit.setObjectName("URL_textEdit")
self.label_4 = QtWidgets.QLabel(self.tab_3)
self.label_4.setGeometry(QtCore.QRect(20, 90, 231, 31))
self.label_4.setObjectName("label_4")
self.URL_Button = QtWidgets.QPushButton(self.tab_3)
self.URL_Button.setGeometry(QtCore.QRect(10, 170, 151, 31))
font = QtGui.QFont()
font.setPointSize(10)
self.URL_Button.setFont(font)
self.URL_Button.setObjectName("URL_Button")
self.textEdit_2 = QtWidgets.QTextEdit(self.tab_3)
self.textEdit_2.setGeometry(QtCore.QRect(340, 20, 171, 41))
self.textEdit_2.setObjectName("textEdit_2")
self.label_5 = QtWidgets.QLabel(self.tab_3)
self.label_5.setGeometry(QtCore.QRect(350, 0, 151, 21))
self.label_5.setObjectName("label_5")
self.tableI_1 = QtWidgets.QTableView(self.tab_3)
self.tableI_1.setGeometry(QtCore.QRect(290, 110, 401, 201))
self.tableI_1.setObjectName("tableI_1")
self.gridLayoutWidget_11 = QtWidgets.QWidget(self.tab_3)
self.gridLayoutWidget_11.setGeometry(QtCore.QRect(290, 320, 401, 241))
self.gridLayoutWidget_11.setObjectName("gridLayoutWidget_11")
self.gridLayout_1 = QtWidgets.QGridLayout(self.gridLayoutWidget_11)
self.gridLayout_1.setContentsMargins(0, 0, 0, 0)
self.gridLayout_1.setObjectName("gridLayout_1")
self.canvas  = MplCanvas(self, width=5, height=4, dpi=100)
self.gridLayout_1.addWidget(self.canvas)
self.tabWidget.addTab(self.tab_3, "")
self.tab = QtWidgets.QWidget()
self.tab.setObjectName("tab")
self.name_textEdit = QtWidgets.QTextEdit(self.tab)
self.name_textEdit.setGeometry(QtCore.QRect(30, 20, 171, 31))
self.name_textEdit.setObjectName("name_textEdit")
self.nub_textEdit = QtWidgets.QTextEdit(self.tab)
self.nub_textEdit.setGeometry(QtCore.QRect(30, 70, 171, 31))
self.nub_textEdit.setObjectName("nub_textEdit")
self.label_6 = QtWidgets.QLabel(self.tab)
self.label_6.setGeometry(QtCore.QRect(10, 0, 211, 21))
self.label_6.setObjectName("label_6")
self.label_7 = QtWidgets.QLabel(self.tab)
self.label_7.setGeometry(QtCore.QRect(10, 40, 251, 41))
self.label_7.setObjectName("label_7")
```

```
self.add_train_Button_3 = QtWidgets.QPushButton(self.tab)
self.add_train_Button_3.setGeometry(QtCore.QRect(40, 110, 161, 31))
font = QtGui.QFont()
font.setPointSize(10)
self.add_train_Button_3.setFont(font)
self.add_train_Button_3.setObjectName("add_train_Button_3")
self.add_test_Button = QtWidgets.QPushButton(self.tab)
self.add_test_Button.setGeometry(QtCore.QRect(40, 150, 161, 31))
font = QtGui.QFont()
font.setPointSize(10)
self.add_test_Button.setFont(font)
self.add_test_Button.setObjectName("add_test_Button")
self.add_final_test_Button = QtWidgets.QPushButton(self.tab)
self.add_final_test_Button.setGeometry(QtCore.QRect(40, 190, 161, 31))
font = QtGui.QFont()
font.setPointSize(10)
self.add_final_test_Button.setFont(font)
self.add_final_test_Button.setObjectName("add_final_test_Button")
self.spinBox = QtWidgets.QSpinBox(self.tab)
self.spinBox.setGeometry(QtCore.QRect(220, 270, 42, 22))
self.spinBox.setObjectName("spinBox")
self.label_11 = QtWidgets.QLabel(self.tab)
self.label_11.setGeometry(QtCore.QRect(10, 260, 151, 41))
self.label_11.setObjectName("label_11")
self.learn_Button = QtWidgets.QPushButton(self.tab)
self.learn_Button.setGeometry(QtCore.QRect(10, 300, 121, 31))
font = QtGui.QFont()
font.setPointSize(10)
self.learn_Button.setFont(font)
self.learn_Button.setObjectName("learn_Button")
self.save_class_Button = QtWidgets.QPushButton(self.tab)
self.save_class_Button.setGeometry(QtCore.QRect(140, 300, 121, 31))
font = QtGui.QFont()
font.setPointSize(10)
self.save_class_Button.setFont(font)
self.save_class_Button.setObjectName("save_class_Button")
self.gridLayoutWidget_12 = QtWidgets.QWidget(self.tab)
self.gridLayoutWidget_12.setGeometry(QtCore.QRect(280, 280, 441, 281))
self.gridLayoutWidget_12.setObjectName("gridLayoutWidget_12")
self.gridLayout_2 = QtWidgets.QGridLayout(self.gridLayoutWidget_12)
self.gridLayout_2.setContentsMargins(0, 0, 0, 0)
self.gridLayout_2.setObjectName("gridLayout_2")
self.canvas2  = MplCanvas(self, width=5, height=4, dpi=100)
self.gridLayout_2.addWidget(self.canvas2)
self.textEdit_3 = QtWidgets.QTextEdit(self.tab)
self.textEdit_3.setGeometry(QtCore.QRect(280, 10, 441, 261))
self.textEdit_3.setObjectName("textEdit_3")
self.textEdit_4 = QtWidgets.QTextEdit(self.tab)
self.textEdit_4.setGeometry(QtCore.QRect(10, 350, 251, 211))
self.textEdit_4.setObjectName("textEdit_4")
self.label_21 = QtWidgets.QLabel(self.tab)
self.label_21.setGeometry(QtCore.QRect(10, 230, 211, 41))
self.label_21.setObjectName("label_21")
```

```
        self.spinBox_3 = QtWidgets.QSpinBox(self.tab)
        self.spinBox_3.setGeometry(QtCore.QRect(220, 240, 42, 22))
        self.spinBox_3.setMinimum(100)
        self.spinBox_3.setMaximum(10000)
        self.spinBox_3.setSingleStep(100)
        self.spinBox_3.setObjectName("spinBox_3")
        self.tabWidget.addTab(self.tab, "")
        self.tab_2 = QtWidgets.QWidget()
        self.tab_2.setObjectName("tab_2")
        self.label_12 = QtWidgets.QLabel(self.tab_2)
        self.label_12.setGeometry(QtCore.QRect(10, 0, 291, 361))
        self.label_12.setScaledContents(False)
        self.label_12.setObjectName("label_12")
        self.label_13 = QtWidgets.QLabel(self.tab_2)
        self.label_13.setGeometry(QtCore.QRect(310, 0, 421, 591))
        self.label_13.setObjectName("label_13")
        self.tabWidget.addTab(self.tab_2, "")
        MainWindow.setCentralWidget(self.centralwidget)
        self.statusbar = QtWidgets.QStatusBar(MainWindow)
        self.statusbar.setObjectName("statusbar")
        MainWindow.setStatusBar(self.statusbar)

        self.retranslateUi(MainWindow)
        self.tabWidget.setCurrentIndex(0)
        QtCore.QMetaObject.connectSlotsByName(MainWindow)

    def retranslateUi(self, MainWindow):
        _translate = QtCore.QCoreApplication.translate
        MainWindow.setWindowTitle(_translate("MainWindow", "MainWindow"))
        self.readFileButton.setText(_translate("MainWindow", "Завантажити файл"))
        self.clasButton.setText(_translate("MainWindow", "Класифікувати"))
        self.saveButton.setText(_translate("MainWindow", "Зберегти результат"))
        self.label.setText(_translate("MainWindow", "<html><head/><body><p><span style=\" font-size:11pt;\">Оберіть класифікатор</span></p></body></html>"))
        self.label_2.setText(_translate("MainWindow", "<html><head/><body><p><span style=\" font-size:11pt;\">Введіть назву файла</span></p></body></html>"))
        self.label_3.setText(_translate("MainWindow", "<html><head/><body><p><span style=\" font-size:12pt;\">Поле для вводу тексту</span></p></body></html>"))
        self.label_4.setText(_translate("MainWindow", "<html><head/><body><p><span style=\" font-size:11pt;\">Введіть URL адресу</span></p></body></html>"))
        self.URL_Button.setText(_translate("MainWindow", "Завантажити URL"))
        self.label_5.setText(_translate("MainWindow", "<html><head/><body><p><span style=\" font-size:11pt;\">Введіть роздільник</span></p></body></html>"))
        self.tabWidget.setTabText(self.tabWidget.indexOf(self.tab_3), _translate("MainWindow", "Класифікація"))
        self.label_6.setText(_translate("MainWindow", "<html><head/><body><p><span style=\" font-size:11pt;\">Введіть назву класифікатора</span></p></body></html>"))
        self.label_7.setText(_translate("MainWindow", "<html><head/><body><p><span style=\" font-size:11pt;\">Введіть кількість кожного підкласу</span></p></body></html>"))
        self.add_train_Button_3.setText(_translate("MainWindow", "Додати тренувальні дані"))
        self.add_test_Button.setText(_translate("MainWindow", "Додати тестові дані"))
        self.add_final_test_Button.setText(_translate("MainWindow", "Додати фінальні тести"))
        self.label_11.setText(_translate("MainWindow", "<html><head/><body><p><span style=\" font-size:11pt;\">Оберіть кількість епох </span></p></body></html>"))
```

```
        self.learn_Button.setText(_translate("MainWindow", "Навчити"))
        self.save_class_Button.setText(_translate("MainWindow", "Зберегти "))
        self.label_21.setText(_translate("MainWindow", "<html><head/><body><p><span style=\" font-size:11pt;\">Оберіть кількість вхідних слів </span></p></body></html>"))
        self.tabWidget.setTabText(self.tabWidget.indexOf(self.tab), _translate("MainWindow", "Створити класифікатор"))
        self.label_12.setText(_translate("MainWindow", "<html><head/><body><p><span style=\" font-family:\'Times New Roman,serif\'; font-size:9pt; font-weight:600;\">Алгоритм дій при класифікації:</span></p><p><span style=\" font-family:\'Times New Roman,serif\'; font-size:9pt;\">1. Завантажити текст:</span></p><p><span style=\" font-family:\'Times New Roman,serif\'; font-size:9pt;\">- завантажити з файлу (.docx, .txt, .excel);</span></p><p><span style=\" font-family:\'Times New Roman,serif\'; font-size:9pt;\">- ввести URL адресу сторінки з коментарями;</span></p><p><span style=\" font-family:\'Times New Roman,serif\'; font-size:9pt;\">- власноруч ввести текст у текстове поле.</span></p><p><span style=\" font-family:\'Times New Roman,serif\'; font-size:9pt;\">2. Обрати класифікатор (існуючий або з новостворених).</span></p><p><span style=\" font-family:\'Times New Roman,serif\'; font-size:9pt;\">3. Ввести роздільник текстів (за необхідністю, якщо</span></p><p><span style=\" font-family:\'Times New Roman,serif\'; font-size:9pt;\"> це не абзац), наприклад (/).</span></p><p><span style=\" font-family:\'Times New Roman,serif\'; font-size:9pt;\">4. Натиснути на кнопку «Класифікувати».</span></p><p><span style=\" font-family:\'Times New Roman,serif\'; font-size:9pt;\">5. За необхідністю можна зберегти результати у .excel, </span></p><p><span style=\" font-family:\'Times New Roman,serif\'; font-size:9pt;\">для цього необхідно дати повну назву файлу</span></p><p><span style=\" font-family:\'Times New Roman,serif\'; font-size:9pt;\"> (D:/…/name) та натиснути на кнопку «Зберегти». </span></p></body></html>"))
        self.label_13.setText(_translate("MainWindow", "<html><head/><body><p><span style=\" font-family:\'Times New Roman,serif\'; font-size:9pt; font-weight:600;\">Алгоритм дій при створенні класифікатора:</span></p><p><span style=\" font-family:\'Times New Roman,serif\'; font-size:9pt;\">1. Ввести назву класифікатора.</span></p><p><span style=\" font-family:\'Times New Roman,serif\'; font-size:9pt;\">2. Ввести кількість кожного з підкласу, наприклад (телефон, ноутбук, планшет, </span></p><p><span style=\" font-family:\'Times New Roman,serif\'; font-size:9pt;\">хороший, поганий – 3, 2).</span></p><p><span style=\" font-family:\'Times New Roman,serif\'; font-size:9pt;\">3. Додати тренувальні дані, обравши файл для кожного класу формату .excel, </span></p><p><span style=\" font-family:\'Times New Roman,serif\'; font-size:9pt;\">де повинні бути підкласи на різних листах, наприклад (у файлі Телефон.excel </span></p><p><span style=\" font-family:\'Times New Roman,serif\'; font-size:9pt;\">є два листи: один з хорошими відгуками, інший з поганими відгуками). Де </span></p><p><span style=\" font-family:\'Times New Roman,serif\'; font-size:9pt;\">обов’язково повинно бути у першому стовбці – текст, у інших відношення </span></p><p><span style=\" font-family:\'Times New Roman,serif\'; font-size:9pt;\">тексту до кожного підкласу, наприклад («Телефон гарний» - 1-0-0-1-0). У </span></p><p><span style=\" font-family:\'Times New Roman,serif\'; font-size:9pt;\">перших рядках – заголовки таблиць (Коментар, Телефон, Ноутбук, Планшет, </span></p><p><span style=\" font-family:\'Times New Roman,serif\'; font-size:9pt;\">Хороший, Поганий). Заповнених даних повинно бути багато та однією мовою, </span></p><p><span style=\" font-family:\'Times New Roman,serif\'; font-size:9pt;\">чим більше тренувальних даних, тим краще.</span></p><p><span style=\" font-family:\'Times New Roman,serif\'; font-size:9pt;\"> 4. Додати тестові дані у одному файлі всі види класів, з позначками віднесення </span></p><p><span style=\" font-family:\'Times New Roman,serif\'; font-size:9pt;\">до кожного підкласу («Телефон гарний» - 1-0-0-1-0). Перший рядок у таблиці -</span></p><p><span style=\" font-family:\'Times New Roman,serif\'; font-size:9pt;\">заголовки (Коментар, Телефон, Ноутбук, Планшет, Хороший, Поганий).</span></p><p><span style=\" font-family:\'Times New Roman,serif\'; font-size:9pt;\">5. Так само додати фінальні тестові данні, які повинні відрізнятися від даних</span></p><p><span style=\" font-family:\'Times New Roman,serif\'; font-size:9pt;\"> на попередніх етапах. </span></p><p><span style=\" font-family:\'Times New Roman,serif\'; font-size:9pt;\">6. Обрати кількість епох (5-10) та кількість слів.</span></p><p><span style=\" font-family:\'Times New Roman,serif\'; font-size:9pt;\">7. Натиснути кнопку «Навчити».</span></p><p><span style=\" font-family:\'Times New Roman,serif\'; font-size:9pt;\">8. В залежності від показників повторити етапи 6 та 7.</span></p><p><span style=\" font-family:\'Times New Roman,serif\'; font-size:9pt;\">9. Натиснути кнопку «Зберегти».</span></p></body></html>"))
        self.tabWidget.setTabText(self.tabWidget.indexOf(self.tab_2), _translate("MainWindow", "Правили користуванням"))


class PresenterMain(QMainWindow, Ui_MainWindow):

    def __init__(self):
        super(PresenterMain, self).__init__()
        self.setupUi(self)
```

```
        self.res = pd.DataFrame()
        self.vec = []
        self.obj = TextData()
        self.obj2 = Learning()
        self.ft1 = ''
        self.ft2 = ''
        lis = self.obj.open_class()
        self.readFileButton.pressed.connect(self.downloadF)
        self.clasButton.pressed.connect(self.classification)
        self.comboBox.addItems(lis)
        self.saveButton.pressed.connect(self.save_res)
        self.add_train_Button_3.pressed.connect(self.add_train)
        self.add_test_Button.pressed.connect(self.add_test)
        self.add_final_test_Button.pressed.connect(self.add_final_test)
        self.learn_Button.pressed.connect(self.learn)
        self.save_class_Button.pressed.connect(self.save_class)

    def add_test(self):
        self.ft1 = self.obj2.get_test_data()

    def add_final_test(self):
        self.ft2 = self.obj2.get_test_data()

    def learn(self):
        name = self.name_textEdit.toPlainText()
        numb = self.nub_textEdit.toPlainText()
        n_w = self.spinBox_3.value()
        n_ep = self.spinBox.value()
        self.obj2.get_param(name, numb, n_w, n_ep)
        X, Y = self.obj2.creat_x_y()
        self.obj2.creat_model()
        history = self.obj.learn_model(X, Y)
        x, y = self.obj2.creat_x_y(self.ft1)
        self.obj2.test_model(x, y)
        x1, y1 =  self.obj2.creat_x_y(self.ft2)
        self.obj2.test_model(x1, y1)

    def add_train(self):
        self.obj2.get_data_train()


    def save_class(self):
        self.obj2.save_model

    def save_res(self):
        name_f = self.File_textEdit.toPlainText()
        self.obj.save_result(self.res, self.vec, name_f)


    def downloadF (self):
        self.obj.showDialog()

    def classification(self):
```

```
        text = self.textEdit.toPlainText()
        urlt = self.URL_textEdit.toPlainText()
        spl = self.textEdit_2.toPlainText()
        if spl != '':
            self.obj.spl = spl
        if text != '' and urlt == '' and self.obj._data == []:
            self.obj._data = text
        elif urlt != '' and text == '' and self.obj._data == []:
            self.obj.open_url(urlt)
        elif text != '' and urlt != '' and self.obj._data == []:
            QMessageBox.critical(self, "Помилка ", "Оберіть один спосіб завантаження даних", QMessageBox.Ok)
        elif  urlt != '' and self.odj._data != [] and text == '' :
            QMessageBox.critical(self, "Помилка ", "Оберіть один спосіб завантаження даних", QMessageBox.Ok)
        elif urlt != '' and self.odj._data != [] and text != '' :
            QMessageBox.critical(self, "Помилка ", "Оберіть один спосіб завантаження даних", QMessageBox.Ok)
        if self.obj._data == []:
            QMessageBox.critical(self, "Помилка ", "Потрібно завантажити дані", QMessageBox.Ok)
        else:
            index = self.comboBox.currentIndex()
            self.res, self.vec, col = self.obj.classification(index, text)
            model1 = pandasModel(self.res)
            self.tableI_1.setModel(model1)
            x = range(len(col))
            self.canvas.axes.bar(x,self.vec, align = 'edge')
            self.canvas.axes.set_xticks(x)
            self.canvas.axes.set_xticklabels(col)
            self.canvas.draw()



def main():
    app = QApplication(sys.argv)
    #appctxt = ApplicationContext()
    window = PresenterMain()
    window.show()
    sys.exit(app.exec_())
    #sys.exit(appctxt.app.exec_())

if __name__ == '__main__':
    main()
```

ДОДАТОК В

Роздатковий матеріал

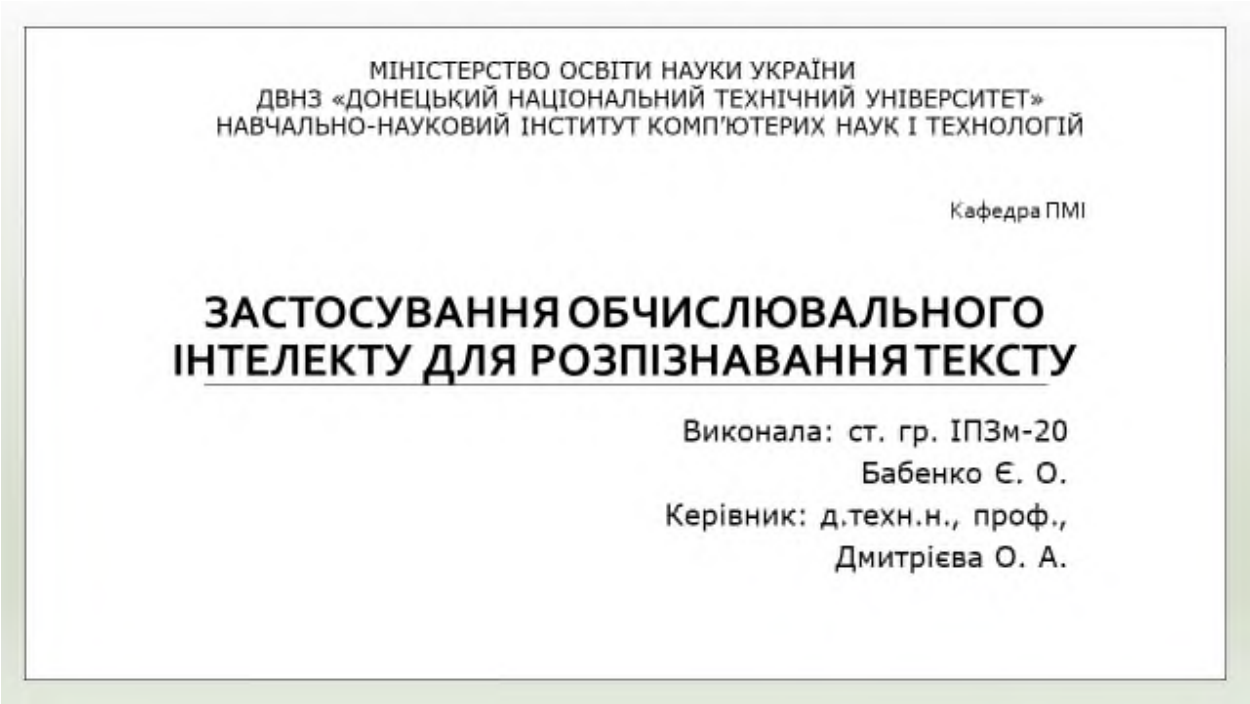

Рисунок В.1 – Лист 1

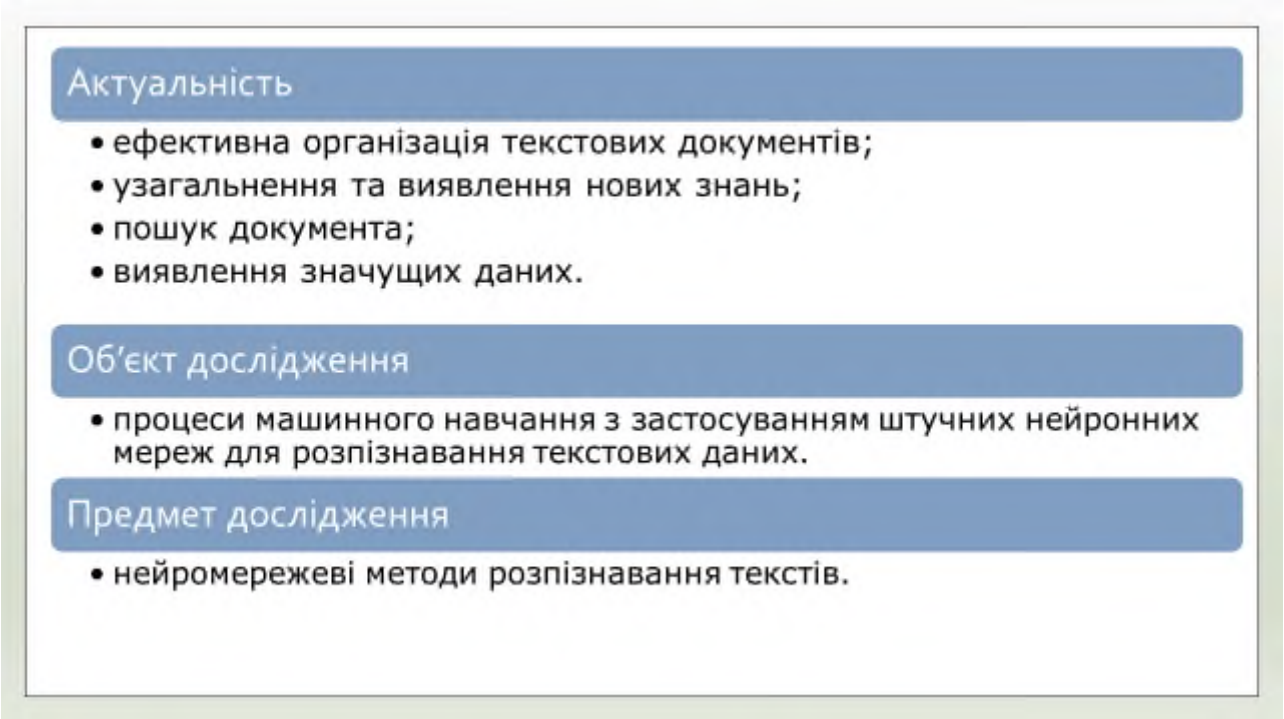

Рисунок В.2 – Лист 2

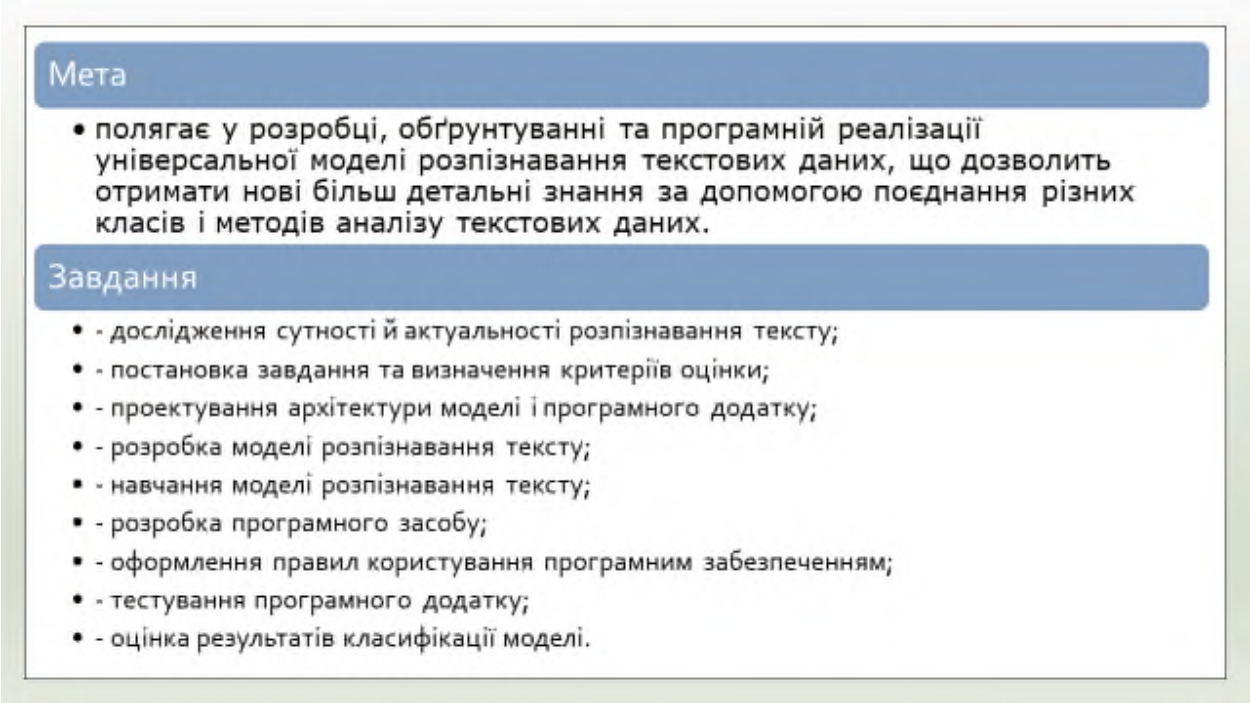

Рисунок В.3 – Лист 3

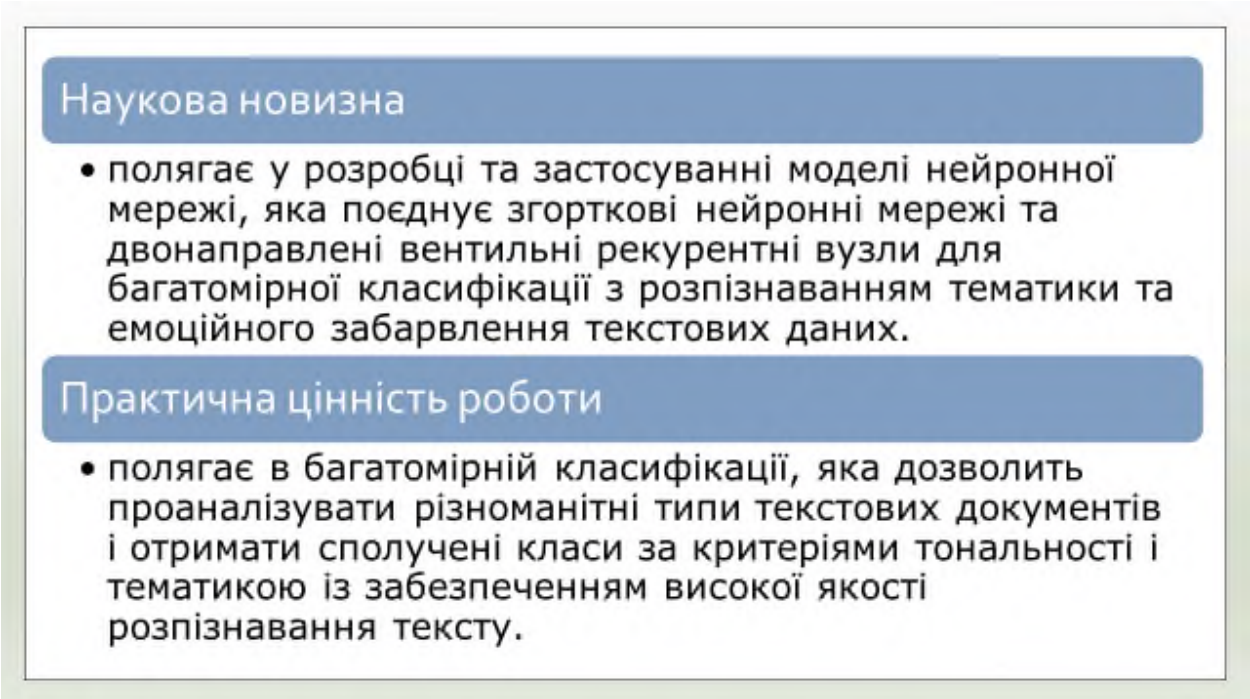

Рисунок В.4 – Лист 4

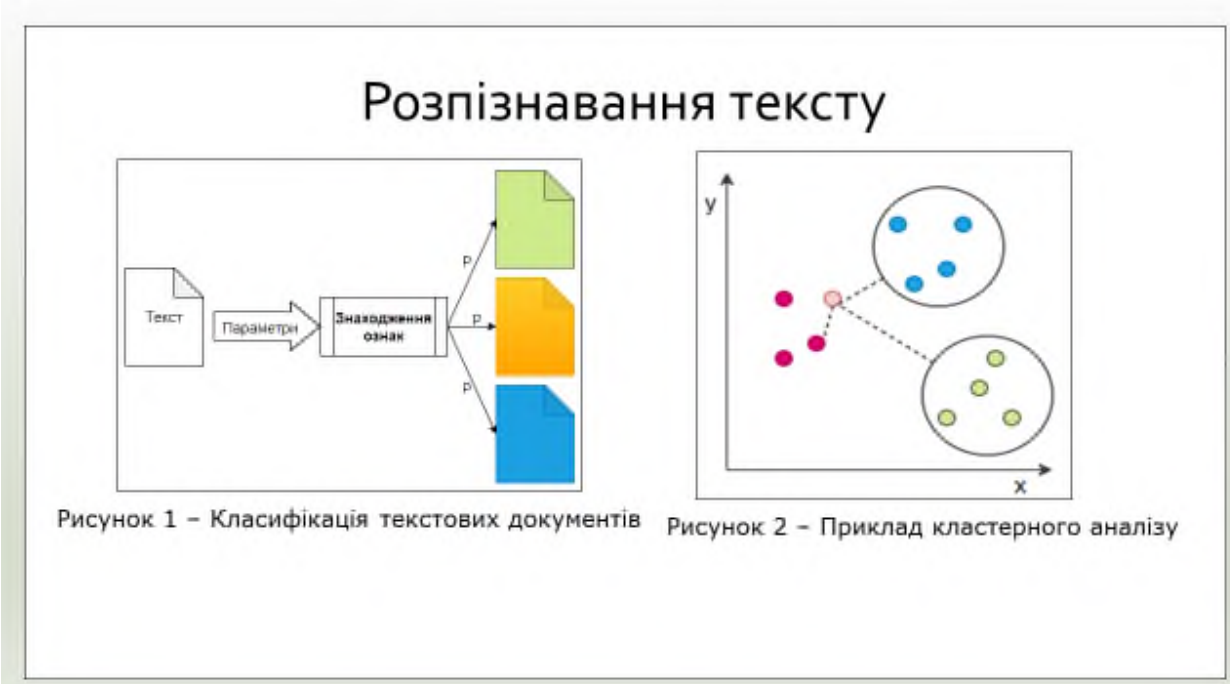

Рисунок В.5 – Лист 5

Рисунок В.6 – Лист 6

Рисунок 4 – Основні підходи у класифікації та кластерному аналізі

Рисунок В.7 – Лист 7

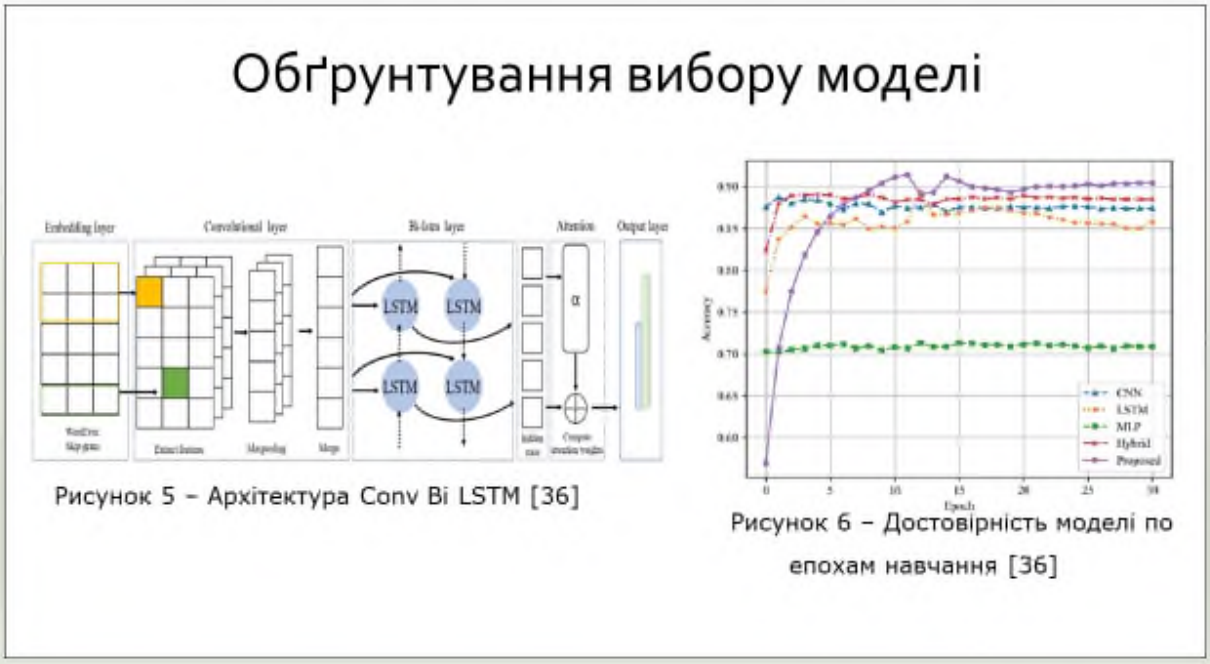

Рисунок 5 – Архітектура Conv Bi LSTM [36]

Рисунок 6 – Достовірність моделі по епохам навчання [36]

Рисунок В.8 – Лист 8

Обґрунтування вибору моделі

Neural Network	Character Input			Word Input		
	P	R	F1	P	R	F1
Bi-GRU	**97.81%**	97.69%	97.75%	**99.22%**	99.10%	**99.16%**
Bi-LSTM	97.69%	**97.88%**	**97.78%**	99.14%	**99.11%**	99.12%
GRU	91.15%	85.81%	88.40%	94.62%	91.87%	93.22%
LSTM	90.65%	86.56%	88.56%	93.99%	91.92%	92.94%

Рисунок 7 – Порівняльний аналіз нейронних мереж за показниками якості [40]

Neural Network	Character Input	Word Input
Bi-GRU	**517 s/epoch**	**521 s/epoch**
Bi-LSTM	562 s/epoch	567 s/epoch
GRU	**269 s/epoch**	**270 s/epoch**
LSTM	292 s/epoch	293 s/epoch

Рисунок 8 – Порівняльний аналіз нейронних мереж за часом навчання [40]

Рисунок В.9 – Лист 9

Модель Conv Bi GRU

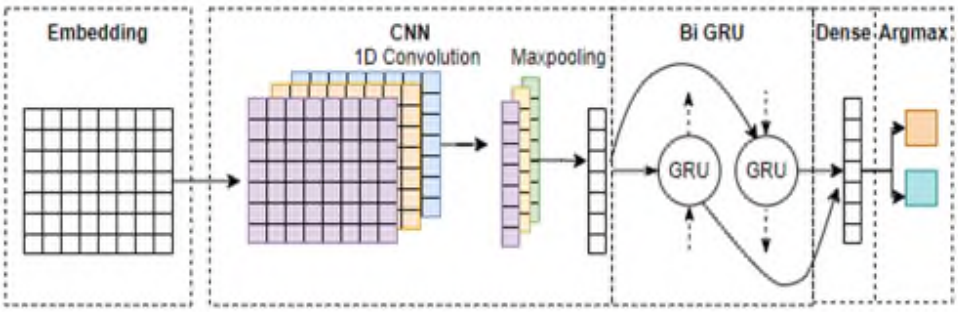

Рисунок 5 – Архітектура моделі класифікації текстів Conv Bi GRU

Рисунок В.10 – Лист 10

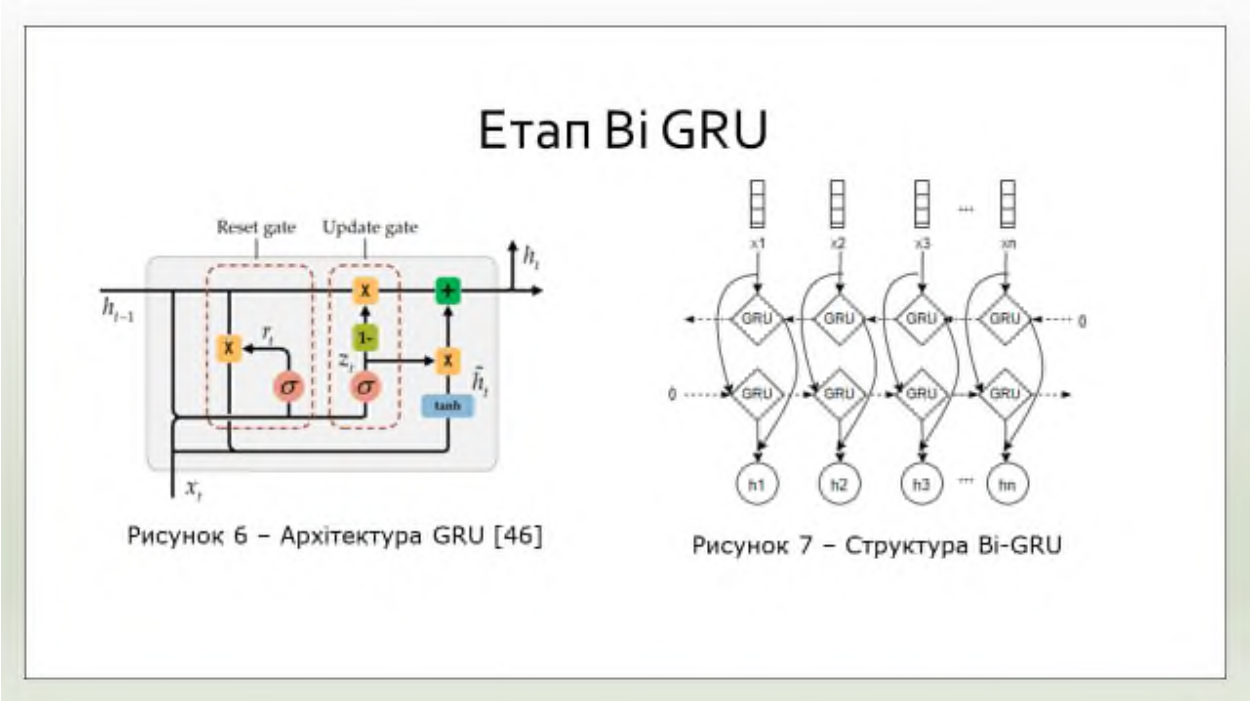

Рисунок 6 – Архітектура GRU [46]

Рисунок 7 – Структура Bi-GRU

Рисунок В.11 – Лист 11

Рисунок 8 – Діаграма варіантів користування

Рисунок В.12 – Лист 12

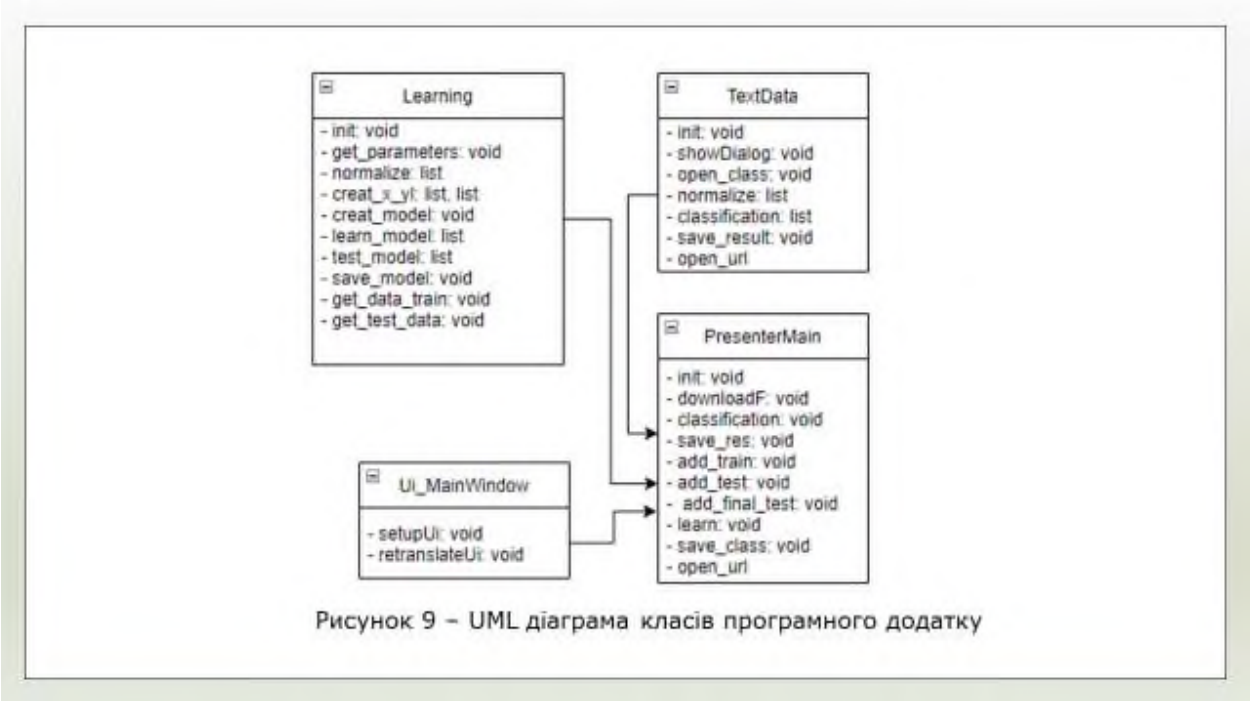

Рисунок В.13 – Лист 13

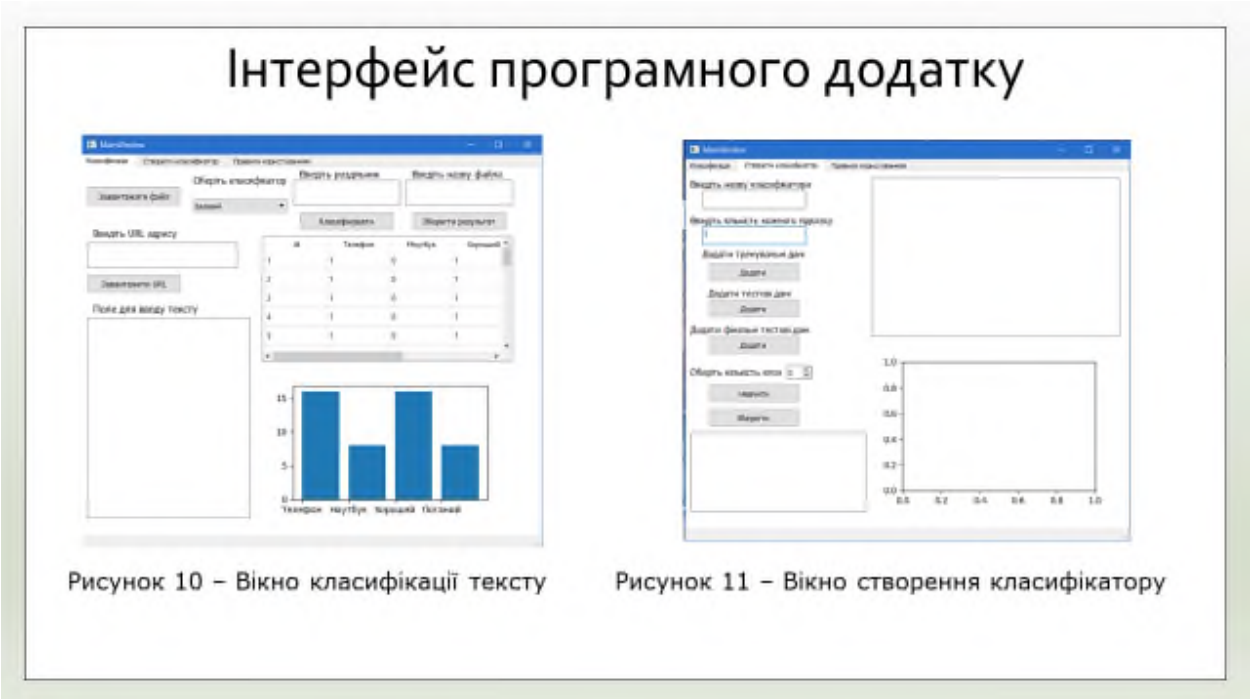

Рисунок В.14 – Лист 14

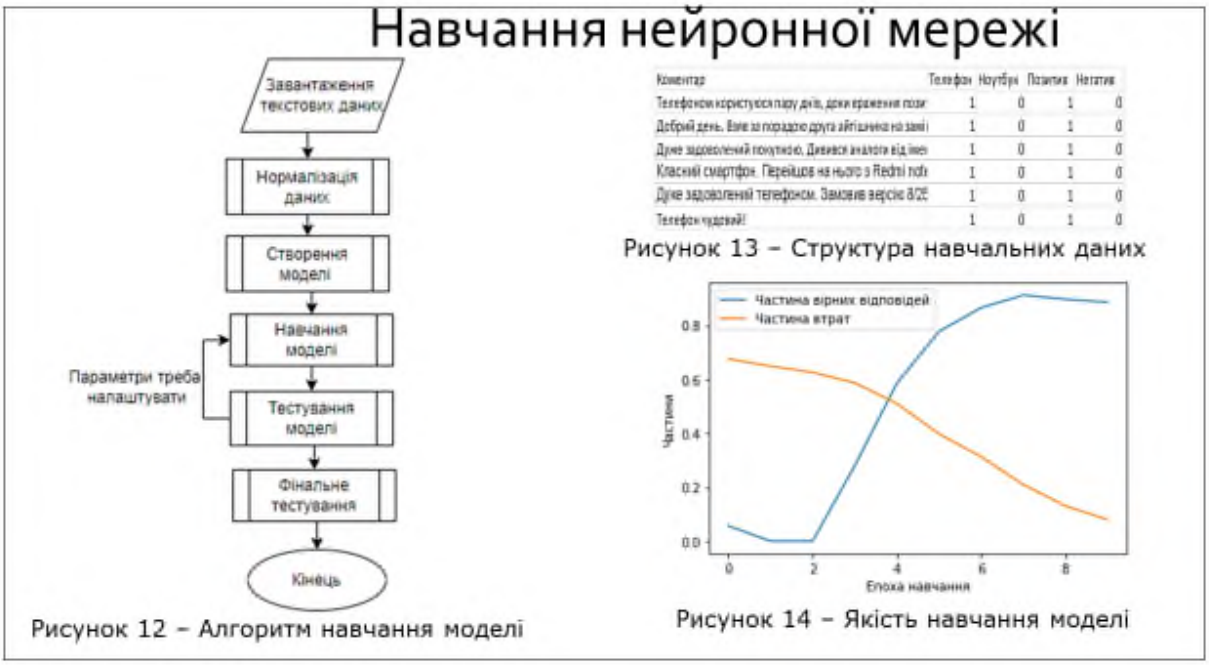

Рисунок В.15 – Лист 15

Таблиця 1 - Порівняльна характеристика програмних засобів

Параметр	MegaIndex	Amazon Comprehend	Mediatoolkit	ABBYY FlexiCapture	Власна модель
Вид додатку	Web-сервіс	Web-сервіс	IOS-сервіс	Програмне забезпечення	Програмне забезпечення
Підтримка мови	Російська, англійська	Більшість мов	Більшість мов	Більшість мов	Українська й інші
Підтримка форматів даних	URL	URL, текстові документи,.	URL	URL, текстові документи, зображення , аудіо-файли	URL, текстові документи та таблиці,.
Сентимент аналіз	-	+	+	-	+
Аналіз тематики	+	+	-	+	+
Добавлення нового класифікатору	-	+	-	+	+
Сортування документів	-	+	-	+	-
Статистика класів	+	+	+	+	+

Рисунок В.16 – Лист 16

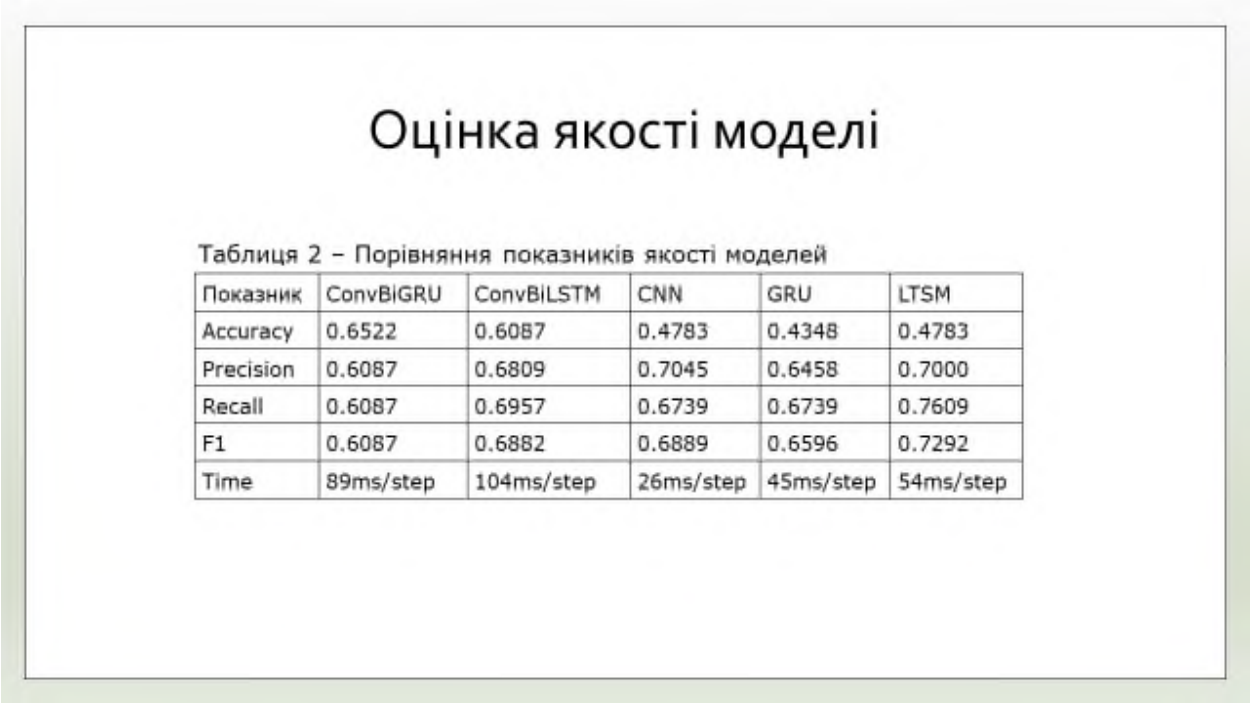

Рисунок В.17 – Лист 17

Рисунок В.18 – Лист 18