Вінницький національний технічний університет
Факультет інформаційних технологій та комп’ютерної інженерії
Кафедра комп’ютерних наук

Пояснювальна записка

до магістерської кваліфікаційної роботи

на тему «Інформаційна технологія класифікації банківських текстів на основі згорткової нейронної мережі»

Виконав: студент 2 курсу,
групи 2КН-18 м
спеціальності 122 «Комп'ютерні науки»
Перородов А. О.

Керівник: к.т.н., доц. Колесницький О.К.

Рецензент: к. т. н., доц. Войтко В. В.

Вінниця
2019

ЗАТВЕРДЖУЮ
Завідувач кафедри____КН____
д.т.н., проф.. Яровий А.А.

_____
(підпис)
“____” _____2019 року

ЗАВДАННЯ

на магістерську кваліфікаційну роботу на здобуття кваліфікації магістра наук зі спеціальності: 122 – «Комп'ютерні науки»

08-22.МКР.032.18.000.ПЗ

Магістранта групи 2КН-18м Переродова Артемія Олексійовича

Тема магістерської кваліфікаційної роботи: «Інформаційна технологія класифікації банківських текстів на основі згорткової нейронної мережі»

Вхідні дані: Обсяг тексту, що класифікується – не більше 400 слів; мова тексту, що класифікується – англійська; тематика текстів – банківська сфера; число класів, до яких може відноситись текст – 12; робота програми в режимі веб-сервісу (Web API); аутентифікація користувача по токену.

Короткий зміст частин магістерської кваліфікаційної роботи:

1. Графічна: Граф-схема алгоритму роботи програмного забезпечення класифікації банківських текстів, структура нейронної мережі; структура інформаційної технології, робочі вікна програми класифікації банківських текстів, результати роботи програми класифікації банківських текстів.
2. Текстова (пояснювальна записка): вступ, аналіз предметної області класифікації банківських текстів, розробка інформаційної технології класифікації банківських текстів на основі нейронної мережі, програмна реалізація інформаційної технології класифікації банківських текстів на основі нейронної мережі, економічна частина, висновки, перелік використаних джерел, додатки.

КАЛЕНДАРНИЙ ПЛАН ВИКОНАННЯ МКР

№ етапу	Назва етапу	Термін виконання		Очікувані результати
		початок	кінець	
1	Аналіз сучасного рівня інформаційних технологій класифікації банківських текстів. Постановка задач дослідження			Аналітичний огляд літературних джерел, задачі досліджень, розділ 1 ПЗ
2	Побудова математичних моделей функціонування нейронної мережі			Математичні моделі, розділ 2
3	Практичне застосування та оцінка ефективності розроблених моделей			розділ 3
4	Підготовка економічної частини			розділ 4
5	Апробація та/або впровадження результатів дослідження			тези доповідей/акт впровадження
6	Оформлення пояснювальної записки, графічного матеріалу та презентації			Пояснювальна записка, графічний матеріал, презентація

Консультанти з окремих розділів магістерської кваліфікаційної роботи

1. Науковий керівник _____ канд. техн. наук, доц., доц. кафедри КН
(підпис) наук. ступінь, вчене звання (посада)
"____" _____20___ р. _____ О. К. Колесницький _____
ініціали та прізвище

2. Економічна частина _____ канд. екон. наук, доц., доц. каф. ЕПВМ
(підпис) наук. ступінь, вчене звання (посада)
___М. В. Бальзан _____
ініціали та прізвище
"____" _____20___ р.

Дата попереднього захисту роботи "____" _____20___ р.

Рецензент _____ канд. техн. наук, доц., доц. кафедри ПЗ
(підпис) наук. ступінь, вчене звання (посада)
_____ В. В. Войтко _____
ініціали та прізвище

Завдання видав
науковий керівник _____ канд. техн. наук, доц., доц. кафедри КН
(підпис) наук. ступінь, вчене звання (посада)
_____О. К. Колесницький _____
ініціали та прізвище
"____" _____20___ р.

Завдання отримав магістрант _____ _____А.О. Переродов_____
(підпис) ініціали та прізвище
"____" _____20___ р.

АНОТАЦІЯ

Дана магістерська кваліфікаційна робота присвячена розв’язанню задачі розробки інтелектуальної інформаційної технології та програмного забезпечення класифікації банківських текстів згортковою нейронною мережею. Було розроблено метод попередньої обробки тексту, інформаційну модель процесу класифікації банківських текстів, визначено архітектуру згорткової нейронної мережі для класифікації текстів та математичну модель згорткової мережі. В ході практичної реалізації інформаційної технології класифікації банківських текстів було обрано такі мови програмування: для модуля класифікації текстів – Python, для модуля аутентифікації – C#, для клієнтського веб-додатку тестування роботи WebAPI – Typescript та бібліотеки Tensorflow та Flask. Розроблене програмне забезпечення класифікації банківських текстів на основі згорткової нейронної мережі порівняно з аналогом має кращу на 7,2% достовірність класифікації.

ABSTRACT

The master's degree work is dedicated to solving the problem of developing intelligent information technology and software for the classification of banking texts with a convolutional neural network. A pre-processing method was developed, an information model for the banking text classification process, a convolutional neural network architecture for text classification, and a mathematical convolution network model were defined. During the practical implementation of banking text classification information technology, the following programming languages were chosen: for the text classification module - Python, for the authentication module – C#, for the WebAPI client Web application testing application - Typescript and the Tensorflow and Flask libraries. Developed software for the classification of banking texts on the basis of convolutional neural network compared to its analogue has a better classification accuracy of 7.2%.

ЗМІСТ

ВСТУП

Актуальність. На поточному етапі розвитку інформаційного суспільства можливість доступу до величезної кількості інформації з будь-якого місця й в будь-який час стала як ніколи доступною. Інформаційні технології стали невід'ємною частиною побуту людини. За наявності доступу до мережі Інтернет легко знайти відповідь на питання, які людство коли-небудь ставило й знайшло на них відповідь. З кожним днем кількість користувачів й питань збільшується й, відповідно, збільшуються обсяги текстової інформації. Для забезпечення швидкого та зручного доступу до цієї інформації виникає потреба у її класифікації.

Класифікація текстової інформації потрібна для вирішення великої кількості задач:

1. Персоніфікація реклами. Контекстна реклама є основним джерелом доходу IT компаній. Вона відображається відвідувачам інтернет-сторінки, сфера інтересів яких потенційно збігається або перетинається з тематикою рекламованого товару або послуги, цільової аудиторії, що підвищує ймовірність їх відгуку на рекламу. Сфера інтересів визначається за текстом інтернет-сторінок переглянутих користувачем.

2. Фільтрація спаму. Спам – це небажані розсилки, які можуть приходити на адресу електронної пошти. Вони можуть містити рекламні пропозиції або комп'ютерні віруси. Завдання боротьби зі спамом полягає в тому, щоб класифікувати всі листи на два класи: спам і не спам.

3. Категоризація сайтів по тематичним каталогам. Дане завдання вирішується пошуковими системами і передбачає обробку документів і віднесення їх до одного з заданих класів.

4. Розпізнавання емоційного забарвлення текстів. Завдання полягає в тому, щоб оцінити відношенню автора до об'єктів, наприклад на основі відгуків про них.

5. Автоматичний розподіл скарг та запитань у службах підтримки. Великі підприємства, які надають різні види послуг, зазвичай мають великі служби підтримки та контакт-центри, на які витрачають багато фінансових ресурсів. Тому процес автоматизації перенаправлення заявок і скарг у відповідні відділи

пришвидшить прийняття відповідних рішень та допоможе витрачати менше ресурсів, як людських, так і фінансових.

В ході аналізу проблеми класифікації текстів у різних сферах діяльності було встановлено, що класифікація текстових документів для їх подальшого перенаправлення у відповідні відділи в банківських установах є досить актуальною проблемою через велику кількість вхідної кореспонденції, яка призводить до перезавантаженості служб банківського моніторингу. Під вхідною кореспонденцією мається на увазі листи, звернення, скарги та інші текстові документи отримані на адресу банку. Перезавантаженість виникає через додаткове візування вхідної кореспонденції людиною, після того як вони були класифіковані програмно. Підвищення точності класифікації дозволить вирішити проблему додаткового візування текстових документів.

Зв'язок роботи з науковими програмами, планами, темами. Магістерська робота виконана відповідно до напрямку наукових досліджень кафедри комп’ютерних наук Вінницького національного технічного університету 22 К1 «Моделі, методи, технології та пристрої інтелектуальних інформаційних систем управління, економіки, навчання та комунікацій» та плану наукової та навчально-методичної роботи кафедри.

Мета і завдання досліджень. Метою магістерської кваліфікаційної роботи є підвищення достовірності класифікації банківських текстів програмними засобами за рахунок застосування згорткових нейронних мереж.

Для досягнення мети розробки необхідно виконати такі задачі:

− провести аналіз проблеми розв’язання задачі класифікації банківських текстів;

− розглянути існуючі методи вирішення задачі класифікації банківських текстів та обрати й обґрунтувати вибір методу, який задовольняє мету даної магістерської кваліфікаційної роботи;

− розробити математичну модель класифікації банківських текстів;

− сформулювати стадії інформаційної технології, розробити структуру та алгоритм роботи програмного засобу;

– виконати програмну реалізацію запропонованої інформаційної технології;

– провести тестування програмного продукту та виконати аналіз отриманих результатів.

Об’єкт дослідження – процес класифікації банківських текстів з використанням згорткових нейронних мереж.

Предмет дослідження – інформаційна технологія та програмні засоби класифікації банківських текстів з використанням згорткових нейронних мереж та достовірність їх роботи.

Методи дослідження. У роботі використані наступні методи наукових досліджень: системного аналізу, інтелектуального аналізу даних, теорії штучних нейронних мереж для реалізації інформаційної технології класифікації банківських текстів, методи математичної статистики для розробки процесу класифікації банківських текстів та обрахунків результатів експериментів із програмним засобом, об’єктно-орієнтованого програмування.

Наукова новизна одержаних результатів.

1. Набула подальшого розвитку інформаційна технологія класифікації банківських текстів, яка відрізняється використанням згорткової штучної нейронної мережі, що дозволило підвищити достовірність класифікації банківських текстів.

2. Удосконалено метод попередньої обробки тексту для подальшого розпізнавання нейронною мережею, який відрізняється використанням процедури видалення стоп-слів перед подачею тексту на вхід нейронної мережі, що дозволило уникнути «зашумлення» ознак і тим самим підвищити достовірність класифікації банківських текстів.

Практичне значення одержаних результатів полягає в тому, що на основі проведених досліджень розроблено програмне забезпечення класифікації банківських текстів.

Запропонована інформаційна технологія сприяє підвищенню достовірності процесу класифікації банківських текстів, зокрема:

– розроблено алгоритм роботи програмного забезпечення класифікації банківських текстів на основі згорткової нейронної мережі;

– розроблено програмні засоби для класифікації банківських текстів на

основі згорткової нейронної мережі.

Достовірність теоретичних положень магістерської кваліфікаційної роботи підтверджується коректністю постановки завдання, коректністю використання математичного апарату методів дослідження, експериментальними дослідженнями тестування програмної реалізації інформаційної технології класифікації банківських текстів. Адекватність розроблених математичних моделей підтверджується результатами експериментальних досліджень.

Особистий внесок здобувача. Усі результати, наведені у магістерській кваліфікаційній роботі, отримані самостійно. У працях, написаних у співавторстві, здобувачу належать: аналіз методів попередньої обробки текстів [1], аналіз процесу класифікації банківських текстів на основі згорткової нейронної мережі та методів підвищення достовірності [2].

Апробація результатів роботи. Результати досліджень апробовані на XI науково-практичній конференції «ІОН-2018», 22–25 травня 2018, Вінниця: ВНТУ [1] та на конференції «Молодь в науці: дослідження, проблеми, перспективи-2020», Вінниця, жовтень 2019 – травень 2020 року [2].

Публікації. За результатами досліджень опубліковано двоє тез доповіді на науково-технічній конференції [1,2].

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ КЛАСИФІКАЦІЇ БАНКІВСЬКИХ ТЕКСТІВ

1.1 Аналіз сучасних методів класифікації банківських текстів

Методи класифікації текстів знаходяться на перетині двох областей – інформаційного пошуку і машинного навчання. Їх схожість полягає в способах представлення самих документів і способах оцінки якості алгоритмів. На сьогоднішній день розроблено велику кількість методів і їх різних варіацій для класифікації текстів. Кожна група методів має свої переваги і недоліки, області застосування, особливості та обмеження.

Основними методами класифікації текстів є:

– наївний баєсівський класифікатор;
– метод k-найближчих сусідів;
– дерева рішень;
– метод опорних векторів;
– методи на основі штучних нейронних мереж.

Розглянемо наївний баєсівський класифікатор, оснований на теоремі Баєса [3]:

$$P(c|d) = \frac{P(d|c)P(c)}{P(d)}, \quad (1.1)$$

де P(c|d) — ймовірність, що документ d належить класу c, саме її і потрібно розрахувати,

P(d|c) — ймовірність зустріти документ d серед всіх документів класу c,

P(c) — абсолютна ймовірність зустріти документ класу c в корпусі документів,

P(d) — абсолютна ймовірність документа d в корпусі документів.

Теорема Байєса дозволяє переставити місцями причину і наслідок. Знаючи з якою ймовірністю причина призводить до якоїсь події, ця теорема дозволяє розрахувати ймовірність того що саме ця причина призвела до нинішньої події.

Мета класифікації полягає в тому, щоб зрозуміти до якого класу належить документ, тому потрібна не сама ймовірність, а найбільш ймовірний клас. Байєсівський класифікатор використовує оцінку апостеріорного максимуму для визначення найбільш вірогідного класу [3]:

$$c_{map} = \arg\max_{c \in C} \frac{P(d|c)P(c)}{P(d)} \quad (1.2)$$

Для реалізації наївного баєсівського класифікатора необхідна навчальна вибірка, в якій проставлені відповідності між текстовими документами і їх класами. Також необхідно зібрати наступну статистику з вибірки, яка буде використовуватися на етапі класифікації:

– відносні частоти класів в корпусі документів (як часто зустрічаються документи того чи іншого класу);

– сумарна кількість слів у документах кожного класу;

– відносні частоти слів у межах кожного класу;

– розмір словника вибірки (кількість унікальних слів у вибірці).

Основним недоліком наївного баєсівського класифікатора для задачі класифікації текстів є те, що якщо на етапі класифікації зустрінеться слово якого ви не бачили на етапі навчання – це призведе до того що документ з цим словом не можна буде класифікувати, так як він буде мати нульову ймовірність по всіх класах.

Також існує проблема при класифікації текстів всередині певної предметної області. Наприклад, якщо класифікувати тексти по таким класам, як «Спорт», «Медицина», «Банківська сфера» – то проблем не буде, адже частота появи «маркувальних» слів одного класу буде значно відрізнятись від частоти появи цих же слів в інших класах. Але, наприклад, якщо класифікувати тексти з банківської сфери по таким класам як «Скарги по кредитам», «Питання по депозитам», «Питання по кредитних картках» - то значна частина понять і термінів буде часто зустрічатись у текстах, які належать до різних класів, а тому кількість «маркувальних» слів кожного

класу буде невеликою, що може призвести до неправильного вибору класу класифікатором.

Розглянемо метод k-найближчих сусідів. Метод k-найближчих сусідів відноситься до метричних методів класифікації. Щоб знайти клас, відповідний документу d, класифікатор порівнює d з усіма документами з навчальної вибірки L, тобто для кожного $d_z \in L$ обчислюється відстань $r(d_z, d)$. Далі з навчальної вибірки вибираються k документів, найближчих до d. Згідно з методом k-найближчих сусідів, документ d вважається належним того класу, який є найбільш поширеним серед сусідів даного документа, тобто для кожного класу c_i обчислюється функція ранжування [3]:

$$CSV(d) = \sum_{d_z \in L_k(d)} p(d_z, d) * \Phi(d_z, c_i), \quad (1.3)$$

де $L_k(d)$ — найближчі k документів із L до d,

$\Phi(d_z, c_i)$ — відомі величини, уже класифіковані по категоріям документи із навчальної вибірки.

Серед переваг даного методу слід виділити можливість оновлення навчальної вибірки без перенавчання класифікатора та відносно просту програмну реалізацію. Недоліками метода є великий час роботи, який пов'язаний з необхідністю повного перебору навчальної вибірки.

Розглянемо метод дерев рішень, який відноситься до логічних методів класифікації. Деревом рішень називають ациклічний граф, за яким здійснюється класифікація об'єктів (в нашому випадку текстових документів), описаних набором ознак. Кожен вузол дерева містить умову розгалуження по однієї з ознак [3]. У кожного вузла стільки розгалужень, скільки значень має обрана ознака. В процесі оцінки регулюється послідовні переходи від одного вузла до іншого відповідно до значень ознак об'єкта. Класифікація вважається завершеною, коли досягнуто один з кінцевих вузлів дерева. Значення цього листа визначає клас, до якого належить даний об'єкт. На практиці зазвичай використовують бінарні дерева рішень, в яких прийняття

рішення переходу по ребрах здійснюється простою перевіркою наявності ознаки в документі. Якщо значення ознаки менше певного значення, вибирається одна гілка, якщо більше або дорівнює, інша. Серед переваг методу слід виділити відносно просту програмну реалізацію. Недоліками є те, що для отримання точних результатів необхідний великий обсяг навчальної вибірки.

Розглянемо метод опорних векторів, який належить до лінійних методів класифікації. Розглянемо множину документів, які необхідно класифікувати. Співставимо їм множину точок в просторі розмірності $|D|$.

Вибірку точок називають лінійно роздільною, якщо точки, які належать до різних класів, можна розділити за допомогою гіперплощини (в двомірному випадку гіперплощиною є пряма лінія). Очевидний спосіб вирішення завдання в такому випадку – провести пряму так, щоб по одну сторону від неї лежали всі точки одного класу, а по іншу - всі точки іншого класу. Тоді для класифікації невідомих точок досить буде подивитися, з якого боку прямої вони знаходяться [3].

Вибірку називають лінійно нероздільною, якщо точки, що належать різним класам, не можна розділити за допомогою гіперплощини. Коли такої роздільною гіперплощини не існує, необхідно перейти від початкового простору ознак документів до нового, в якому навчальна вибірка виявиться лінійно роздільною [3].

На практиці структура даних часто буває невідома і дуже рідко вдається побудувати роздільну гіперплощину, а значить, неможливо гарантувати лінійну роздільність вибірки. Можуть існувати такі документи, які алгоритм віднесе до одного класу, а в дійсності вони повинні належати до протилежного. Такі дані називаються викидами, вони створюють похибку метода, тому було б краще їх ігнорувати. У цьому полягає суть проблеми лінійної нероздільності.

Метод опорних векторів є досить якісним методом, але лише для випадку бінарної класифікації, тобто за допомогою нього не можна класифікувати тексти більш ніж на два класи.

Ще одним методом, що широко використовується для класифікації текстів є штучні нейронні мережі. Штучні нейронні мережі (ШНМ) – математичні моделі спроектовані за прикладом дії біологічних нейронних мереж [4]. Основою ШНМ є

модель людського мозку, який складається із мільярдів нейронів що з’єднанні синапсами. Аналогічно, штучні нейронні мережі складаються з обчислювальних елементів які називають штучними нейронами. Зв’язки між нейронами визначають характеристики як мозку так і штучних нейронних мереж. Важливою перевагою нейронних мереж над звичайними алгоритмами є їх здатність до навчання. Навчання складається із подачі на вхід мережі багатьох тренувальних прикладів, кожен з яких складається з набору входів та бажаних виходів. Найбільш поширеним методом навчання нейронних мереж є алгоритм зворотнього поширення помилки. Оскільки для навчання нейронних мереж потрібні лише тренувальні дані, характер взаємозв’язків між входами та виходами не повинен бути відомий. Це є значною перевагою особливо в випадках коли такий взаємозв’язок є складним. Також, нейронні мережі здатні до узагальнення – у разі успішного навчання, нейронна мережа поверне правильний результат даних, що не входять до навчальної вибірки.

Класичні нейронні мережі прямого поширення складаються з вхідного шару, вихідного шару та проміжних шарів. Сигнал розповсюджується послідовно від вхідного шару нейронів по проміжним шарам до вихідного. Прикладом такої структури є багатошаровий персептрон. Недоліком класичних мереж прямого поширення є те, що кожен нейрон одного шару з’єднаний з кожним нейроном наступного шару. Для задачі класифікації текстів це є неприйнятим, так як призведе до великої кількості зв’язків мережі, так як кожне слово тексту перетворюється у вектор довжиною у десятки (частіше за все, сотні) чисел. Наприклад, для класифікації текстів об’ємом не більше 400 слів на 12 класів та довжиною вектора слова 100 кількість зв’язків із 1 прихованим шаром зі 100 нейронів становитиме 4001200 зв’язків (400 * 10 + 100 * 12).

Для задачі класифікації текстів найчастіше використовуються згорткові нейронні мережі, які дозволяють знизити складність мережі за рахунок операцій згортки та агрегування [5], що знизить складність мережі, але не знизить якість класифікації.

Приклад архітектури згорткової нейронної мережі для задачі класифікації текстів на 2 класи зображено на рисунку 1.1.

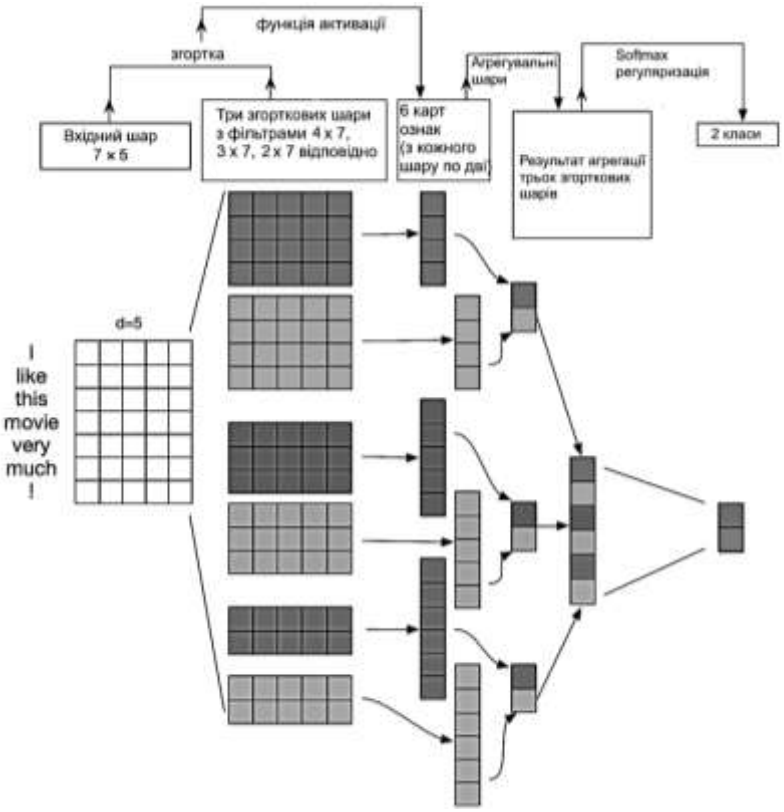

Рисунок 1.1 – Приклад архітектури згорткової мережі для задачі класифікації текстів

Таким чином, найкращим з методів класифікації текстів є метод на основі згорткових нейронних мереж, тому саме його оберемо для використання у даній роботі.

1.2 Аналіз методів векторизації слів

Векторизація слова – це перетворення слова у вектор дійсних чисел фіксованої довжини. Задача перетворення слова у вектор є досить актуальною проблемою при обробці тексту.

Основними методами, які використовуються для перетворення слів у вектор, є:

- one-hot encoding;
- word2vec;
- glove.

Метод one-hot encoding є найбільш простим методом векторизації слів. В даному методі кожне слово кодується за допомогою вектора фіксованої довжини, що дорівнює кількості використовуваних слів у вибірці. Кожен вектор складається з нулів і однієї одиниці. Недоліком цього методу є величезна довжина вектору слова, так як в основному вибірка слів складається з десятків, а то і сотень тисяч слів.

Word2vec - це метод розрахунку векторних представлень слів, який реалізує дві основні архітектури: [6]

– Continuous Bag of Words (CBOW);

– Skip-gram.

На вхід подається корпус тексту, а на виході виходить набір векторів слів. Знаходження зв'язків між контекстами слів згідно з припущенням, що слова, що знаходяться в схожих контекстах, мають тенденцію означати схожі речі, тобто бути семантично близькими. Більш формально завдання стоїть так: максимізація косинусної близькості між векторами слів (скалярний добуток векторів), які з'являються поруч один з одним, і мінімізація косинусної близькості між векторами слів, що не з'являються поруч один з одним. Поруч один з одним в даному випадку означає в близьких контекстах.

Косинусна міра близькості (косинусна схожість) - це міра подібності між двома векторами. Косинусна схожість між векторами A і B обчислюється за формулою: [6]

$$\text{similarity} = \frac{A * B}{\|A\| \, \|B\|} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} (A_i)^2} \times \sqrt{\sum_{i=1}^{n} (B_i)^2}} \quad (1.4)$$

CBOW і Skipgram - це нейромережеві архітектури, які описують, як саме нейромережа «навчається» на даних і «запам'ятовує» представлення слів. Принципи у обох архітектур різні. Принцип роботи CBOW - передбачення слова по заданому контексті, а skip-gram навпаки - передбачення контексту по заданому слову. Skipgram модель працює повільніше, але зазвичай за допомогою неї досягається краща якість класифікації текстів.

Недоліком методу word2vec є те, що для навчання моделі word2vec хорошої якості потрібен дуже великий корпус текстів, але в мережі Інтернет доступні вже “навчені” на великих об’ємах даних моделі.

Ще одним методом, який використовується для векторизації слів є метод Glove. Нехай об’єм словника даних рівний V. Усі слова, які зустрічаються в даних нумеруються від 1 до V. Складається матриця «слово-слово» $X \in R^{V \times V}$. Приклад матриці слово-слово зображено на рисунку 1.2.

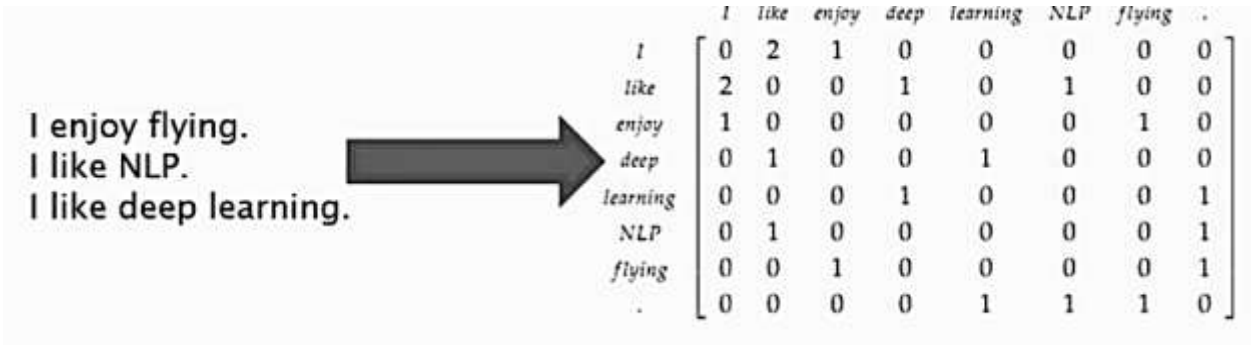

Рисунок 1.2 – Приклад матриці «слово-слово», яка утворюється в ході роботи метода Glove

Недоліком цього методу є те, що матриця «слово-слово» буде дуже розрідженою при об’ємному словнику. А додавання нових слів у матрицю призводить до необхідності перерахунку усієї матриці.

В ході аналізу методів перетворення слова у вектор фіксованої довжини було розглянуто такі методи, як one-hot encoding, word2vec, glove. Для вирішення задачі класифікації текстів для перетворення слова в вектор був обраний метод word2vec, так як він забезпечує схожість векторів семантично близьких слів та має відносно невелику розмірність, що зменшує складність обчислень.

1.3 Аналіз існуючих програм-аналогів

На сьогоднішній день існує не так багато програм, які виконують автоматичну класифікацію текстових документів.

Розглянемо програму «Sentence Classification CNN» [7] – див. рис. 1.3. Програма працює на основі згорткової нейронної мережі. Серед переваг даної програми слід виділити простоту реалізації, що зменшує ймовірність виникнення помилок. Недоліком програми є невисока достовірність класифіаціїї, а також обмежений функціонал – по-перше, вона класифікує не тексти, а речення, а, по-друге, дана програма призначена для локального використання без клієнт-серверної архітектури, що ускладнює можливість її практичного застосування в якості сервісу, який надає послуги класифікації третім сторонам.

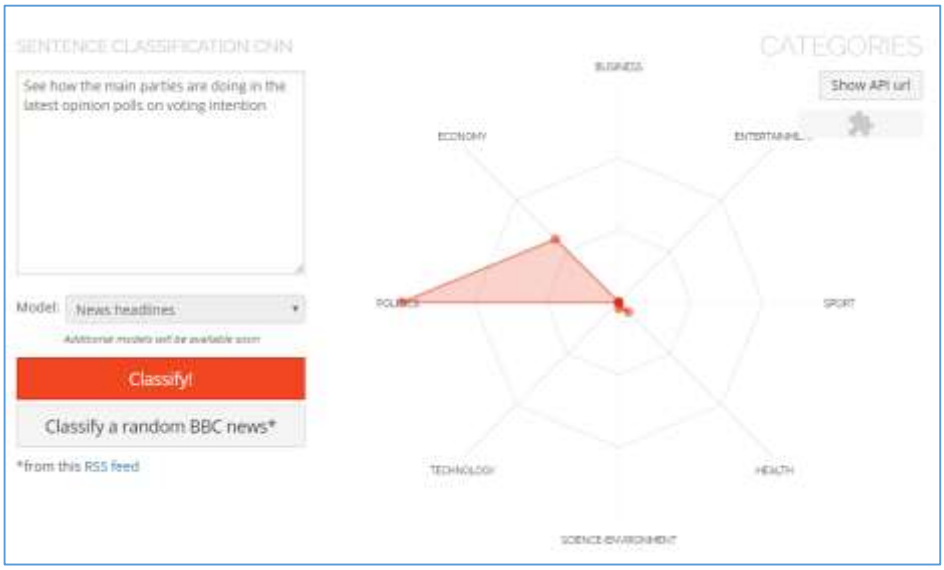

Рисунок 1.3 – Вид вікна програми «Sentence Classification CNN»

Також було проаналізовано роботу ще однієї програми класифікації текстів – «Multiclass CNN Classifier» [8] – див. рис. 1.4. Дана програма працює на основі нейронної мережі. Перевагою даної програми є те, що вона має гарну швидкодію, але не забезпечує належної достовірності класифікації. Однією з причин цього є те, що ця програма не використовує готових векторизаторів слів, а формує власні вектори слів на основі навчальної вибірки. Тому може виникнути проблема «незнайомих» слів після навчання моделі. Ще одним недоліком є те, що дана програма не видаляє стоп-слова перед подачею тексту на вхід нейронної мережі, що призводить до «зашумлення» ознак.

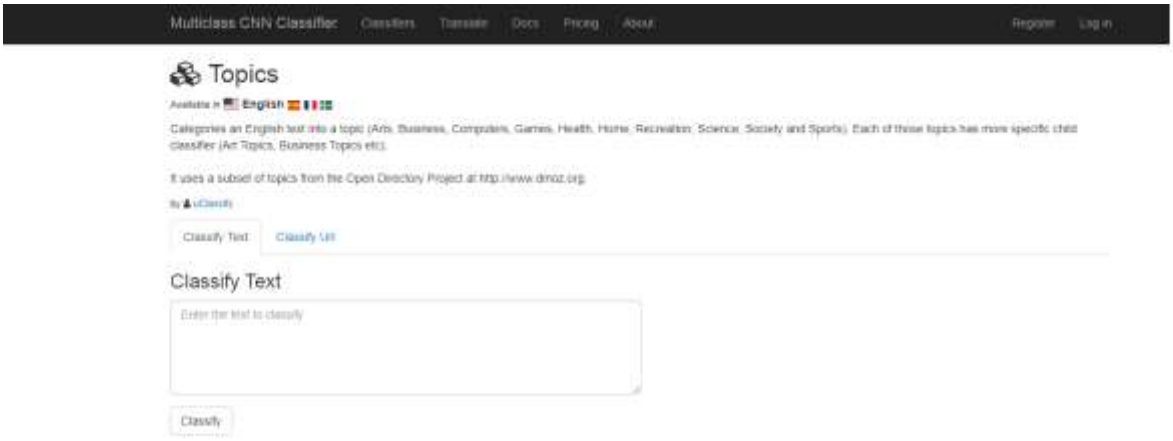

Рисунок 1.4 – Вид вікна програми «Multiclass CNN Classifier»

У результаті аналізу програм-аналогів було визначено їх недоліки та зясовано, що для подолання цих недоліків потрібно використовувати методи класифікації на основі згорткових нейронних мереж, вдосконалювати процес попередньої обробки тексту, і таким чином, підвищити достовірність класифікації текстів. Також потрібно розширити функціональні можливості розроблюваної програми шляхом застосування клієнт-серверної архітектури.

1.4 Постановка задачі

У даній роботі необхідно реалізувати програму класифікації банківських текстів. Класифікація вирішує наступну задачу: задано кінцеву множину текстів, для кінцевої підмножини яких відомо до якого класу вони відносяться. Ця підмножина називається навчальною вибіркою. Класова приналежність інших об'єктів не відома. Потрібно побудувати інформаційну технологію та її програмну реалізацію, здатну класифікувати довільний текст з початкової множини.

Класифікувати текст – значить, вказати номер (або найменування класу), до якого відноситься даний текст. В задачі класифікації текстів об’єктами є текстові документи. Формальна постановка задачі класифікації текстів виглядає так:

$D = \{ d_1, \dots, d_n \}$ – множина текстових документів. Кожний документ $d \in D$ являє собою послідовність слів $W_d = \{w_1, \dots, w_{n_d}\}, n_d$ – довжина документа d.

$Y = \{y_1, \dots, y_n\}$ – кінцева множина міток класів.

$y^*: D \to Y$ – невідома цільова залежність, значення якої відомі тільки на об'єктах кінцевої навчальної вибірки $D^m = \{(d_1, y_1), \dots, (d_m, y_m)\}$.

Потрібно розробити програму, в якій буде реалізовано алгоритм $a: D \to Y$, який здатний класифікувати довільний об'єкт $d \in D$.

Етапи процесу класифікації текстів зображено на рисунку 1.5.

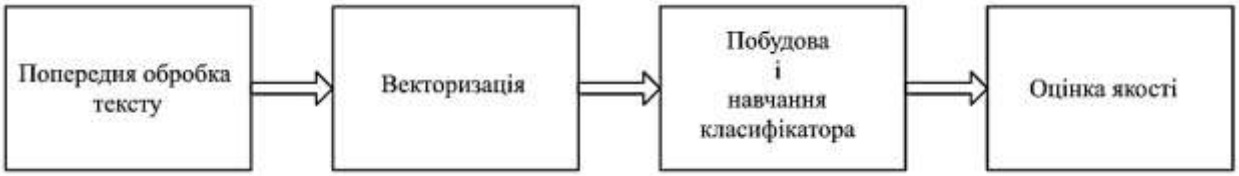

Рисунок 1.5 – Етапи процесу класифікації текстів

Проаналізувавши вихідні дані, сучасні методи класифікації текстів та існуючі програми-аналоги можна сформулювати функціональні вимоги до програми класифікації банківських текстів.

Програма класифікації банківських текстів повинна виконувати класифікацію тексту на англійській мові з банківської тематики, обсягом не більше 400 слів. Програма повинна працювати на основі згорткової нейронної мережі, проводити попередню обробку тексту (видалення стоп-слів та стоп-символів) та word2vec векторизацію, що підвищать достовірність класифікації.

Програма класифікації банківських текстів повинна працювати як веб-сервіс (Web API) через стандартні HTTP-запити та використовувати аутентифікацію по токену. Веб-сервіс повинен підтримувати такий формат обміну даними як JSON. Веб-сервіс повинен працювати під управлінням операційної системи Windows XP/7/8/10/Server, Linux, які є найбільш поширеними серед операційних систем. Ємність ОЗУ залежить від навантаження на веб-сервер, але повинно бути не меншою за 512 Мбайт.

Надійність функціонування програми і її функціональну стійкість визначають вхідні дані, які передаються в тілі HTTP-запиту, тому необхідно передбачити перевірку вхідних даних на правильність.

1.5 Висновок

Для задачі класифікації текстів було обрано штучні нейронні мережі, адже їх використання дає змогу підлаштовувати ознаки, за якими класифікуються тексти, в ході навчання нейронної мережі, а також класифікувати тексти всередині конкретної предметної області, де область перетину множини слів кожного класу є досить великою. Також використання нейронних мереж сприяє мінімізації можливості появи незнайомих слів в тексті за рахунок можливості використання готових (навчених на дуже великій навчальній вибірці) векторизаторів слів. Для задачі класифікації текстів було обрано згорткову нейронну мережу, адже за допомогою шарів згортки можна зменшити вхідні дані в рази, і, таким чином, зменшити складність обчислень. Для векторизації слів було обрано алгоритм word2vec, перевагою якого є те, що він перетворює слова у вектори відносно невеликої розмірності, а також існує достатня кількість моделей, навчених на великих об’ємах текстів, завдяки чому можливість появи незнайомого слова в тексті зменшується. Також було проаналізовано існуючі програми-аналоги та зроблено постановку задачі. На основі вихідних даних, аналізу сучасних методів класифікації текстів та існуючих програм-аналогів було розроблено функціональні вимоги до програми класифікації банківських текстів.

2 РОЗРОБКА ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ КЛАСИФІКАЦІЇ БАНКІВСЬКИХ ТЕКСТІВ НА ОСНОВІ НЕЙРОННОЇ МЕРЕЖІ

2.1 Метод попередньої обробки тексту

Оскільки задача класифікації текстів буде вирішена за допомогою нейронних мереж і кожне слово перед подачею на вхід нейронної мережі буде перетворено у вектор фіксованої довжини, то для підвищення ефективності і достовірності класифікації пропонується провести попередню обробку тексту, яка включає такі етапи, як:

- видалення стоп-символів;
- видалення стоп-слів.

Видалення стоп-слів та стоп-символів в більшості випадків значно впливає на достовірність класифікації [9].

Видалення стоп-символів – це видалення будь-яких символів, що є не літерами і не цифрами. Але набори стоп-символів можуть бути різними і залежати від цілей і завдань дослідження. Для задачі класифікації текстів було визначено наступну множину стоп-символів:

$$SC = \{\ «"», «!», «\#», «\$», «\%», «\&», «'», «(», «)», «*», «+», «“», «”», «-», «,», «.», «/», «:», «;», «<», «=», «>», «?», «@», «\», «^», «_», «`», «{», «|», «}», «~», «\»\ \} \quad (2.1)$$

Видалення стоп-слів – це видалення слів, які не несуть ніякого змістового навантаження. До них відносяться службові частини мови, артиклі, допоміжні слова і т. д. Фрагмент множини стоп-слів:

SW= { «do», «does», «did», «a», «an», «the», «and», «but», «if», «or», «that», «this», «those», «these», «could», «will», «would», «shall» } (2.2)

2.2 Інформаційна модель процесу класифікації банківських текстів

Одним із етапів процесу проектування інформаційної технології класифікації текстів є розробка інформаційної моделі.

Інформаційна модель процесу класифікації текстів – це модель, що описує істотні для даного процесу параметри та змінні величини, зв'язки між ними, та його вхідні та вихідні значення. Виходячи з визначення інформаційної моделі, її можна подати у вигляді кортежа:

$$IMTC = < TXT, ss, sw, va, ca, R >, \text{ де} \quad (2.3)$$

TXT– текст, який потрібно класифікувати,

ss – множина стоп-символів,

sw – множина стоп-слів,

va – алгоритм векторизації,

ca – алгоритм класифікації,

R – результат класифікації.

Схему інформаційної моделі процесу класифікації текстів зображено на рисунку 2.1.

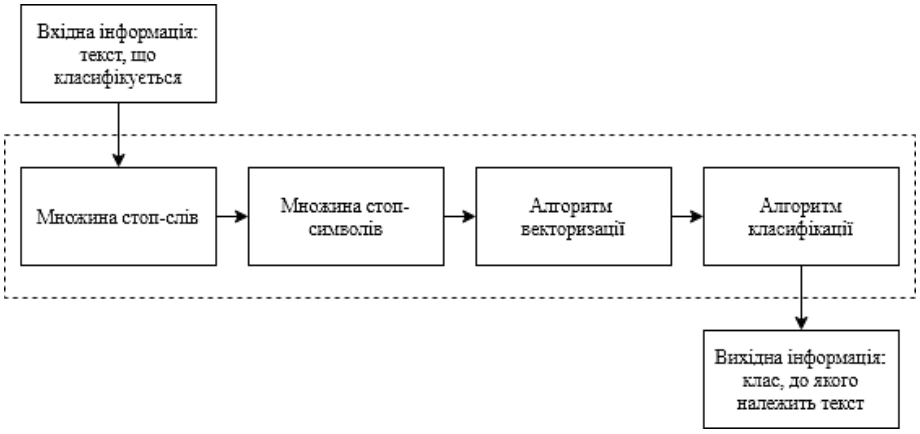

Рисунок 2.1 – Схема інформаційної моделі процесу класифікації текстів

2.3 Розробка архітектури згорткової нейронної мережі

Згорткова нейронна мережа (ЗНМ) складається з шарів входу та виходу, згорткових шарів та шарів агрегації. Згортковий і агрегувальний шари вважаються шарами двовимірної розмірності, а вихідний шар, як правило, представлений вектором із простору $\mathbb{R}^1$. Узагальнена структурна схема згорткової нейронної мережі для класифікації текстів представлена на рис. 2.2.

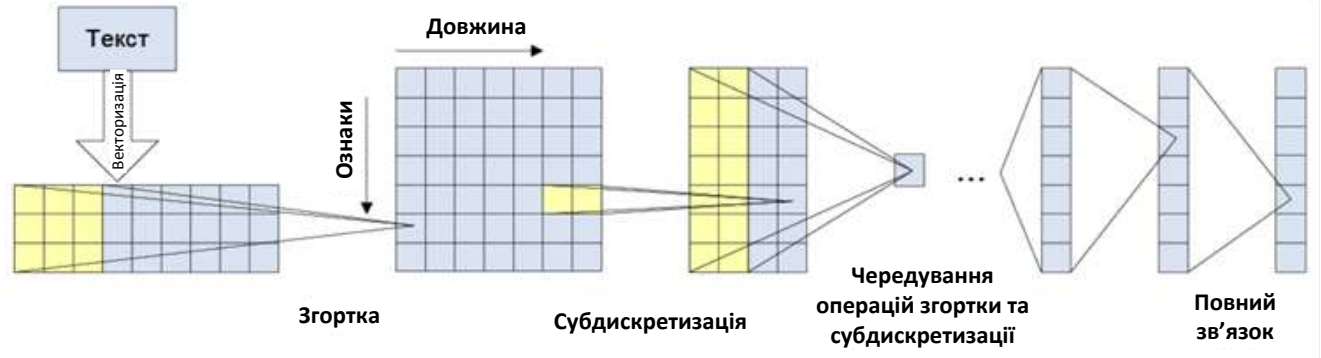

Рисунок 2.2 – Узагальнена структурна схема згорткової нейронної мережі для класифікації текстів

Для задачі класифікації текстів розмірність вхідного шару залежить від максимально можливої кількості слів в тексті та обраної довжини вектора слова. На вхід нейронної мережі подаватимуться слова, які були отримані шляхом векторизації за допомогою алгоритму word2vec [6]. Кожне слово – це вектор з N чисел, де N = 100.

Відповідно до вихідних даних максимально можлива кількість слів у тексті – 400, тому розмірність вхідного шару 400 × 100. Якщо кількість слів є меншою за максимально можливу, то відсутні слова будуть заповнюватись «пустими» векторами.

Згортковий шар є основним будівельним блоком ЗНМ. Параметри шару складаються з набору фільтрів для навчання (так званих ядер згортки), які мають невеличке рецептивне поле. В згортковій нейронній мережі кожен згортковий шар має декілька рівнів (кількість рівнів дорівнює кількості фільтрів). Кожен рівень представлений двовимірним масивом. Вихід кожного рівня в подальшому називатиметься картою ознак.

Для задачі класифікації текстів буде використано три згорткових шари з фільтрами розмірністю 3 × N, 4 × N та 5 × N відповідно, де N = 100 і є довжиною вектора слова. Кількість фільтрів – 32 для кожного шару. Відповідно до цього кожен згортковий шар утворить 32 карти ознак. Оскільки ширина «вікна» фільтра дорівнює N, то кількість стовпців карти ознак дорівнює одиниці. Кількість рядків карти ознак обчислюється наступним чином:

$$H = M - FH + 1, \text{де} \tag{2.4}$$

M – максимально можлива кількість слів у тексті,

FH – висота «вікна» фільтра.

Отже, згорткові шари з фільтрами розмірністю 3 × N, 4 × N та 5 × N утворять по 32 карти ознак розмірністю 396 × 1, 395 × 1, 394 × 1 відповідно.

В якості функції активації для згорткових шарів буде використано функцію ReLU – ненасичувальну передавальну функцію, яка посилює нелінійні властивості функції ухвалення рішення і мережі в цілому, не зачіпаючи рецептивних полів згорткового шару [5]:

$$f(x) = \max(0, x) \tag{2.5}$$

Іншим важливим поняттям ЗНМ є агрегування, яке є різновидом нелінійного зниження дискретизації. Існує декілька нелінійних функцій для реалізації агрегування, серед яких найпоширенішою є максимізаційне агрегування:

$$\hat{c} = \max\{c\} \tag{2.6}$$

Воно розділяє вхідну матрицю на набір прямокутників без перекриттів, і для кожної такої підобласті виводить її максимум. Ідея полягає в тому, що точне положення ознаки не так важливе, як її грубе положення відносно інших ознак. Агрегувальний шар слугує поступовому скороченню просторового розміру представлення для зменшення кількості параметрів та об'єму обчислень у мережі. В архітектурі ЗНМ є звичним періодично вставляти агрегувальний шар між послідовними згортковими шарами [5].

Для задачі класифікації текстів кожен агрегувальний шар перетворюватиме вихід згорткового шару у матрицю розмірністю 32 х 1, що значно знижуватиме розмірність, а відповідно і складність обчислень.

Після кількох згорткових та максимізаційно агрегувальних шарів, високорівневі міркування в нейронній мережі здійснюються повноз'єднаними шарами. Нейрони у повноз'єднаному шарі з'єднуються з усіма нейронами попереднього шару.

Після проведення операцій згортки та агрегування буде виділено по 32 ознаки з кожної зв’язки згорткового та агрегувального шарів, які будуть подані на вихідний повнозвязний шар.

Відповідно до вихідних даних кількість класів до яких може відноситись текст – 11, тому кількість нейронів вихідного повнозвязного шару – 11. Для того щоб нормалізувати значення ймовірності належності тексту до того чи іншого класу буде використана Softmax функція. Функція задається наступним чином [10]:

$$\sigma(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \quad (2.7)$$

Softmax функція «стискає» K-вимірний вектор z із довільним значенням компонент до K-вимірного вектора $\sigma(z)$ з дійсними значеннями компонент в області [0, 1], що в сумі дають одиницю.

2.4 Математична модель згорткової нейронної мережі

Розглянемо математичну модель згорткової мережі для класифікації текстів. Кількість шарів згорткової мережі обчислюється наступним чином:

$$L = 2a + 2, \text{ де} \quad (2.8)$$

$a \in \mathbb{Z}$ – кількість згорткових шарів.

Вихід кожного згорткового шару зв'язаний з агрегувальним шаром, що пояснює множення кількості згорткових шарів на 2 у формулі 2.5. Також до загальної кількості додається вхідний та вихідний шари.

Розглянемо згортковий шар l. Карта ознак n згорткового шару l обчислюється наступним чином:

$$y_n^l = f_l(\mathbb{x} \oplus \mathbb{w}_n^l + b_n^l), \text{ де} \quad (2.9)$$

f_l – функція активації шару l

$\mathbb{x}$ – вхідний шар,

$\mathbb{w}_\mathrm{n}^\mathrm{l} = \{w_n^l(i,j)\}$ – n-ий фільтр шару l,

$\mathrm{b}_\mathrm{n}^\mathrm{l}$– порогові значення, які додаються до карти ознак n шару l,

$\oplus$ – операція двовимірної згортки.

Припустимо, що розмірність вхідного шару дорівнює $\mathrm{H^x} \times \mathrm{W^x}$, а розмірність фільтра (ядра згортки) дорівнює $\mathrm{r^l} \times \mathrm{c^l}$, тоді розмірність вихідної карти ознак $\mathrm{y_n^l}$ обчислюється наступним чином:

$$(H^x - r^l + 1) \times (W^x - c^l + 1) \quad (2.10)$$

Розглянемо агрегувальний шар a. Карти ознак n, обчислені в згорткових шарах l, подаються на шари агрегації, де розбиваються на блоки розміром $h \times v$. Після чого до кожного блоку застосовується функція агрегування і в результаті отримаємо матрицю $\mathbb{z}_n^a = \{z_n^a(i,j)\}$ розмірністю $(H^x - r^l + 1) - \mathrm{h} + 1 \times (W^x - c^l + 1) - v + 1$, елементами якої будуть відповідні значення результатів агрегації блоків. Карта ознак n агрегувального шару a обчислюється за формулою 2.13.

$$y_n^a = f_a(\mathbb{z}_n^a + b_n^{\mathrm{a}}) \text{, де} \quad (2.11)$$

f_a – функція активації шару a,

$\mathrm{b_n^a}$– порогові значення, які додаються до карти ознак n шару a.

Після проходження карт ознак через шари агрегації вони об’єднуються та подаються на вихідний повнозвязний шар. Розглянемо вихідний повнозвязний шар L. Значення вихідного нейрону обчислюється наступним чином:

$$y_i^L = f_L(\sum_{m=1}^{n} y_m^a + b_n^{\mathrm{L}}) \text{, де} \quad (2.12)$$

f_L – функція активації шару L,

$\mathrm{b_n^L}$– порогові значення, які додаються до карти ознак m шару L.

Таким чином, виходом згорткової нейронної мережі є вектор наступного вигляду:

$$\mathbb{y} = [y_1^L, y_2^L, \dots, y_{N^L}^L]. \quad (2.13)$$

2.5 Метод навчання згорткової нейронної мережі

Метою навчання мережі алгоритмом зворотного поширення помилки є корекція ваг зв'язків для максимізації точності результату. При навчанні передбачається, що для кожного вхідного вектора існує парний йому цільової вектор, що задає необхідний вихід. Разом вони називаються навчальною парою. Мережа навчається на багатьох парах.

Позначимо за T і OUT очікуваний та реальний виходи нейронної мережі відповідно. А за p та q нейрон з якого виходить зв'язок та у який входить зв'язок відповідно.

Введемо величину σ, яка дорівнює різниці між необхідним T_q і реальним OUT_q виходами, помноженої на похідну логістичної функції активації:

$$\sigma_q = f' * (T_q - OUT_q), \text{ де} \tag{2.14}$$

f'– похідна функції активації,

T_q– очікуване значення виходу мережі,

OUT_q– реальне значення виходу мережі.

Тоді, ваги шару після корекції після корекції буду дорівнювати:

$$\omega_{p-q}(i+1) = \omega_{p-q}(i) + \alpha * \sigma_q * OUT_p, \text{ де} \tag{2.15}$$

i – номер поточної ітерації,

ω_{p-q} – величина зв'язк нейрона p з нейроном q,

α – швидкість навчання,

OUT_p – вихід нейрона p.

На першому кроці навчання мережі вагові коефіцієнти ініціалізуються довільними значеннями. Після цього, сигнал поширюється від попередніх шарів до наступних.

Другим кроком навчання мережі є поширення помилки, отриманої на першому кроці, в зворотному направленні, тобто від наступних шарів до попередніх. При цьому відбувається корегування ваг і порогових значень кожного нейрона за формулами 2.16 та 2.17.

2.6 Структура інформаційної технології класифікації банківських текстів на основі згорткової нейронної мережі

Структура інформаційної технології класифікації банківських текстів на основі згорткової нейронної мережі, детально описана у попередніх підпунктах, зображена на рис. 2.3.

Для роботи інформаційної технології класифікації банківських текстів на основі згорткової нейронної мережі вхідною є інформація у вигляді тексту (ASCII коди символів). Потім здійснюється процес видалення стоп-символів та стоп-слів з метою зменшення «зашумлення» ознак, які потім подаються на вхід нейронної мережі. Далі відбуваться процес векторизації слів тексту за методом word2vec. Класифікація банківських текстів буде виконуватись згортковою нейронною мережею на основі попереднього її навчання на основі відповідної навчальної вибірки. Тому відбуваться процес створення цієї навчальної вибірки текстів. Далі відбувається процес формування (ініціалізації) та навчання згорткової нейронної мережі, а потім процес подачі векторизованого тексту на вхід згорткової нейромережі. Далі відбувається процес класифікації тексту згортковою нейронною мережею, а потім - процес виведення результату класифікації тексту.

Рисунок 2.3 - Структура інформаційної технології класифікації банківських текстів на основі згорткової нейронної мережі

Таким чином, розроблена структура інформаційної технології класифікації банківських текстів на основі згорткової нейронної мережі може бути використана для подальшої розробки програмних засобів.

2.7 Декомпозиція програмної реалізації класифікації текстів

Програмна реалізація класифікації текстів виконуватиме ряд задач:

- попередня обробка тексту (видалення стоп-слів та стоп-символів);
- перетворення тексту у масив векторів фіксованої довжини;

- власне класифікація тексту по класам;
- аутентифікація запитів по токену.

Тому буде доцільно провести декомпозицію програмної реалізації.

Декомпозиція – це процес поділу системи на елементи, зручні для якихось операцій з нею, а саме поділ до елементів, які приймаються за неподільні об’єкти. Це означає що всю сукупність інформації, яка характеризує систему, і всю сукупність зв`язків між елементами системи, неможливо сприйняти в цілому і повністю, звідси, додержуючись методу декомпозиції, для швидкого впровадження інформаційної системи необхідно дотримуватися принципу “добре структурованої системи”, і тому головна мета декомпозиції – поділ системи на простіші частини, таким чином зменшуючи її складність [11].

Програмну реалізацію класифікації текстів можна розділити на складові по ряду задач, які були описані вище та, відповідно, сформувати 4 модулі: модуль попередньої обробки тексту, модуль векторизації тексту, модуль класифікації, модуль аутентифікації.

2.8 UML-моделювання

UML (англ. Unified Modeling Language) — уніфікована мова моделювання. Є невід'ємною частиною уніфікованого процесу розробки програмного забезпечення. UML є мовою широкого профілю, це відкритий стандарт, що використовує графічні позначення для створення абстрактної моделі системи, яка називається UML-моделлю. UML не є мовою програмування, але в засобах виконання UML-моделей як інтерпретованого коду можлива кодогенерація [12].

Основними причинами використання UML є:

− можливість представити систему у такому вигляді, щоб її можна було легко перевести в програмний код;

− підвищення супроводжуваності проекту і полегшення розробки документації.

В процесі проектування програми класифікації текстів будуть розроблені наступні UML-діаграми:

− UML-діаграма активності модуля попередньої обробки тексту;

− UML-діаграма активності модуля перетворення тексту у масив векторів фіксованої довжини;

− UML-діаграма модуля аутентифікації;

− загальна UML-діаграма активності програми класифікації текстів;

− загальна UML-діаграма послідовностей програми класифікації текстів.

UML-діаграма активності потрібна для того, щоб схематично зобразити алгоритм вирішення деякої задачі, яка виконується системою.

UML-діаграма послідовностей потрібна для того, щоб зобразити взаємодії об'єктів (елементів системи) в динаміці, тобто впорядкованих за часом виконання певних дій.

UML-діаграму активності модуля попередньої обробки тексту зображено на рисунку 2.4.

Під токенізацією тексту мається на увазі розбиття тексту на елементи (слова і символи). Після токенізації тексту кожен елемент перевіряється на рахунок входження до заборонених множин, наведених в формулах 2.1 і 2.2 відповідно. Якщо елемент входить до забороненої множини – то видаляємо його і переходимо до наступного слова чи символу, якщо таке є.

UML-діаграму активності модуля перетворення тексту у масив векторів фіксованої довжини зображено на рисунку 2.5.

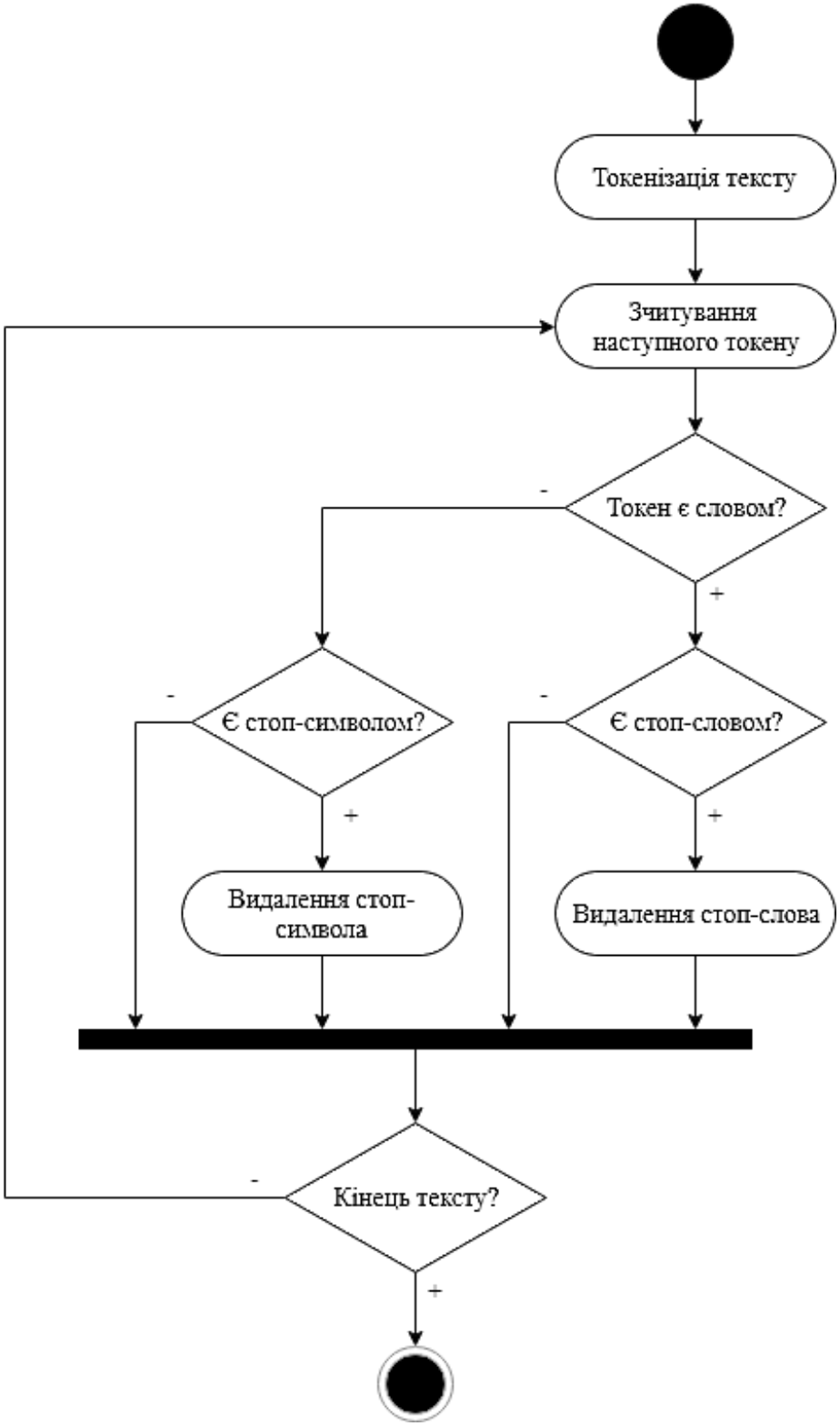

Рисунок 2.4 – UML-діаграма активності модуля попередньої обробки тексту

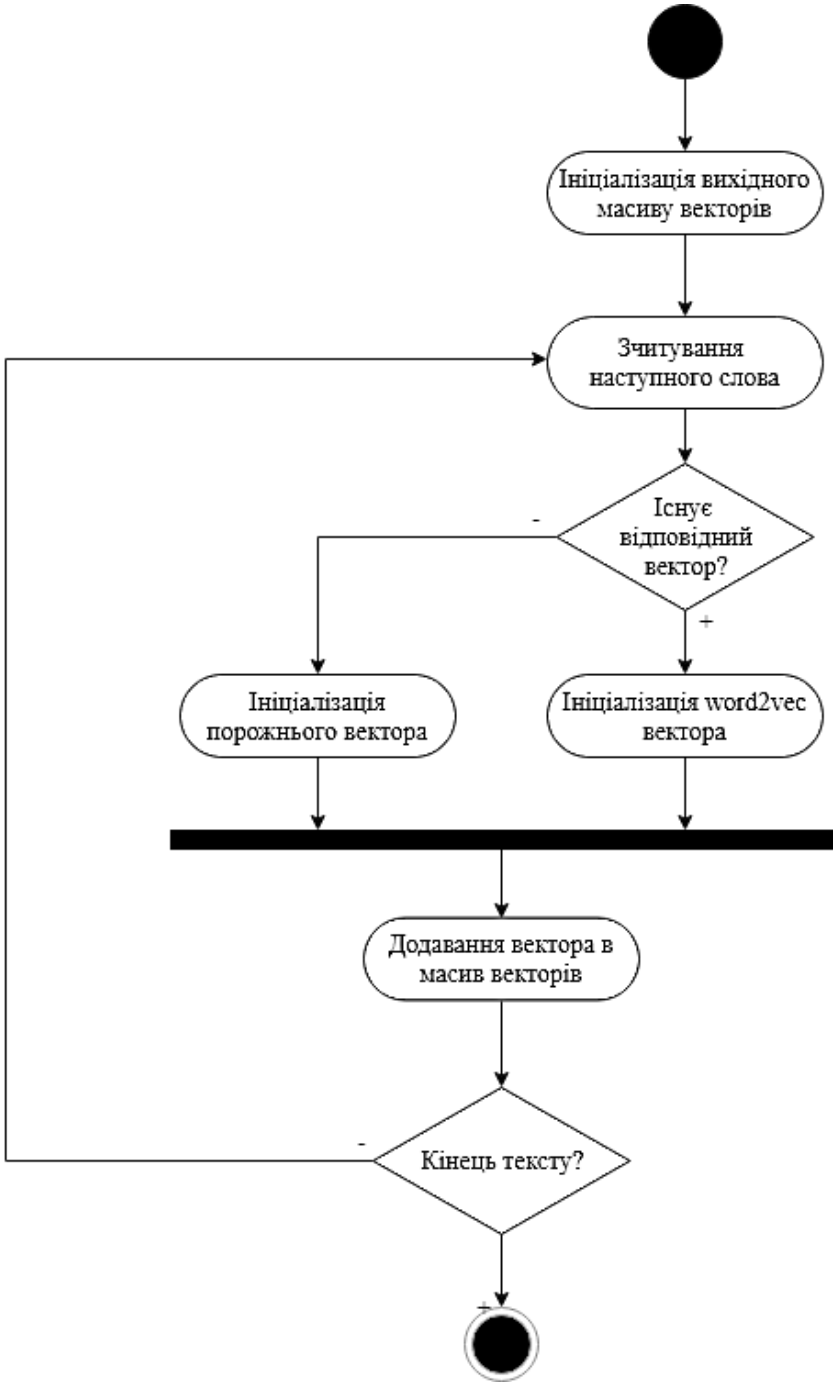

Рисунок 2.5 – UML-діаграма активності модуля перетворення тексту у масив векторів фіксованої довжини

UML-діаграму активності модуля аутентифікації зображено на рисунку 2.6.

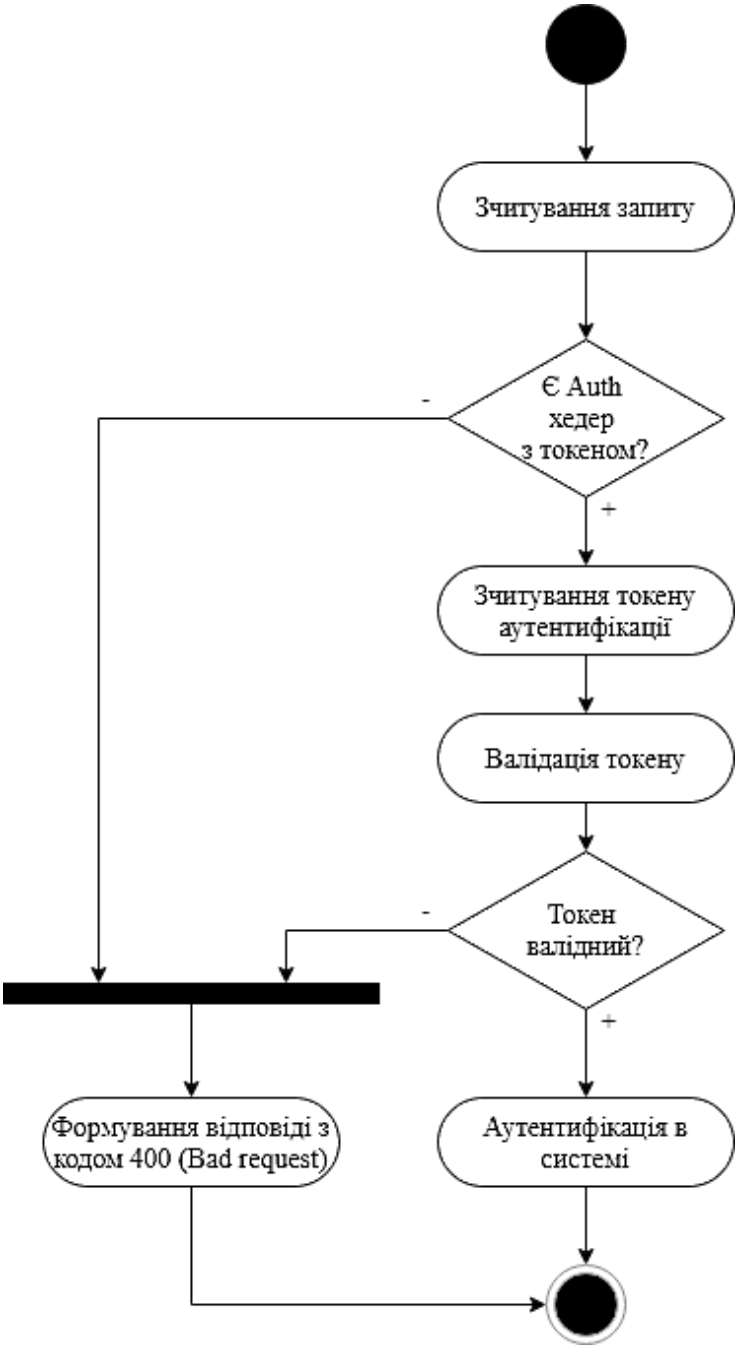

Рисунок 2.6 – UML-діаграма активності модуля аутентифікації

Загальну UML-діаграму активності програми класифікації текстів зображено на рисунку 2.7.

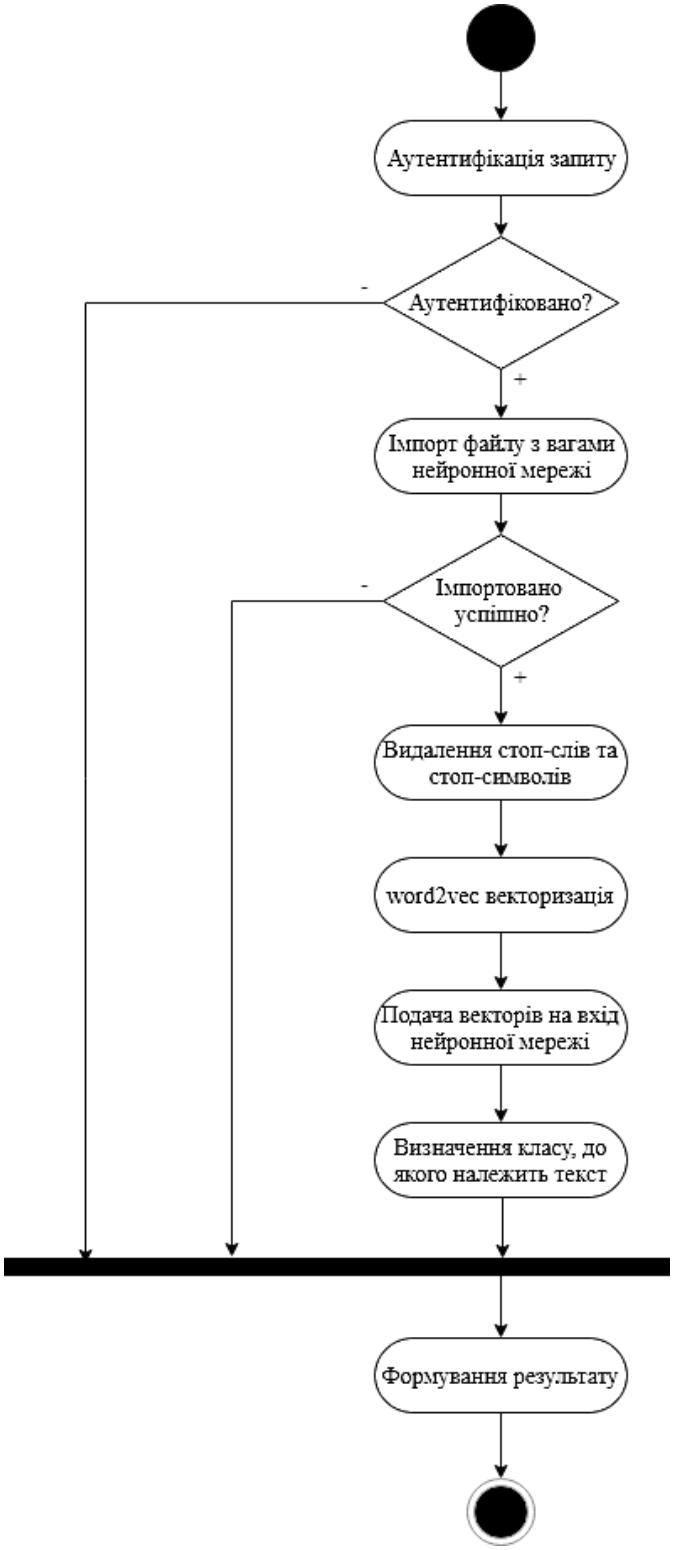

Рисунок 2.7 – Загальна UML-діаграма активності програми класифікації текстів

Загальну UML-діаграму послідовностей програми класифікації текстів зображено на рисунку 2.8.

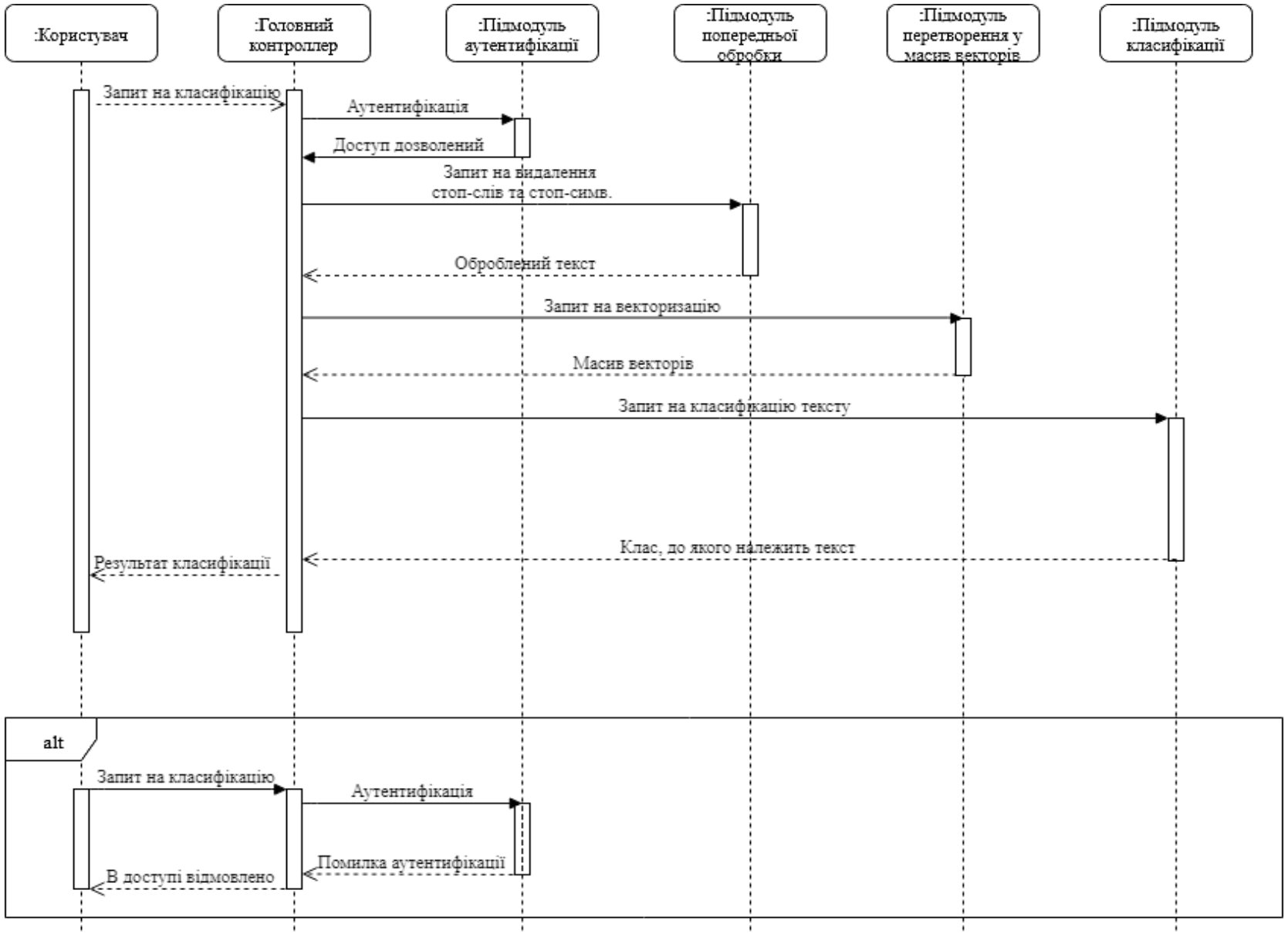

Рисунок 2.8 – Загальна UML-діаграма послідовностей програми класифікації текстів

2.9 Висновок

В даному розділі було розроблено розроблено метод попередньої обробки тексту, інформаційну модель процесу класифікації банківських текстів, визначено архітектуру згорткової нейронної мережі для класифікації текстів та математичну модель згорткової мережі. Мережа складатиметься з вхідного шару розмірністю 400×100, 3 згорткових шарів з фільтрами 3×100, 4×100 та 5×100, 3 агрегувальних шарів та повнозвязного вихідного шару. Кількість фільтрів для кожного згорткового шару – 32. Функція, яка буде використана в шарах агрегування – максимізаційна (max-pooling). Для розподілення ймовірності між класами для вихідного шару в якості функції активації буде використана функція Softmax. Для згорткових та агрегувальних шарів функція активації – ReLU. Метод, за яким буде проходити навчання мережі – це метод зворотного поширення помилки, в основі якого лежить стохастичний градієнтний спуск. Також було визначено набори стоп-символів та стоп-слів, які будуть видалятись з тексту перед подачею на вхід нейронної мережі, що допоможе підвищити достовірність класифікації. У підсумку було розроблено структуру інформаційної технології класифікації банківських текстів на основі згорткової нейронної мережі та розроблено алгоритми роботи модулів програмного забезпечення класифікації банківських текстів.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ КЛАСИФІКАЦІЇ БАНКІВСЬКИХ ТЕКСТІВ НА ОСНОВІ НЕЙРОННОЇ МЕРЕЖІ

3.1 Обґрунтування вибору мови програмування

Вибір мови програмування залежить не тільки від її можливостей і сфери застосування, але і від великої кількості інших факторів, таких як: тип ліцензії, наявність зручних середовищ програмування, об’єм спільноти, кількість та якість відповідних бібліотек та фреймворків.

Програмну реалізацію інформаційної технології класифікації текстів умовно можна поділити на три частини:

- програмна реалізація процесу навчання нейронної мережі;
- програмна реалізація обробки запитів (WebAPI);
- програмна реалізація клієнтського веб-додатка для тестування WebAPI.

Для програмної реалізації процесу навчання нейронної мережі та основних модулей WebAPI було обрано мову програмування Python.

Python – це інтерпретована об'єктно-орієнтована мова програмування високого рівня з динамічною типізацією. Структури даних високого рівня разом із динамічною семантикою та динамічним зв'язуванням роблять її привабливою для швидкої розробки програм, а також як засіб поєднання існуючих компонентів. Python підтримує модулі та пакети модулів, що сприяє модульності та повторному використанню коду. Інтерпретатор Python та стандартні бібліотеки доступні як у скомпільованій так і у вихідній формі на всіх основних платформах. В мові програмування Python підтримується кілька парадигм програмування, зокрема: об'єктно-орієнтована, процедурна, функціональна та аспектно-орієнтована [13].

Основною перевагою Python є велика кількість бібліотек та фреймворків для наукових досліджень в області машинного навчання, зокрема – в області нейронних мереж. Також Python має ефективні структури даних високого рівня

та простий, але ефективний підхід до обєктно-орієнтованого програмування, що робить код лаконічним та простим для сприйняття, порівняно, наприклад, з C++. Елегантний синтаксис Python, динамічна обробка типів, а також те, що це інтерпретована мова, роблять її ідеальною для написання скриптів та швидкої розробки прикладних програм у багатьох галузях на більшості платформ. Інтерпретатор мови Python може бути розширений функціями та типами даних, розробленими на C чи C++ (або на іншій мові, яку можна викликати із C).

В якості середовища програмування для Python було обрано IDE PyCharm, що має ряд зручностей, таких як: підсвічування синтаксису, автодоповнення, менеджер пакетів, інтеграція з системами контроля версій і т. д. Також це середовище програмування має студентський тип ліцензії, що робить його використання безкоштовним для студентів.

Для програмної реалізації модуля аутентифікації було обрано мову програмування C# – об’єктно-орієнтовану мову програмування, яка забезпечує зручні інструменти для аутентифікації користувача.

Для програмної реалізації клієнтського веб-додатка для тестування WebAPI було обрано Typescript, що є надмножиною Javascript.

TypeScript є зворотньо сумісною мовою з JavaScript. Фактично, після компіляції програму на TypeScript можна виконувати в будь-якому сучасному б

р Переваги над JavaScript:

а − можливість явного визначення типів (статична типізація);

у − підтримка використання повноцінних класів (як в традиційних об'єктно-

орієнтованих мовах);

е − підтримка підключення модулів.

р Наведені переваги підвищують швидкість розробки програм, легкість

сприйняття, рефакторинг і повторне використання коду, зменшують час пошуку

помилок на етапі розробки та компіляції, а також швидкодію програм.

б

о

в

В якості середовища для написання клієнтського веб-додатку для тестування та модуля аутентифікації WebAPI було обрано IDE – Visual Studio.

3.2 Використання фреймворків та бібліотек

Для програмної реалізації процесу навчання нейронної мережі було використано бібліотеку Tensorflow.

TensorFlow – відкрита програмна бібліотека для машинного навчання цілій низці задач, розроблена компанією Google для задоволення її потреб у системах, здатних будувати та тренувати нейронні мережі для виявлення та розшифровування образів та кореляцій, аналогічно до навчання й розуміння, які застосовують люди. TensorFlow було початково розроблено командою Google Brain для внутрішнього використання в Google, поки її не було випущено під відкритою ліцензією Apache 2.0 в 2015 році [14]. Tensorflow надає прикладний програмний інтерфейс для декількох мов програмування, зокрема для Python.

Для програмної реалізації основних підмодулів WebAPI було використано мікрофреймворк Flask.

Flask – фреймворк для створення WebAPI для мови програмування Python, що використовує набір інструментів Werkzeug, а також шаблонізатор Jinja2. Відноситься до категорії так званих мікрофреймворків – мінімалістичних каркасів веб-прикладних програм, які реалізують лише базові можливості [15].

Для реалізації процедури аутентифікації було обрано JSON Web Token – це стандарт токена доступу на основі JSON. Використовується для верифікації запитів. JWT складається з трьох частин: заголовка, вмісту і підпису [16].

Заголовок – це JSON елемент, який описує до якого типу токену належить даний і які методи шифрування використовувались. Вміст – складається з елемента JSON який описує твердження. Структура підпису визначається стандартом JSON Web Signature. Щоб згенерувати підпис, заголовок та вміст кодуються в Base64, записуються в один рядок через крапку, а потім цей рядок

хешується. Заголовок, вміст і підпис кодуються в Base64 і розділяються в токені крапкою. JWT передається в заголовках HTTP, зазвичай двома способами: в полі Authorization як Bearer-Token та в полі Cookie.

3.3 Програмна реалізація процесу навчання нейронної мережі

Для навчання нейронної мережі спочатку необхідно завантажити множину текстів та на основі цієї множини сформувати кінцевий масив міток класів. Фрагмент функції, яка завантажує множину текстів та повертає вихідний масив текстів та міток класів до яких вони належать, та масив всіх можливих міток класів:

```
def load_data_and_labels(filename):
df = pd.read_csv(filename, compression='zip',
dtype={'consumer_complaint_narrative': object})
selected = ['product', 'consumer_complaint_narrative']
non_selected = list(set(df.columns) - set(selected))
df = df.drop(non_selected, axis=1)
df = df.dropna(axis=0, how='any', subset=selected)
df = df.reindex(np.random.permutation(df.index))
labels = sorted(list(set(df[selected[0]].tolist())))
encoded_labels = np.zeros((len(labels), len(labels)), int)
np.fill_diagonal(encoded_labels, 1)
label_dict = dict(zip(labels, encoded_labels))
x_raw = df[selected[1]].apply(lambda x: remove_stop_symbols(x))
.apply(lambda x: remove_stop_words(x)).tolist()
y_raw = df[selected[0]].apply(lambda y: label_dict[y]).tolist()
return x_raw, y_raw, labels
```

Функція pd.read_csv зчитує файл з множиною текстів та записує його в змінну df. Потім проводиться вибір полів з множини даних, які потрібні для процесу навчання мережі, а зайві поля видаляються викликом функції df.drop. Після цього формується масив міток класів у вигляді векторів, де один елемент

дорівнює одиниці, а інші – нулю. В кінці функції проводиться видалення стоп-символів та стоп-слів викликами функцій remove_stop_symbols та remove_stop_words відповідно.

Після того, як множина текстів була завантажена, потрібно розбити її на навчальну та тестову вибірки у пропорції 80% : 20%.

```
x_, x_test, y_, y_test = train_test_split(x, y, test_size=0.2)
```

Далі необхідно створити об’єкт згорткової нейронної мережі з бібліотеки TensorFlow та ініціалізувати його:

```
cnn = TextCNN(
   sequence_length=x_train.shape[1],
   num_classes=y_train.shape[1],
   vocab_size=len(vocab_processor.vocabulary_),
   embedding_size=params['embedding_dim'],
   filter_sizes=list(map(int, params['filter_sizes'].split(","))),
   num_filters=params['num_filters')
```

Перший та другий параметр об’єкту TextCNN – це розмірність вхідних даних (400 та 100 відповідно). Третій параметр вказує на кількість слів у словнику word2vec (залежить від кількості текстів, на якій була натренована модель). Четвертий параметр вказує на довжину вектора, в який буде перетворено кожне слово вхідного тексту. П`ятий параметр – це масив з розмірністю згорткових фільтрів (3 × 100, 4 × 100 та 5 × 100). Шостий параметр вказує на кількість фільтрів (32 для кожного згорткового шару).

Після ініціалізації мережі починається процес навчання. Процес навчання складається з певної кількості ітерацій. На кожній ітерації обирається деякий текст з навчальної вибірки та подається на вхід нейронної мережі. Відповідно на кожній ітерації проводиться видалення стоп-слів та стоп-символів та

веткоризація тексту, який буде поданий на вхід мережі. Фрагмент функції, яка виконує видалення стоп-символів:

```
stop_symbols = ["[^A-Za-z0-9(),!?\'\`]", "\'s", "\'ve", "n\'t", "\'re",
"\'d", "\'ll", ",", "!","\(","\)","\?", "\s{2,}"];
stop_words = ["do", "does", "did", "a", "an", "the", "and", "but", "if",
"or", "that", "this", "those", "these", "could", "will", "would",
"shall"];
def remove_stop_symbols(s):
   for stop_sym in stop_symbols:
      s = re.sub(stop_sym, " \'s", s)
   return s.strip().lower();
def remove_stop_words(s):
   for stop_sym in stop_words:
      s = re.sub(stop_sym, " \'s", s)
   return s.strip().lower();
```

Фрагмент функції, яка проводить word3vec векторизацію тексту та доповнення нульовими векторами у тому випадку, коли текст має кількість слів, меншу за максимальну:

```
word2vec_model = gensim.models.KeyedVectors.load_word2vec_format(
   os.path.abspath("GoogleNews-vectors-negative100.bin"), binary=True)
max_document_length = max([len(x.split(' ')) for x in x_raw])
logging.info('The maximum length of all sentences:
{}'.format(max_document_length))
vocab_processor =
learn.preprocessing.VocabularyProcessor(max_document_length,
vocabulary={k: v.index for k,v in word2vec_model.vocab.items()})
x = np.array(list(vocab_processor.transform(x_raw)))
y = np.array(y_raw)
```

Процес навчання мережі складається з повторення даного фрагменту коду:

```
optimizer = tf.train.AdamOptimizer(1e-3)
grads_and_vars = optimizer.compute_gradients(cnn.loss)
train_op = optimizer.apply_gradients(grads_and_vars,
global_step=global_step)
def train_step(x_batch, y_batch):
   feed_dict = {
      cnn.input_x: x_batch,
      cnn.input_y: y_batch,
      cnn.dropout_keep_prob: params['dropout_keep_prob']}
   _, step, loss, acc = sess.run([train_op, global_step, cnn.loss,
cnn.accuracy], feed_dict)
```

Дана функція на кожній ітерації обчислює значення виходу нейронної мережі на основі тексту з навчальної вибірки та на основі порівняння значення виходу нейронної мережі з бажаним результатом (співставлення міток класів) проводить корекцію ваг мережі на основі методу стохастичного градієнтного спуску. Ці всі операції виконуються після виклику функції sess.run всередині бібліотеки Tensorflow.

3.4 Програмна реалізація WebAPI

WebAPI складається з набору POST методів, які можна викликати через HTTP-протокол. Основним методом API є метод /predict. Тілом даного метода є текст, який потрібно класифікувати.

Першим кроком є імпорт файлу з параметрами нейронної мережі та файу з вагами нейронної мережі, який був отриманий в процесі навчання нейронної мережі:

```
params = json.loads(open('./parameters.json').read())
checkpoint_dir = './trained_model'
if not checkpoint_dir.endswith('/'):
    checkpoint_dir += '/'
checkpoint_file = tf.train.latest_checkpoint(checkpoint_dir +
```

```
'checkpoints')
logging.critical('Loaded the trained model: {}'.format(checkpoint_file))
```

Виклик функції json.loads проводить завантаження моделі з параметрами, які були отримані в процесі навчання нейронної мережі.

Після цього текст, який був переданий в тілі POST-методу проходить такі ж кроки, як і під час навчання нейронної мережі: видалення стоп-слів та стоп-символів, перетворення тексту у масив векторів фіксованої довжини та доповнення нульовими векторами у тому випадку, коли текст має кількість слів, меншу за максимальну.

Після цього буде ініціалізовано об’єкт нейронної мережі та обчисленно значення ймовірностей належності тексту до кожного з класів. В результату буде сформовано відповідь та повернено програмі-клієнту:

```
resp = Response(labels[int(all_predictions[0])])
return resp
```

Для того, щоб реалізувати можливість аутентифікації користувачів необхідно зберігати дані про них в базі даних. Серед моделей даних, які можуть бути використані для зберігання даних про користувачів, було обрано реляційну, як таку, що надає велику кількість систем управління реляційними базами даних [17].

Основними вимогами до системи управління базами даних є: висока надійність, можливість розширюватися по мірі наповнення інформацією, без помітного зменшення швидкодії операцій із записами в багатокористувацькому режимі.

Проаналізувавши означені вимоги, можна зробити висновок про доцільність застосування СУБД MS SQL SERVER для маніпулювання даними про користувачів системи.

База даних складається з однієї таблиці, яка містить дані про користувачів. Атрибути таблиці «Користувачі»: «Ід», «Дата реєстрації», «Дата останього входу в систему», «Ім’я», «Прізвище», «Пароль», «Емейл», «Роль».

Фрагмент функції модуля аутентифікації, яка виконує реєстрацію нового користувача:

```
[HttpPost("signup")]
        public IActionResult SignUp([FromBody]SignUpRequest request)
        {
            var user = this._authManager.CreateUser(request);

            if (user == null)
                return BadRequest();

            return Ok();
        }
```

Це метод контролера, який обробляє запити на реєстрацію нових користувачів. Для того, щоб зберегти читабельність коду UserController, реалізацію функцій, зокрема функції реєстрації користувача, покладено на AuthManager. Фрагмент функції CreateUser з AuthManager:

```
public User CreateUser(SignUpRequest request)
        {
            string pwdHash = String.Empty;
            using (var sha = SHA256Managed.Create())
            {
                pwdHash =
String.Concat(sha.ComputeHash(Encoding.UTF8.GetBytes(request.Password))
                        .Select(item => item.ToString("x2")));
            }

            User newUser = new User
            {
```

```
            Name = request.Name,
            Surname = request.Surname,
            Email = request.Email,
            Password = pwdHash,
            Role = "User"
        };
        this._context.Users.Add(newUser);
        this._context.SaveChanges();
        return newUser;
    }
```

Дана функція приймає на вхід об’єкт User який містить такі поля як Name, Surname, Email, Password та Role.

Для того, щоб зменшити вірогідність використання конфіденційних даних зловмисниками, пароль користувача хешується за допомогою хеш-функції SHA-256. Це також захищає користувача від можливості отримання його паролю адміністратором бази даних.

Виклики функцій this._context.Users.Add та this._context.SaveChanges створюють нового користувача в базі даних та зберігають зміни відповідно.

3.5 Програмна реалізація клієнтського веб-додатку для тестування WebAPI

Для того, щоб перевірити роботу WebAPI було створено клієнтський веб-додаток, який приймає текст, введений користувачем, робить запит на сервер, та повертає результат. Головну сторінку веб-додатку для тестування роботи WebAPI зображено на рисунку 3.1.

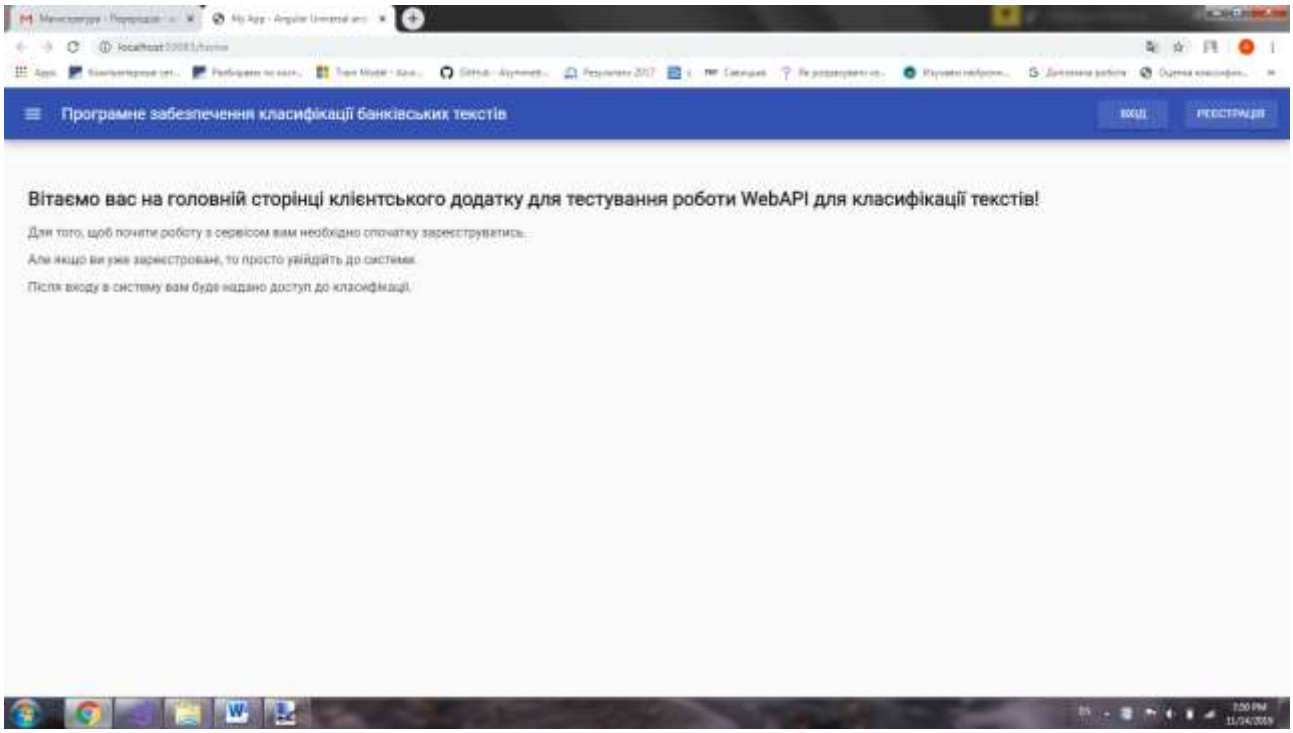

Рисунок 3.1 – Головна сторінка веб-додатку для тестування роботи WebAPI

Фрагмент функції, яка виконує дану процедуру приймання тексту, введеного користувачем, робить запит на сервер, та повертає результат:

```
classify() {
    if (this.wordCount == 0) {
            this.snackBar.open('Введіть текст для класифікації!', null, {
duration: 3000 });
            return; }
        this.api.request({
            body: this.textToClassify,
            method: 'POST',
            url: 'http://localhost:5000/api/predict',
            withAuthorization: true,
            responseType: 'text'
        })
            .subscribe(res => {
                this.dialog.open(ResultDialogComponent, {
                    width: '250px',
                    data: {
```

```
                        class: res}});
            }, err => {
                this.snackBar.open('Під час класифікації сталась помилка.
Спробуйте пізніше.', null, { duration: 3000 });
            });
```

Спочатку проводиться валідація вхідних даних за допомогою оператора if, якщо вхідний текст не містить жодного символу, то буде виведено повідомлення про помилку за допомогою виклику функції this.snackBar.open. Якщо вхідні дані коректні – то за допомогою функції this.api.request буде зроблено запит до WebAPI.

3.6 Результати тестування WebAPI

Для того, щоб отримати доступ для класифікації, спочатку необхідно увійти до системи. Сторінку входу до системи зображено на рисунку 3.2.

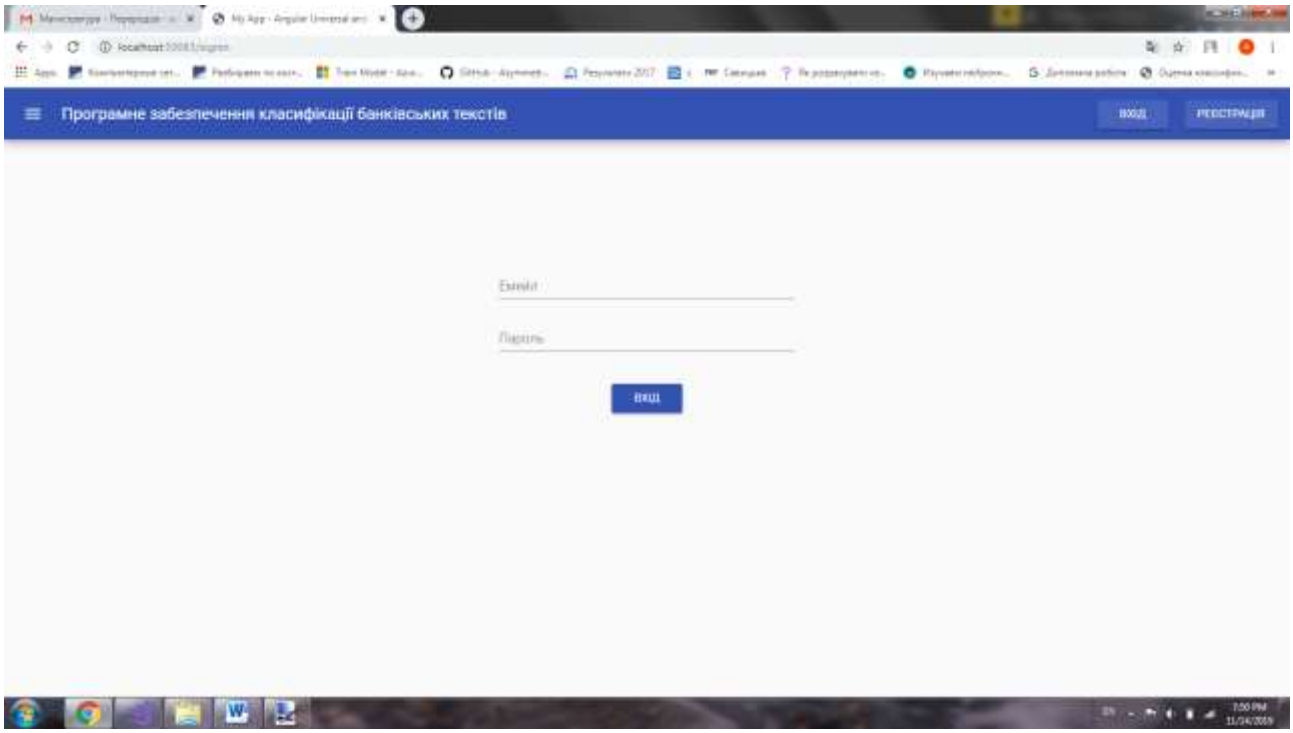

Рисунок 3.2 – Сторінка входу до системи

Після входу до системи користувача буде перенаправлено на сторінку класифікації текстів. Сторінку класифікації текстів зображено на рисунку 3.3.

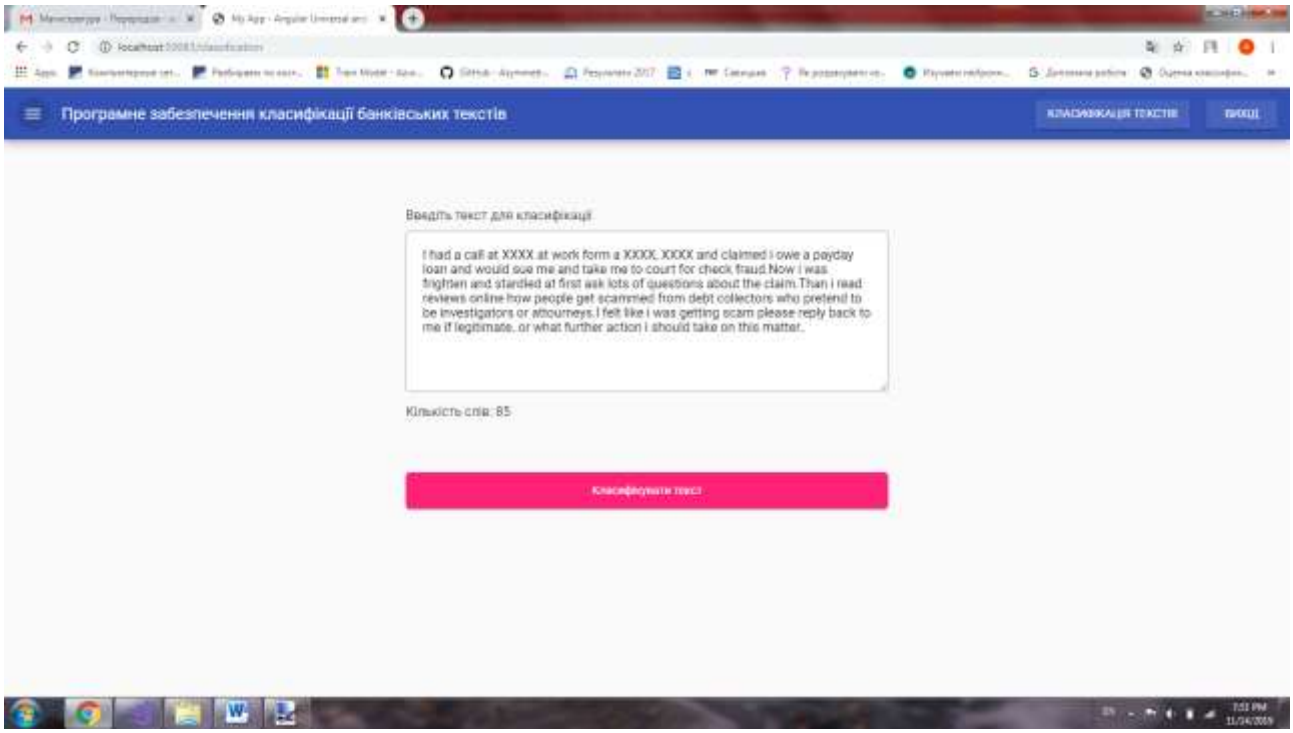

Рисунок 3.3 – Сторінка класифікації текстів

Програма класифікації текстів перед тим як обробити запит проводить валідацію запиту на предмет присутності самого тексту. Тому у випадку натиснення кнопки «Класифікувати текст», попередньо не заповнивши поле з текстом, користувач отримає помилку зображену на рисунку 3.4:

Якщо користувач заповнив поле з текстом, то після натиснення кнопки «Класифікувати текст» буде виконано запит на сервер. Результатом виконання запиту є класс, до якого належить введений текст. Результат виконання запиту зображено на рисунку 3.5:

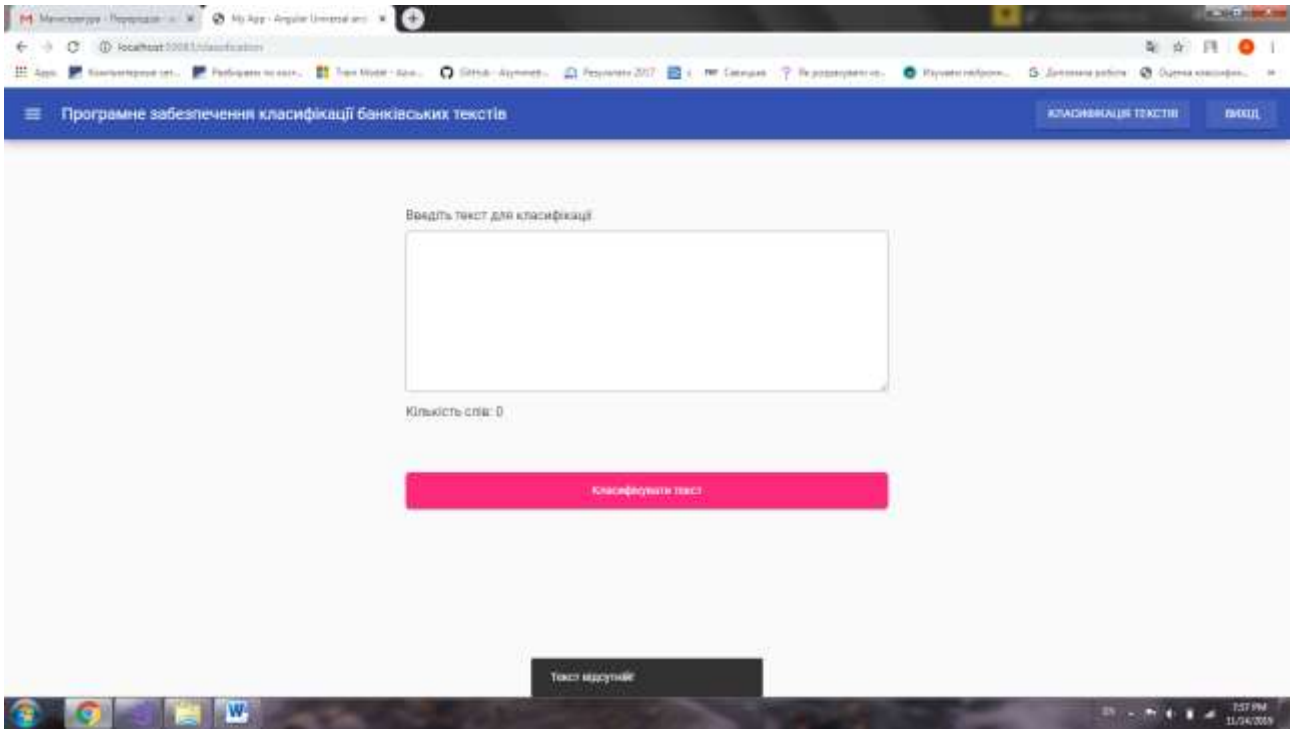

Рисунок 3.4 – Повідомлення про помилку при натисненні кнопки «Класифікувати текст» з пустим текстовим полем

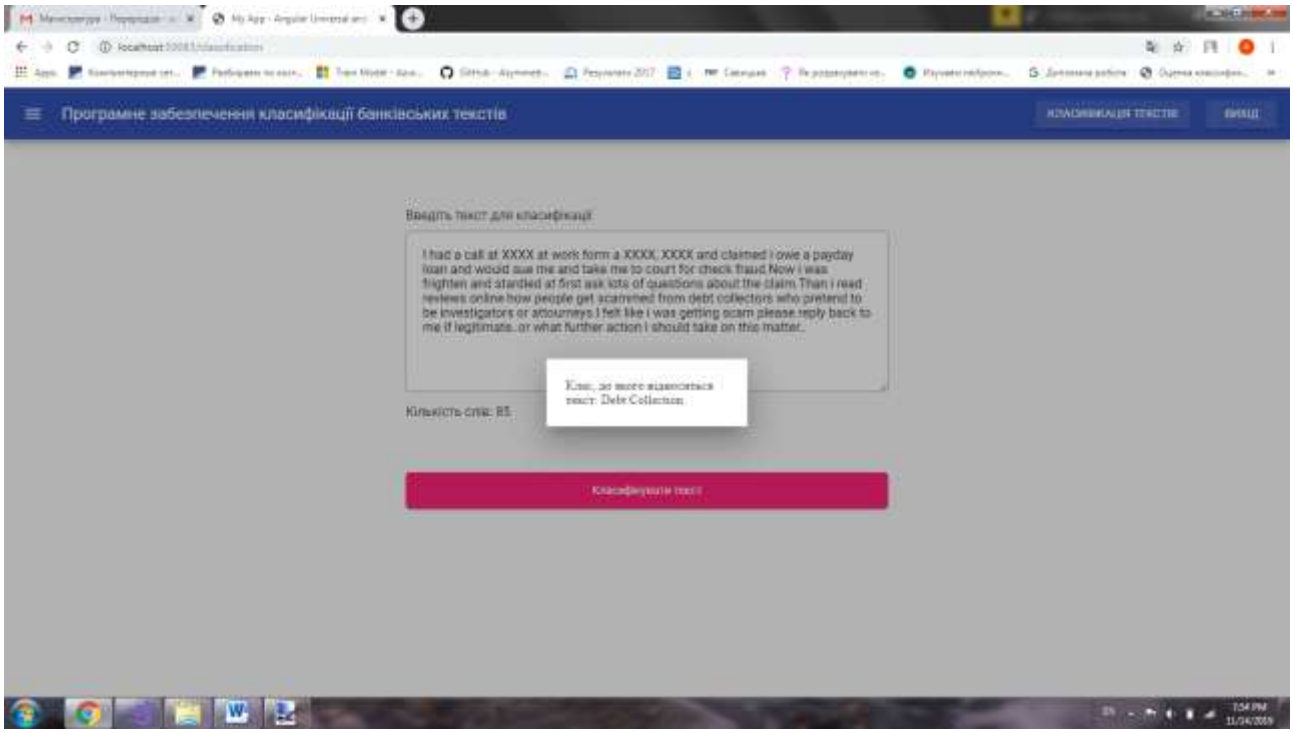

Рисунок 3.5 – Результат класифікації тексту

3.7 Результати навчання нейронної мережі

Навчання нейронної мережі проводилось на CPU Intel I7 4700 MQ (два ядра по два потоки). Досягнена максимальна достовірність класифікації у 85.3%. Графік залежності між точністю класифікації та кількістю класифікованих текстів зображено на рисунку 3.6.

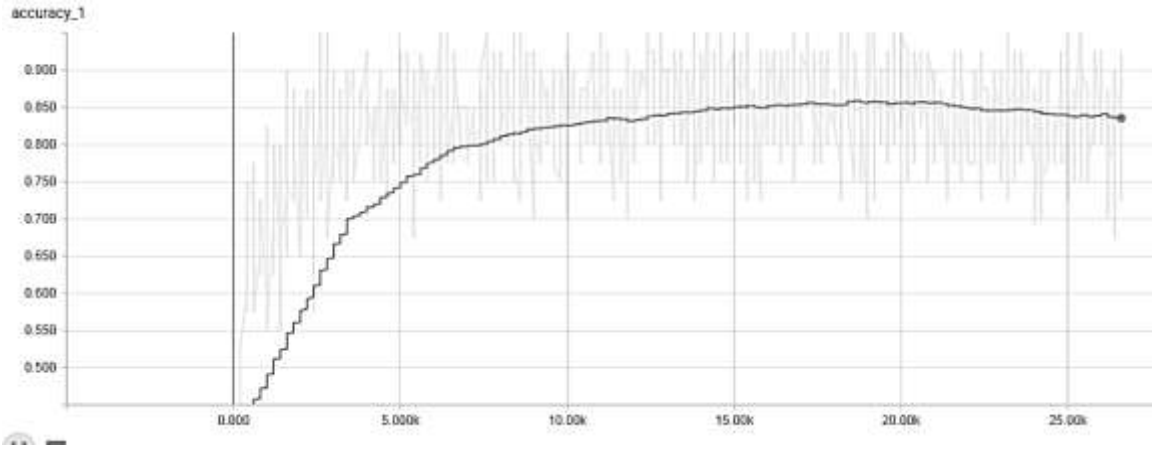

Рисунок 3.6 – Графік залежності між достовірністю класифікації та кількістю класифікованих текстів

Графік залежності між значенням функції втрат та кількістю класифікованих текстів зображено на рисунку 3.7.

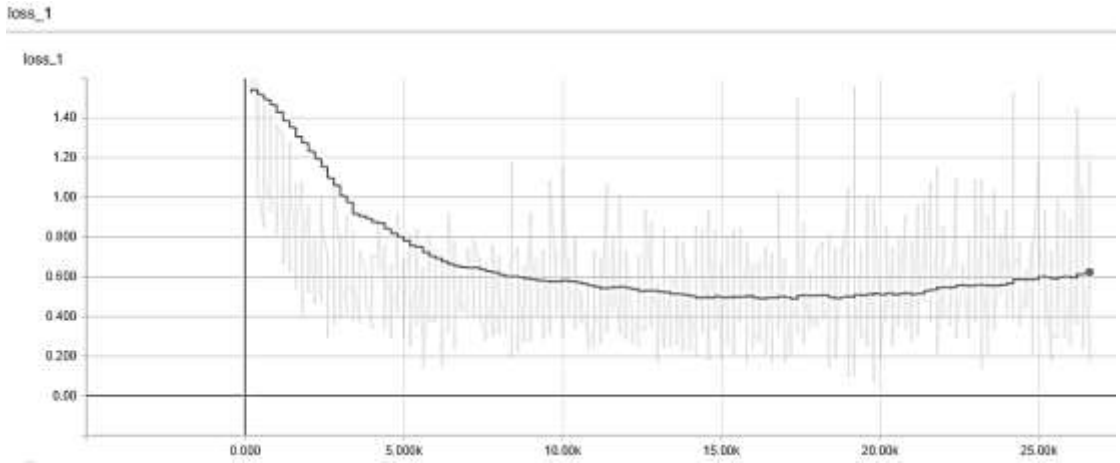

Рисунок 3.7 – Графік залежності функції втрат від кількості класифікованих текстів

Функцію втрат вдалось знизити до значення в 0.45. Графіки були побудовані за допомогою інструмента Tensorboard на основі файлу з метаданими, який був отриманий під час тренування нейронної мережі.

3.8 Порівняння результатів роботи з програмами-аналогами

Порівняння результатів роботи програми класифікації текстів з програмами-аналогами наведено у таблиці 3.1.

Таблиця 3.1 – Порівняння результатів роботи програми класифікації текстів з програмами-аналогами

Характеристика | Засіб	«Multiclass CNN Classifier»	«Sentence Classification CNN»	Програма класифікації текстів
Достовірність класифікації	78.1%	75.8%	85.3%
Видалення стоп-символів	+	+	+
Видалення стоп-слів	-	-	+
Використання готових векторизаторів, навчених на великій вибірці даних	-	-	+
Робота у режимі WebAPI	-	-	+
Аутентифікація користувача	-	-	+

Достовірність класифікації визначається за формулою 3.1.

$$accuracy = \frac{Nt}{Nall} * 100\% , \quad (3.1)$$

де Nt — кількість правильно класифікованих текстів тестової вибірки,

$Nall$ — загальна кількість текстів тестової вибірки,

Із табл. 3.1 видно, що розроблена програма має вищу на 7,2% достовірність розпізнавання (85.3%), ніж перша аналогічна програма (78.1%) та вищу на 9,5% достовірність розпізнавання (85.3%), ніж друга аналогічна програма (75.8%), а значить достовірність класифікації банківських текстів покращена як мінімум на 7,2%, тобто мета роботи досягнута.

Отже, після проведення порівняння розробленої програми класифікації банківських текстів на основі нейронної мережі з аналогами, можна сказати, що розроблена програма має переваги у достовірності розпізнавання та більшу функціональність.

Інструкцію до користування програмою наведено в додатку А, деякі ілюстрації до програми (у т.ч. скріншоти) наведено в додатку В.

3.9 Висновок

В ході практичної реалізації інформаційної технології класифікації банківських текстів у даному розділі було обрано такі мови програмування: для модуля класифікації текстів – Python, для модуля аутентифікації – C#, для клієнтського веб-додатку тестування роботи WebAPI – Typescript та бібліотеки Tensorflow та Flask. Також було описано роботу основних частин програми та наведено відповідні фрагменти коду. В результаті було розроблено програмне забезпечення класифікації банківських текстів на основі згорткової нейронної мережі, яке порівняно з аналогом має кращу на 7,2% достовірність класифікації банківських текстів. Таким чином, мета роботи досягнута – достовірність класифікації банківських текстів підвищена. Також було наведено результати навчання згорткової нейронної мережі та результати роботи програми класифікації текстів. Розроблена програма повністю відповідає завданню, що підтверджується її тестуванням.

4 ЕКОНОМІЧНА ЧАСТИНА

4.1 Визначення комерційного потенціалу розробки

Метою проведення технологічного аудиту є оцінювання комерційного потенціалу розробки. Для проведення технологічного аудиту було залучено 2-х незалежних експертів. Такими експертами будуть к.т.н., доц. каф КН Колесницький О.К. та к.т.н., проф. каф. КН Месюра В.І.

Здійснюємо оцінювання комерційного потенціалу розробки за 12-ма критеріями за 5-ти бальною шкалою [18].

Результати оцінювання комерційного потенціалу розробки наведено в таблиці 4.1.

Таблиця 4.1 – Результати оцінювання комерційного потенціалу розробки

Критерії	Прізвище, ініціали, посада експерта	
	1. Колесницький О.К.	2. Месюра В.І.
	Бали, виставлені експертами:	
1	4	4
2	3	3
3	3	4
4	4	3
5	3	4
6	4	4
7	3	3
8	4	4
9	4	4
10	4	3
11	3	4
12	3	4
Сума балів	$СБ_1 = 43$	$СБ_2 = 44$
Середньоарифметична сума балів $\overline{СБ}$	$\overline{СБ} = \frac{\sum_{1}^{3} СБ_i}{2} = 43{,}5$	

Отже, з отриманих даних таблиці 4.1 видно, що нова розробка має високий рівень комерційного потенціалу.

4.2 Прогнозування витрат на виконання науково-дослідної роботи та конструкторсько–технологічної роботи

Для розробки нового програмного продукту необхідні такі витрати [18].

Основна заробітна плата для розробників визначається за формулою (4.1):

$$З_о = \frac{М}{Т_р} \cdot t, \qquad (4.1)$$

де М- місячний посадовий оклад конкретного розробника;

$Т_р$ - кількість робочих днів у місяці, $Т_р$ = 22 дні;

t - число днів роботи розробника, t = 45 днів.

Розрахунки заробітних плат для керівника і програміста наведені в таблиці 4.2.

Таблиця 4.2 – Розрахунки основної заробітної плати

Працівник	Оклад М, грн.	Оплата за робочий день, грн.	Число днів роботи, t	Витрати на оплату праці, грн.
Науковий керівник	5500	250	7	1750
Інженер-програміст	3500	159,09	45	7159,05
Всього:				8909,05

Розрахуємо додаткову заробітну плату:

$$З_{дод} = 0,1 \cdot 8909,05 = 890,9 \text{ (грн.)}$$

Нарахування на заробітну плату операторів НЗП розраховується як 37,5…40% від суми їхньої основної та додаткової заробітної плати:

$$Н_{зп}= (З_о + З_р)\cdot\frac{\beta}{100}, \quad (4.2)$$

$$Н_{зп}= (8909,05+890,9)\cdot\frac{36,3}{100} = 3557,38 \text{ (грн.)}.$$

Розрахунок амортизаційних витрат для програмного забезпечення виконується за такою формулою:

$$А=\frac{Ц\cdot Н_а}{100}\cdot\frac{Т}{12}, \quad (4.3)$$

де Ц – балансова вартість обладнання, грн;

$Н_а$ – річна норма амортизаційних відрахувань % (для програмного забезпечення 25%);

Т – Термін використання (Т=3 міс.).

Таблиця 4.3 – Розрахунок амортизаційних відрахувань

Найменування програмного забезпечення	Балансова вартість, грн.	Норма амортизації, %	Термін використання, міс.	Величина амортизаційних відрахувань,грн
Персональний комп’ютер	10000	25	3	625
Всього:				625

Розрахуємо витрати на комплектуючі. Витрати на комплектуючі розрахуємо за формулою:

$$K = \sum_1^n Н_і \cdot Ц_і \cdot К_і, \quad (4.4)$$

де n – кількість комплектуючих;

Ні - кількість комплектуючих і-го виду;

Ці – покупна ціна комплектуючих і-го виду, грн;

Кі – коефіцієнт транспортних витрат (приймемо Кі = 1,1).

Таблиця 4.4 - Витрати на комплектуючі, що були використані для розробки ПЗ.

Найменування матеріалу	Одиниці виміру	Ціна, грн.	Витрачено	Вартість витрачених матеріалів, грн.
Флешка	шт.	180	1	180
Пачка паперу	уп.	120	1	120
Ручка	шт.	10	1	10
Всього з урахуванням транспортних витрат				341

Витрати на силову електроенергію розраховуються за формулою:

$$В_е=В\cdot П\cdot Ф\cdot К_п\ ; \qquad (4.5)$$

де В – вартість 1кВт-години електроенергії (В=1,7 грн/кВт);

П – установлена потужність комп'ютера (П=0,6кВт);

Ф – фактична кількість годин роботи комп'ютера (Ф=150 год.);

$К_п$ – коефіцієнт використання потужності ($К_п< 1$, $К_п = 0{,}8$).

$$В_е= 1{,}7 \cdot 0{,}6\cdot 150\cdot 0{,}8 = 122{,}4 \text{ (грн.)}$$

Розрахуємо інші витрати $В_{ін}$.

Інші витрати $І_в$ можна прийняти як (100…300)% від суми основної заробітної плати розробників та робітників, які були виконували дану роботу, тобто:

$$В_{ін} = (1..3) \cdot (З_о + З_р). \qquad (4.6)$$

Отже, розрахуємо інші витрати:

$$В_{ін} = 1,5 * (8909,05 + 890,9) = 14699,92 \text{ (грн).}$$

Сума всіх попередніх статей витрат дає витрати на виконання даної частини роботи:

$$В = З_о + З_д + Н_{зп} + А + К + В_е + І_в$$

В = 8909,05+890,9+3557,38+625+341+122,4+14699,92= 29145,65 (грн.)

Розрахуємо загальну вартість наукової роботи $B_{заг}$ за формулою:

$$В_{заг} = \frac{В_{ін}}{\alpha} \quad (4.7)$$

де α – частка витрат, які безпосередньо здійснює виконавець даного етапу роботи, у відн. одиницях = 1.

$$В_{заг} = \frac{29145,65}{1} = 29145,65$$

Прогнозування загальних витрат ЗВ на виконання та впровадження результатів виконаної наукової роботи здійснюється за формулою:

$$ЗВ = \frac{В_{заг}}{\beta} \quad (4.8)$$

де β – коефіцієнт, який характеризує етап (стадію) виконання даної роботи.

Отже, розрахуємо загальні витрати:

$$ЗВ = \frac{29145,65}{0,9} = 32384,05 \text{ (грн.)}$$

4.3 Прогнозування комерційних ефектів від реалізації результатів розробки.

Спрогнозуємо отримання прибутку від реалізації результатів нашої розробки. Зростання чистого прибутку можна оцінити у теперішній вартості грошей. Це забезпечить підприємству (організації) надходження додаткових коштів, які дозволять покращити фінансові результати діяльності .

Оцінка зростання чистого прибутку підприємства від впровадження результатів наукової розробки. У цьому випадку збільшення чистого прибутку підприємства $\Delta П_i$ для кожного із років, протягом яких очікується отримання позитивних результатів від впровадження розробки, розраховується за формулою:

$$\Delta П_i = \sum_1^n (\Delta П_я \cdot N + П_я \Delta N)_i \qquad (4.9)$$

де $\Delta П_я$ – покращення основного якісного показника від впровадження результатів розробки у даному році;

N – основний кількісний показник, який визначає діяльність підприємства у даному році до впровадження результатів наукової розробки;

ΔN – покращення основного кількісного показника діяльності підприємства від впровадження результатів розробки;

$П_я$ – основний якісний показник, який визначає діяльність підприємства у даному році після впровадження результатів наукової розробки;

n – кількість років, протягом яких очікується отримання позитивних результатів від впровадження розробки.

В результаті впровадження результатів наукової розробки витрати на виготовлення інформаційної технології зменшаться на 35 грн (що автоматично спричинить збільшення чистого прибутку підприємства на 35 грн), а кількість користувачів, які будуть користуватись збільшиться: протягом першого року – на 150 користувачів, протягом другого року – на 125 користувачів, протягом

третього року – 100 користувачів. Реалізація інформаційної технології до впровадження результатів наукової розробки складала 500 користувачів, а прибуток, що отримував розробник до впровадження результатів наукової розробки – 250 грн.

Спрогнозуємо збільшення чистого прибутку від впровадження результатів наукової розробки у кожному році відносно базового.

Отже, збільшення чистого продукту $\Delta П_1$ протягом першого року складатиме:

$$\Delta П_1 = 35 \cdot 500 + (250 + 35) \cdot 150 = 60250 \text{ грн.}$$

Протягом другого року:

$$\Delta П_2 = 35 \cdot 500 + (250 + 35) \cdot (150 + 125) = 95875 \text{ грн.}$$

Протягом третього року:

$$\Delta П_3 = 35 \cdot 500 + (250 + 35) \cdot (150 + 125 + 100) = 124375 \text{ грн.}$$

4.4 Розрахунок ефективності вкладених інвестицій та період їх окупності

Визначимо абсолютну і відносну ефективність вкладених інвестором інвестицій та розрахуємо термін окупності.

Абсолютна ефективність $Е_{абс}$вкладених інвестицій розраховується за формулою:

$$Е_{абс} = (ПП - PV), \qquad (4.10)$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному із років, протягом яких виявляються результати виконаної та впровадженої НДДКР, грн;

т – період часу, протягом якого виявляються результати впровадженої НДДКР, 3 роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні; для України цей показник знаходиться на рівні 0,1;

t – період часу (в роках) від моменту отримання чистого прибутку до точки 2, 3,4.

Рисунок, що характеризує рух платежів (інвестицій та додаткових прибутків) буде мати вигляд, рисунок 4.1.

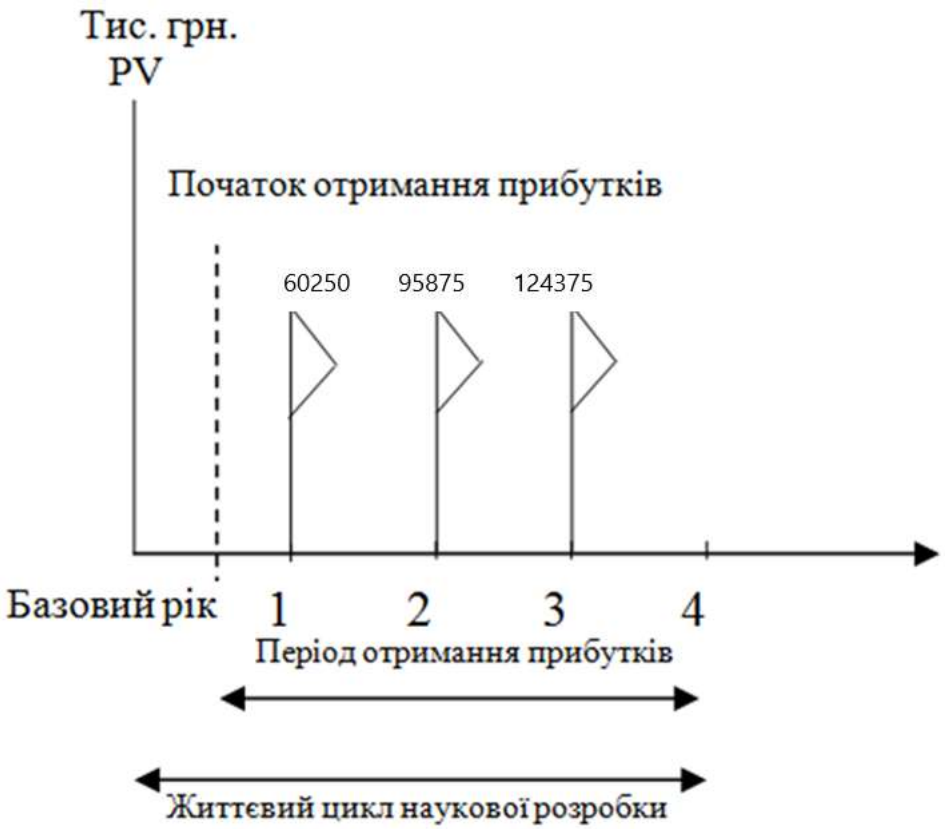

Рисунок 4.1 – Вісь часу з фіксацією платежів, що мають місце під час розробки та впровадження результатів НДДКР

Розрахуємо вартість чистих прибутків за формулою:

$$ПП = \sum_1^m \frac{\Delta П_i}{(1+\tau)^t} \quad (4.11)$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному із років, протягом яких виявляються результати виконаної та впровадженої НДДКР, грн;

т – період часу, протягом якого виявляються результати впровадженої НДДКР, роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні; для України цей показник знаходиться на рівні 0,1;

t – період часу (в роках) від моменту отримання чистого прибутку до точки.

Отже, розрахуємо вартість чистого прибутку:

$$ПП = \frac{32384,05}{(1+0,1)^0} + \frac{60250}{(1+0,1)^2} + \frac{95875}{(1+0,1)^3} + \frac{124375}{(1+0,1)^4} = 239159,52 \text{ (грн.)}$$

Тоді розрахуємо $E_{абс}$:

$$E_{абс} = 239159,52 - 32384,05 = 206775,47 \text{ грн.}$$

Оскільки $E_{абс} > 0$, то вкладання коштів на виконання та впровадження результатів НДДКР буде доцільним.

Розрахуємо відносну (щорічну) ефективність вкладених в наукову розробку інвестицій $E_в$ за формулою:

$$E_в = \sqrt[T]{1 + \frac{E_{абс}}{PV}} - 1 \qquad (4.12)$$

де $E_{абс}$ – абсолютна ефективність вкладених інвестицій, грн;

PV – теперішня вартість інвестицій PV = ЗВ, грн;

$T_ж$ – життєвий цикл наукової розробки, роки.

Тоді будемо мати:

$$E_в = \sqrt[3]{1 + \frac{206775,47}{32384,05}} - 1 = 0,94 \text{ або } 94\ \%$$

Далі, розраховану величина $E_в$ порівнюємо з мінімальною (бар'єрною) ставкою дисконтування $\tau_{мін}$, яка визначає ту мінімальну дохідність, нижче за яку інвестиції вкладатися не будуть. У загальному вигляді мінімальна (бар'єрна) ставка дисконтування $\tau_{мін}$ визначається за формулою:

$$\tau = d + f,$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; в 2019 році в Україні d = 0,2;

f – показник, що характеризує ризикованість вкладень, величина f = 0,1.

$$\tau = 0{,}2 + 0{,}1 = 0{,}3$$

Оскільки $E_в$ = 94% > $\tau_{мін}$ = 0,3 = 30%, то у інвестор буде зацікавлений вкладати гроші в дану наукову розробку.

Термін окупності вкладених у реалізацію наукового проекту інвестицій. Термін окупності вкладених у реалізацію наукового проекту інвестицій $T_{ок}$ розраховується за формулою:

$$T_{ок} = \frac{1}{E_в}$$

$$T_{ок} = \frac{1}{0{,}94} = 1{,}06 \text{ року}$$

Обрахувавши термін окупності даної наукової розробки, можна зробити висновок, що фінансування даної наукової розробки буде доцільним.

4.5 Висновок

В даному розділі було проведено економічне обґрунтування доцільності розробки програми класифікації банківських текстів, яка створена у результаті науково-технічної діяльності. Незалежними експертами було здійснено оцінювання комерційного потенціалу розробки, за результатами якого було визначено, що нова розробка має високий рівень комерційного потенціалу, оскільки середньоарифметична сума балів становить 43,5. Також було виконано прогнозування витрат на виконання розробки, де розраховано основну заробітну плату кожного із розробників, додаткову заробітну плату всіх розробників, нарахування на заробітну плату, амортизацію обладнання, комп'ютерів та приміщень, витрати на допоміжні матеріали, витрати на силову електроенергію тощо. Загальна сума витрат на виконання означених робіт склала 32384,05 грн. Визначено, що абсолютна ефективність вкладених інвестицій становить 206775,47 грн, і це свідчить про те, що вкладання коштів на виконання та

впровадження результатів НДДКР є доцільним. Було розраховано відносну (щорічну) ефективність вкладених в наукову розробку інвестицій – 94 %, її величина більша за мінімальну (бар'єрну) ставку дисконтування, отже інвестор буде зацікавлений у фінансуванні даної наукової розробки. Проведено розрахунок терміну окупності, який складає 1,06 року. В загальному можна зробити висновок, що фінансування розробки програми класифікації банківських текстів, яка створена у результаті науково-технічної діяльності є висококонкурентоспроможним.

ВИСНОВКИ

В результаті виконання магістерської кваліфікаційної роботи було розв’язано задачу розробки інтелектуальної інформаційної технології та програмного забезпечення класифікації банківських текстів згортковою нейронною мережею.

У першому розділі магістерської роботи була поставлена проблема недостатньої достовірності сучасного процесу класифікації банківських текстів. Були розглянуті основні методи та програмні засоби, які виконують класифікації банківських текстів та визначені їх недоліки. Для задачі класифікації текстів було обрано згорткову нейронну мережу, адже за допомогою шарів згортки можна зменшити вхідні дані в рази, і, таким чином, зменшити складність обчислень. Для векторизації слів було обрано алгоритм word2vec, перевагою якого є те, що він перетворює слова у вектори відносно невеликої розмірності. Також було проаналізовано існуючі програми-аналоги та зроблено постановку задачі. На основі вихідних даних, аналізу сучасних методів класифікації текстів та існуючих програм-аналогів було розроблено функціональні вимоги до програми класифікації банківських текстів.

У другому розділі магістерської кваліфікаційної роботи було розроблено метод попередньої обробки тексту, інформаційну модель процесу класифікації банківських текстів, визначено архітектуру згорткової нейронної мережі для класифікації текстів та математичну модель згорткової мережі. Мережа складатиметься з вхідного шару розмірністю 400 ×100, 3 згорткових шарів з фільтрами 3 ×100, 4 ×100 та 5 ×100, 3 агрегувальних шарів та повнозвязного вихідного шару. Кількість фільтрів для кожного згорткового шару – 32. Функція, яка використана в шарах агрегування – максимізаційна (max-pooling). Для розподілення ймовірності між класами для вихідного шару в якості функції активації використана функція Softmax. Для згорткових та агрегувальних шарів – функція активації ReLU. Метод, за яким буде проходити навчання мережі – метод зворотного поширення помилки, в основі якого лежить стохастичний

градієнтний спуск. У підсумку було розроблено структуру інформаційної технології класифікації банківських текстів на основі згорткової нейронної мережі та розроблено алгоритми роботи модулів програмного забезпечення класифікації банківських текстів.

У третьому розділі в ході практичної реалізації інформаційної технології класифікації банківських текстів було обрано такі мови програмування: для модуля класифікації текстів – Python, для модуля аутентифікації – C#, для клієнтського веб-додатку тестування роботи WebAPI – Typescript та бібліотеки Tensorflow та Flask. Також було описано роботу основних частин програми та наведено відповідні фрагменти коду. В результаті було розроблено програмне забезпечення класифікації банківських текстів на основі згорткової нейронної мережі, яке порівняно з аналогом має кращу на 7,2% достовірність класифікації. Таким чином, мета роботи досягнута – достовірність класифікації банківських текстів підвищена. Також було наведено результати навчання згорткової нейронної мережі та результати роботи програми класифікації текстів. Розроблена програма повністю відповідає завданню, що підтверджується її тестуванням.

У четвертому розділі було проведено економічне обґрунтування доцільності розробки програми класифікації банківських текстів, нова розробка має високий рівень комерційного потенціалу- середньоарифметична сума балів становить 43,5. Загальна сума витрат на виконання означених робіт склала 32384,05 грн., абсолютна ефективність вкладених інвестицій становить 206775,47 грн. Відносна (щорічна) ефективність вкладених в наукову розробку інвестицій – 94 %, отже інвестор буде зацікавлений у фінансуванні даної наукової розробки. Термін окупності складає $1,06$ року. В загальному можна зробити висновок, що фінансування розробки програми з використанням згорткової нейронної мережі є економічно доцільним проектом.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Переродов А. О., Петришин С. І. Методи перетворення слова у вектор фіксованої довжини для задачі класифікації текстів // Тези XI науково-практичної конференції «ІОН-2018», 22–25 травня, 2018. – Вінниця: ВНТУ, 2018.

2. Переродов А. О., Колесницький О. К., Денисов І. К. Інформаційна технологія класифікації банківських текстів на основі згорткової нейронної мережі // Матеріали конференції «Молодь в науці: дослідження, проблеми, перспективи-2020», Вінниця, 2019. [Електронний ресурс]. Режим доступу: https://conferences.vntu.edu.ua/index.php/mn/mn2020/paper/view/8394. Дата звернення: Листопад 2019.

3. Методи автоматичної класифікації текстів – [Електронний ресурс]. – Режим доступу: http://www.swsys.ru/index.php?page=article&id=4252

4. Штучні нейронні мережі – [Електронний ресурс]. – Режим доступу: http://archive.ws-conference.com/wp-content/uploads/pw0060

5. Згорткові нейронні мережі – [Електронний ресурс]. – Режим доступу: http://ru.datasides.com/code/cnn-convolutional-neural-networks/

6. GloVe: Global Vectors for Word Representation – [Електронний ресурс]. – Режим доступу: https://nlp.stanford.edu/pubs/glove.pdf

7. Sentence Classification CNN – [Електронний ресурс]. – Режим доступу: https://github.com/umairqadir97/Sentense-Classification-with-CNN

8. Multiclass CNN Classifier – [Електронний ресурс]. – Режим доступу: https://github.com/jiegzhan/multi-class-text-classification-cnn

9. M. Toman. Influence of Word Normalization on Text Classification / M. Toman, R. Tesar, K. Jezek. – Plzen, Czech Republic, 2006. – 5 p.

10. Softmax функція – [Електронний ресурс]. – Режим доступу: http://matlab.exponenta.ru/neuralnetwork/book2/20/softmax.php

11. Штучний нейрон – [Електронний ресурс]. – Режим доступу: http://helpiks.org/5-72193.html

12. Декомпозиція інформаційних систем – [Електронний ресурс]. – Режим доступу: http://helpiks.org/1-27670.html

13. UML-діаграми у процесі проектування ПО – [Електронний ресурс]. – Режим доступу: http://baumanki.net/lectures/10-informatika-i-programmirovanie/273-uml/3457-2-vidy-diagramm-uml.html

14. The Quic Python Book / Vernon L. Ceder - In Proc. of the 24-th VLDB Conf., New York, 1996. – P. 10.

15. Tensorflow – [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/Tensorflow

16. Flask – [Електронний ресурс]. – Режим доступу: https://ru.wikipedia.org/wiki/Flask_(веб-фреймворк)

17. JSON Web Token – [Електронний ресурс]. – Режим доступу: https://jwt.io/

1 Методичні вказівки до виконання студентами-магістрантами економічної частини магістерських кваліфікаційних робіт / Уклад. В. О. Козловський – Вінниця: ВНТУ, 2012. – 22 с.