НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО”

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

До захисту допущено:
Завідувач кафедри
_____ Сергій СТІРЕНКО
(підпис)
“__” _____2022 р.

Дипломний проєкт

на здобуття ступеня бакалавра
за освітньо-професійною програмою “Комп’ютерні системи та мережі”
спеціальності 123 “Комп’ютерна інженерія”

на тему: Сервіс аналізу емоційного забарвлення текстових повідомлень

Виконав: студент 4 курсу, групи ІО-82
(шифр групи)
Шендріков Євгеній Олександрович _____
(прізвище, ім’я, по батькові) (підпис)

Керівник доцент, к.т.н. Болдак А. О. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант (нормоконтроль) проф., д.т.н., Сімоненко В. П. _____
(назва розділу) (посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

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

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2022 р.

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО”

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

Рівень вищої освіти – перший (бакалавр)

Освітньо-професійна програма

“Комп’ютерні системи та мережі”

спеціальність 123 “Комп’ютерна інженерія”

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Сергій СТІРЕНКО
(підпис)
“__” _____2022 р.

–

ЗАВДАННЯ
на бакалаврський дипломний проєкт студента

Шендрікова Євгенія Олександровича

1. Тема проєкту *Сервіс аналізу емоційного забарвлення текстових повідомлень*
керівник проєкту Болдак Андрій Олександрович, доцент, к.т.н.
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)
затверджені наказом по університету від _____ 2022 року №_____

2. Термін здачі студентом закінченого проєкту _*11 червня 2022 р.*

3. Вихідні дані до проєкту *технічна документація, теоретичні дані*

4. Зміст розрахунково-пояснювальної записки (перелік питань, які розробляються)

Розділ 1. Огляд предметної області

Розділ 2. Огляд моделей для класифікації настроїв та методи їх оцінювання

Розділ 3. Реалізація і порівняння моделей класифікації та розробка веб-сервісу для сентимент-аналізу

5. Перелік графічного матеріалу (з точним позначенням обов'язкових креслень) *схема бази даних (структурна схема), діаграма архітектури веб-сервісу (функціональна схема), блок схема алгоритму роботи сервісу (принципова схема).*

6. Консультанта проєкту, з вказівкою розділів проєкту, які до них вносяться

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
Нормоконтроль	д.т.н., проф. Сімоненко В.П.		

7. Дата видачі завдання_____

Календарний план

№ п/п	Найменування етапів дипломного проєкту	Терміни виконання етапів проєкту	Примітки
1.	*Затвердження теми проєкту*	*13.12.2021-17.12.2021*	
2.	*Вивчення та аналіз завдання*	*17.12.2021-18.03.2022*	
3.	*Розробка архітектури та загальної структури системи*	*18.03.2022-25.03.2022*	
4.	*Розробка структур окремих підсистем*	*25.03.2022-8.04.2022*	
5.	*Програмна реалізація системи*	*8.04.2022-15.04.2022*	
6.	*Оформлення пояснювальної записки*	*15.04.2022-19.05.2022*	
7.	*Захист програмного продукту*	*25.05.2022*	
8.	*Передзахист*	*13.06.2022*	
9.	*Захист*	*22.06.2022*	

Студент-дипломник _____ Євгеній ШЕНДРІКОВ
(підпис)

Керівник проєкту _____ Андрій БОЛДАК

АНОТАЦІЯ

У даному бакалаврському дипломному проєкті було розроблено сервіс для аналізу емоційного забарвлення текстових повідомлень для української та англійської мов. Основна задача сервісу полягає у сентимент-аналізі мотиваційних листів, звітів, офіційних документів, коментарів тощо. Це робиться для того, щоб надати оцінювачу можливість зосередитися на найважливіших висновках і, таким чином, полегшити робоче навантаження та звільнити себе від трудомісткої та нудної роботи.

У проєкті також було оглянуто, розроблено та порівняно найпопулярніші моделі класифікації з використанням машинного та глибинного навчання. А найкращі моделі були використані для роботи сервісу.

Програмна реалізація моделей класифікації була розроблена на мові Python. Користувацький інтерфейс, у вигляді веб-сервісу, було створено за допомогою веб-фреймворку Django.

ANNOTATION

In the given bachelor's thesis, a service for the analysis of the sentimental component of text messages for the Ukrainian and English languages has been developed. The main task of the service lies in the sentiment analysis of motivational letters, reports, official documents, comments, etc. It is performed to allow the assessor to focus on the most important conclusions and thus facilitate the workload and free themselves from time-consuming and tedious work.

In the thesis, the most popular machine and deep learning classification models were reviewed, developed, and compared. The best models were used for the service performance.

The software implementation of classification models was developed in Python. The user interface in the form of a web service was created using the Django web framework.

ВІДОМІСТЬ ПРОЄКТУ
ДО ДИПЛОМНОГО ПРОЄКТУ

на тему: «*Сервіс аналізу емоційного забарвлення текстових повідомлень*»

Київ – 2022 року

№ рядка	Формат	*Значення*	*Найменування*	Кіл. листів	№ екземпляра	*Додаток*
			Документація загальна			
			Знову розроблена			
1	*А4*	***ІАЛЦ.467800.001 ВП***	Сервіс аналізу емоційного	1		
			забарвлення текстових			
			повідомлень			
			Відомість проєкту			
2	*А4*	***ІАЛЦ.467800.002 ТЗ***	Сервіс аналізу емоційного	3		
			забарвлення текстових			
			повідомлень			
			Технічне завдання			
3	*А4*	***ІАЛЦ.467800.003 ПЗ***	Сервіс аналізу емоційного	107		
			забарвлення текстових			
			повідомлень			
			Пояснювальна записка			
4	*А4*	***ІАЛЦ.467800.004 Д1***	Сервіс аналізу емоційного	1		
			забарвлення текстових			
			повідомлень			
			Діаграма бази даних			
5	*А3*	***ІАЛЦ.467800.005 Д2***	Сервіс аналізу емоційного	1		
			забарвлення текстових			
			повідомлень			
			Архітектура веб-сервісу			
6	*А4*	***ІАЛЦ.467800.006 Д3***	Сервіс аналізу емоційного	1		
			забарвлення текстових			
			повідомлень			
			Послідовність дій роботи			
			сервісу			

					ІАЛЦ.467800.001 ВП			
Зм.	*Арк.*	*№ докум.*	*Підпис*	*Дата*				
Розробив		*Шендріков Є.О.*			*Сервіс аналізу емоційного забарвлення текстових повідомлень*	*Літ.*	*Аркуш*	*Аркушів*
Перевірив		*Болдак А.О.*					1	1
Реценз.					*Відомість проєкту*	*«КПІ імені Ігоря Сікорського» ФІОТ, гр. ІО-82*		
Н. Контр.		*Сімоненко В.П.*						
Затв.		*Стіренко С.Г.*						

ТЕХНІЧНЕ ЗАВДАННЯ
ДО ДИПЛОМНОГО ПРОЄКТУ

на тему: «*Сервіс аналізу емоційного забарвлення текстових повідомлень*»

Київ – 2022 року

ЗМІСТ

1. НАЙМЕНУВАННЯ ТА ОБЛАСТЬ ЗАСТОСУВАННЯ

Найменування розробки: «Сервіс аналізу емоційного забарвлення текстових повідомлень».

Область застосування: збір зворотного зв’язку та різного роду документів від користувачів з наступним їх сентимент-аналізом оцінювачами.

2. ПІДСТАВИ ДЛЯ РОЗРОБКИ

Основою для розробки є завдання на здобуття ступеня «бакалавр комп’ютерної інженерії», затверджене кафедрою обчислювальної техніки Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського».

3. МЕТА ТА ПРИЗНАЧЕННЯ РОЗРОБКИ

Метою проєкту є створення унікального веб-сервісу для аналізу емоційного забарвлення текстових документів, що допоможе оцінювачу зосередитися на важливих висновках.

4. ДЖЕРЕЛА РОЗРОБКИ

Джерелом інформації є науково-технічна література, публікації у періодичних виданнях та електронні статті у мережі Інтернет.

5. ТЕХНІЧНІ ВИМОГИ

5.1. Вимоги до розроблюваного продукту

- Коректна попередня обробка даних.
- Висока точність моделі сентимент-аналізу для української та англійської мов.
- Сумісність з будь-якою сучасною операційною системою з будь-яким сучасним браузером для перегляду веб-сторінок.
- Зручний користувацький інтерфейс веб-сервісу.

5.2. Вимоги до програмного забезпечення

- Сучасні версії операційних систем: Linux, Windows, MacOS.

- Будь-який сучасний веб-браузер.
- Наявність доступу до мережі Internet.
- Python 3.8+.

5.3. Вимоги до апаратної частини

Мінімально:

- Процесор: Intel Core i5 і вище.
- Оперативна пам’ять: 8 Гб і вище.
- Наявність доступу до мережі Internet.

6. ЕТАПИ РОЗРОБКИ

Вивчення необхідної літератури	24.12.2021
Складання та узгодження технічного завдання	20.01.2022
Написання теоретичної основи	20.04.2022
Розробка архітектури та програмна реалізація	10.05.2022
Тестування та виправлення помилок	12.05.2022
Оформлення документації дипломного проєкту	20.05.2022
Попередній захист та проходження нормативного контролю	08.06.2022
Захист дипломного проєкту	22.06.2022

ПОЯСНЮВАЛЬНА ЗАПИСКА
ДО ДИПЛОМНОГО ПРОЄКТУ

на тему: «*Сервіс аналізу емоційного забарвлення текстових повідомлень*»

Київ – 2022 року

ЗМІСТ

ПЕРЕЛІК ТЕРМІНІВ ТА СКОРОЧЕНЬ

СА	Сентимент-аналіз
API	Application Programming Interface
URL	Uniform Resource Locator
NLP	Natural Language Processing
NLTK	Natural Language Toolkit
РНМ	Рекурентна нейронна мережа
ДКЧП	Довга короткочасна пам'ять
ЗНМ	Згорткова нейронна мережа
КНН	К-найближчих сусідів
ОВМ	Опорно-векторні машини
ROC	Receiver operating characteristic
SQL	Structured query language
HTML	HyperText Markup Language
CSS	Cascading Style Sheets
ОП	Оперативна пам’ять

ВСТУП

Почуття і думки є суттєвими ознаками людського існування. Уміння виражати і розуміти емоції має безпосередній вплив на наше пізнання та поведінку й відіграє важливу роль у нашому повсякденному житті. Рішення, які ми приймаємо, тісно пов'язані з настроями та ставленням як до нас самих, так і до інших.

У цю епоху великих даних кожна людина стає одним соціальним сенсором, і загалом завдяки цьому ми можемо краще зрозуміти наше соціально-економічне середовище. Щоб зрозуміти суспільство, багато дослідників доклали чимало зусиль. Одна з таких цікавих і популярних тем в цій галузі – аналіз настроїв (сентимент-аналіз, СА). Аналіз настроїв і емоцій не є новими, їх вивчають і застосовують у широкому діапазоні майже в кожній діловій та соціальній сфері.

Гарний приклад використання громадського настрою можна було побачити під час Олімпійських ігор у Лондоні 2012 року. Відомий спортсмен Дейлі Томпсон створив світлове шоу, яке виставили на лондонське око. Аналізатор настроїв у режимі реального часу класифікував твіти про Олімпійські ігри на позитивні та негативні категорії та засвітив лондонське око відповідно до відсотка позитивних твітів.

Ще одним прикладом є дослідження Асура та Хубермана [1] в результаті якого їм вдалося успішно спрогнозували касові збори фільмів за допомогою методів сентимент-аналізу, використовуючи дані Twitter.

Щодня в Інтернеті публікується велика кількість різного роду інформації: люди публікують відгуки про товари, висловлюють політичні погляди та діляться своїми почуттями в соціальних мережах. Таким чином, природно, що ця інформація про настрої не тільки впливає на нашу поведінку, але й має значний вплив на наш процес прийняття рішень. Перш ніж зробити покупку або відвідати місцевий ресторан, більшість людей перевіряють відгуки в Інтернеті та читають відгуки клієнтів. На платформах соціальних

медіа, таких як Twitter і Facebook, люди також діляться поглядами на особисті важливі теми, трейдери висловлюють свою торгову позицію, а політики передають свої повідомлення виборцям.

Актуальність дослідження тісно пов'язана з тим, що багато компаній залежать від своїх користувачів і тому повинні прислухатися до того, що вони говорять про свої послуги та продукти. Один із способів зробити це – аналіз настроїв. Згідно з опитуваннями, проведеним у квітні 2013 р., 90% рішень клієнтів залежать від онлайн-оглядів [2]. Дослідження 2017 року [3] показують, що 88% довіри клієнтів базується на оглядах персональних рекомендацій в Інтернеті.

Крім того, попередні дослідження показують, що на даний момент є вкрай мало робіт з сентимент-аналізу для українських текстів, таким чином створення моделі для аналізу настроїв текстів українською є вкрай актуальним та потрібним.

Мета бакалаврського дипломного проєкту полягає у дослідженні, як глибинне навчання можна застосувати для аналізу настроїв українських та англійських текстів і їх поділу на позитивні та негативні групи за допомогою різних методів машинного навчання, а також у перевірці, який класифікатор та архітектура працює найкраще.

Об'єктом дослідження даного бакалаврського диплому є класифікатори машинного навчання та моделі нейронних мереж. **Предмет** дослідження – веб-сервіс для сентимент-аналізу офіційних документів та зворотного зв'язку.

Модель машинного навчання буде навчатися на наборі даних Amazon. Причина, чому використовується саме цей набір даних, полягає в тому, що Amazon надає великі обсяги загальнодоступних даних, а також тому, що набір містить відгуки з різних категорій, що дозволить використовувати впроваджену та навчену модель для класифікації або аналізу настроїв текстів майже будь-якої тематики, що допомогло б автоматизувати та прискорити процес отримання огляду продукту чи компанії.

Оскільки для цього експерименту немає доступного набору даних для навчання української моделі, деяку частину часу для цього дипломного проєкту буде відведено для підготування потрібної кількості навчальних даних.

Дані будуть зібрані шляхом перекладу корпусу даних Amazon на українську. Переклад відгуків буде відбуватися за допомогою скрипта, який буде відправляти запити на API Google Translate і отримувати переклад.

При цьому ми розуміємо, що це не є найкращим варіантом для підготовки корпусу даних, оскільки українська та англійська мови відрізняються, коли справа доходить до побудови речень та опису емоцій у тексті. Однак, на схожу тему було проведено дослідження з експериментом [4]. Їхні результати показують, що перекладені таким чином тексти дійсно дають непогані результати для навчання, але певною мірою сентимент у реченні може мати інше значення. Щоб володіти стовідсотковою впевненістю, ви повинні потім вручну перевіряти кожен переклад.

При цьому, точність все одно буде набагато вищою, ніж якщо б ми використовували власні думки щодо того, чи вважаємо ми певні відгуки негативними чи позитивними. Через це та обмеження часу, використання перекладача для створення корпусу даних для навчання української моделі вважаємо обґрунтованим.

Кінцевим результатом дипломного проєкту стане створення веб-сервісу для аналізу мотиваційних листів, повідомлень зворотного зв’язку та іншого роду ділових документів. Ми створимо цілком нову запропоновану методику оцінки зворотного зв’язку на основі аналізу настроїв.

Дана спеціалізація сервісу обґрунтовується тим, що у поєднанні зі статистичною обробкою відповідей інструменти штучного інтелекту можуть допомогти обробити звіти/документи, щоб оцінити настрої у роботах та зауваженнях, а також допомогти витягти необхідну інформацію з текстових частин. Таким чином, це допоможе знизити робоче навантаження для

оцінювача документів, допомагаючи йому зосередитися на важливій інформації.

Застосування аналізу настроїв для ділових документів та зворотного зв'язку має багато можливостей. Робота для оцінювача значно полегшиться, а відгуки та роботи студентів можуть бути розглянуті миттєво. З часом можна буде спостерігати за відгуками студентів, і потім визначити закономірності, що дозволить викладачу внести відповідні зміни.

У цьому дослідженні нашими **завданнями** є:

1. Огляд предметної області аналізу настроїв;
2. Аналіз існуючих рішень;
3. Розгляд, розробка та порівняння класифікаторів машинного навчання та моделей глибинного навчання для англійської та української мов;
4. Розробка веб-сервісу для сентимент-аналізу мотиваційних листів та зворотного зв'язку.

В проєкті ми використали такі **методи дослідження**: аналіз літератури, моделювання, опис, порівняння та формалізація.

Наукова новизна дослідження полягає у самій постановці й розробці проблеми аналізу настроїв ділових документів та зворотного зв'язку, оскільки попередні дослідження переваг використання сентимент-аналізу у даній області фактично відсутні, тому створення сервісу, який вирішить цю проблему є не тільки виправданим, а й виключно необхідним.

Практичне значення. Цей дипломний проєкт буде корисний в першу чергу для викладачів та їхніх студентів, збір зворотного зв'язку від студентів в режимі реального часу з наступним сентимент-аналізом дозволить покращити спілкування між викладачем і студентами, дозволяючи лекторові мати загальний підсумок думки студентів щодо курсу, матеріалу, лекції тощо. Крім того, це дозволить покращити процес навчання загалом та розуміти поведінку студентів.

Система буде візуалізувати відгуки студентів у змістовний спосіб, надаючи викладачу найважливішу інформацію зі зворотного зв’язку. Викладачі можуть використовувати цю інформацію, щоб змінити свій стиль викладання або оцінювання.

Зрештою, система може привести до корисної інформації, яку можна використовувати для моніторингу прогресу студентів, і, наприклад, для зміни навчальних матеріалів.

Також створені моделі машинного навчання можуть бути використані компаніями для аналізу відгуків своїх клієнтів і за потреби внесення корективів у свої продукти.

Даний проєкт також буде корисною для тих, кому цікаво дізнатися більше про машинне навчання, особливо про використання та роботу різних класифікаторів та архітектур для сентимент-аналізу тексту.

Структура дипломного проєкту. Проєкт складається зі вступу, 3 розділів, висновків, списку використаних джерел та літератури, додатків (6 с.). Загальний обсяг дослідження – 107 стр.

У першому розділі проводиться повний огляд предметної області, де детально описується кожен етап процесу аналізу настроїв, його завдання, складності, відкриті проблеми, а також наводиться опис існуючих рішень та проводиться їх порівняння.

У другому розділі відбувається ґрунтовний огляд моделей машинного та глибинного навчання для класифікації настроїв та методи їх оцінювання.

У третьому розділі надано обґрунтування обраних засобів розробки, описано процес розробки та порівняння моделей класифікації, створення та опис функціональності веб-сервісу, а також його подальше розгортання та тестування можливостей.

У висновках підбиваються підсумки щодо розробленого проєкту, а також описуються напрямки майбутньої роботи.

РОЗДІЛ 1
ОГЛЯД ПРЕДМЕТНОЇ ОБЛАСТІ

1.1. Визначення

Згідно зі словником Мерріам-Вебстер [5], слово «сентимент» має три шари значень:

• Прихильність (пристрасть);

• Емоції або витончені почуття;

• Ідея, забарвлена емоціями.

За визначенням, весь автоматичний аналіз властивостей такого роду потрапляє в діапазон аналізу настроїв. За словами Лю [6], сентимент аналіз – це область дослідження, яка аналізує думки людей, настрої, оцінки, ставлення та емоції щодо таких сутностей, як продукти, послуги, організації, особи, проблеми, події, теми та їх атрибути. Цей термін можна використовувати як взаємозамінні з *«opinion mining»*.

Фарзіндар [7] розділяє аналіз почуттів і аналіз емоцій, щоб підкреслити тонку різницю, у нього аналіз емоцій більш прискіпливо класифікується на тонку деталізацію.

Також автор поділяє емоції на шість класів: гнів, огида, страх, радість, смуток, здивування, які найбільш широко використовуються в літературі. Хоча наразі немає консенсусу щодо того, скільки класів емоцій слід використовувати.

Розрізнення аналізу почуттів і аналізу емоцій виходить за рамки цієї роботи. У нашому дослідженні всі семантичні орієнтації трьох шарів, виражені щодо певних сутностей, зараховуються як настрої.

1.1.1. Компоненти настрою

Сентимент можна розділити на різні компоненти: власник, ціль, аспект і полярність. Кожен компонент відповідає певним завданням у системі.

Власник (англ. *holder*) позначає сутність, яка володіє настроєм.

Ціль (англ. *target*) визначає об’єкт, обраний як мету настрою.

Полярність (англ. *polarity*) є властивістю настроїв. Полярність може бути двократною (позитивною і негативною) або трикратною (позитивною, негативною і нейтральною).

Аспект (англ. *aspect*) визначає конкретну частину або ознаку цілі, до якої висловлюються настрої.

Візьмемо для прикладу наступне речення [8]:

Стів Джобс сказав, що Microsoft просто не має смаку.

Настрої в цьому реченні можна проаналізувати, оскільки власник (*«Стів Джобс»*) висловив думки щодо цілі (*«Microsoft»*), а полярність настроїв негативна (*«не має смаку»*).

Аспект також є важливим компонентом настрою. Візьмемо для прикладу наступний текст огляду компанії [9]:

KindGeek, як компанія, має позитивну, орієнтовану на людей культуру. Переваги та баланс між роботою і життям чудові. Однак міжкомандне спілкування може бути складним.

Користувач дав загальну оцінку KindGeek щодо чотирьох аспектів: культура, переваги, баланс між роботою та особистим життям і спілкування. У той час як перші три аспекти отримують позитивну оцінку, останній отримує негативну оцінку, як описано нижче:

Культура – позитивно (*«позитивна, орієнтована на людей, перспективне мислення»*).

Переваги – позитивно (*«чудові»*).

Баланс роботи та особистого життя – позитивно (*«чудово»*).

Спілкування – негативно (*«складне»*).

У підсумку, власник, мета, полярність і аспект – це чотири основні компоненти настрою.

Вони працюють разом, щоб передати почуття, виражені природною мовою.

1.1.2. Рівні навчання

Аналіз настроїв можна розділити на категорії відповідно до деталізації тексту. Попередня робота в основному зосереджена на трьох рівнях:

- Рівень документа/тексту

Аналіз цього рівня полягає в тому, щоб визначити, чи є настрої, виражені в цілому документі, позитивними чи негативними. Наприклад [10], враховуючи огляди продуктів, система зможе оцінити загальну полярність настроїв. Аналіз на рівні документа передбачає, що фрагмент тексту виражає настрої щодо однієї цілі. Хоча це зазвичай стосується оглядів продуктів, фільмів, ресторанів тощо, це, ймовірно, не стосується ситуацій, коли документ критикує декілька об'єктів.

- Рівень речення

Аналіз цього рівня полягає в тому, щоб визначити, чи є думки, висловлені в реченні, позитивними чи негативними. Аналіз рівня речення можна провести двома способами.

Один із способів – просто розглядати такий аналіз як завдання 2-сторонньої класифікації, де мітки є позитивними та негативними.

Другий спосіб полягає в тому, щоб спочатку виявити суб'єктивність у реченні, щоб відокремити тексти з наявною думкою від текстів без думок, а потім класифікувати ці суб'єктивні тексти за допомогою однієї з двох позначок (позитивної чи негативної).

Завдання аналізу на рівні речення полягає в тому, що кожне окреме речення семантично та синтаксично пов'язане з іншими частинами тексту. Тому для виконання цього завдання потрібна як локальна, так і глобальна контекстна інформація. Янг і Карді [11] аналізують огляд продуктів на рівні речення та успішно вирішують цю проблему.

- Рівень аспекту/ознак /сутності

На відміну від аналізу на рівні документа або речення, аналіз на рівні аспектів досліджує, що подобається або, що ненавидить власник у цілі. Завдання такого дрібнозернистого аналізу поділяються на три частини [12]:

(1) виділення ознак цілі;

(2) визначення полярності за ознаками;

(3) підсумовування загальної оцінки.

Аналіз настроїв на аспектному рівні є одним із найскладніших завдань порівняно з іншими рівнями аналізу.

Крім того, дослідження можна проводити також на рівні фраз [13], речення [14] або на рівні слів [15].

У нашому проєкті ми розглядаємо завдання класифікації тональності лише на рівні документа, де огляд розглядається як ціла інформаційна одиниця.

1.1.3. Завдання

Більшість завдань з сентимент-аналізу визначається сентимент-компонентою, якої він стосується. За допомогою сучасних технологій дослідники проводили широку роботу, починаючи від виявлення власника/цілі, класифікації настроїв, виділення аспектів, виявлення спаму тощо.

У нашому проєкті ми зосередимося на класифікації настроїв на рівні документа, що підходить найбільше для нашого навчального корпусу даних.

1.2. Етапи процесу сентимент-аналізу

Процес сентимент-аналізу включає в себе 4 кроки – це збір даних, попередня обробка, класифікація та визначення сентименту та представлення результатів [16], як показано на рис. 1.1.

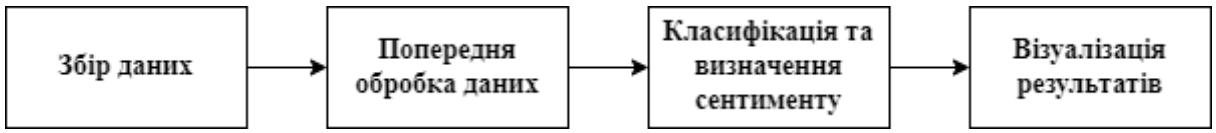


Рис. 1.1: *Етапи процесу аналізу настроїв*

1.2.1. Збір даних

Збір даних є першим кроком у аналізі настроїв. Збір даних зазвичай проводиться із таких джерел, як Twitter, Facebook, блоги та комерційні веб-сайти, на кшталт amazon.com та alibaba.com тощо.

У даному проєкті ми використовували навчальний набір даних від Amazon, який містить відгуки на товари з 25 різних категорій і вважається одним з найкращих корпусів для створення моделей по сентимент-аналізу.

1.2.2. Попередня обробка

Вважається, що 80% світових даних є неструктурованими [17]. Таким чином, отримання інформації з неструктурованих даних є важливою частиною аналізу даних.

Попередня обробка тексту (документів) – це операція, яка перетворює вхідний документ у інший вихідний формат. Основна мета таких операцій, як правило, полягає в тому, щоб спростити і нормалізувати текст, щоб усунути неоднозначність (з точки зору моделі). Ці види операцій зазвичай є простими операціями, заснованими на правилах, які вилучають елементи. Однак є ще один тип операцій попередньої обробки, які додають ознаки, тому документи стають багатшими. Ці операції зазвичай використовують певний лінгвістичний алгоритм, і дуже часто додається зовнішня модель або набір даних. Нижче ми обговоримо етапи попередньої обробки, які були використані в нашому проєкті.

Нормалізація рядків

Цей крок включає всі звичайні операції на основі регулярних виразів, наприклад:

- перетворення в нижній регістр;
- видалення чи перетворення в слова;
- видалення розділових знаків;
- видалення пробілів;
- видалення символів не пов’язаних з алфавітом вхідної мови.

Це важливий крок, оскільки за допомогою цього ми робимо документи більш доступними для токенізації, усуваючи багато *«зашумлення»*.

Токенізація

Під час токенізації документ розбивається на слова, тому функція повертає список слів. Звісно, взяття окремих слів, а не речень, руйнує зв'язки між словами. Однак це поширений метод, який використовується для аналізу великих наборів текстових даних. Комп'ютерам ефективно та зручно аналізувати текстові дані, перевіряючи, які слова з'являються в огляді та скільки разів ці слова зустрічаються, і цього достатньо для отримання детальних результатів. Може здатися, що це тривіальний крок, але можуть виникнути деякі неоднозначні випадки. Наприклад,

- Що робити з дефісами? Наприклад *«він будь-де»* – має 2 токени чи 3?
- Існують сутності, які мають бути одним токеном, але за допомогою простих правил їх можна розділити. IP-номери, назви моделей автомобілів, номери телефонів тощо. Розпізнавання об'єктів завжди є проблемою, пов'язаною з предметною областю.
- Це також може бути проблемою, пов'язаною з мовою. Наприклад, німецька мова використовує багато складних іменників, наприклад *Rechtsschutzversicherungsgesellschaften*. Виходом тут може стати стемінг.

Звичайною практикою зазвичай є *«видалення URL-адрес, видалення стоп-слів, видалення цифр, повернення слів, які містять повторювані літери, до їх початкової форми, і розширення абревіатур на оригінальне слово»* [18]. Такий підхід був використаний для токенізації в нашому проєкті.

Видалення стоп-слів

Стоп-слово є дуже поширеним терміном, який передає дуже мало значення. Відсіювання цих слів зменшує *«зашумлення»* в текстах і допомагає зробити документи більш конкретними.

Стоп-слова залежать від мови та предметної області; перелік останніх зазвичай складається вручну відповідно до проблеми. Прикладами стоп-слів є *«попід»*, *«та»*, *«від»*, *«аби»* тощо. Під час роботи з проблемами класифікації фільтрація стоп-слів зазвичай є хорошою стратегією.

Стемінг / Лематизація

Стемінг – це процес скорочення слова до основи шляхом відкидання допоміжних частин, таких як закінчення чи суфікси, зберігається лише корінь слова. Результати стемінгу іноді дуже схожі з процесом визначення кореня слова, але його алгоритми засновані на інших принципах. Тому слово після обробки алгоритмом стематизації може відрізнятись від морфологічного кореня слова.

Наприклад:

бігом, бігаю, бігати → біг

швидко, швидкий, швидкі → швидк

Лематизація – це подібний процес, коли слова зводяться до леми, яка є словниковою формою слова.

Наприклад:

іменник [*плечима – плече*]

дієслово [*ходили – ходити*]

прикметник [*смішним – смішний*]

Тут потрібно додати дві речі. По-перше, в англійській мові це не дуже важливий крок, оскільки англійська мова має відносно невеликі тенденції до аглютинації або відмінювання, тому словникова форма дуже часто використовується як є. Однак українська мова містить в собі значну словомодифікацію і тут це дуже важливий крок.

По-друге, лематизація двох різних слів може звести їх до однієї базової форми, що *«зменшить високу розмірність простору ознак у класифікації тексту»* [19]. Це може бути корисним, але також може бути проблематичним,

коли різні форми передають важливі характеристики для класифікації документів.

1.2.3. Класифікація настроїв

У цьому підрозділі ми розглянемо найпопулярніші ресурси та методології, що використовуються у класифікації настроїв. Загалом методи можна розділити на три класи: методи на основі лексики, на основі машинного навчання та на основі правил.

Методи на основі лексики

- **Лексикон настроїв**

Лексика сентиментів відноситься до списку слів або фраз, які передають позитивну або негативну інформацію про полярність. Лексикон є дуже важливим ресурсом для аналізу настроїв. Він надає інформацію про настрої щодо найменшої мовної одиниці. Навіть методи, засновані на машинному навчанні, можуть покладатися на лексикон настроїв при розробці ознак. Правильне використання добре розробленого лексикону покращить роботу системи аналізу настроїв. У цій частині ми представимо найпопулярніші лексикони, які використовуються як у промисловості, так і в наукових колах.

Лексикон суб'єктивності MPQA (MPQA subjective lexicon) [13] є частиною MPQA Opinion Corpus. Лексикон доступний на умовах Ліцензії GNU. Кожен запис представляє слово та його довжину, частину мови та полярність. Він надає дуже вичерпну кількість інформації, яка має значення для різних галузей дослідження.

SentiWordNet [20] додає реальні оцінки настроїв до кожного синсету WordNet, щоб позначити його полярність настроїв (позитивну, негативну та об’єктивну). Крім частини мови також включається контекстна інформація. Однією з переваг SentiWordNet є те, що він використовує семантичний ресурс для покращення структури лексикону. Ще одна перевага полягає в тому, що він присвоює як позитивні, так і негативні оцінки одному слову.

VADER Sentiment Lexicon [21] – це вичерпний список «золотих стандартів» слів-настроїв, які особливо застосовуються до мікроблогів та інших текстових даних соціальних мереж. Забезпечуючи як полярність, так і інтенсивність, VADER перевірено експертами. Окрім звичайних слів у словнику, він також дає інформацію про емотікони, сленг (*«nah»*, *«meh»* тощо) та акроніми (*«LOL»*, *«LMAO»* тощо).

Лексикон настроїв від Бін Лю [12] є одним із найпопулярніших лексиконів сентиментів для англійської мови. Він містить 2003 позитивних і 4783 негативних слів. Лексикон відмінно справляється з практичними завданнями, оскільки містить орфографічні помилки, сленгові та веб-мовні варіанти записів.

Окрім згаданих лексиконів, дослідники, як правило, створюють індивідуальні лексикони та адаптують їх відповідно до своїх потреб. Відомо два типи підходів: на основі словника та на основі корпусу.

Підходи на основі словників використовують лексичні бази даних, такі як WordNet, для розширення створеного вручну набору початкових даних. Автоматичне розширення досліджуватиме попарні відношення слів і створюватиме словник належного розміру. Перша робота такого типу використана в [12].

Незважаючи на те, що підхід на основі словника може генерувати велику кількість слів для настроїв, ці слова зазвичай не залежать від контексту та предметної області. Корпусні підходи зазвичай можуть вирішити такі проблеми.

У роботі [22] використовуються лінгвістичні зв'язки для визначення полярності прикметників. Основою цієї роботи є «узгодженість настроїв» природної мови, коли люди схильні використовувати «і» для поєднання слів із аналогічною семантичною орієнтацією, наприклад. *«красивий і розумний»* – це доволі стандартна фраза, у той час як *«красивий і огидний»* ми навряд почуємо у реальному спілкуванні. Розширенням цього методу є [23]. У цій роботі автор

досліджує узгодженість міжреченевих і внутрішньоречиневих настроїв. Це дослідження виявилося корисним у створенні слів, залежних від предметної області.

- **Алгоритми класифікації на основі лексиконів**

У методах, заснованих на знаннях, які також називаються класифікацією настроїв на основі лексиконів, метою є побудова або використання існуючих лексиконів настроїв із зазначеними мітками настроїв для слів або фраз у тексті. Настрій документа визначається домінуючими компонентами (словами чи фразами). Базові техніки включають мажоритарне голосування, оцінку документа з граничним значенням і простий підрахунок слів [24].

Цей підхід не вимагає ніякої підготовки (крім формування лексикону, якщо потрібно). Однак для отримання знань зі слів, які не завжди доступні, потрібні потужні лінгвістичні ресурси. Ху і Лю [12] створили лексикон, використовуючи лише WordNet і список позначених початкових прикметників. Їх метод витягує й автоматично позначає синоніми (одна полярність) та антоніми (протилежна полярність). Цей процес дозволяє списку перетворитися на лексикон. Недоліком цього підходу є те, що він застосовний лише в мовах, які доступні у WordNet.

У будь-якому випадку, метод, заснований на знаннях, може бути складним через «шум» у текстових даних, тоді як створення правил вручну для поєднання інформації про слова, отримані з лексиконів настроїв, вимагає часу та зусиль.

Метод вилучення лексикону називається *«frequentiment»* і доведено [24], що він у 3-5 разів швидше, ніж навчання під керівництвом. Хоча це дуже інформативно та багатообіцяюче, подібних відомих робіт для перевірки його ефективності в текстах англійською та українською мовою не проводилося.

Через відсутність існуючих алгоритмів класифікації на основі лексиконів для українською мови, а також через складність та тривалість

розробки такого методу вручну – в нашому проєкті методи на основі лексики не використовувались.

Методи машинного навчання

Класифікація настроїв за своєю природою є різновидом завдання двосторонньої категоризації тексту. Категоризація тексту зазвичай класифікує дані на кілька попередньо визначених категорій. Це добре вивчена сфера з дуже зрілими рішеннями та застосуваннями. Більшість досліджень як з категоризації тексту, так і з аналізу настроїв належать до методології на основі машинного навчання. У цьому розділі ми коротко розглянемо як контрольовані, так і неконтрольовані методи, а також їх комбінацію.

- **Методи навчання під керівництвом**

Модель Першою роботою з використанням машинного навчання для аналізу настроїв є [25]. Експериментовані в цій роботі моделі були широко використані пізніше, а саме наївний баєсів класифікатор [26], максимальна ентропія і метод опорних векторів [27, 28].

Особливість (Ознака) З часів фундаментальної роботи Панга [25] все більше алгоритмів та особливостей активно розроблялися та застосовувалися в аналізі настроїв. Ці особливості включали частоту термінів в уніграмах і n-грамах, слова сентименту, правила, положення слова, міри довжини, а також інші лінгвістичні особливості [6].

У більшості випадків контрольоване машинне навчання є правильним підходом до аналізу настроїв. Основним завданням тут є побудова класифікатора. Найбільш використовуваними контрольованими алгоритмами є метод опорних векторів, наївний баєсів класифікатор, логістична регресія, випадковий ліс та метод k-найближчих сусідів. Було показано, що контрольовані техніки перевершують техніку без нагляду за продуктивністю [25]. Контрольовані методи можуть використовуватися, як самостійно, так і в комбінації.

Самий такий метод навчання було обрано в нашому проєкті, а всі алгоритми згадані в абзаці вище було порівняно та реалізовано при розробці. Більш детально про це буде описано в розділах 2 та 3.

- **Методи навчання без нагляду**

У техніці без нагляду класифікація здійснюється за допомогою функції, яка порівнює ознаки тексту з лексиконами, так званих, дискримінаційних слів (англ. *discriminatory-word*), полярність яких визначається перед їх використанням. Наприклад, починаючи з позитивних і негативних лексиконів слів, їх можна шукати в тексті, відповідно до сентименту який шукається, і занотовувати їх кількість. Тоді, якщо в документі більше позитивних лексиконів, він позитивний, інакше негативний.

Дещо інший підхід застосовує Терні [10] який використовував просту неконтрольовану техніку для класифікації оглядів як рекомендованих (палець вгору) або нерекомендованих (палець вниз) на основі семантичної інформації фраз, що містять прикметник або прислівник. Він обчислює семантичну орієнтацію фрази за взаємною інформацією фрази зі словом «відмінно» мінус взаємна інформація тієї ж фрази зі словом «погано». З індивідуальної семантичної орієнтації фраз обчислюється середня семантична спрямованість огляду. Огляд рекомендується, якщо середня семантична спрямованість позитивна, не рекомендується в іншому випадку.

- **Комбіновані прийоми**

Є також підходи, які використовують комбінацію інших підходів. Один з таких підходів застосовується у роботі [29]. Вони починають з двох лексиконів слів та немаркованих даних. За допомогою двох дискримінаційних лексиконів слів (негативного та позитивного) вони створюють псевдодокументи, що містять усі слова вибраного лексикону. Після цього вони обчислюють косинус подібності між цими псевдодокументами та немаркованими документами. На основі косинус-подібності документу

присвоюється позитивний або негативний настрій. Потім вони використовують їх для навчання наївного баєсового класифікатора.

Існують також інші типи комбінованих підходів, які доповнюють один одного в тому сенсі, що різні класифікатори використовуються таким чином, що один класифікатор сприяє роботі іншого [30].

Методи, засновані на правилах

Перші системи автоматичної категоризації тексту значною мірою покладалися на методи інженерії знань [31], де застосовувався набір створених людиною логічних правил, однак побудова такої експертної системи зазвичай є достатньо трудомістким, тривалим та дорогим процесом.

Вивчення аналізу настроїв з'являється після того, як категоризація тексту стала майже «вирішеною проблемою». Тому в більшості досліджень використовується методологія, яка базується на машинному навчанні. Нам відомо дуже мало заснованих на чисто правилах методів чи систем. Більшість правил включені до системи на основі лексики для підвищення продуктивності, як, наприклад, у VADER [21], який ми вже згадували раніше.

Концептуально більшість методів, заснованих на лексиці, можна розглядати як «неконтрольовані» або «напівконтрольовані» методи навчання.

У цьому дослідженні ми зосередилися на методах навчання під керівництвом. Ми розглянемо найуспішніші моделі та архітектури, ефективність яких була доведена у літературі.

1.2.4. Візуалізація результатів

Основна мета аналізу величезної кількості даних полягає в тому, щоб перетворити неструктурований текст в корисну інформацію, а потім відобразити її за допомогою діаграм, графіків, малюнків тощо. У нашому проєкті для цього буде розроблено спеціальний веб-сервіс.

Ці візуальні звіти дійсно важливі, тому що саме через них ви бачите детальні результати на основі аспектів. Це дає вам уявлення про те, які сфери потребують вашої уваги більше, ніж інші.

1.3. Складність та відкриті проблеми сентимент-аналізу

Аналіз емоцій та настроїв – це наукове поле з безліччю неперевернутих каменів і можливостей. Через різноманітну природу людських емоцій все ще існує багато проблем та можливостей для розробки унікальних систем або вдосконалення та збагачення існуючих систем ефективними модифікаціями та можливостями.

Деякі основні проблеми, а також можливі напрямки в майбутньому для дослідження виявлення та аналізу настроїв із текстів наведено нижче.

1.3.1. Виявлення сарказму

Виявлення сарказму з тексту є дуже складним завданням, і максимальна точність, досягнута на даний момент, не дуже велика. Правильне виявлення сарказму залежить від правильного використання структури речення, виявлення точних емоцій за реченням, розуміння контексту речення та багатьох інших параметрів.

Наприклад, *«Гарний парфум. Скільки ти в ньому маринувався»* – це сарказм, який важко виявити для існуючої автоматичної системи виявлення сарказму.

1.3.2. Визначення стійкості думки

Думки мають різні сильні сторони. Деякі з них дуже сильні: *«Цей турнікет – сміття»*, і деякі з них слабкі: *«Я думаю, що цей бронік в порядку»*. Хоча можна створити початковий список слів слабкої або сильної думки залежно від потреби програми, це все одно неможливо для комп'ютерів, коли сила думки змішується з позицією цієї думки і повністю змінює полярність документа.

1.3.3. Виявлення множинних настроїв

У більшості робіт з виявлення настроїв дослідники зосереджуються на первинному настрої в тексті. Речення, що виражають декілька настроїв, або відкидаються, або позначаються першою емоцією, яка була виявлена.

Наприклад, якщо людина пише: *«Сценарій був поверхневим, артисти зіграли жахливо, якість звуку така собі, але мені сподобалося»* або *«Це робить мене щасливим і сумним водночас»*, тоді система повинна мати можливість розпізнавати обидві емоції за часовим та/або просторовим виміром.

1.3.4. Виявлення мережі настроїв на основі тексту

Існуючі дослідницькі роботи, засновані на текстових настроях у соціальних мережах, здебільшого зосереджені на публікації користувача. Настрої коментарів або відповідей на ці дописи можуть дати нам уявлення про схожість і відмінність настроїв користувачів у мережі. Це можна використовувати для виявлення цілих мереж настроїв.

1.4. Огляд існуючих готових рішень та їх обмеження

В даний час існує досить велика кількість готових програм, які певним чином виконують аналіз настроїв. Розглянемо деякі з них.

1.4.1. TweetViz

TweetViz – це веб-інструмент, який отримує твіти, що містять певні ключові слова, надані користувачем. Він розташовує їх уздовж двох осей: одна описує приємне та неприємне, а інша описує, наскільки сильні почуття. Також сервіс пропонує 8 різних способів представлення згаданих твітів, у тому числі представлення їх відповідно до їх настрою чи шкали часу, або у вигляді хмари тегів.

Розраховується оцінка загальної емоції, що міститься в твіті, і кожен твіт відображається у вигляді кола, розташованого за настроєм. Як ми бачимо на рис. 1.2 неприємні твіти пофарбовані в синій колір зліва, а приємні – в зелений справа.

Щоб оцінити настрої твіту, використовується словник настроїв. У кожному твіті здійснюється пошук конкретних слів, які містяться в словнику. Потім рейтинги задоволення слів об'єднуються, щоб дати загальний рейтинг всього твіту.

Це не враховує сарказм, і саме тому твіти на кшталт *«Мені подобається, як заряд моєї батареї розряджається з 40% до 10% за одну годину»*, класифікуються як «щасливий». Крім того, сервіс часто не враховує різні аспекти твіту.

Посилання: https://www.csc2.ncsu.edu/faculty/healey/tweet_viz/tweet_app/

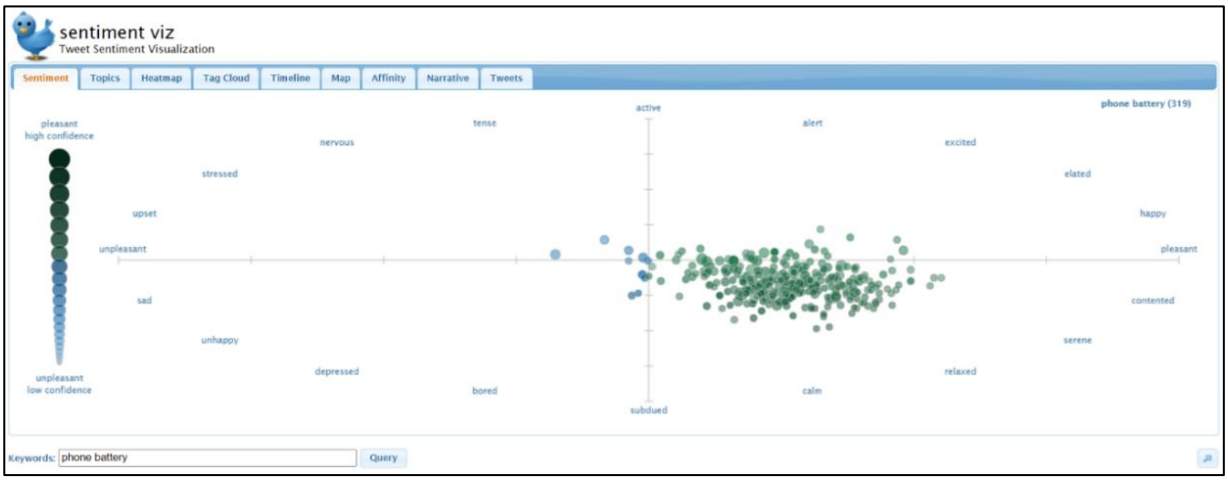


Рис. 1.2: *Приклад роботи TweetViz*

1.4.2. Social Mention

Social Mention – це веб-платформа, яка об'єднує вміст із багатьох різних джерел, як-от Facebook, Twitter, Google, YouTube тощо. Вона намагається виміряти, що люди думають про певну тему в Інтернеті. Як ми бачимо на рис. 1.3, тут також докладено зусиль, щоб класифікувати всі отримані документи на 3 категорії настроїв (позитивні, нейтральні, негативні). Незалежно від теми запиту, здається, майже завжди більше половини результатів класифікуються як нейтральні і немає доступної інформації про те, як саме обчислюється ці настрої.

Посилання: http://socialmention.com/

1.4.1. Stanford NLP

Stanford NLP – відкрита демо-модель Стенфордського університету, що дозволяє користувачам отримувати лінгвістичні анотації для тексту, включаючи межі лексем і речень, частини мови, іменовані об'єкти, числові та часові значення, аналіз настрою та інше (рис. 1.4).

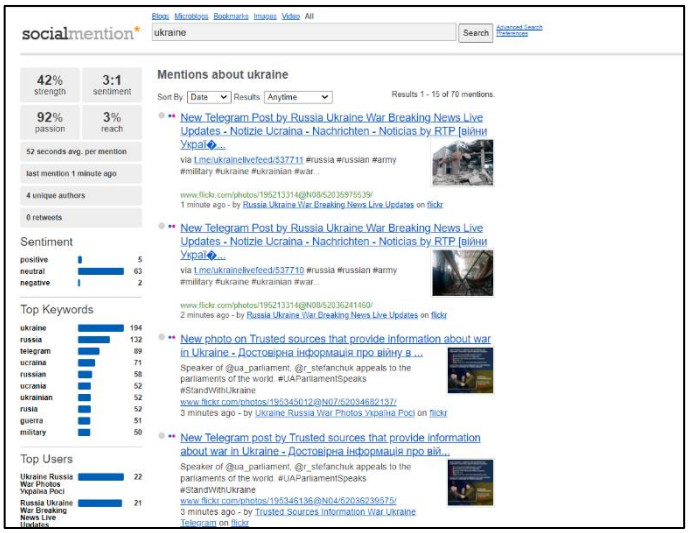


Рис. 1.3: *Приклад роботи Social Mention*

Робота системи полягає в застосуванні рекурсивних нейронних мереж. Зараз CoreNLP підтримує 8 мов: арабську, китайську, англійську, французьку, німецьку, угорську, італійську та іспанську.

Посилання: https://corenlp.run/

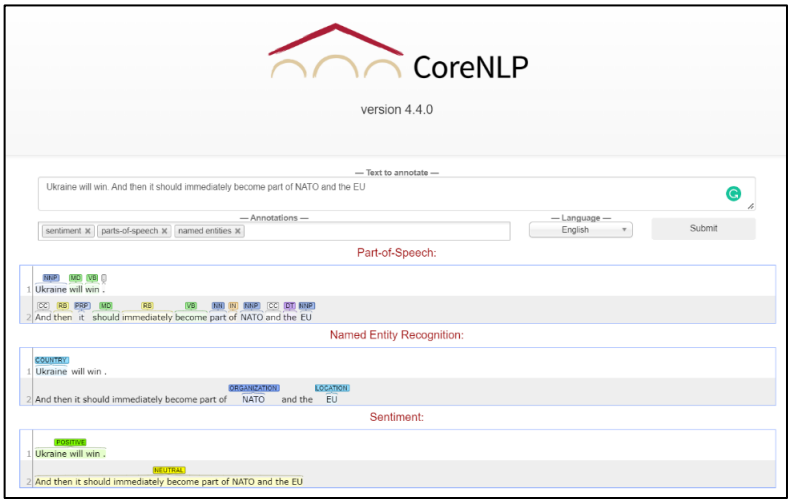


Рис. 1.4: *Приклад роботи Stanford NLP*

1.4.2. MonkeyLearn

MonkeyLearn надає простий графічний інтерфейс, де користувачі можуть створювати налаштовану класифікацію тексту, навчаючи моделі машинного навчання, такі як аналіз настроїв, виявлення тем, виділення ключових слів тощо. MonkeyLearn можна інтегрувати з сотнями інших програм завдяки прямій інтеграції та відкритому API.

На рис. 1.5 показано приклад роботи аналізатору настроїв.

Він також надає своїм клієнтам інструмент «хмара слів», який повідомляє, які слова найчастіше використовуються в кожному тегу категоризації. Для компаній це може бути корисно тим, що у випадку коли буде помічено, що один продукт постійно вказано під негативним тегом категоризації, це означає, що з цим продуктом є проблема, яке викликає незадоволення у клієнтів і з цим треба щось робити.

Посилання: https://monkeylearn.com/sentiment-analysis-online/

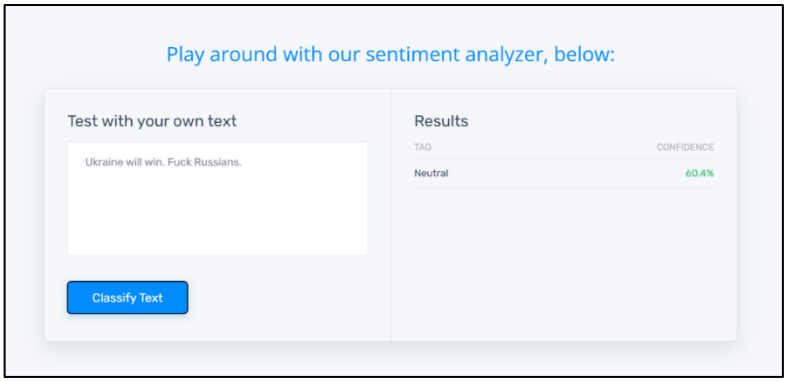


Рис. 1.5: *Приклад роботи MonkeyLearn*

1.5. Порівняння готових рішень

Табл. 1.1: *Порівняння готових рішень*

Сервіс / Особливість	TweetViz	Social Mention	Stanford NLP	Monkey Learn
Аналіз за ключовими словами	+	+	-	-
Аналіз довільного тексту	-	-	+	+
Аналіз почуттів	+	-	-	-
Можливість аналізувати тексти будь-якої тематики	+	+	+	+
Доступно декілька мов для аналізу	-	-	+	+
Висока швидкість аналізу	-	+	+	+
Наявний опис алгоритму сентимент-аналізу сервісу	+	-	+	+
Наявність відкритого API	-	-	+	+
Безкоштовні послуги	+	+	+	-

ВИСНОВОК ДО ПЕРШОГО РОЗДІЛУ

У цьому розділі ми надали широкий огляд області аналізу настроїв і розглянули попередні дослідження, пов’язані з нашим проєктом. Ми надали огляд процесу, як відбувається сентимент-аналіз, включаючи етапи збору даних, попередньої обробки, класифікації та візуалізації результатів.

Ми детально розглянули основні підходи до класифікації настроїв, починаючи від підходу, заснованого на лексиці, і закінчуючи підходом, заснованим на навчанні.

Крім того, ми обговорили найчастіші складності, пов'язані з аналізом настрою та виклики, які потребують додаткових досліджень.

Наприкінці розділу ми розглянули найпопулярніші готові рішення та проблеми, з якими вони стикаються. Головні проблеми включають: відсутність підтримки української мови, проблеми аналізу довільного тексту будь-якого розміру на різну тематику, складний інтерфейс та застарілий дизайн.

В процесі огляду готових додатків ми не змогли знайти сервіс, який би допомагав викладачам оцінювати зворотний зв’язок від студентів в режимі реального часу. Тому розробка такого сервісу буде надзвичайно потрібною.

У наступному розділі ми обговоримо класифікатори машинного навчання та моделі глибинного навчання, що були використані в нашому проєкті, а також надамо опис характеристикам їх оцінювання.

РОЗДІЛ 2
ОГЛЯД МОДЕЛЕЙ ДЛЯ КЛАСИФІКАЦІЇ НАСТРОЇВ ТА МЕТОДИ ЇХ ОЦІНЮВАННЯ

2.1. Класифікатори машинного навчання

Існує безліч доступних класифікаторів на основі машинного навчання, які можна використовувати для аналізу настроїв. Деякі з найпопулярніших алгоритмів аналізу даних, які використовуються для класифікації і дають найкращу точність – це наївний баєсів класифікатор, k-найближчих сусідів, випадковий ліс, логістична регресія та метод опорних векторів.

Далі у цьому розділі ми розглянемо кожен з цих контрольованих алгоритмів машинного навчання більш детально, а у 3 розділі ми реалізуємо та зробимо порівняльний аналіз кожного з цих класифікаторів для аналізу настроїв на нашому наборі даних для англійської та української мови.

2.1.1. Наївний баєсів класифікатор

Наївний класифікатор Баєса є одним із найпростіших і найбільш часто використовуваних алгоритмів машинного навчання для класифікації тексту. Це ймовірнісний класифікатор, заснований на добре відомій теоремі Баєса.

Даний класифікатор може добре працювати з будь-яким набором даних, який можна відобразити у вигляді списків ознак. Ознакою може бути будь-що, що може бути присутнім або відсутнім. У випадку документів ознаками можуть бути слова.

Він передбачає, що всі змінні є незалежними одна від одної, а це означає, що наявність чи відсутність певної ознаки абсолютно не пов’язане з наявністю будь-якої іншої ознаки, тому його і називають наївним.

Приклад: якщо фрукт визначається за кольором, формою та смаком, то помаранчевий, кулястий і солодкий фрукт розпізнається як апельсин. Таким чином, кожна ознака окремо сприяє визначенню того, що це апельсин, не залежно одне від одного.

Багато проєктів використовували наївний баєсів метод як перший підхід до класифікації тексту через його простоту. Такі інструменти, як WEKA, MALLET і NLTK, включають даний підхід як один із своїх класифікаторів машинного навчання для оцінки досліджень.

У випадку великих наборів даних дуже легко створити наївну модель Баєса без будь-якої ітеративної оцінки параметрів, яка також працює дуже добре в порівнянні з іншими передовими методами класифікації.

Теорему Баєса можна обчислити, виконуючи апостеріорну ймовірність. Як ми вже казали, наївний баєсів класифікатор припускає, що вплив значення предикату (X) на даний клас (Y) не залежить від значень інших предикатів. Це також відомо як умовна незалежність.

$$P(X|Y) = \frac{P(Y|X) * P(X)}{P(Y)} \quad (2.1)$$

P(X|Y) — апостеріорна ймовірність: ймовірність гіпотези X щодо спостережуваної події Y.

P(Y|X) – ймовірність імовірності: ймовірність доказу, якщо вірогідність гіпотези є істинною.

P(X) — попередня ймовірність: ймовірність висунення гіпотези до спостереження за доказами.

P(Y) — гранична ймовірність: ймовірність доказів.

Коли X містить n атрибутів, умовно незалежних один від одного, заданих Y, ми маємо:

$$P(X_1 \dots X_n|Y) = \prod_{i=1}^{n} P(X_i|Y) \quad (2.2)$$

Враховуючи, що Y є будь-якою змінною з дискретним значенням, а ознаки $X_1 \dots X_n$ є будь-якими дискретними або реальними змінними, наша мета полягає в тому, щоб навчити класифікатор, який буде виводити розподіл ймовірності за можливими значеннями Y для кожного нового екземпляра X, що ми просимо його класифікувати. Щоб обчислити ймовірність того, що Y прийме будь-яке з k-го можливих значень, ми використовуємо таке рівняння:

$$P(Y = y_k | X_1 \dots X_n) = \frac{P(Y = y_k) \prod_i P(X_i | Y = y_k)}{\sum_j P(Y = y_j) \prod_i P(X_i | Y = y_j)} \quad (2.3)$$

З огляду на новий екземпляр $x_{new} = \langle X_1 \dots X_n \rangle$, рівняння 2.3 показує, як обчислити ймовірність того, що Y набуде будь-якого заданого значення, враховуючи спостережувані значення атрибутів X_{new} і розподіли P(Y) і $P(X_i|Y)$ оцінюється за даними навчання.

Щоб вибрати найбільш імовірне значення Y, ми використовуємо таке правило класифікації:

$$Y \leftarrow \arg\max_{y_k} P(Y = y_k) \prod_i P(X_i | Y = y_k) \quad (2.4)$$

2.1.2. Метод опорних векторів

Це метод навчання під наглядом, у якому створюється функція відображення (англ. *mapping function*) з доступного набору навчальних даних. Метод опорних векторів (англ. *SVM, support vector machines*) широко застосовується для задачі класифікації та регресії, яка класифікує як лінійну, так і нелінійну [32].

Базовий метод опорних векторів приймає набір вхідних даних і прогнозує для кожного заданого входу до якого з двох можливих класів він належить, класи в нашому випадку можуть бути позитивними або негативними.

Він працює на основі концепції пошуку оптимальної гіперплощини, яка відокремлює всі точки даних одного класу від даних іншого класу.

Точки, найближчі до гіперплощини, є опорним вектором. Великою перевагою методу опорних векторів є те, що при його використанні важче опинитися в ситуаціях перенавчання.

Рис. 2.1 показує лінійну класифікацію, при цьому зелені точки вказують точки даних типу 1, і червоні вказують точки даних типу -1.

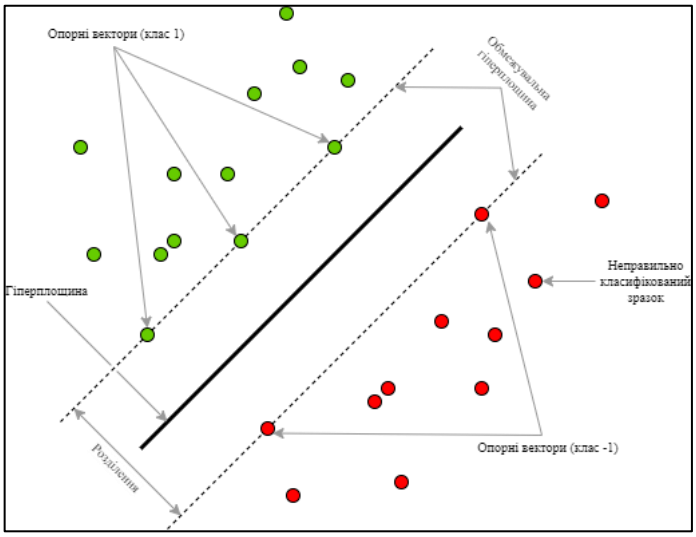


Рис. 2.1: *Гіперплощина ОВМ (опорно-векторних машин)*

Згідно [33], метод опорних векторів є універсальним навчальним засобом, який можна використовувати також для вивчення поліноміальних класифікаторів, і він має здатність навчатися незалежно від розмірності простору ознак. Даний метод дуже корисний у вирішенні питань, пов'язаних із класифікацією текстів шляхом лінійного їх розділення, як це пропонує [33].

Загалом задача даного класифікатора спрямована на знаходження гіперплощини з великим розділенням ($\omega^T * x + b$), яка розділяє два значення класу $y \in \{+1, -1\}$ відповідно до простору ознак, представленого x. Якщо дані лінійно розділені, оптимальною є гіперплощина, яка максимізує розділення між позитивними та негативними спостереженнями в наборі навчальних даних, утвореному N спостереженнями. У загальному випадку завдання вивчення ОВМ з набору даних формалізується у вигляді наступної задачі оптимізації:

$$\min_{\omega,b} \frac{1}{2}\omega^T\omega + C\sum_{i}^{N}\xi_i \tag{2.5}$$

за умови $y_i(\omega^T x_i + b) \geq 1 - \xi_i, \forall i \in \{1, \dots, N\}, \xi_i \geq 0\ \forall i \in \{1, \dots, N\}$

Цільова функція задачі поєднує довжину вектору параметрів і похибки $\sum_i^N \xi_i$. Параметр С називають «параметром регуляризації м'якого розділення», він контролює чутливість ОВМ до можливих викидів.

Також можна змусити ОВМ ефективно знаходити нелінійні шаблони, використовуючи метод ядра. Для цього використовується функція φ(x), яка відображає простір ознак x у просторі високої розмірності. Цей високовимірний простір є гільбертовим простором, а добуток φ(x) · φ(x') можна представити як ядровий метод K(x, x').

Популярним ядровим методом є ядро радіальної базисної функції (англ. *RBF, radial basis function*):

$$K(x, x') = \exp\left(-\frac{\|x - x'\|^2}{2\sigma^2}\right) \tag{2.6}$$

в якому σ (σ > 0) є вільним параметром, який потрібно налаштувати для кожної конкретної задачі. Використовуючи ядра, гіперплощина обчислюється у просторі високої розмірності ($\omega^T * \varphi(x) + b$). Подвійне формулювання ОВМ дозволяє замінити кожен крапковий добуток функцією ядра, як показано в наступному виразі:

$$\max_{\alpha} \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} \alpha_i \alpha_j y_i y_j * K(x_i, x_j) \tag{2.7}$$

за умови $0 \leq \alpha_i \leq C, \forall i \in \{1, \ldots, N\}$:

$$\sum_{i=1}^{N} \alpha_i y_i = 0$$

де параметри $\alpha_i, i \in \{1, \ldots, N\}$ відповідають множникам Лагранжа задачі оптимізації з обмеженнями. Після визначення параметрів α можна класифікувати нове спостереження x_j відповідно до такого виразу:

$$\text{sign}\left(\sum_{i=1}^{N} \alpha_i y_i * K(x_i, x_j) + b\right) \tag{2.8}$$

Розрахунок члена зміщення b змінюється відповідно до алгоритму вирішувача [34]. Зручною властивістю ОВМ є те, що значення α будуть відрізнятися від нуля лише для певного (зазвичай невеликого) числа

прикладів, відомих як опорні вектори. Таким чином, ОВМ потрібно лише оцінити функцію ядра між x_j і опорними векторами.

Одним із недоліків використання методу опорних векторів є те, що він не здатний розрізняти слова, які мають різне значення в різних реченнях, і тому не можна створити конкретні «лексикони на основі предметної області» [33].

Приклад: припустимо, ми бачимо єнота, який також має деякі особливості панди, тож якщо нам потрібна модель, яка може точно визначити, єнот це чи панда, то таку модель можна створити за допомогою алгоритму ОВМ. Спочатку ми потренуємо нашу модель з великою кількістю зображень панд і єнотів, щоб вона могла дізнатися про різні особливості панд і єнотів, а потім випробуємо її з нашою твариною. Отже, оскільки опорний вектор створює розділення рішення між цими двома даними (єнот і панда) і вибирає крайні випадки (вектори підтримки), він побачить крайній випадок єнота і панди. На основі опорних векторів він буде класифікувати тварину як єнота.

2.1.3. K-найближчих сусідів

K-найближчих сусідів (K-НС) – це один із найпростіших алгоритмів машинного навчання, заснований на техніці навчання під керівництвом. Алгоритм K-НС зберігає всі доступні дані та класифікує нову точку даних на основі подібності. Це означає, що коли з'являються нові дані, їх можна легко віднести до певної категорії за допомогою алгоритму K-НС.

Алгоритм K-НС виконує пошук для визначення місцезнаходження найближчих сусідів з метою класифікації n-вимірних елементів відповідно до їхньої схожості з різними n-вимірними елементами.

У сфері машинного навчання аналіз найближчого сусіда реалізується для ідентифікації шаблонів даних, не вимагаючи ідентичного шаблону, що відповідає будь-яким збереженим елементам або шаблонам. N-вимірні елементи, які не є аналогічними, віддалені один від одного, тоді як розмірні елементи, які є аналогічними, близькі один до одного [35].

Алгоритм К-НС можна визначити як підхід до класифікації елементів на основі найближчих навчальних екземплярів, розташованих у просторі ознак. К-НС – це форма відкладеного навчання або навчання на основі прикладів, у якому функція внутрішньо округляється, а всі обчислення відкладаються до класифікації. К-НС вважається основним і найпростішим підходом до класифікації, якщо попереднього уявлення про розподіл даних немає [36].

Алгоритм К-НС можна використовувати як для регресії, так і для класифікації, але в основному він використовується для задач класифікації. К-НС є непараметричним алгоритмом, це означає, що він не робить жодних припущень щодо базових даних.

Його також називають алгоритмом ледачого навчання (англ. *lazy learning*), оскільки він не вчиться з навчального набору відразу, а зберігає набір даних, а під час класифікації і кожному запиту призначає клас, що представляє більшість класів запиту k-найближчих сусідів в наборі навчальних даних.

Основним типом К-НС, якщо K = 1, є принцип найближчого сусіда (НС). У цій техніці кожен елемент має бути класифікований відповідно до суміжних елементів [36]. Рис. 2.2 ілюструє принцип рішення К-НС, коли K = 4 і K = 1 для групи елементів, розподілених на пару класів. Крім того, у К-НС, якщо є нова точка $d = \{d_1, d_2, \dots, d_n\}$, то k спостережень у наборі навчальних даних динамічно вказуються на основі d (який є k найближчими сусідами). Сусіди визначаються мірою дивергенції або відстанню, яку можна розрахувати серед спостережень залежно від автономних змінних. Отже, евклідову відстань між точками $z = (z_1, z_2, \dots, z_n)$, і $d = (d_1, d_2, \dots, d_n)$ можна представити так [35]:

$$\sqrt{(z_1 - d_1)^2 + (z_2 - d_2)^2 + \cdots + (z_n - d_n)^2} \qquad (2.9)$$

Для кожного запитаного n-вимірного елемента (наприклад, елемента з n атрибутами) обчислення евклідових відстаней виконується між запитаним елементом і кожним елементом у наборі навчальних даних, а мітка класу більшості з k найближчого навчального набору даних призначається

запитуваному елементу [35]. Більше того, рис. 2.2 є прикладом класифікації K-НС у разі застосування з наступними чотирма класами нижче $\{L_1, L_2, L_3, L_4\}$ [35].

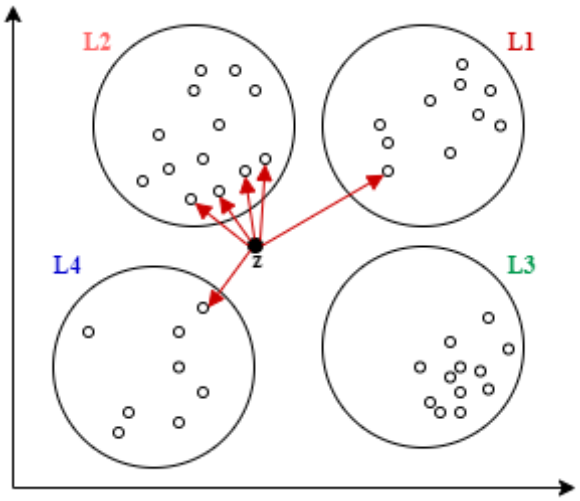


Рис. 2.2: *Класифікація KNN у випадку чотирьох класів* $\{L_1, L_2, L_3, L_4\}$

На рис. 2.2 K = 6 (червоні стрілки) і $z = (z_1, z_2, \ldots, z_n)$. Мета полягає в тому, щоб визначити мітку класу з наступних чотирьох класів $\{L_1, L_2, L_3, L_4\}$ для z (що символізує невідомі дані). Отже, із шести найближчих сусідів один знаходиться в межах L_1, інший — у межах L_4, а чотири — в межах L_2. Отже, z буде віднесено до L_2 на основі більшості голосів, оскільки це переважаючий клас [35].

Приклад: припустимо, у нас є зображення істоти, схожого на єнота і панду, але ми хочемо знати, чи це єнот чи панда. Отже, для цієї ідентифікації ми можемо використовувати алгоритм К-НС, оскільки він працює на мірі подібності. Наша модель К-НС знайде схожі характеристики нового набору даних із зображеннями єнотів та панд і на основі найбільш подібних ознак помістить його в категорію єнотів або панд.

2.1.4. Логістична регресія

Логістична регресія – це один з найпопулярніших алгоритмів машинного навчання, який підпадає під техніку контрольованого навчання. Даний тип регресії обробляє зв'язок між категоріальною залежною змінною та численними незалежними змінними та оцінює ймовірність існування події за

допомогою логістичної кривої, яка поєднується з даними. Логістична регресія має дві моделі: поліноміальну та бінарну.

Бінарна логістична регресія використовується в ситуації, коли незалежна змінна є двійковою і категоричною/безперервною. З іншого боку, поліноміальна логістична регресія використовується, якщо незалежна змінна не є бінарною і містить більше 2 класів (наприклад, позитивний, негативний і нейтральний) [37].

Логістична регресія може бути використана для класифікації спостережень за допомогою різних типів даних і може легко визначити найбільш ефективні змінні, що використовуються для класифікації, оскільки логістична функція, на якій базується модель, забезпечує оцінки, які повинні бути між областю нуля та одиниці. Наступна формула ілюструє логістичну функцію:

$$f(z) = \frac{1}{1 + e^{-z}} \tag{2.10}$$

Логістична регресія дуже схожа на лінійну регресію, за винятком того, як вони використовуються. Лінійна регресія використовується для розв'язування задач регресії, тоді як логістична регресія використовується для вирішення завдань класифікації.

На рис. 2.3 зображено логістичну функцію. Якщо z дорівнює +∞ в правій частині графіка, то це означає, що f(z) дорівнює 1, і навпаки, якщо z дорівнює −∞ в лівій частині графіка, то це означає, що f(z) дорівнює 0. Отже, область визначення знаходиться між $0 \leq f(z) \geq 1$ незалежно від значення z (вхідного), а e символізує експоненціальну функцію [38].

Тому результат повинен бути категоріальним або дискретним значенням. Це може бути або так, або ні, 0 або 1, істина чи хибність тощо, але замість того, щоб давати точне значення як 0 та 1, він дає ймовірні значення, які лежать між 0 та 1 .

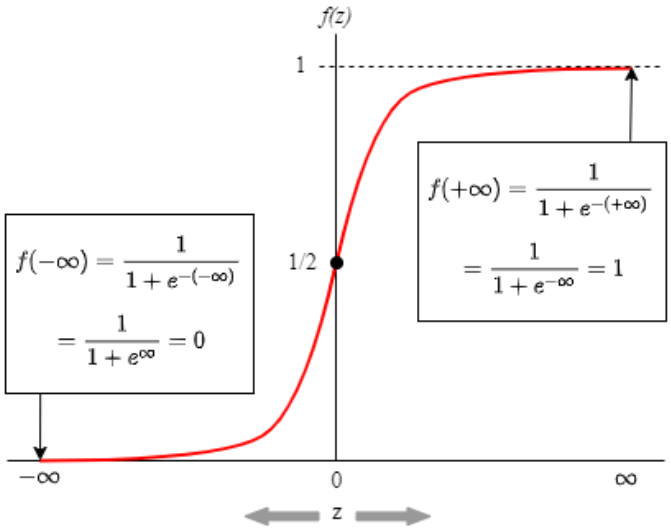


Рис. 2.3: *Логістична функція*

Приклад: наведено набір даних, який містить інформацію про різних користувачів, отриману із соціальних мереж. Існує компанія з виробництва смартфонів, яка нещодавно випустила новий флагман. Тому компанія хотіла перевірити, скільки користувачів із набору даних хоче придбати смартфон.

2.1.5. Випадковий ліс

Випадковий ліс – популярний алгоритм машинного навчання, який також належить до методики навчання з наглядом. Його можна використовувати як для задач класифікації, так і для задач регресії. Він заснований на концепції ансамблевого навчання, що являє собою процес об'єднання кількох класифікаторів для вирішення складної проблеми та покращення продуктивності моделі.

Як випливає з назви, *«випадковий ліс – це класифікатор, що містить ряд дерев рішень для різних підмножин даного набору даних і бере середнє значення, щоб підвищити точність прогнозування цього набору даних»* [39]. Замість того, щоб покладатися на одне дерево рішень, випадковий ліс бере прогноз з кожного дерева на основі більшості голосів передбачень і прогнозує кінцевий результат.

Більша кількість дерев у лісі призводить до більшої точності та запобігає проблемі перенавчання. Через високу точність методу, випадковий ліс,

можливо, є одним з найкращих підходів до навчання, оскільки він конкурує з іншими взірцевими методами, такими як метод опорних векторів [39].

У випадкових лісах кожне дерево будується залежно від вибірки, яка випадково вибирається з вхідних даних, які зазвичай є підвибіркою або бутстреповою вибіркою (англ. *bootstrap sample*) [40].

Навчання кожного дерева рішень здійснюється через цю підгрупу, яка випадковим чином вибирається з набору навчальних даних [39]. Кожне дерево дає «голосування» за рішення щодо кожної тестової вибірки, і остаточна класифікаційна категорія обирається на основі вибору категорії з найбільшою кількістю голосів [39]. Рис. 2.4 ілюструє класифікатор випадкових лісів з k деревами рішень:

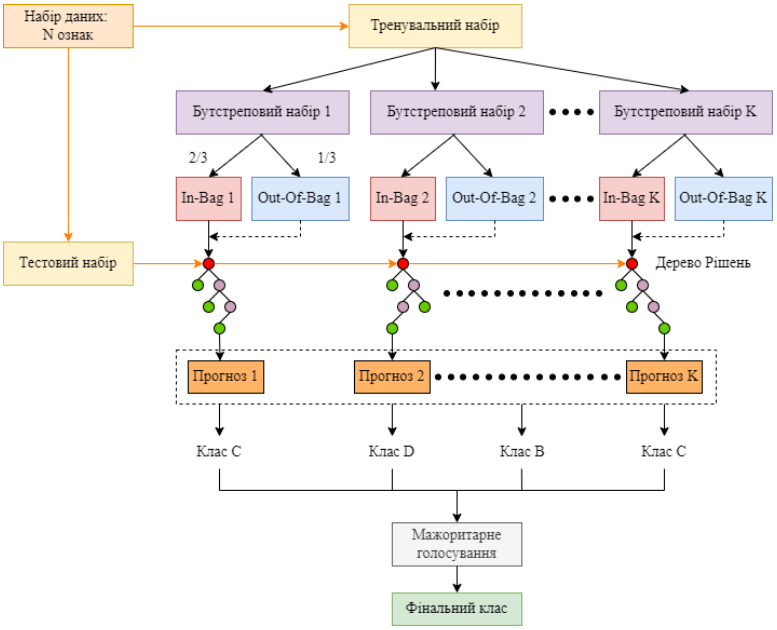


Рис. 2.4: *Структура класифікатора випадкових лісів*

Оскільки випадковий ліс поєднує кілька дерев, щоб передбачити клас набору даних, можливо, що деякі дерева рішень можуть передбачити правильний результат, а інші – ні. Але разом усі дерева передбачають правильний вихід.

Стратегія створення випадкового лісу підсумована таким чином [41]:

1. Створюємо N підгруп із навчальних даних $\{z_1, z_2, \ldots, z_N\}$ шляхом застосування випадкової вибірки із заміною на Z.

2. Щоб виростити будь-яке дерево, застосовуємо алгоритм дерева рішень для кожного навчального набору даних $z_i(1 \leq i \leq N)$, виконуємо випадкову вибірку для підгрупи W з ознакою Q у кожному вузлі ($Q \ll N$), обчислюємо усі розділи (англ. *partitions*) в підпросторі W_i і, нарешті, обираємо оптимальний розділ як ознаку розділення (англ. *partitioning feature*) для створення дочірнього вузла. Повторюємо цю процедуру до тих пір, поки не буде досягнуто критеріїв зупинки, і буде отримано дерево $t_i(Z_i, W_i)$, створене за допомогою навчальних даних Z_i в підпросторі W_i.
3. Ансамблюємо N дерев $\{t_1(Z_1, W_1), t_2(Z_2, W_2), \ldots, t_F(Z_F, W_F)\}$, щоб створити випадковий ліс і застосувати до цих дерев мажоритарне голосування (більшістю), щоб створити рішення щодо класифікації.

Існує кілька основних параметрів, пов'язаних з алгоритмом, наприклад, кількість N дерев для створення випадкового лісу та кількість Q об'єктів, які випадково відбираються для створення дерева рішень. Загалом, дослідники пропонують [41] брати параметр N рівним 100, тоді як параметр Q розраховується за допомогою $[Q = \log_2 N + 1]$. У разі об'ємних даних необхідно приймати великі N та Q.

Приклад: припустимо, що є набір даних, який містить кілька зображень овочів. Цей набір даних надається класифікатору випадкових лісів. Набір даних розбивається на підмножини і надається кожному дереву рішень. Під час фази навчання кожне дерево рішень дає результат прогнозу, а коли виникає нова точка даних, на основі більшості результатів класифікатор випадкового лісу прогнозує остаточне рішення.

2.2. Моделі глибинного навчання

Моделі, представлені в цьому підрозділі, є більш складними і в основному засновані на глибинному навчанні. Властивість глибинного навчання, що робить його відмінним інструментом для класифікації, полягає в тому, що воно вивчає глибинні нейронні мережі як модель класифікації,

тобто нейронні мережі з багатьма шарами, які зазвичай навчаються наскрізь (англ. *end-to-end*). Використання кількох шарів дозволяє моделям перетворювати дані (поступово) у форму, де легко вирішити конкретне завдання класифікації.

У цих моделях кожен шар є функцією, яка діє на вихід попереднього шару, тому ми можемо сказати, що мережа є ланцюгом складених функцій, а також що ланцюжок оптимізовано для виконання конкретного завдання. Тобто, нейронні мережі перетворюють дані через існуючі шари, полегшуючи вирішення їхнього завдання. Ці перетворені версії даних називаються *представленнями*.

Представлені та описані моделі у цьому підрозділі були також використані в нашому проєкті, на додачу до тих, які ми детально розглянули в попередньому підрозділі. Кожну з моделей було також реалізовано та зроблений порівняльний аналіз у розділі 3. Традиційно нейронні мережі поділяють на дві категорії. Рекурентні нейронні мережі (РНМ, англ. *RNN*) і згорткові нейронні мережі (ЗНМ, англ. *CNN*).

Почнемо з РНМ, причина, чому вони називаються рекурентними, (повторюваними), полягає в тому, що вони виконують одне і те ж завдання для кожного елемента вхідної послідовності, а вихідні дані залежать від попередніх обчислень. Теоретично РНМ мають можливість використовувати інформацію в довільно довгих послідовностях, але на практиці вони обмежуються оглядом назад лише кількома кроками [42]. Іншими словами, РНМ мають два джерела входу, а саме теперішнє та недавнє минуле, які разом дають нам можливість визначити, як реагувати на нові дані, як ми це робимо в житті (рис. 2.5 а).

Однак, РНМ досі страждають від проблеми зникання градієнту (англ. *vanishing gradients*) [43]. Проблема зникнення градієнтів виникає при навчанні нейронних мереж за допомогою методів на основі градієнтів, наприклад, при використанні алгоритму зворотного поширення (англ. *backpropagation*) [44].

Метод зворотного поширення використовується в поєднанні з методом оптимізації, таким як градієнтний спуск, і він намагається зменшити похибки між вихідним і бажаним результатом. Зокрема, для кожного навчального прикладу вагові коефіцієнти прихованих шарів змінюються, щоб мінімізувати помилку, обчислену між прогнозом моделі та правильним значенням. Як випливає з назви, ці модифікації здійснюються в зворотному напрямку, тобто від вихідного шару, через кожен прихований шар, до першого прихованого шару. Проблема зникнення градієнтів передбачає втрату чутливості (англ. *sensitivity*) протягом часу, оскільки нові входи перезаписують активації прихованих шарів, викликаючи забуття перших входів [43].

Щоб усунути цю проблему, був представлений новий варіант РНМ під назвою **довга короткочасна пам’ять** (ДКЧП, англ. *LSTM*) [45]. ДКЧП може обирати збереження пам’яті протягом довільних періодів часу, але також забувати, якщо це необхідно (рис. 2.5 б). Беручи до уваги ці зміни, ДКЧП є однією з найбільш використовуваних моделей РНМ в задачах обробки природної мови (англ. *NLP*).

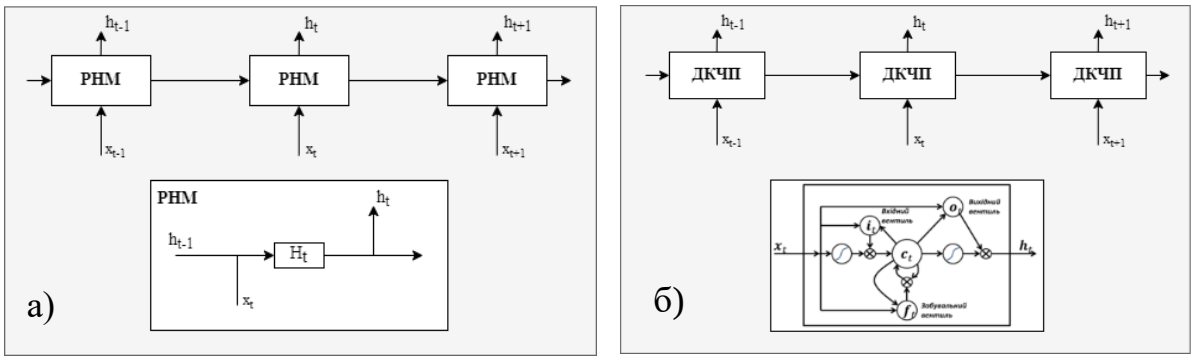


Рис. 2.5: *Приклади а) традиційної архітектури рекурентної нейронної мережі (РНМ) та б) архітектури довгої короткочасної пам'яті (ДКЧП)*

Що стосується згорткових нейронних мереж, вони зазвичай складаються з кількох шарів згорток із застосуванням нелінійних функцій активації до результатів. Якщо бути більш детальним, згортки над вхідним шаром використовуються з метою обчислення вихідних даних.

На відміну від нейронних мереж прямого поширення (англ. *feedforward neural network*), де кожен вхідний нейрон з'єднаний з кожним вихідним нейроном наступного шару, в ЗНМ використання згорток призводить до багатьох локальних з'єднань, де кожна область входу з'єднана з нейроном на виході [46]. Кожен згортковий шар застосовує різні фільтри, а в кінці об’єднує їх результати. Значення цих фільтрів автоматично вивчаються на етапі навчання.

2.2.1. Огляд шарів

У цьому підрозділі представлені всі шари, що складають архітектури моделей глибинних мереж, кожен з яких має власний підрозділ, де зроблено опис відповідної корисності, призначення та функціонування.

Шари вбудовування

Шар вбудовування (англ. *embedding layer*) має на меті перетворити індекси слів у щільні представлення (англ. *dense representations*). У цьому випадку доступні три різні варіанти:

(1) Вивчити вагові коефіцієнти вбудовування з нуля, тобто вкладання слів ініціалізуються випадковими значеннями і вдосконалюються в процесі навчання.

(2) Використовувати попередньо навчені вбудовування, зберігаючи ваги вбудовування статичними.

(3) Використовувати попередньо навчені вкладання, але замість того, щоб підтримувати статичні ваги, вони будуть покращені під час навчального процесу.

Шари виключення

Шар виключення (англ. *dropout layers*) несе відповідальність за те, щоб допомогти моделям запобігти перенавчанню. Як ми вже згадували, глибинні нейронні мережі складаються з кількох нелінійних прихованих шарів, що робить ці моделі дуже виразними. Існування цих типів шарів дозволяє моделям вивчати дуже складні відносини між входами та виходами.

Беручи до уваги обмежені навчальні дані, багато взаємозв’язків будуть результатом зашумлення вибірки (англ. *sampling noise*), тобто деякі зв’язки існуватимуть у навчальному наборі, але не в тестовому наборі. Коли це відбувається, модель погано узагальнює, що призводить до поганих результатів з точки зору ефективності передбачення.

Щоб уникнути проблеми перенавчання, було розроблено багато методів, одним з яких є метод виключення.

По суті, метод виключення змушує нейронну мережу вивчати кілька незалежних представлень одних і тих же даних (виключення нейронів під час фази навчання), з метою зменшення залежності від навчального набору. Таким чином, шар виключення буде встановлювати на нуль певну частину своїх вхідних даних.

Шари ДКЧП

Шар ДКЧП в основному реалізує рекурентну модель ДКЧП, яка є одним із найбільш використовуваних варіантів РНМ через той факт, що цей підхід враховує три додаткові аспекти, які значно покращують продуктивність передбачення.

По-перше, мережа має контроль, щоб вирішувати, коли дозволити вхідному сигналу ввійти в нейрон.

По-друге, це здатність контролювати, коли запам'ятовувати те, що було обчислено на попередніх проміжках часу.

По-третє, цей метод також має здатність контролювати, які частини інформації має виводити система. Ці покращення проілюстровані на рис. 2.5 б).

Щільні шари

Щільний шар (англ. *dense layer*) являє собою звичайний повністю пов'язаний шар. Зокрема, ідея цього шару полягає в тому, щоб з’єднати кожен нейрон мережі з кожним нейроном у сусідніх шарах.

Згорткові шари

Згортковий шар є найбільш релевантним у випадку моделей ЗНМ. Ці шари складаються з набору фільтрів з невеликим рецептивним полем. Під час процесу навчання кожен фільтр згортається на всю глибину вхідного об'єму. Цей процес виконується обчисленням точкового добутку між записами фільтра та входом, щоб створити карту активації кожного фільтра.

Ключовим моментом є те, що мережа вивчає фільтри, які активуються, коли вони бачать певний тип функції в певному просторовому положенні на вході.

Вихід згорткового шару представлений стеком карт активації, створених кожним фільтром. Кожну позицію стека можна розглядати як вихід нейрона, який дивиться на невелику область на вході.

Шари агрегування та згладжування

Шар агрегування (англ. *pooling layer*) зазвичай використовується між послідовними згортковими шарами в моделі ЗНМ. Їх мета полягає в поступовому зменшенні просторового розміру представлення, що також призведе до зменшення параметрів мережі та обчислень. Завдяки цьому, ці шари також є важливим засобом боротьби з проблемою перенавчання.

Найбільш традиційною функцією для виконання цієї операції є максимізаційне агрегування (англ. *max pooling*). Задача полягає в тому, щоб розділити вхідне представлення на набір областей, що не перекриваються, і для кожної з них вибрати максимальне значення.

Таким чином, наприкінці кожен елемент нового представлення є максимальним значенням області у початковому вхідному представленні. Ідея, яка за цим стоїть, полягає в тому, що як тільки функція була виявлена, її точне розташування не дуже важливе в порівнянні з важливістю її розташування щодо інших ознак.

Шар згладжування (англ. *flattening layer*), як видно з назви, згладжує вхід. Точніше, ці шари зводять усі розміри своїх входів в один вимір.

Шари активації

Шар активації (англ. *activation layer*) просто застосовує функцію активації до виходу. Функція активації в нейронних мережах, крім обмеження вихідних даних певним діапазоном, лінійно розбиває нейронну мережу, дозволяючи їй вивчати складніші функції, ніж лінійна регресія.

Нейрон без функції активації еквівалентний нейрону з лінійною функцією активації, наприклад $f(x) = x$. Якщо ці функції не додають ніякої нелінійності, вся мережа еквівалентна одному лінійному нейрону. Тому немає сенсу будувати багатошарову мережу з лінійними функціями активації.

Більше того, враховуючи, що один лінійний нейрон не здатний працювати з нероздільними даними, незалежно від того, наскільки глибока багатошарова мережа, він ніколи не зможе вирішити жодну нелінійну проблему. Беручи це до уваги, роль функцій активації в нейронній мережі полягає у створенні нелінійної межі рішення за допомогою нелінійної комбінації вхідних вагових даних.

2.2.2. Архітектури моделей

Даний підрозділ описує архітектури моделей, кожна з яких має власний підрозділ. В описі кожної моделі, крім її складу (тобто шарів), також наводяться специфікації кожного компонента.

Модель ДКЧП (Стохастичний градієнтний спуск)

Популярним алгоритмом навчання нейронних мереж є алгоритм оптимізації стохастичного градієнтного спуску. Він передбачає використання поточного стану моделі для прогнозування, порівняння прогнозу з очікуваними значеннями та використання різниці як оцінки градієнта помилки. Цей градієнт помилки потім використовується для оновлення ваг моделі, і процес повторюється.

Градієнт помилки є статистичною оцінкою. Чим більше навчальних прикладів буде використано в оцінці, тим точнішою буде ця оцінка і тим більша ймовірність, що ваги мережі будуть скориговані таким чином, щоб

покращити продуктивність моделі. Покращена оцінка градієнта помилки забезпечується ціною того, що потрібно використовувати модель, щоб зробити багато інших передбачень, перш ніж оцінку можна буде обчислити, і, у свою чергу, оновити вагові коефіцієнти.

З іншого боку, використання меншої кількості прикладів призводить до менш точної оцінки градієнта помилки, яка сильно залежить від конкретних використаних навчальних прикладів.

Це призводить до «зашумленої» оцінки (англ. *noisy estimate*), яка, у свою чергу, призводить до «зашумлених» оновлень ваг моделі, наприклад, до багатьох оновлень з, можливо, досить різними оцінками градієнта помилки [47]. Тим не менш, ці «зашумлені» оновлення можуть призвести до швидшого навчання, а іноді й до більш надійної моделі.

Кількість навчальних прикладів, що використовуються для оцінки градієнта помилки, є гіперпараметром для алгоритму навчання, який називається **розміром пакету** (англ. *batch size*) або просто **пакетом** (англ. *batch*).

Наприклад, розмір пакету 32 означає, що 32 вибірки з навчального набору даних будуть використані для оцінки градієнта помилки до оновлення ваг моделі. Одна епоха навчання означає, що алгоритм навчання зробив один прохід через навчальний набір даних, де приклади були розділені на випадково вибрані групи за розміром пакету.

Навчальний алгоритм, де розмір пакету встановлюється на загальну кількість навчальних прикладів, називається **пакетним градієнтним спуском** (англ. *batch gradient descent*), а навчальний алгоритм, де розмір пакету встановлюється на 1 навчальний приклад, називається **стохастичним градієнтним спуском** (англ. *stochastic gradient descent*).

Конфігурація з розміром пакету посередині (наприклад, більше 1 прикладу і менше, ніж кількість прикладів у навчальному наборі даних)

називається **міні-пакетним градієнтним спуском** (англ. *minibatch gradient descent*).

Враховуючи, що дуже великі набори даних часто використовуються для навчання нейронних мереж глибинного навчання, розмір пакету мало коли встановлюється на розмір навчального набору даних. Тому у нашій роботі ми обрали для реалізації підходи з міні-пакетним та стохастичним градієнтним спуском.

Щоб краще зрозуміти цю концепцію, на рис. 2.6 представлена архітектура даної моделі ДКЧП, вона складається з одного шару вбудовування, одного шару ДКЧП та одного щільного шару.

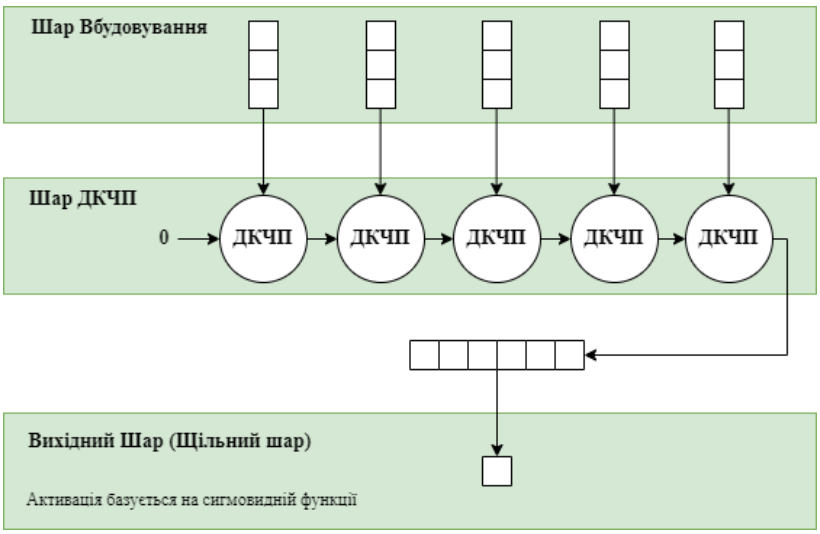


Рис. 2.6: *Модель ДКЧП*

Модель ДКЧП (Міні-пакетний градієнтний спуск)

Як ми вже з'ясували у попередньому підрозділі, для міні-пакетного градієнтного спуску розмір пакету встановлюється у значення більше ніж 1 і менше, ніж загальна кількість прикладів у наборі навчальних даних.

Менші розміри пакетів використовуються з 2 основних причин:

- Менші розміри пакетів є «зашумленими», забезпечуючи ефект регуляризації та меншу похибку узагальнення;

- Менші розміри пакетів полегшують розміщення однієї партії навчальних даних у пам’яті (тобто при використанні графічного процесора).

Тим не менш, розмір пакету впливає на швидкість навчання моделі та стабільність процесу навчання. Це важливий гіперпараметр, який повинен бути добре зрозумілий і налаштований фахівцем глибинного навчання.

Інший популярний підхід для визначення цього параметру: взяти його як 1024 і зменшувати в порядку зменшення степені двійки (512, 256, 128 і т.д.) та дивитись при якому параметрі будуть найкращі результати. Невеликі розміри пакетів, наприклад 32 чи 64, як правило, працюють найкраще [48].

Архітектура даної моделі ДКЧП має такий самий вигляд як і для попереднього випадку (див. рис. 2.6).

Складені шари ДКЧП

Складені шари ДКЧП – це просто стек з кількох шарів ДКЧП. Ідея полягає в тому, щоб сформувати глибинну мережу не лише з точки зору шарів, але й у рекурентних розмірах. Ідея полягає в тому, що вищі шари ДКЧП можуть фіксувати абстрактні поняття в послідовності, які можуть допомогти в певному завданні, у нашому випадку, аналізі настроїв.

Для кращого розуміння на рис. 2.7 представлена архітектура моделі стека 3 ДКЧП, вона складається з одного шару вбудовування, 3 шарів ДКЧП та одного щільного шару.

Двонаправлені ДКЧП

Двонаправлена модель ДКЧП була введена з метою покращення продуктивності стандартних РНМ. Моделі РНМ не мають доступу до майбутньої інформації про поточний стан. Щоб вирішити це обмеження, двонаправлена модель ДКЧП з’єднує приховані шари протилежних напрямків до одного виходу, щоб вихідний шар міг отримати доступ до інформації з минулого та майбутнього станів. Одна з таких архітектур представлена на рис. 2.8.

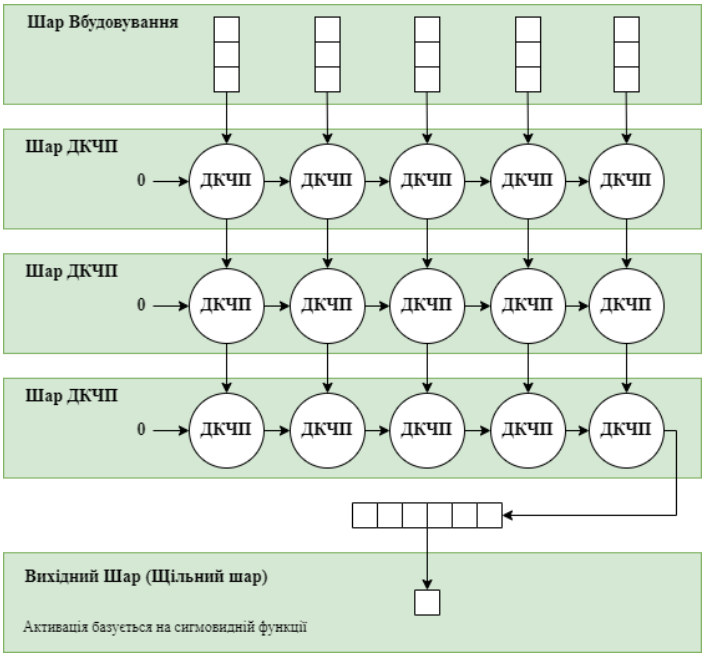


Рис. 2.7: *Стек з 3 шарів ДКЧП*

Наша двонаправлена модель ДКЧП складається з одного шару вбудовування, двох шарів ДКЧП з двонаправленим спрямуванням, шару виключення та щільного шару.

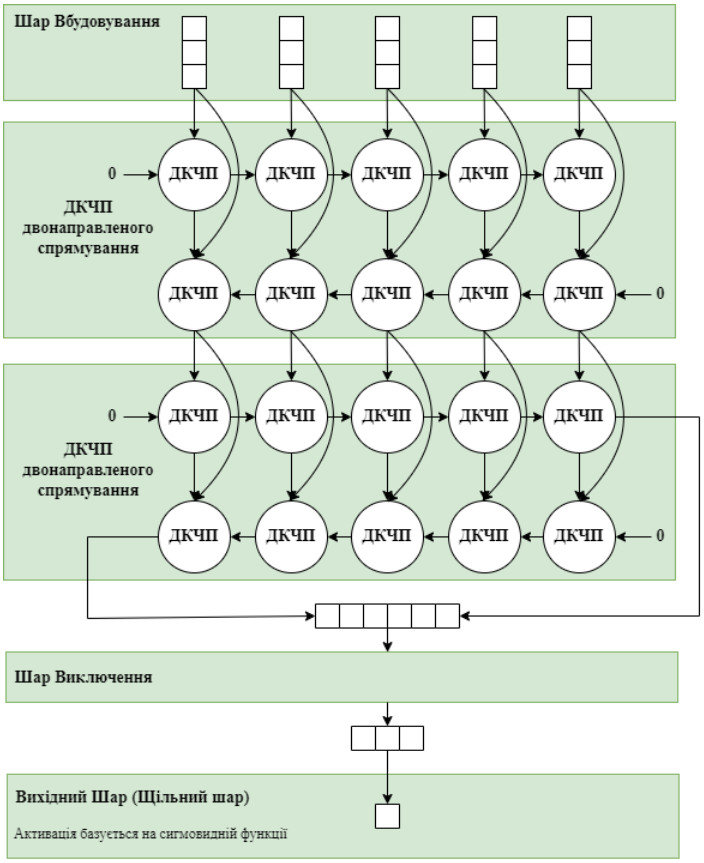


Рис. 2.8: *Архітектура двонаправленої ДКЧП*

Комбінована мережа (ЗНМ-ДКЧП)

Модель ЗНМ-ДКЧП (англ. *CNN-LSTM*) полягає, як випливає з назви, у поєднанні архітектури ЗНМ з архітектурою ДКЧП. Попередні дослідження [49] показують, що іноді поєднання різних моделей може дати нам більш потужну загальну модель, яка покращує ефективність передбачення. Це забезпечується завдяки тому, що кожна з комбінованих моделей надає свої специфічні особливості, що призводить до більш повної моделі [50].

Моделі згорткових нейронних мереж застосовують згортки над входами для обчислення виходів. На відміну від інших нейронних мереж, які лише з'єднують кожен вхідний нейрон з кожним вихідним нейроном попереднього шару, у ЗНМ кожен шар застосовує різні фільтри до даних, і під час цього процесу модель вивчає значення своїх фільтрів на основі завдання, яке ми хочемо виконати. Наприкінці цього процесу результати кожного шару об'єднуються. Однак, оскільки моделі ЗНМ здатні витягувати локальну інформацію, але можуть не охоплювати довгі залежності – ми виконуємо об'єднання з ДКЧП, щоб спробувати гарантувати, що це обмеження зникне. Комбіновану архітектуру можна побачити на рис. 2.9.

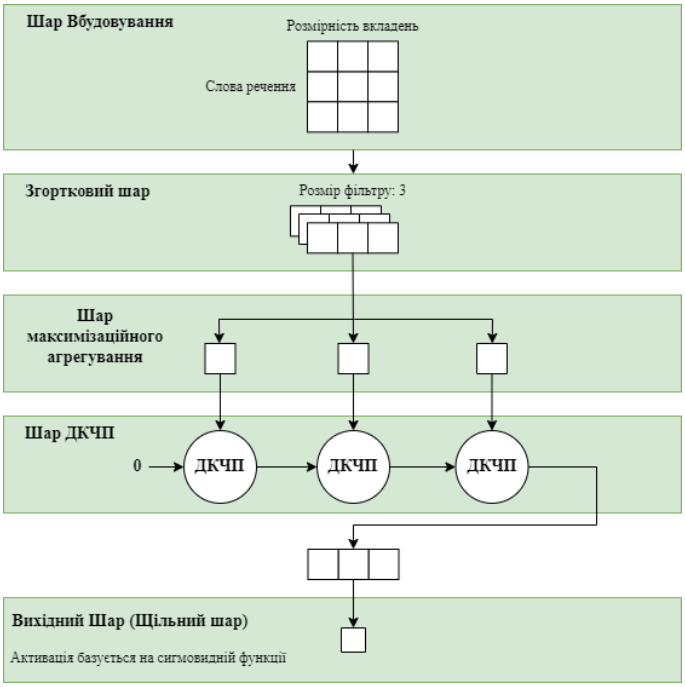


Рис. 2.9: *Архітектура ЗНМ-ДКЧП*

З рис. 2.9 видно, що модель складається з одного шару вбудовування, одного згорткового шару, одного шару максимізаційного агрегування, одного шару ДКЧП та одного щільного шару.

2.3. Характеристики оцінки якості моделі

Далі ми представляємо показники продуктивності, які використовуються для оцінки класифікаторів у нашому проєкті. Розглядаючи двійковий класифікатор f, розгорнутий на тестовому наборі даних, можна розрахувати чотири можливі результати:

- правильно класифіковані позитивні спостереження або *істинно позитивні* (ІП, англ. *TP, true positive*);
- правильно класифіковані негативні спостереження або *істинно негативні* (ІН, англ. *TN, true negative*);
- негативні спостереження неправильно класифіковані як позитивні або *хибно позитивні* (ХП, англ. *FP, false positive*);
- позитивні спостереження, помилково класифіковані як негативні або *хибно негативні* (ХН, англ. *FN, false negative*).

1. **Істинно позитивні** (ІП) показники представляють кількість правильно класифікованих екземплярів для позитивного класу, поділену на загальну кількість екземплярів, тобто випадків, класифікованих як позитивні, які насправді є позитивними. ІПР – істинно позитивний рівень.

$$\text{ІПР} = \frac{\text{ІП}}{\text{П}} = \frac{\text{ІП}}{\text{ІП} + \text{ХН}}$$

2. **Істинно негативні** (ІН) представляють кількість правильно класифікованих екземплярів для негативного класу, поділену на загальну кількість екземплярів, тобто екземплярів, класифікованих як негативні, які насправді є негативними. ІНР – істинно негативний рівень.

$$\text{ІНР} = \frac{\text{ІН}}{\text{Н}} = \frac{\text{ІН}}{\text{ІН} + \text{ХП}}$$

3. **Хибно позитивні** (ХП) результати представляють кількість неправильно класифікованих екземплярів для позитивного класу, поділену на

загальну кількість екземплярів, тобто випадків, класифікованих як позитивні, які насправді є негативними. ХПР – хибнопозитивний рівень.

$$\text{ХПР} = \frac{\text{ХП}}{\text{Н}} = \frac{\text{ХП}}{\text{ХП} + \text{ІН}}$$

4. **Хибно негативні** (ХН) являють собою кількість неправильно класифікованих екземплярів для негативного класу, поділену на загальну кількість екземплярів, тобто випадків, класифікованих як негативні, які насправді є позитивними. ХНР – хибнонегативний рівень.

$$\text{ХНР} = \frac{\text{ХН}}{\text{П}} = \frac{\text{ХН}}{\text{ХН} + \text{ІП}}$$

Ці результати зазвичай відображаються за допомогою матриці невідповідностей С, як, наприклад, у таблиці 2.1.

Табл. 2.1: *Класифікаційна матриця невідповідностей*

	$\boldsymbol{y = +1}$	$\boldsymbol{y = -1}$
$\boldsymbol{f(x) = +1}$	ІП	ХП
$\boldsymbol{f(x) = -1}$	ХН	ІН

Іншими найпоширенішими показниками оцінки, які використовуються в аналізі настроїв, є точність (англ. *accuracy*), влучність (англ. *precision*), повнота (англ. *recall*) та F-міра.

Показник **точності** вказує, яка частка екземплярів було правильно класифіковано для всіх класів. Для задач бінарної класифікації вона обчислюється так:

$$Accuracy = \frac{\text{ІП} + \text{ІН}}{\text{ІП} + \text{ІН} + \text{ХП} + \text{ХН}}$$

і узагальнюється для багатокласових задач до такого виразу:

$$Accuracy = \frac{\sum_{i=1}^{L} C_{ii}}{N},$$

де L – кількість класів;

N – загальна кількість прикладів;

C_{ii} – комірка у матриці невідповідностей.

Влучність представляє частку правильно класифікованих позитивних спостережень над усіма спостереженнями, класифікованими як позитивні:

$$Precision = \frac{\text{ІП}}{\text{ІП} + \text{ХП}}$$

Повнота (також відома як чутливість або ІПР) – частка правильно класифікованих спостережень певного типу над усіма спостереженнями обраного типу, наприклад, скільки істинно позитивних випадків було класифіковано як позитивні:

$$Recall = \frac{\text{ІП}}{\text{ІП} + \text{ХН}}$$

F-міра — це показник точності, який поєднує в собі влучність та повноту, тобто є їх середнім гармонійним. Найпоширенішою F-мірою є міра F1:

$$F = 2 * \frac{Precision * Recall}{Precision + Recall}$$

Влучність, повноту та F-міру для класифікатора отримують як середнє значення цих значень для всіх класів.

Крім вище зазначених параметрів у нашому дослідженні для оцінки якості бінарної класифікації ми також використовували ROC-криву.

ROC-крива (робоча характеристика приймача, англ. *receiver operating characteristic*) – це графік, що показує продуктивність моделі класифікації для задач бінарної класифікації. Ця крива відображає два параметри: істинно позитивний та хибнопозитивний рівень.

Крива ROC відображає співвідношення ІПР та ХНР за різних порогів класифікації. Зниження порога класифікації дозволяє класифікувати більше елементів як позитивні, тим самим збільшуючи кількість хибно позитивних та істинно позитивних спостережень.

Площа під ROC-кривою (англ. *AUC – area under ROC curve*) – кількісна інтерпретація ROC, надає сукупний показник ефективності за всіма можливими порогами класифікації.

Чим вище AUC, тим краще продуктивність моделі при розрізненні позитивних і негативних класів. Коли AUC = 1, то класифікатор здатний ідеально розрізняти всі позитивні та негативні точки класу правильно, при цьому значення 0.5 відповідає звичайному вгадуванню. Розраховується за формулою для хибнонегативного рівня.

ВИСНОВОК ДО ДРУГОГО РОЗДІЛУ

У цьому розділі ми описали усі необхідні компоненти, залучені до вирішення завдань класифікації для аналізу настроїв. Зокрема, у цій главі були описані найпопулярніші класифікатори, а також було надано короткий огляд основних архітектур моделей глибинного навчання та шарів, що використовуються в глибинних нейронних мережах.

Обрані класифікатори машинного навчання та моделі глибинного навчання були прийняті завдяки їх чудовій продуктивності, а також результатам при використанні в завданнях класифікації, які виконували інші дослідники.

У наступному розділі ми представимо оціночні експерименти, а також зробимо порівняльний аналіз для кожного конкретного класифікатора та моделей глибинного навчання.

РОЗДІЛ 3
РЕАЛІЗАЦІЯ І ПОРІВНЯННЯ МОДЕЛЕЙ КЛАСИФІКАЦІЇ ТА РОЗРОБКА ВЕБ-СЕРВІСУ ДЛЯ СЕНТИМЕНТ-АНАЛІЗУ

3.1. Обґрунтування обраних засобів розробки

3.1.1. Python

Задля створення моделі для сентимент-аналізу текстів, а також для розробки веб-сервісу ми використовували потужності мови Python. Щоб реалізувати цілі машинного навчання, ми повинні використовувати мову програмування, яка є стабільною, гнучкою та має доступні інструменти. Все це і пропонує Python.

Хороший вибір бібліотек є однією з головних причин, чому Python є найпопулярнішою мовою програмування для машинного навчання. Він має складні інструменти для представлення та дослідження результатів: matplotlib, pandas і Jupyter. І, звичайно, має зрілі пакети, що надають необхідні інструменти для машинного навчання: низькорівневі, високопродуктивні числові маніпуляції (numpy, sciру) та набори інструментів машинного навчання. Python також простий у використанні, навчанні, і є платформо-незалежним. Іншою причиною, яка приваблює до використання Python, є його простота та легкість навчання. Python написаний за допомогою простого коду і може легко створювати моделі для машинного навчання.

3.1.2. Jupyter Notebook

Jupyter Notebook – це веб-додаток з відкритим вихідним кодом, який забезпечує інтерактивне обчислювальне середовище для розробки додатків для машинного навчання, дата аналізу та багато іншого.

Нотбуки Jupyter об’єднують в собі програмний код, обчислювальні результати, пояснювальний текст і багатий вміст в одному документі. Вони

вже давно стали стандартом для використання при роботі з машинним навчанням.

3.1.3. PyCharm

PyCharm – одна з найпопулярніших IDE для Python, розроблена JetBrains, яка використовується для виконання програм мовою Python. PyCharm надає багато корисних функцій, таких як перевірка коду, процес дебагу, підтримка різних фреймворків програмування, керування пакетами тощо.

Однією з особливостей PyCharm є те, що він включає підтримку Django, а саме на ньому і буде реалізовано наш веб-сервіс. Також завдяки можливості включення функцій JavaScript в PyCharm, його можна вважати найкращою IDE для Django.

3.2. Обґрунтування обраних бібліотек та фреймворку

3.2.1. Огляд бібліотек для розробки моделей класифікації

NumPy

Бібліотека NumPy є фундаментальним інструментом для вивчення машинного навчання. Багато його функцій дуже корисні для виконання будь-яких математичних або наукових обчислень. Оскільки відомо, що математика є основою машинного навчання, більшість математичних завдань можна виконати за допомогою NumPy.

NumPy розшифровується як Numerical Python і є однією з найкорисніших наукових бібліотек у програмуванні на Python. Вона забезпечує підтримку великих багатовимірних об’єктів масиву та різноманітні інструменти для роботи з ними. Різні інші бібліотеки, які ми використовували, такі як Pandas, Matplotlib і Scikit-learn, створені на основі цієї.

Pandas

Pandas – це бібліотека з відкритим кодом, яка надає структури даних та інструменти аналізу даних. Важливою приміткою щодо Pandas є його висока продуктивність і простота у використанні.

Pandas використовуються для розширеної структури та аналізу даних. Він дозволяє об’єднувати, фільтрувати та збирати дані з інших зовнішніх джерел (наприклад, Excel, CSV).

Matplotlib

Matplotlib – одна з найпопулярніших і найстаріших бібліотек для візуалізації у Python, яка використовується в машинному навчанні, де допомагає зрозуміти величезну кількість даних за допомогою різних візуалізацій.

Основні бібліотеки, засновані на Matplotlib, такі як Seaborn, яку ми теж використовували в процесі роботи, можуть створювати більш складні типи графіків і діаграм.

NLTK

Бібліотека NLTK є однією з найпопулярніших і надає доступні інтерфейси для більш ніж 50 корпусів і лексичних джерел, зіставлених з алгоритмами машинного навчання, а також надійний вибір синтаксичних аналізаторів і утиліт.

Окрім забезпечення аналізу настроїв, алгоритми NLTK включають розпізнавання іменованих об’єктів, токенізацію, позначення частини мови, стемінг, лематизацію та семантичний механізм міркувань (англ. *semantic reasoning*).

NLTK також може похвалитися гарним вибором розширень сторонніх розробників, величезна різноманітність деяких його інструментів (наприклад, бібліотека має 9 бібліотек для стемінгу на відміну від єдиного стемера SpaCy) може зробити фреймворк схожим на певний пакет архівних матеріалів NLP за останні 15 років.

Позитивною стороною цього є те, що жоден конкурент NLTK не може похвалитися такою повною та корисною базою документації, а також іншою корисною літературою та інтернет-ресурсами. Недоліком є те, що ця

бібліотека може бути досить повільною, а також важкою у використанні і не має підтримки української мови.

Ми використовували цю бібліотеку на етапі попередньої обробки даних.

Gensim

Gensim виник із роботи двох студентів у лабораторії обробки природних мов у Чеській Республіці приблизно в 2010 році і став одним із найбільш масштабованих та потужних варіантів проєктів NLP.

Як і NLTK, Gensim є комплексним і достатньо потужним інструментом. Його оригінальна та високо оптимізована реалізація моделей машинного навчання word2vec від Google робить його сильним претендентом на включення до проєкту аналізу настроїв, як базової основи, чи як ресурсу бібліотеки.

В основному Gensim використовують для пошуку подібності в текстових документах і тематичному моделюванні. Це чудова бібліотека для виявлення подібності між двома документами за допомогою векторного моделювання простору та тематичного моделювання. Він розглядає вміст документів як послідовності векторів і кластерів. Потім Gensim класифікує їх і забезпечує візуалізацію.

Він також має неймовірну оптимізацію використання пам’яті та швидкість обробки. Ось чому він може обробляти великі обсяги текстових даних достатньо швидко.

Ми використовували дану бібліотеку для конвертування GloVe моделі у формат word2vec.

Scikit-Learn

Scikit-learn (Sklearn) – це одна з найкорисніших та найнадійніших бібліотек для машинного навчання на Python. Вона надає вибір ефективних інструментів для машинного навчання та статистичного моделювання, включаючи класифікацію, регресію, кластеризацію та зменшення розмірності. Ця бібліотека побудована на NumPy, SciPy і Matplotlib.

Майже всі популярні алгоритми навчання з керівництвом, такі як логістична регресія, метод опорних векторів, випадковий ліс та інші розглянуті в попередньому розділі алгоритми, є частиною scikit-learn. Бібліотека також має простий у використанні інтерфейс для кожного типу об'єкта моделі, що полегшує швидке створення прототипів та експериментування з моделями.

Саме з цієї бібліотеки ми використовували вже готову реалізацію класифікаторів машинного навчання описаних у попередньому розділі, а також саме з її допомогою створювали тренувальні та тестові набори і робили класифікаційні звіти.

Keras

Keras був розроблений, щоб дозволити інженерам глибинного навчання дуже швидко створювати й експериментувати з різними моделями. І якщо нам треба отримати швидкі результати з різними моделями, як у нашому випадку, найкращим вибором є Keras.

Keras має прості у використанні модулі практично для кожного аспекту процесу побудови моделі нейронної мережі, включаючи моделі, шари, оптимізатори і методи втрат. Саме для всього цього ми і використали дану бібліотеку. Ці прості у використанні функції забезпечують швидке навчання, надійне тестування, швидке експериментування та гнучке розгортання моделі, на додачу до багатьох інших функцій.

Окрім ЦП комп'ютера, бібліотека також використовує графічний процесор, що дозволяє здійснювати швидші обчислення. Ми використовували дану бібліотек для всієї роботи з моделями глибинного навчання, описаних у попередньому розділі.

Seaborn

Seaborn – це бібліотека для створення статистичних графіків на Python. Вона ґрунтується на matplotlib і тісно взаємодіє зі структурами даних pandas. Архітектура Seaborn дозволяє нам швидко вивчити та зрозуміти свої дані.

Seaborn захоплює цілі масиви даних і виконує всі внутрішні функції, потрібні для семантичного мапінгу та статистичної агрегації для перетворення даних на інформативні графіки.

Ми використовували бібліотеку для створення інформативних гістограм.

WordCloud

Багато разів ми могли бачити картинку, наповнену великою кількістю слів різного розміру, які відображають частоту або важливість кожного слова. Це називається хмара тегів або WordCloud. Цей інструмент є досить зручним для вивчення текстових даних і для того, щоб зробити звіт більш живим, тому ми його використали при розробці для аналізу слів на етапі попередньої обробки даних.

Pickle

Pickle – це модуль Python, який використовується для серіалізації об'єкта Python у двійковий формат та десеріалізації його назад до об'єкта Python. Модуль pickle відстежує об’єкти, які він уже серіалізував, так що пізніші посилання на той самий об’єкт не будуть серіалізовані знову, що дозволяє пришвидшити час виконання.

Через зручність, а також можливість збереження моделі за дуже короткий час ми використали даний модуль в роботі.

3.2.2. Огляд фреймворку та бібліотек для розробки веб-сервісу

Django

Django – це безкоштовний фреймворк з відкритим вихідним кодом для веб-додатків на Python і дуже гнучкий інструмент веб-розробки, який можна використовувати для створення будь-якого типу веб-сайтів або програм, які потрібні.

Це чудовий вибір практично для будь-якого проєкту веб-розробки. Фреймворк існує вже 11 років і пройшов етапи значного вдосконалення. Django має вбудовані функції, які чудово підходять для захисту

конфіденційних даних, транзакцій та аутентифікації користувачів. Панель адміністратора тут йде за замовчуванням, що дуже допоможе нам при керуванні нашою програмою.

Django чудово створює веб-сайти, які можуть обробляти велику кількість трафіку та транзакцій. Сайти на його базі набагато краще адаптуються до змін, не турбуючись про загальну функціональність веб-сайту.

Django не тільки чудово підходить для створення потужного, масштабованого інтерфейсного веб-контенту, але він також має високу здатність створювати програми, які можуть працювати на стороні сервера, щоб забезпечити розширені та потужні функції, яких немає у більшості веб-сайтів. Тому він і був обраний нами при створенні веб-додатку для сентимент-аналізу.

PDFMiner

PDFMiner – це інструмент для вилучення інформації з документів PDF. На відміну від інших інструментів, пов’язаних із PDF, він повністю зосереджений на отриманні та аналізі текстових даних. PDFMiner дозволяє отримати точне розташування тексту на сторінці, а також іншу інформацію, таку як шрифти або рядки. Він включає в себе конвертер PDF, який може перетворювати PDF-файли в інші текстові формати, тому був використаний нами в проєкті – щоб витягувати текстовий контент для його подальшої обробки.

Gunicorn

Gunicorn – сервер додатків Python. Він перетворює HTTP-запити на те, що Python може зрозуміти. Gunicorn реалізує інтерфейс шлюзу веб-сервера (англ. *WSGI*), який є стандартним інтерфейсом між програмним забезпеченням веб-сервера та веб-додатками.

Він швидкий, оптимізований і розроблений для виробництва. Також добре перевірений, зокрема щодо його функціональності як сервера додатків. Був використаний нами для деплою нашого додатку на Heroku.

Psycopg

Psycopg – найпопулярніший адаптер бази даних PostgreSQL для мови програмування Python. Його основними характеристиками є повна реалізація специфікації Python DB API 2.0 і безпека потоків (кілька потоків можуть використовувати одне з'єднання). Був використаний для підтримки нашої бази даних.

WhiteNoise

Завдяки кільком рядкам конфігурації WhiteNoise дозволяє веб-програмі обслуговувати власні статичні файли, що робить його автономним блоком, який можна розгорнути в будь-якому місці, не покладаючись на nginx, Amazon S3 або будь-який інший зовнішній сервіс. Це особливо корисно для Heroku, тому ми його використали.

3.3. Розробка моделі класифікації

3.3.1. Вибір корпусу даних

Огляд продуктів Amazon є улюбленою мішенню для оцінки алгоритмів сентиментального аналізу. Аналіз відгуків клієнтів має багато практичних застосувань, оскільки мільйони людей щодня планують свої покупки на основі оцінок товарів, створених за відгуками.

У даному проєкті ми використовуємо набір оглядів продуктів із набору даних Amazon, зібраних Марком Дредзе та Джоном Блітцером [51], який містить огляди з 25 різних категорій.

Позитивними ми будемо вважати огляди з рейтингом 3 і більше (з 5), негативними відповідно рейтинги менше 3.

Загальна кількість оглядів, які ми будемо використовувати для навчання складає близько 30 тисяч. Для навчання української моделі ми використовували перекладені огляди Amazon з англійської. Для цього був написаний спеціальний скрипт, який робить запити на API Google Translate і витягує переклад. Це звісно не найкращий вибір, однак готових наборів даних українською нам знайти не вдалось.

3.3.2. Обробка даних

На цьому етапі ми виконали перетворення даних з вхідного формату XML у представлення в датафреймі pandas для зручності обробки, а потім виконали кожен пункт описаний нами раніше у розділі 1.2.2, а точніше нормалізацію рядків, токенізацію, видалення стоп-слів, стемінг та лематизацію за допомогою засобів бібліотеки NLTK.

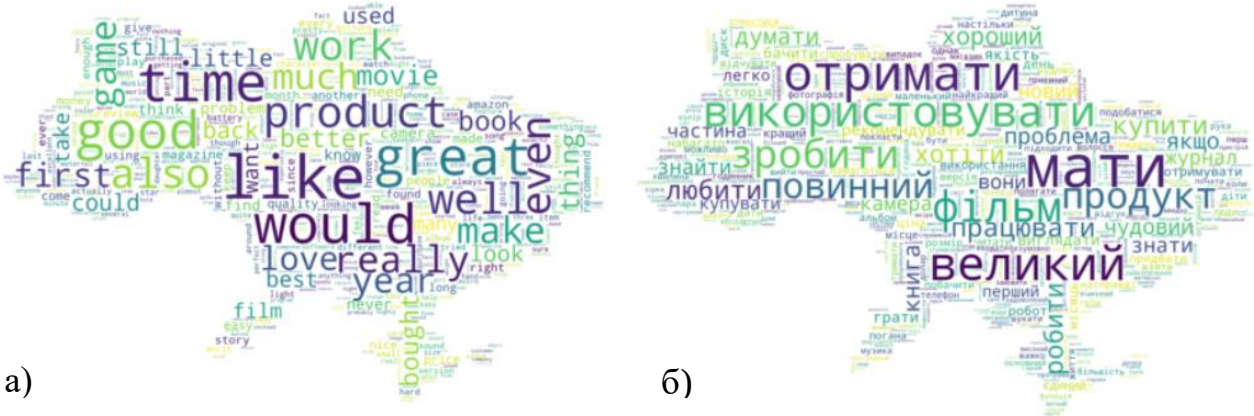
Опісля ми провели ще одну операцію з очищення від зайвих символів, що включало видалення цифр, слів менше 3 символів та некириличних символів для української моделі.

Оскільки в бібліотеці NLTK немає підтримки української мови, ми пішли іншим шляхом.

Для видалення стоп-слів ми скористалися вже готовим достатньо великим списком українських стоп-слів [52].

Для стемінгу ми написали власну реалізацію PorterStemmer, а для лематизації скористались рішенням яке пропонує бібліотека pymorphy2, де є підтримка української, однак поки тільки на експериментальному рівні.

Візуалізацію найбільш вживаних слів після обробки для англійського та українського корпусів можна побачити на рис. 3.1 та рис 3.2. Для візуалізації ми використали бібліотеки WordCloud та Seaborn.



а) б)

Рис. 3.1: *Хмара найбільших вживаних слів в оглядах для англійського (а) та українського (б) набору даних*

Далі ми перетворили наші навчальні документи у вигляд кортежів. Чому кортеж? Кортежі незмінні, тобто вони мають фіксований розмір за своєю природою, тоді як списки є динамічними, їх можна змінювати. Далі нам не потрібно буде додавати або видаляти елементи, тому кортежі є чудовим варіантом, які, на додачу, працюють швидше, ніж списки.

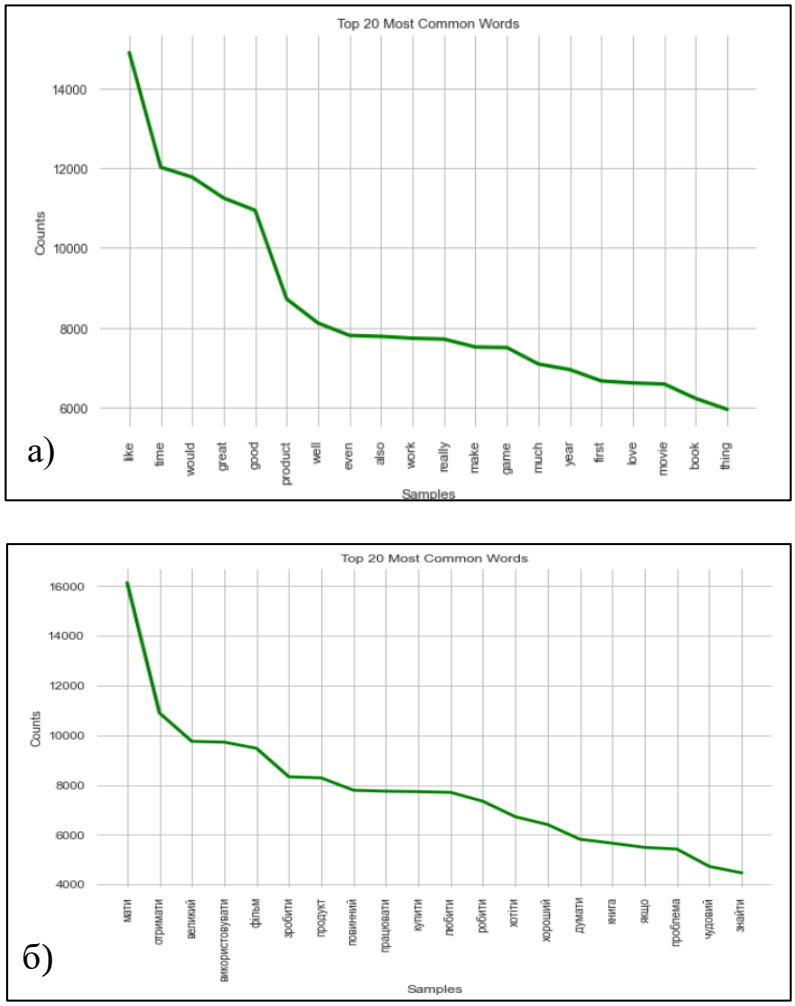


Рис. 3.2: *Графік найбільших вживаних слів та їх кількості в оглядах для англійського (а) та українського (б) набору даних*

Робота по створенню моделі була розбита на 3 частини, однак в кожній нам потрібно було працювати з даними, які ми тут обробили, тому ми використовували бібліотеку *pickle*, призначену для серіалізації/десеріалізації, щоб потім без проблем користуватися цими даними в інших частинах проєкту.

Наступним кроком ми розділили набір даних, щоб виконати неупереджену оцінку моделі та визначити недонавчання чи перенавчання. Для

цього ми використовували модуль *scikit-learn* або *sklearn*. Він має багато пакетів для науки про дані та машинного навчання, але зараз ми зосередимося на пакеті *model_selection*, зокрема на функції *train_test_split()*.

На цьому наша підготовка даних була завершена і ми перейшли до створення, тренування та тестування класифікаторів.

3.3.3. Навчання і порівняння класифікаторів машинного навчання

Для початку ми скористались швидким підходом до наївної баєсової класифікації, щоб побачити, чи можуть наші дані після попередньої обробки дати нам хороші результати, щоб ми могли їх використовувати для більш складних класифікаторів.

Реалізацію наївного баєсового класифікатора ми взяли з модуля *classify* бібліотеки *NLTK*. Точність класифікатора зображена у табл. 3.1.

Табл. 3.1: *Точність при використанні наївного баєсового класифікатора*

Корпус даних	**Точність класифікатора**
Українська мова	*80.38%*
Англійська мова	*81.26%*

Результати достатньо хороші, щоб ми могли продовжити дослідження.

Для подальшої роботи нам потрібно створити вектори слів. Для простоти ми використовували попередньо навчену модель. Google зміг навчити модель Word2Vec на величезному наборі даних Google News, який містив понад 100 мільярдів різних слів. Google створив 3 мільйони вектор-слів із цієї моделі, кожне має розмірність 300 [53]. Ми використали ці вектори для даних англійською.

Ми також знайшли модель Word2Vec з українськими словами-векторами, кожне з яких має розмірність 300. Цей зразок вектор-слів створено на основі художньої літератури. Ми обрали лематизовану версію цієї моделі, яка ідеально підійшла для наших оброблених даних [54].

Ідея word2vec полягає в наступному:

1. Взяти 3-шарову нейронну мережу (1 вхідний шар + 1 прихований шар + 1 вихідний шар);
2. Дати їй слово і навчити передбачати сусіднє слово;
3. Видалити останній (вихідний шар) і зберегти вхідний і прихований шари;
4. Ввести слово зі словникового запасу. Результатом, наданим на прихованому шарі буде «вкладення слова» у вхідне слово.

Після завантаження моделі ми почали аналізувати наші вектори. Нашим першим підходом була модель середнього вбудовування слова.

Суть цього підходу полягає в тому, щоб взяти середнє значення всіх вектор-слів із речення, щоб отримати один 300-вимірний вектор, який представляє тон усього речення, яким ми подаємо моделі і намагаємося отримати швидкий результат.

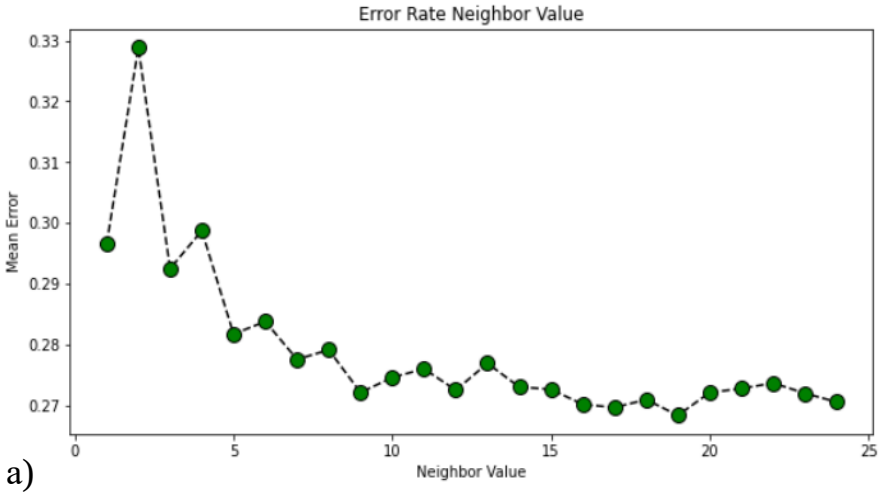
Для подальшої роботи ми розділили наш корпус на набори для навчання, тестування та розробки. Після чого перейшли до аналізу різних моделей класифікації.

KNN модель

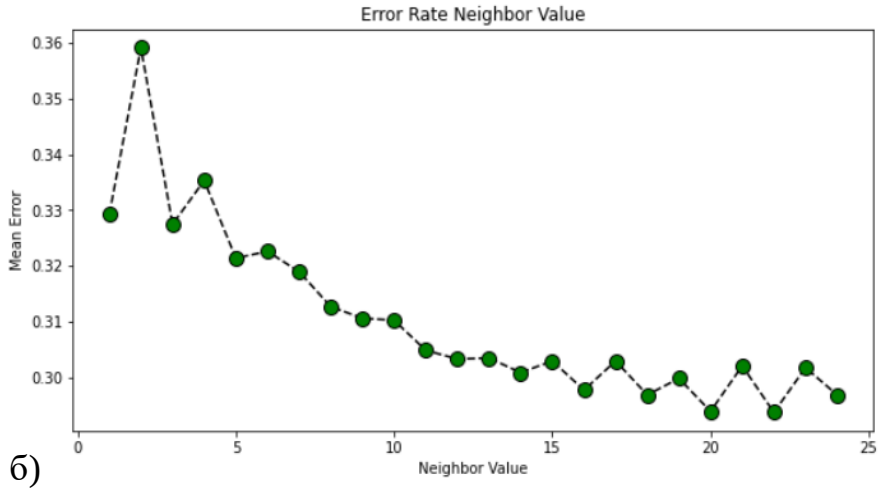
При розробці ми використовували готове рішення класифікатора з модуля *KNeighborsClassifier* бібліотеки *sklearn*. Ми створили об'єкт класифікатора K-НС, передаючи число аргументів сусідів у функцію *KNeighborsClassifier().*

Один із способів допомогти знайти найкраще значення сусідів – побудувати графік значень сусідів та відповідної частоти помилок для набору даних.

Ми побудували графік середньої помилки для прогнозованих значень тестового набору для всіх значень сусідів від 1 до 25 (рис. 3.3). Для цього обчислили середнє значення помилки для всіх прогнозованих значень, де сусід знаходиться в діапазоні від 1 до 25.



а)



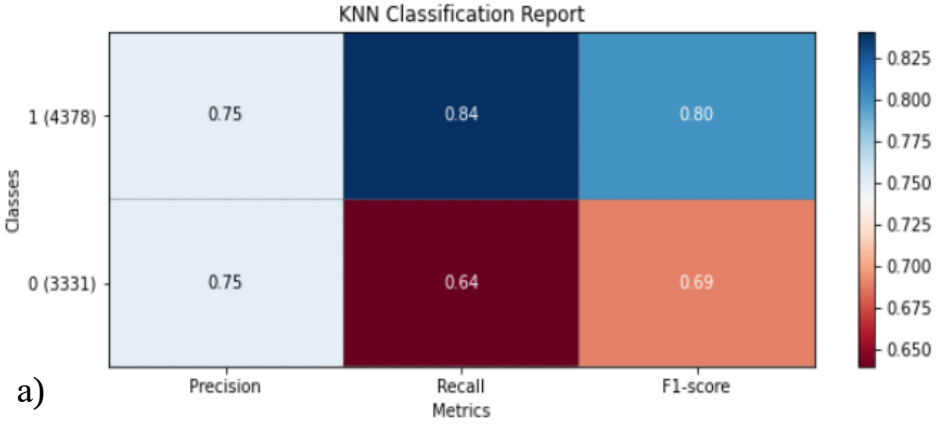
б)

Рис. 3.3: *Коефіцієнт помилки значення сусіда для англійського (а) та українського (б) набору даних*

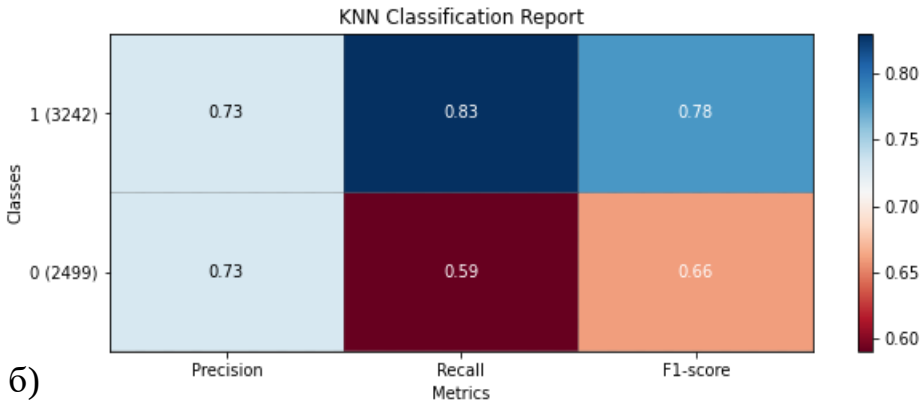
Як бачимо з рис.3.3, найкраще було взяти k=20 для української моделі, та k=19 для англійської. В результаті тренування ми отримали наступні результати (табл. 3.2). Класифікаційні звіти у вигляді теплокарти з усіма важливими характеристиками оцінювання моделі (влучність, повнота, F-міра) наведені на рис. 3.4.

Табл. 3.2: *Точність при використанні класифікатора k-найближчих сусідів*

Корпус даних	**Точність класифікатора**
Українська мова	*72.91%*
Англійська мова	*75.41%*



а)



б)

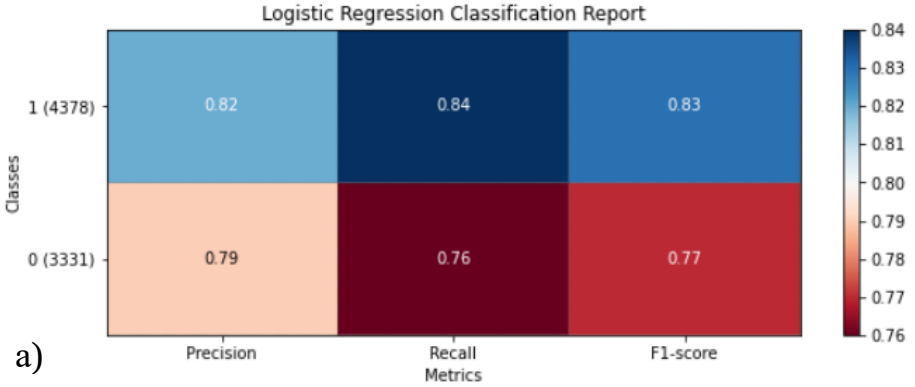
Рис. 3.4: *Класифікаційний звіт для k-найближчих сусідів для англійського (а) та українського (б) набору даних*

Логістична регресія

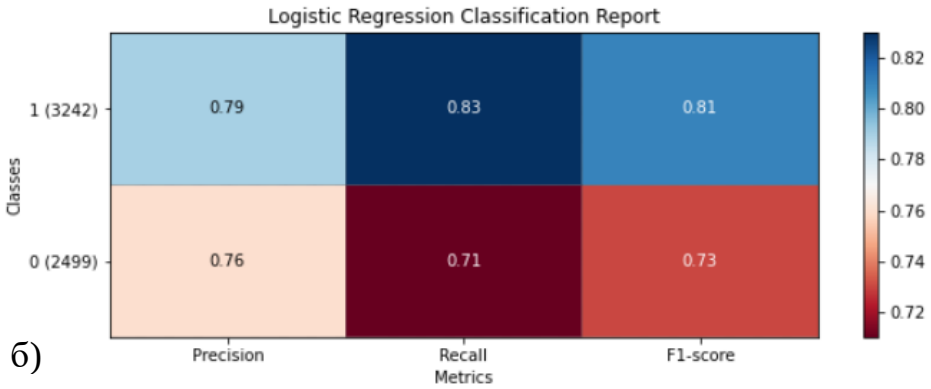
Для імплементації даного алгоритму ми взяли готове рішення з модуля *LogisticRegression* бібліотеки *sklearn*. Результати тренування зображені у табл. 3.3. Класифікаційні звіти для даного методу наведені на рис 3.5.

Табл. 3.3: *Точність при використанні для класифікації логістичної регресії*

Корпус даних	**Точність класифікатора**
Українська мова	*77.7%*
Англійська мова	*80.58%*



а)



б)

Рис. 3.5: *Класифікаційний звіт для логістичної регресії для англійського (а) та українського (б) набору даних*

Випадковий ліс

Бібліотека *sklearn* надає готове рішення і для цього алгоритму, який ми взяли з модуля *RandomForestClassifier*.

Результати тренування зображені у табл. 3.4.

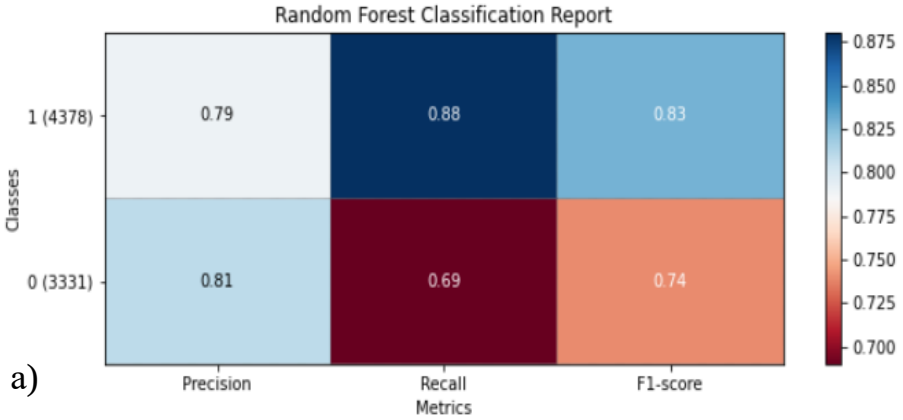
Класифікаційні звіти у вигляді теплокарти наведені на рис 3.6.

Табл. 3.4: *Точність при використанні для класифікації випадкового лісу*

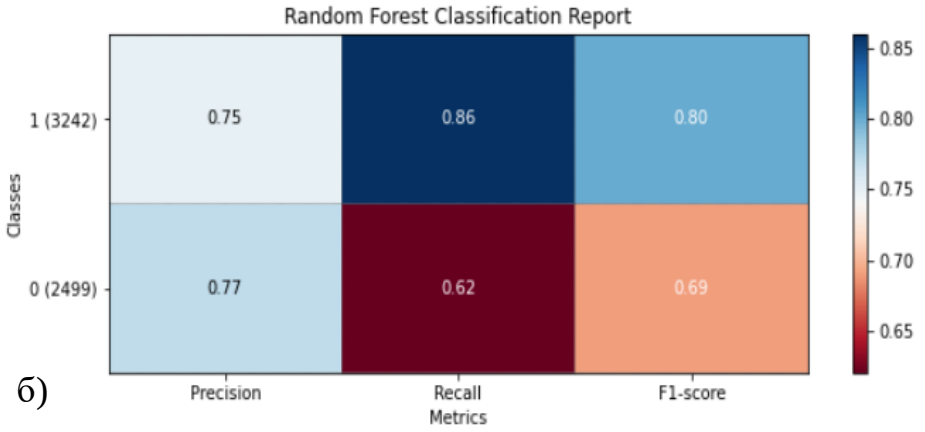
Корпус даних	**Точність класифікатора**
Українська мова	*75.74%*
Англійська мова	*79.54%*

Метод опорних векторів

Реалізація даного алгоритму була взята з модуля *svm* бібліотеки *sklearn*. Результати навчання опорно-векторних машин (ОВМ) на наших даних показано у табл. 3.5.



а)



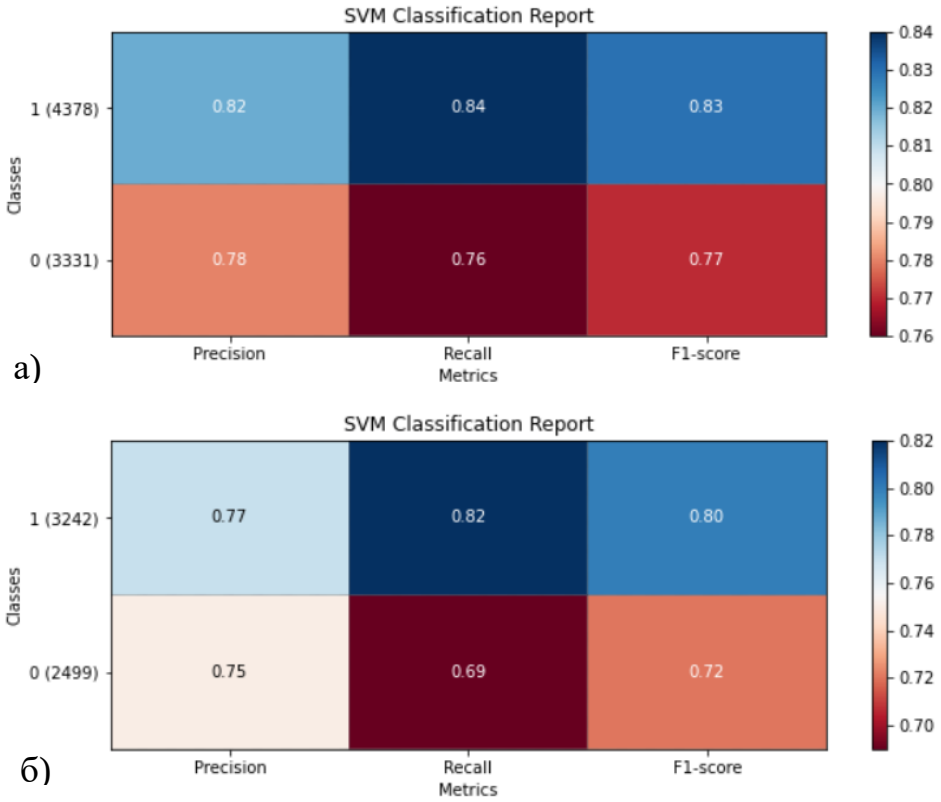
б)

Рис. 3.6: *Класифікаційний звіт для випадкового лісу для англійського (а) та українського (б) набору даних*

Табл. 3.5: *Точність при використанні для класифікації ОВМ*

Корпус даних	**Точність класифікатора**
Українська мова	*75.05%*
Англійська мова	*79.36%*

Класифікаційні звіти для розглянутого методу наведені на рис 3.7.



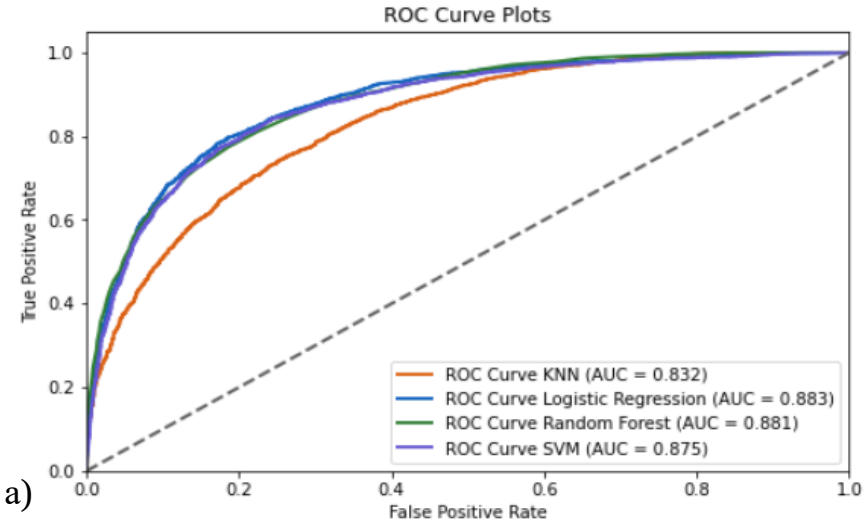
а)

б)

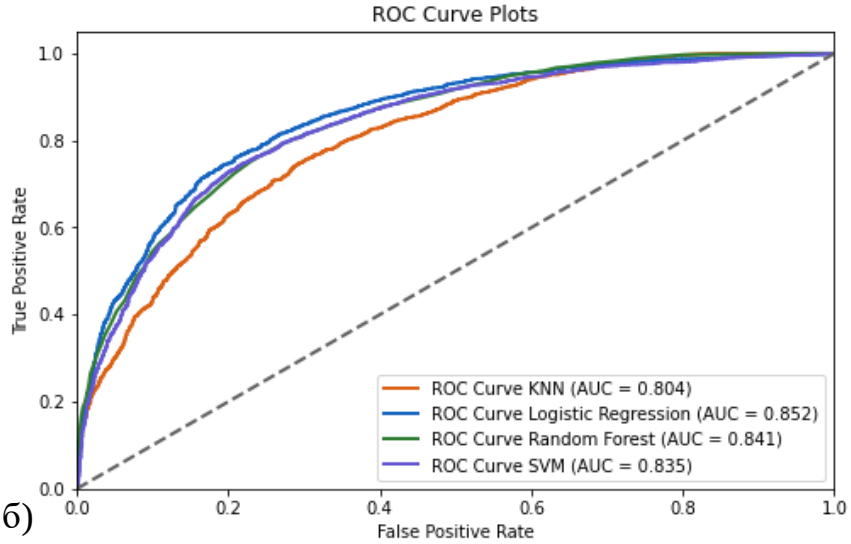
Рис. 3.7: *Класифікаційний звіт для методу опорних векторів для англійського (а) та українського (б) набору даних*

Порівняння класифікаторів машинного навчання

Корисним інструментом для прогнозування ймовірності бінарного результату є крива робочої характеристики приймача, або ROC-крива, саме вона була використана для порівняння наших класифікаторів (рис. 3.8).



а)



б)

Рис. 3.8: *Порівняння результатів класифікаторів машинного навчання за допомогою ROC-кривої для англійського (а) та українського (б) набору даних*

Виходячи з даних дослідження, а також порівняльного аналізу на рис. 3.8, ми можемо зробити висновок, що найкращою є модель логістичної регресії з AUC = 85.2% для української моделі та 88.3% для англійської, і відповідно показником точності 77.7% для української та 80.58% для англійської моделі.

3.3.4. Навчання та порівняння моделей глибинного навчання

У цій частині ми реалізували описані у розділі 2.2.2 моделі глибинного навчання. Такий підхід мав дати нам найкращі результати.

Підготовка вхідних даних

Модель Word2Vec від Google дуже важко обробляти з наявною у нас оперативною пам'яттю для задач глибинного навчання, тому ми використовували меншу модель, а саме GloVe [55], модель містить 400000 слів, тому ми отримали матрицю вбудовування 400000x100, яка містить усі значення вектор-слів.

Для української моделі нам вдалося лишити модель Word2Vec (бо вона була меншої розмірності), однак це досягалося повним завантаженням ОП.

Для налаштування моделей ми потребували параметру вхідної розмірності, який визначається максимальною кількістю слів для розгляду. Для цього ми провели аналіз і визначили оптимальну кількість слів (рис. 3.9).

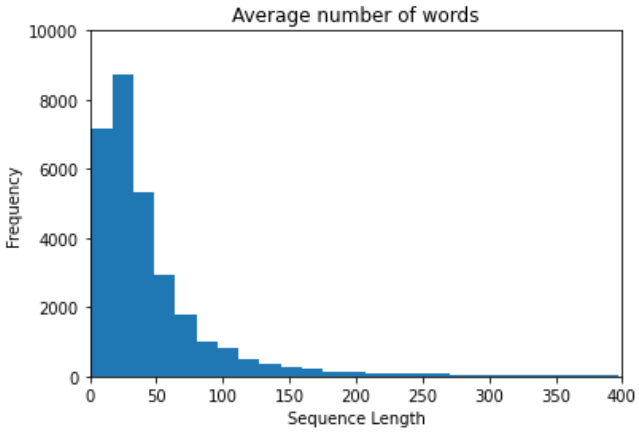


Рис. 3.9: *Середня кількість слів у оглядах*

Виходячи з даних гістограми (рис. 3.9) ми можемо з упевненістю сказати, що більшість оглядів матиме менше 100 слів, що є максимальним значенням довжини послідовності, яку ми встановили.

Далі ми написали власну функцію для вбудовування речень. Ми відновлювали вбудовування кожного речення із середнім вектором, який його складає: якщо слово є в словнику word2vec – додавали його векторне представлення, якщо нема – додавали нульовий вектор. Наступним кроком ми

вирівнювали довжину речення під 100 слів, а в кінці додавали вектор речення до списку X, до списку Y ми додавали мітку класу.

Для подальшої роботи ми розділили наш корпус на набори для навчання, тестування та розробки і перейшли до реалізації моделей глибинного навчання.

Реалізація моделей глибинного навчання

- **Модель ДКЧП (Стохастичний градієнтний спуск)**

Для шару вбудовування потрібно вказати 2 основні параметри даних: вхідна та вихідна розмірності.

Вхідна розмірність визначається як максимальна кількість слів для розгляду в представленні (для нашої роботи було прийнято значення 100, як ми зазначали вище). Вихідна розмірність визначається як розмір вкладення слова (у нас це 300 для української моделі, та 100 для англійської)

Шар ДКЧП має ті самі 2 параметри, як і для шару вбудовування. Вхідна розмірність визначається як вхідна та вихідна розмірності шару вбудовування. Вихідна розмірність було встановлено у 128 (таке значення було прийнято як оптимальне внаслідок кількох експериментів з моделлю).

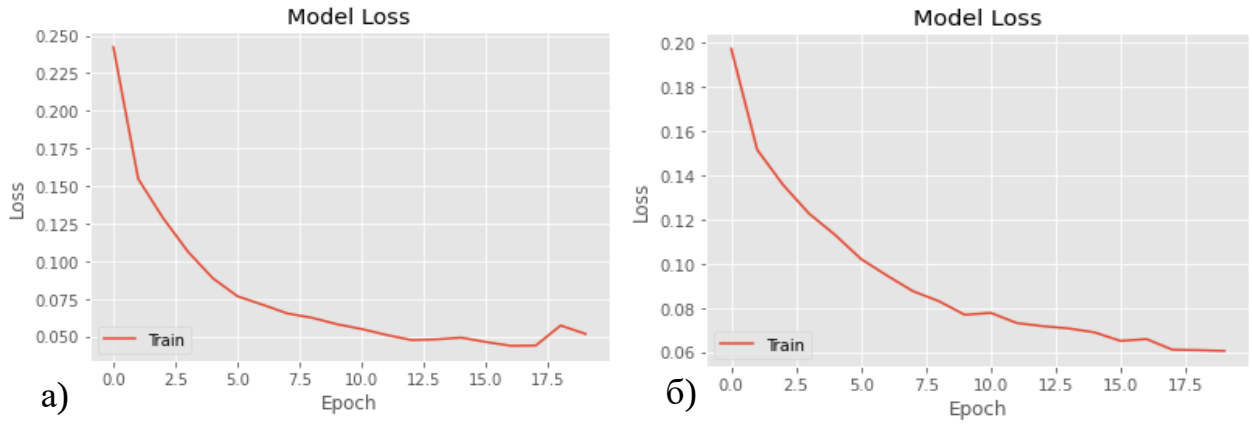
Шар щільності має також 2 параметри: розмірність виходу та активація. Вихідна розмірність представляє кількість вихідних розмірів моделі і вибирається відповідно до конкретного завдання. У нашому випадку необхідно встановити значення в 1. Активація встановлюється на сигмовидну функцію. Загалом мережа має один вхідний шар, прихований шар із 128 одиницями та один вихідний шар.

Конфігурація для тренування також включає в себе оптимізатор Адама, функцію втрат (середньоквадратична похибка) та метрику (бінарна точність). Кількість епох для навчання в кожній моделі становить 20.

Графік втрат моделі зображено на рис. 3.10.

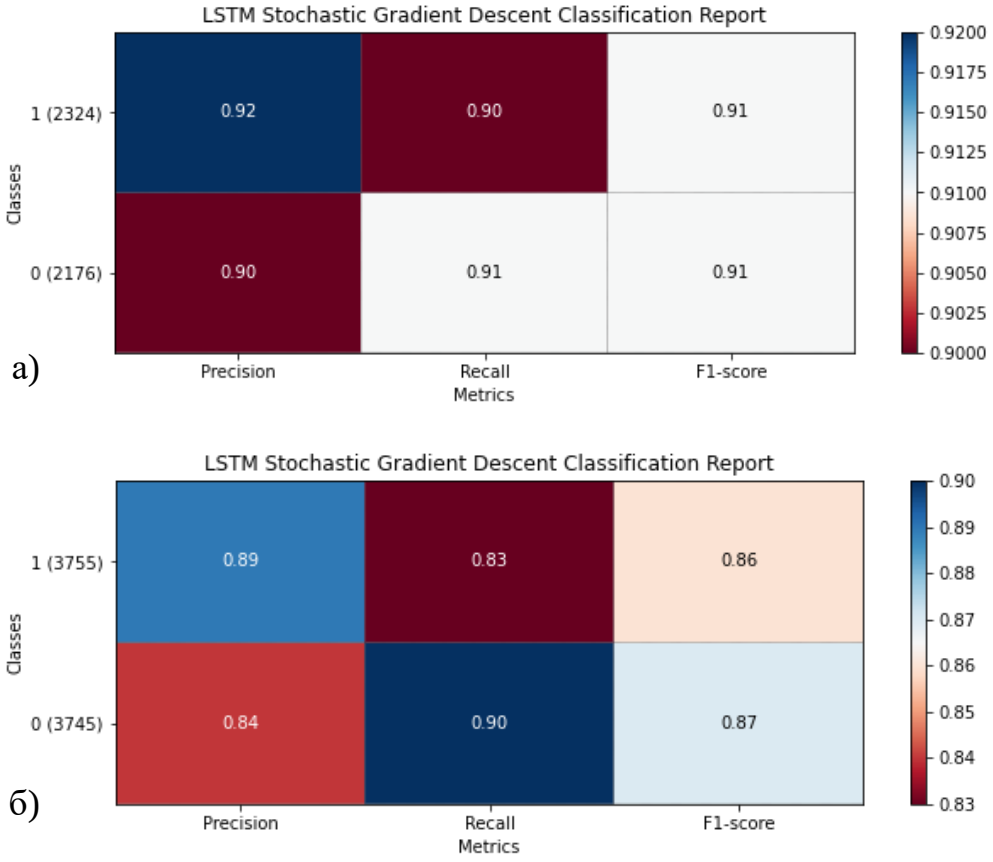
Класифікаційний звіт у вигляді теплокарти наведено на рис. 3.11.

Точність моделі описано в табл. 3.6.



а) б)

Рис. 3.10: *Втрати моделі ДКЧП (стохастичний градієнтний спуск) для англійського (а) та українського (б) набору даних*



а)

б)

Рис. 3.11: *Класифікаційний звіт для моделі ДКЧП (стохастичний градієнтний спуск) для англійського (а) та українського (б) набору даних*

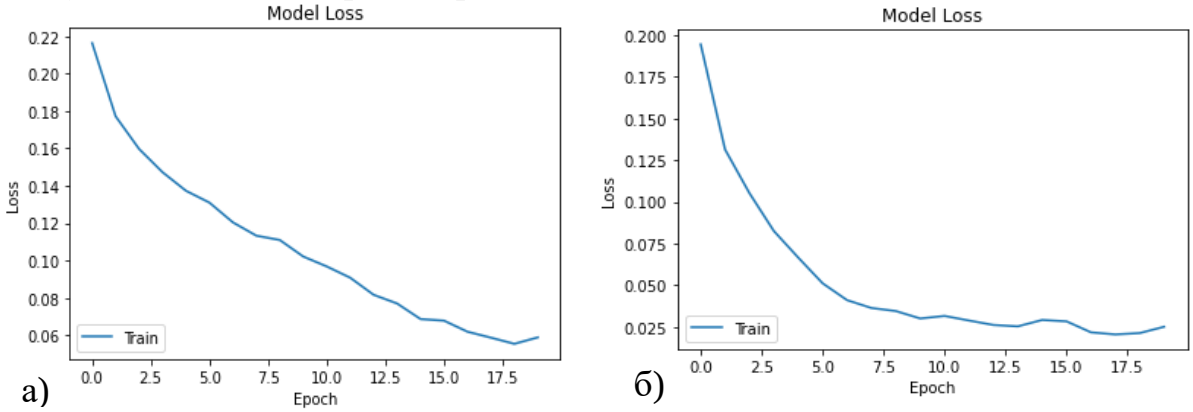
Табл. 3.6: *Точність моделі ДКЧП (стохастичний градієнтний спуск)*

Корпус даних	**Точність моделі**
Українська мова	*86.42%*
Англійська мова	*91.82%*

➢ **Модель ДКЧП (Міні-пакетний градієнтний спуск)**

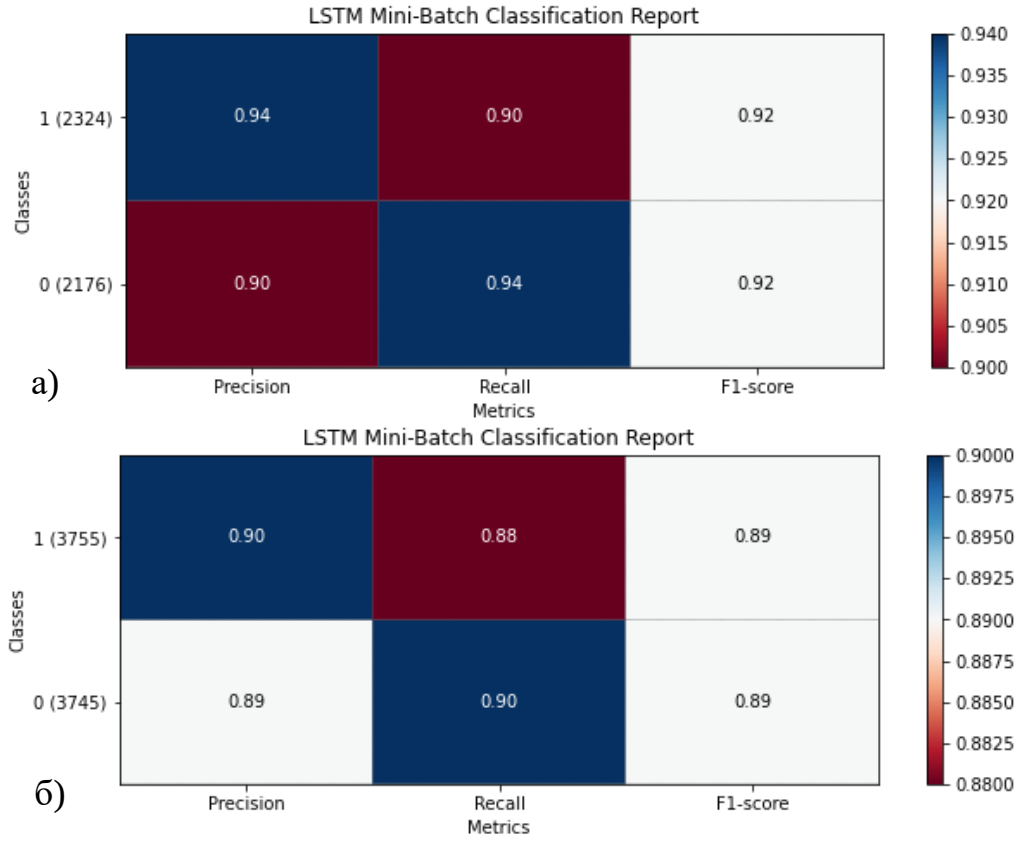
Дана модель має ті самі параметри, як і для випадку попередньої моделі, окрім розміру пакета, тут він встановлюється у 128 (дослідження показали, що для цієї моделі це найоптимальніший розмір).

Графік втрат для цієї моделі зображено на рис. 3.12. Класифікаційний звіт у вигляді теплокарти на рис. 3.13. Точність моделі описано в табл. 3.7.



а) б)

Рис. 3.12: *Втрати моделі ДКЧП (міні-пакетний градієнтний спуск) для англійського (а) та українського (б) набору даних*



а)

б)

Рис. 3.13: *Класифікаційний звіт для моделі ДКЧП (міні-пакетний градієнтний спуск) для англійського (а) та українського (б) набору даних*

Табл. 3.7: *Точність моделі ДКЧП (міні-пакетний градієнтний спуск)*

Корпус даних	Точність моделі
Українська мова	*88.48%*
Англійська мова	*90.67%*

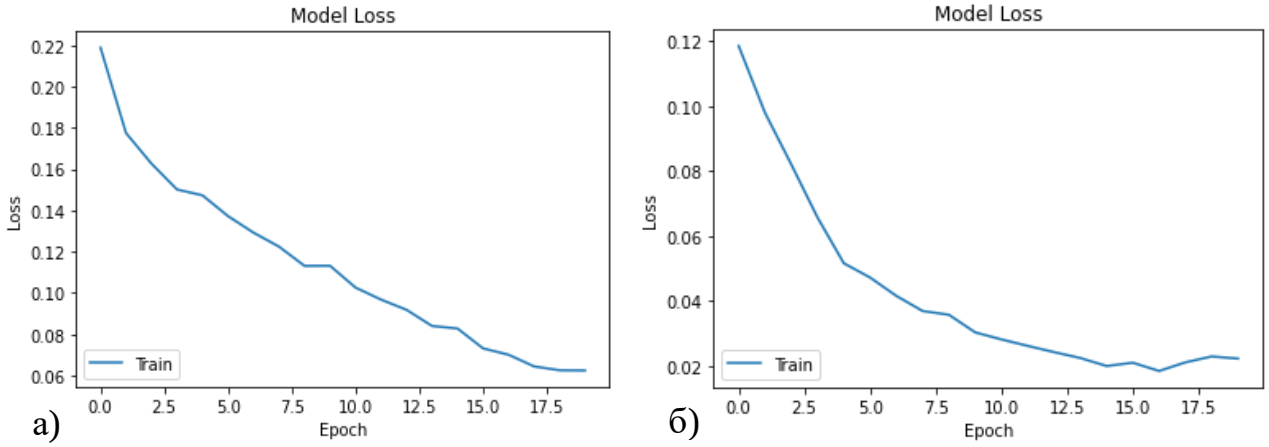
➢ **Складені шари ДКЧП**

Дана модель має ті самі конфігурації для тренування, шару вбудовування та шару щільності, що і в попередніх моделях. Розмір пакету також лишився 128. Вихідна розмірність першого шару ДКЧП становить 100 одиниць, для 2 та 3 по 32. Також перші 2 шари стеку ДКЧП мають параметр *return_sequence=True*, який повертає останній вихід у вихідній послідовності, а не повну послідовність.

Графік втрат для цієї моделі зображено на рис. 3.14.

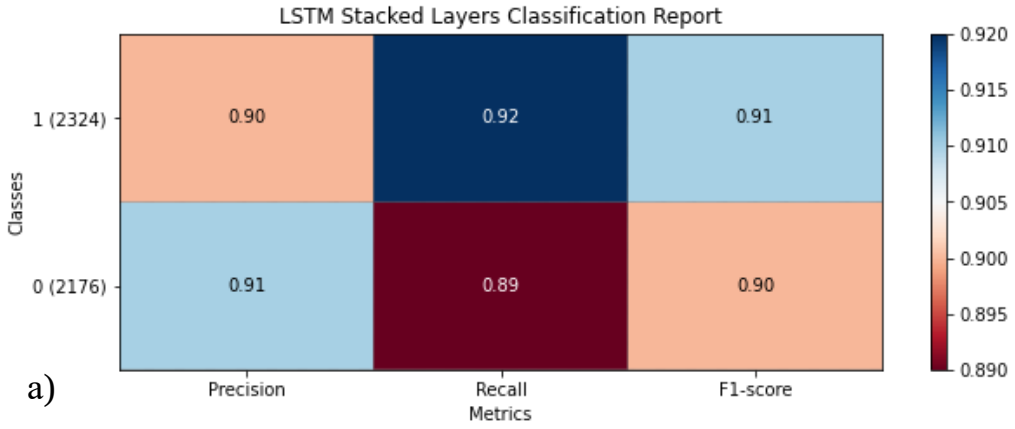
Класифікаційний звіт у вигляді теплокарти наведено на рис. 3.15.

Точність моделі описано в табл. 3.8.

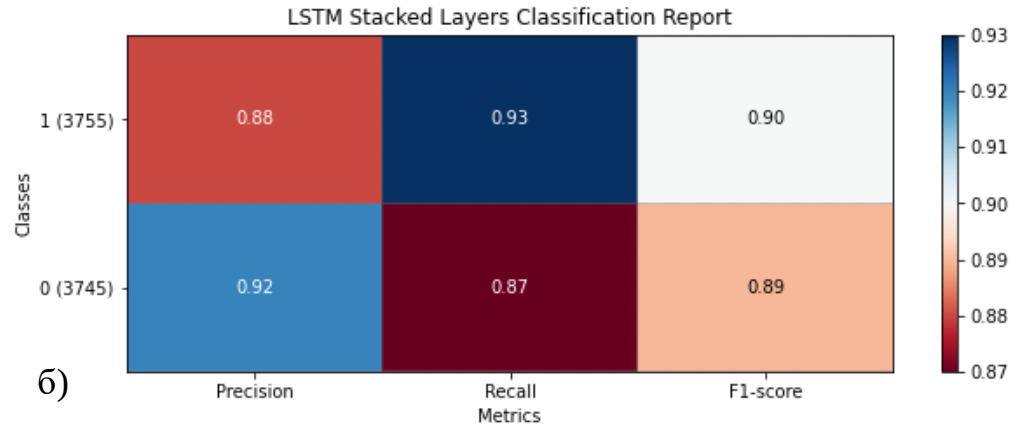


а) б)

Рис. 3.14: *Втрати моделі зі складених шарів ДКЧП для англійського (а) та українського (б) набору даних*



а)



б)

Рис. 3.15: *Класифікаційний звіт для моделі зі складених шарів ДКЧП для англійського (а) та українського (б) набору даних*

Табл. 3.8: *Точність моделі зі складених шарів ДКЧП*

Корпус даних	**Точність моделі**
Українська мова	*89.39%*
Англійська мова	*90.98%*

- **Двонаправлені ДКЧП**

Як і для попередніх моделей без змін лишаються налаштування для шару вбудовування та шару щільності.

Вихідна розмірність обох шарів двонаправленої ДКЧП становить 64 одиниці.

Також для даної моделі було додано шар виключення, мета якого полягає у випадковому встановленні частини вхідних одиниць на 0 у процесі кожного оновлення під час навчання, що допомагає запобігти перенавчанню моделі.

Оптимальним значенням параметру частки одиниць, які потрібно прибрати є 0.5 (50%) [56].

Для функції втрат при конфігурації тренування даної моделі була обрана двійкова перехресна ентропія [56].

Графік втрат для цієї моделі зображено на рис. 3.16.

Класифікаційний звіт у вигляді теплокарти наведено на рис. 3.17.

Точність моделі описано в табл. 3.9.

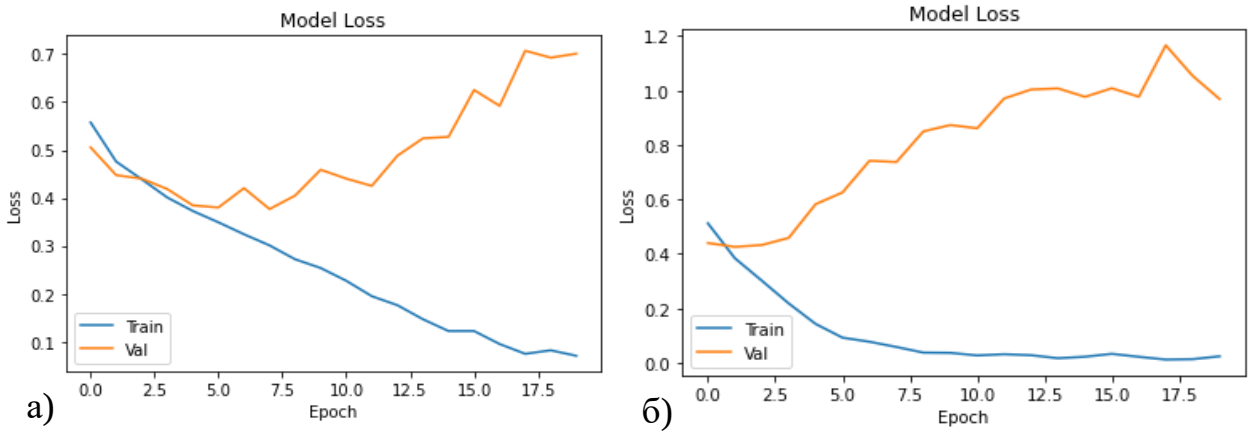


Рис. 3.16: *Втрати моделі двонаправленої ДКЧП для англійського (а) та українського (б) набору даних*

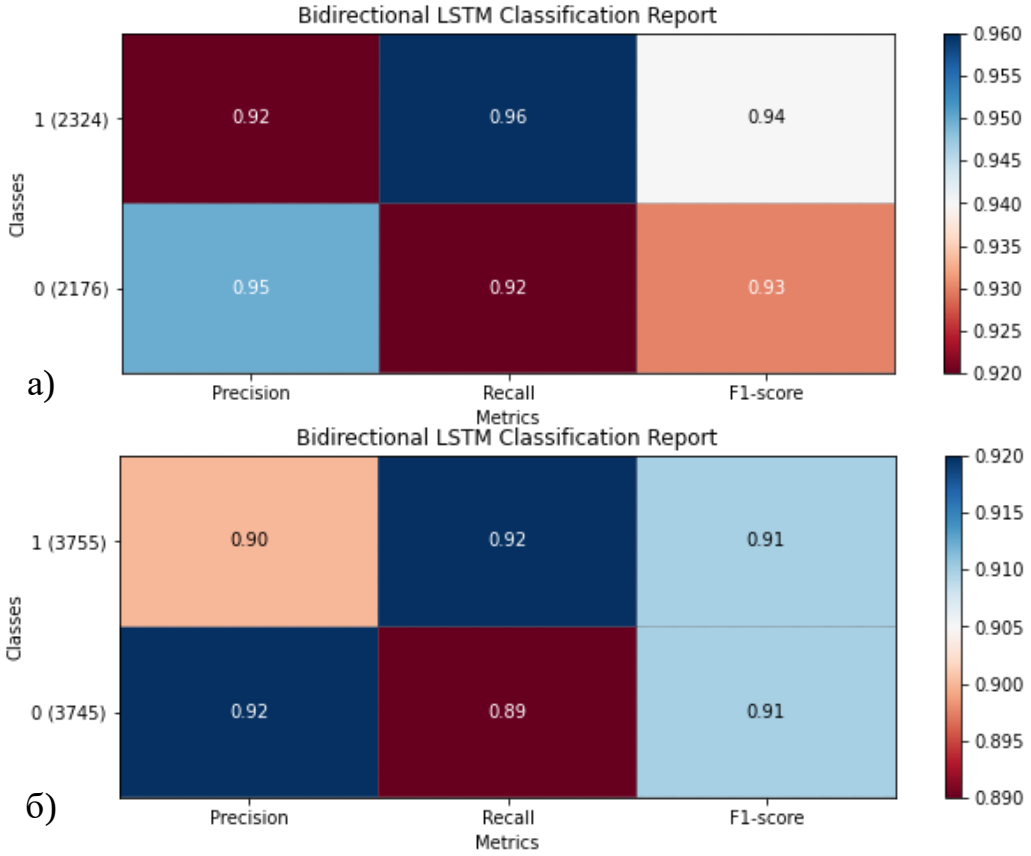


Рис. 3.17: *Класифікаційний звіт для моделі двонаправленої ДКЧП для англійського (а) та українського (б) набору даних*

Табл. 3.9: *Точність моделі двонаправленої ДКЧП*

Корпус даних	**Точність моделі**
Українська мова	*90.93%*
Англійська мова	*93.31%*

➢ **Комбінована мережа (ЗНМ-ДКЧП)**

Шари вбудовування та щільності мають залишаються без змін. Параметри для навчання, такі як у попередній моделі. Відмінності полягають у шарах згортки, ДКЧП, виключення та максимізаційного агрегування.

У згорткових шарах є 5 важливих параметрів: кількість згорткових ядер, довжина фільтра та активація. Кількість згорткових ядер обрана як 128, довжина фільтра 4, активація визначається як функція ReLU (шару зрізаних лінійних вузлів).

Шари *maxpooling* (максимізаційного агрегування) мають лише один параметр, який є довжиною пулу.

Довжина пулу визначається як розмір області, до якої застосовується максимізаційне агрегування. У даній архітектурі цьому параметру було присвоєно значення 3.

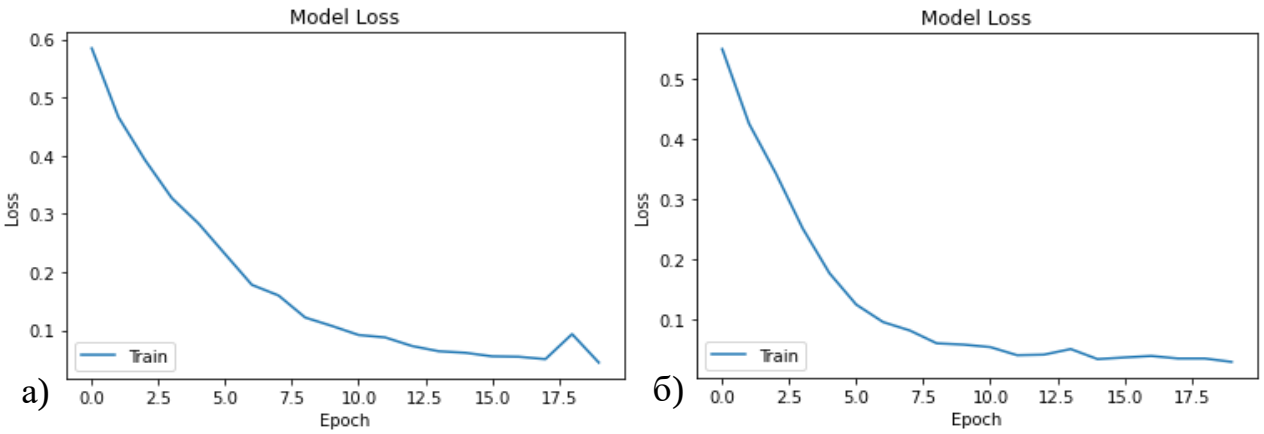
Вихідна розмірність шару ДКЧП: 32 одиниці.

Для шару виключення ми обрали значення 0.2 для параметру частки одиниць, які потрібно прибрати, оскільки при ньому точність виявилась кращою.

Графік втрат для цієї моделі зображено на рис. 3.18.

Класифікаційний звіт у вигляді теплокарти наведено на рис. 3.19.

Точність моделі описано в табл. 3.10.



а) б)

Рис. 3.18: *Втрати комбінованої моделі ЗНМ-ДКЧП для англійського (а) та українського (б) набору даних*

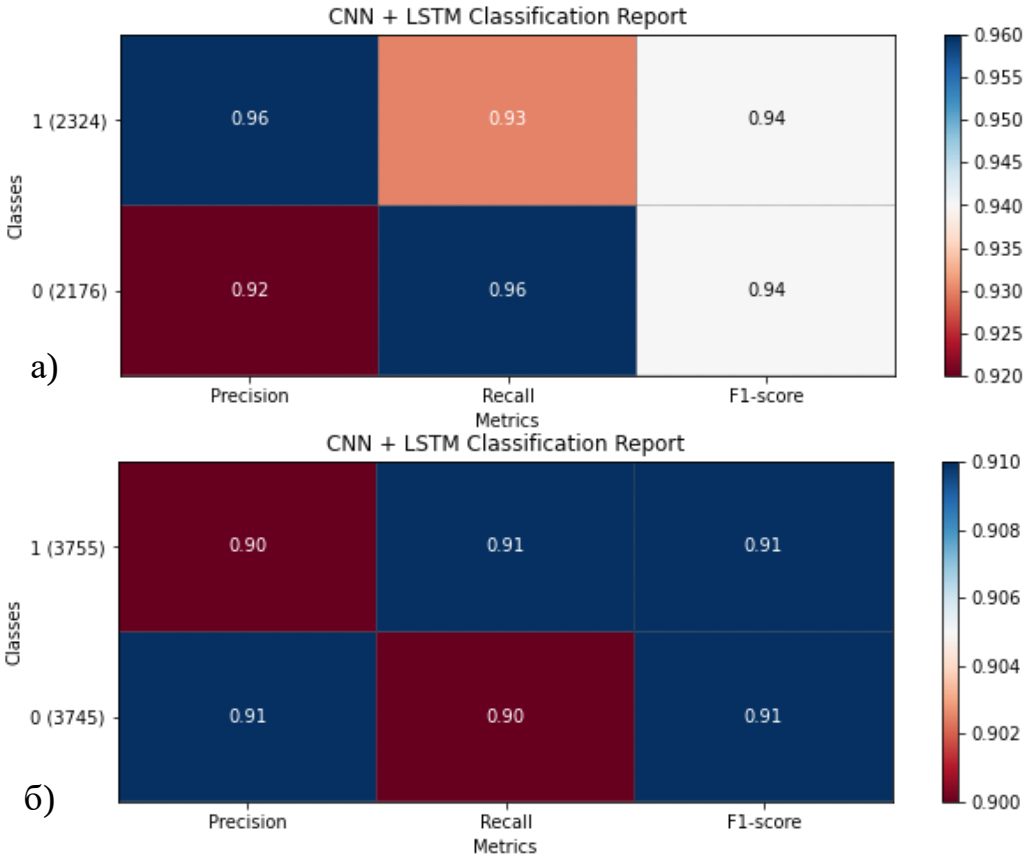


Рис. 3.19: *Класифікаційний звіт для комбінованої моделі ЗНМ-ДКЧП для англійського (а) та українського (б) набору даних*

Табл. 3.10: *Точність комбінованої моделі ЗНМ-ДКЧП*

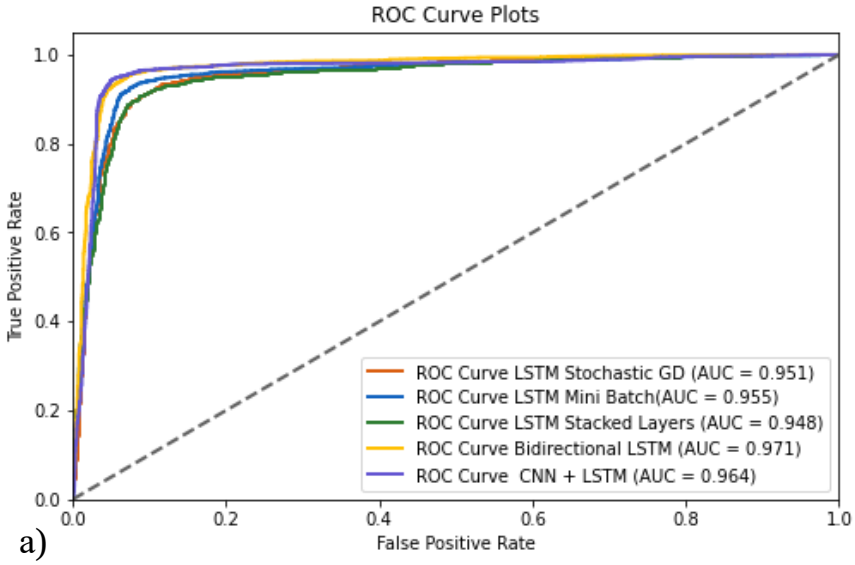
Корпус даних	**Точність моделі**
Українська мова	*90.15%*
Англійська мова	*94.48%*

Порівняння моделей глибинного навчання

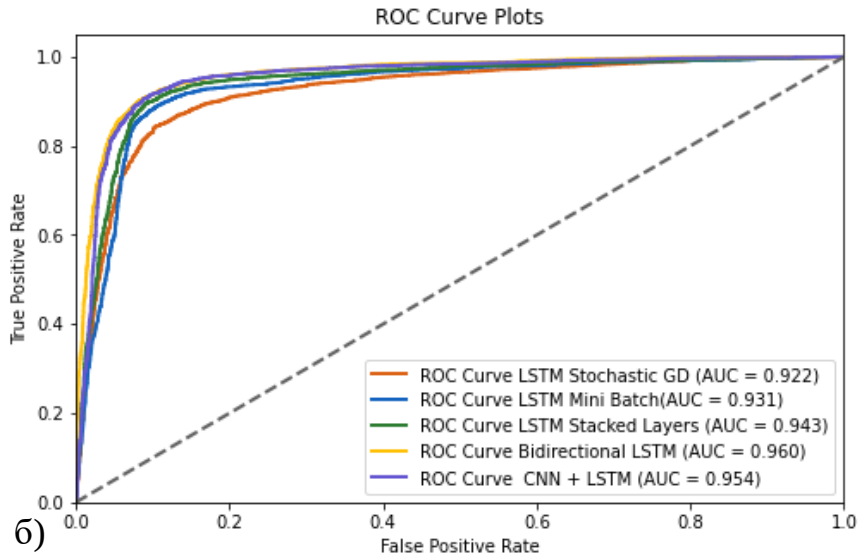
Порівняння для моделей глибинного навчання також базувалося на ROC-кривій, результати представлені на рис. 3.20.

Як ми бачимо, усі моделі показали неймовірно хороші результати, досягнувши 90-95% точності для всіх моделей і для обох наборів даних. Всі наші моделі можуть бути запущені у виробництво.

Звісно, англійські моделі показали трохи кращі результати, однак це тільки тому, що наш набір даних українською формувався перекладом оглядів англійською, що тягне за собою певні проблеми, які ми вже обговорювали.



а)



б)

Рис. 3.20: *Порівняння результатів моделей глибинного навчання за допомогою ROC-кривої для англійського (а) та українського (б) набору даних*

Моделі, які показали найкращі результати і для англійських, і для українських текстів – це двонаправлені ДКЧП та комбінована модель ЗНМ-ДКЧП, що не дивно, оскільки комбіновані моделі, як правило, дають найліпші результати. Так само і з моделями двонаправлених ДКЧП, які чудово показують себе на практиці [57, 58]. Самі ці 2 моделі ми використаємо для нашого веб-сервісу з сентимент-аналізу.

3.4. Розробка веб-сервісу

3.4.1. Опис функціональності проєкту

Як ми зазначали у вступі, мета нашого веб-сервісу полягає у аналізі настроїв ділових документів та зворотного зв’язку від студентів, це робиться для того, щоб дати викладачеві загальний висновок думки студентів щодо курсу чи матеріалу (якщо ми використовуємо зворотній зв’язок).

З іншої сторони, аналіз документів, наприклад, мотиваційних листів, надає ґрунтовну інформацію про настрій з яким писався даний документ, що краще допоможе розуміти відправника.

Сервіс може аналізувати документи українською та англійською мовою.

Загалом наш сервіс є багатокористувацьким та багатогруповим. Користувачі, які не зареєстровані, можуть заповнити форму (повне ім'я, електронна адреса, примітка (необов'язково), кому буде присвоєний документ (необов'язково)), після чого сформується унікальне посилання і користувач зможе завантажити свої документи (максимум 3) і подивитись згодом на коментарі викладача та результати аналізу його документів.

На веб-сервісі можна створювати безліч груп, кожна з яких буде мати принаймні одного адміністратора (викладача), групи можна створювати під конкретне завдання чи під потоки студентів.

Тільки адміністратор може створювати групи, проводити аналіз документів та писати коментарі.

3.4.2. Структура проєкту

Наш проєкт був реалізований за допомогою фреймворку Django і складається з 3 основних частин (функціональна схема проєкту наведена у Додатку 2):

- Базові налаштування та маршрутизація проєкту Django;
- Застосунок для аналізу настрою з яким буде взаємодіяти користувач, і в який входять необхідні утиліти для обробки вхідних документів, підготовки даних та сентимент-аналізу.
- Шаблони (HTML), а також необхідні стилі (CSS) для роботи веб-інтерфейсу.

Початкові налаштування

Для початку ми створили та активували віртуальне середовище Python, куди встановили описані на початку цього розділу бібліотеки для веб-сервісу. Після чого командою `django-admin startproject sentimento` створюємо каркас проєкту Django.

Каркас проєкту стандартно складається з наступного (рис. 3.21):

- ***manage.py***: адміністративна утиліта командного рядка Django для проєкту. Ми запускаємо адміністративні команди за допомогою `python manage.py <command> [options]`.
- Підтеки з іменем *sentimento*, яка містить наступні файли:
  - ***__init__.py***: порожній файл, який повідомляє Python, що ця папка є пакетом Python.
  - ***asgi.py***: точка входу для ASGI-сумісних веб-серверів для обслуговування нашого проєкту.
  - ***settings.py***: містить налаштування проєкту Django, які змінюються під час розробки веб-програми.
  - ***urls.py***: використовується для загальної маршрутизації в проєкті Django, також змінюється в процесі розробки.
  - ***wsgi.py***: точка входу для WSGI-сумісних веб-серверів для обслуговування проєкту. Цей файл лишається без змін

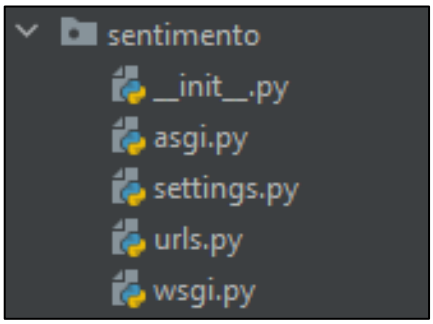


Рис. 3.21: *Стандартна структура основи Django проєкту*

Наступним кроком ми створили порожню базу даних для розробки виконавши команду `python manage.py migrate`. За замовчуванням створюється база даних SQLite, однак в нашому випадку ми прописали

необхідні налаштування для створення бази даних PostgesSQL, яка підтримується та сумісна з Heroku.

Сервер запускається командою `python manage.py runserver`, за замовчуванням працює на 8000 порту.

Застосунок з інтерфейсом користувача для аналізу настроїв

Щоб створити потрібний застосунок ми запустили команду: `python manage.py startapp analyzer` у кореневій папці нашого проєкту. Дана команда створює пакет з назвою analyzer, який містить декілька файлів коду та одну підтеку.

Серед файлів ми бачимо ***views.py*** (що містить функції для визначення сторінки у нашій веб-програмі), ***urls.py*** (маршрутизація всередині застосунку) та ***models.py*** (містить класи, що визначають наші об'єкти даних).

Підтека має назву ***migrations***, міграції використовується адміністративною утилітою Django для керування версіями бази даних.

Є також файли: ***apps.py*** (конфігурація програми), ***admin.py*** (для створення адміністративного інтерфейсу) і ***tests.py*** (для створення тестів). Також ми створили модулі ***forms.py*** (для роботи з формами HTML) та ***utils.py*** (містить утиліти для опрацювання документів, їх попередньої обробки).

У наступних підпунктах ми більш детально оглянемо найбільш важливі модулі, а саме ***models.py***, ***views.py*** та ***utils.py.***

- **Опис моделей бази даних**

Для потреб нашого сервісу ми створили 4 основні моделі, опис яких наведено нижче. Для аутентифікації, та загалом процесу реєстрації/логіну ми використали стандартний бекенд Django «з коробки» та модель AUTH_USER_MODEL.

- **Task** – для збереження інформації про завдання (документ, мотиваційний лист тощо), яке необхідно перевірити оцінювачу. Містить наступні сутності:
  - *USERNAME* – ім'я студента, який залишив завдання;

- *TASK_LIST* – зовнішній ключ на модель **TaskList**; містить дані до якого списку належить дане завдання;
- *EMAIL* – електронна пошта користувача;
- *CHECKED* – булеве значення, яке повідомляє чи було перевірене завдання;
- *CHECKED_DATE* – дата перевірки документа (встановлюється автоматично);
- *ASSIGNED_TO* – зовнішній ключ на модель **AUTH_USER_MODEL**, містить інформацію кому з оцінювачів було присвоєно завдання;
- *NOTE* – містить короткий опис до завдання (необов'язково);
- *PRIORITY* – пріоритет завдання (встановлюється викладачем, за замовчуванням 1).

- **TaskList** – для збереження завдань в певному списку. Містить наступні сутності:
  - *NAME* – назва списку;
  - *SLUG* – містить назву, зручну для сприйняття, яка буде використовуватись для веб-адреси даного списку (встановлюється автоматично);
  - *GROUP* – зовнішній ключ на модель **Group**, до якої буде присвоєно список завдань.
- **Comment** – для збереження коментарів під певним завданням. Містить наступні сутності:
  - *TASK* – зовнішній ключ на модель **Task**; містить інформацію про завдання до якого додано коментар;
  - *AUTHOR* – зовнішній ключ на модель **AUTH_USER_MODEL**; містить інформацію про те, який користувач залишив коментар;

- *DATE* – дата створення коментаря (чи його змінення, встановлюється автоматично);
- *BODY* – текст коментаря.

- **Attachment** – для збереження файлів, які будуть завантажені для конкретного завдання, містить наступні сутності:
  - *TASK* – зовнішній ключ на модель **Task**; містить інформацію про завдання до якого додано вкладення;
  - *TIMESTAMP* – дата завантаження вкладення (встановлюється автоматично);
  - *FILE* – зберігає в собі файли документів, які потім можна завантажувати та аналізувати.

Повна діаграма бази даних зображена у Додатку 1.

➢ **Опис функцій-обробників сторінок**

В нашому проєкті ми створили достатньо багато вузькопрофільних функції, що визначають сторінки (рис. 3.22), серед них:

- ***add_list.py*** – дозволяє користувачам додавати новий список до групи, в якій вони перебувають (лише оцінювачі) можуть додавати списки.
- ***del_list.py*** – для видалення всього списку з завданнями (доступ до цього мають лише оцінювачі).
- ***delete_task.py*** – використовується для видалення конкретного завдання. Перенаправляє до перегляду списку, в якому було завдання.
- ***import_csv.py*** – дозволяє імпортувати спеціально відформатований CSV до збережених завдань.
- ***list_detail.py*** – для відображення завдань у списку та керування ними.
- ***list_lists.py*** – домашня сторінка застосунку, відображає список списків завдань, які може переглядати користувач, для оцінювачів є можливість додавати нові списки.

- ***prediction.py*** – розроблено для передбачення аналізу настрою для даного вкладення PDF, повертає конкретний відсоток позитивності результату. Переспрямовує на сторінку прогнозу.
- ***remove_attachment.py*** – дозволяє видалити раніше розміщений об'єкт вкладення (якщо є необхідні дозволи).
- ***reorder_tasks.py*** – обробляє переупорядкування завдань (пріоритети).
- ***search.py*** – призначено для пошуку завдань, які користувач має дозвіл переглядати.
- ***task_detail.py*** – використовується для перегляду деталей завдання. Дозволяє редагувати деталі завдання вчителям та опрацьовувати нові коментарі до завдання.
- ***toggle_done.py*** – дозволяє перемикати статус завдання з «перевіреного» на «потребує перевірки» або навпаки. Перенаправляє до списку з якого надійшло завдання.

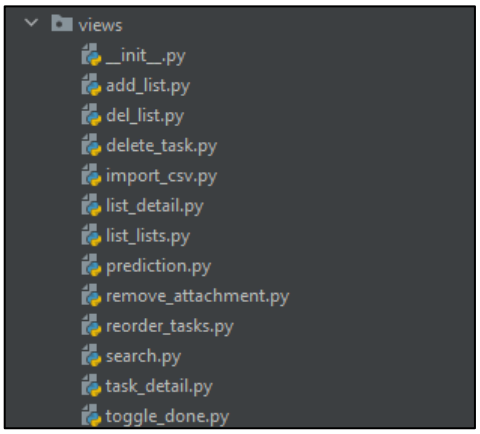


Рис. 3.22: *Структура функцій-обробників сторінок (views) для застосунку з сентимент-аналізу текстів*

➢ **Опис модуля утиліт для обробки документів та аналізу настроїв**

Даний модуль містить весь необхідний функціонал для сентимент-аналізу документу. Принцип роботи нашого сервісу для сентимент-аналізу документів описано у Додатку 3.

Для конвертації документів з PDF у звичайне текстове представлення ми використовували бібліотеку PDFMiner, для завантаження моделі глибинного

навчання застосовувався Keras, для роботи з word2vec – Gensim, а попередня обробка даних здійснювалася за допомогою бібліотек NLTK, PyMorphy та власних функцій.

Загальну структура нашого застосунку можна побачити на рис. 3.23

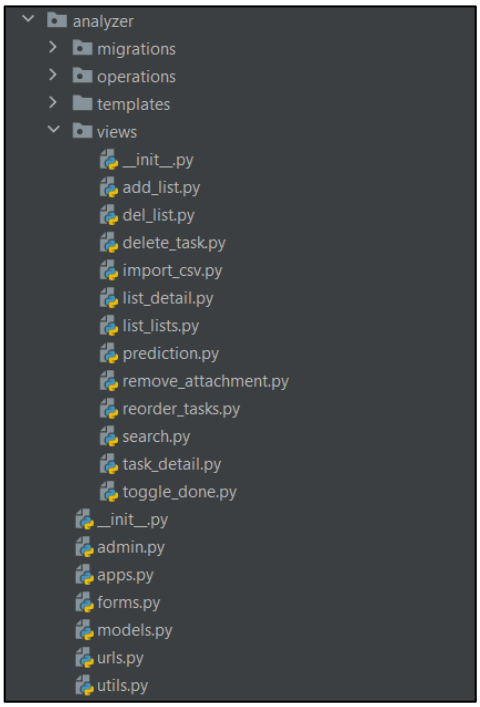


Рис. 3.23: *Структура застосунку для сентимент-аналізу текстів*

Шаблони веб-сторінок застосунку

У Django шаблон – це файл HTML, який містить заповнювачі для значень, які надає код під час виконання. Механізм шаблонів дбає про заміни під час візуалізації сторінки та забезпечує автоматичне екранування, щоб запобігти атакам XSS. Таким чином, код стосується лише значень даних, а шаблон – лише розмітки.

Шаблони Django надають гнучкі параметри, такі як успадкування шаблонів, що дозволяє визначити базову сторінку із загальною розміткою, а потім розширити цю базу за допомогою спеціальних доповнень.

Структура наших шаблонів наведена на рис. 3.24 (а). У папці static містяться необхідні статичні файли (стилі, скрипти і т.д.) для підтримки шаблонів; структура на рис. 3.24 (б).

Розглянемо детальніше наші шаблони:

- ***base.html*** – базова сторінка із загальною розміткою, включаючи навігаційну панель, рядок пошуку та імпортування скриптів, стилів.
- ***list_lists.html*** – дозволяє переглядати списки списків завдань розділених по групам.
- ***add_list.html*** – форма для додавання нового списку завдань.
- ***del_list.html*** – форма видалення списку завдань (разом з завданнями).
- ***list_detail.html*** – дозволяє переглядати усі завдання, що були призначені конкретному оцінювачу, при цьому оцінювач не має доступ до завдань інших оцінювачів.
- ***task_detail.html*** – дозволяє переглядати деталі завдання призначеного конкретному оцінювачу, доступний функціонал для видалення, створення, оновлення завдань та переведення їх у стан перевірених. Також тут можна встановлювати пріоритети завданням.
- ***task_edit.html*** – форма для створення (чи оновлення) завдання та прикріплення відповідних документів.
- ***predict.html*** – сторінка для відображення результатів сентимент-аналізу певного документу.
- ***import_csv.html*** – форма для завантаження нового завдання у CSV форматі.
- ***search_results.html*** – містить результати пошуку завдань призначених конкретному оцінювачу.

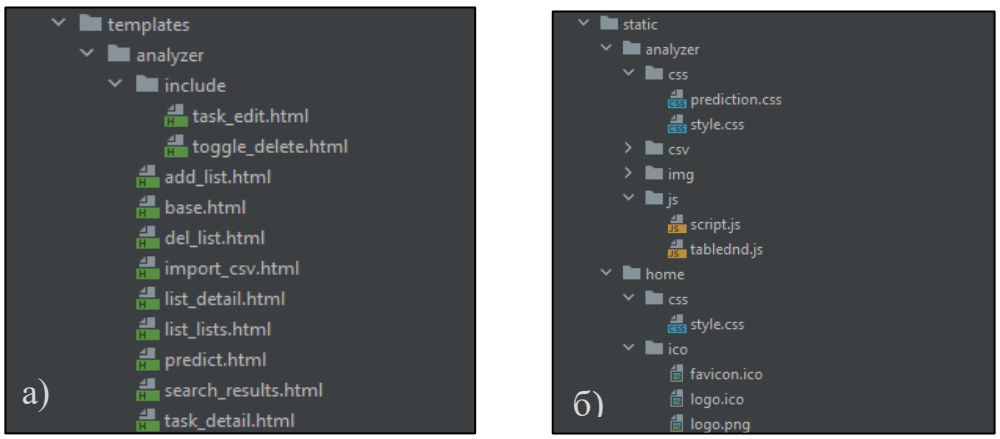


Рис. 3.24: *Структура шаблонів (а) та статичних файлів (б) проєкту*

3.4.3. Розгортання проєкту

Для розгортання нашого веб-сервісу ми обрали Heroku. Heroku – це платформа хмарних сервісів, популярність якої зросла в останні роки. Спочатку він підтримував лише програми Ruby, але тепер його можна використовувати для розміщення програм з багатьох програмних середовищ, включаючи Django.

Для розгортання проєкту, в першу чергу, нам потрібно було створити репозиторій додатку на GitHub, після чого додати файл з іменем *Procfile* у корені нашого GitHub репозиторію (файл необхідний для оголошення типи процесів та точки входу програми). Сервер веб-додатків у нашому випадку – це Gunicorn. Існує безліч способів обслуговування статичних файлів у виробничому середовищі, Heroku рекомендує використовувати WhiteNoise для цих задач, у нашому проєкті ми так і зробили.

Також нам був необхідний файл *requirements.txt*, який містить список бібліотек, які необхідні для запуску нашого сервісу, та файл *runtime.txt*, який повідомляє Heroku, яку мову програмування та її версію ми збираємось використовувати. Після чого наша попередня робота була завершена. Далі ми отримали акаунт Heroku, встановили клієнта і за допомогою команд `heroku create`, `git push heroku master` та `heroku run python manage.py migrate` запустили наш проєкт на сервері. На рис. 3.25 відображено інтерфейс Heroku, який показує приєднану базу даних, дані про розгортання проєкту та статус додатку.

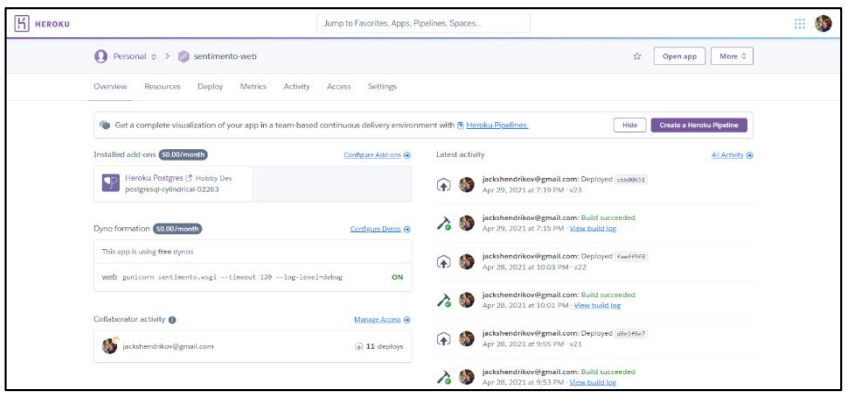


Рис. 3.25: *Інтерфейс Heroku*

3.4.4. Можливості користувачів різних рівнів

Користувачі нашого сервісу поділяються на 2 типи: звичайні користувачі (студенти) та адміністратори (викладачі, оцінювачі). У адміністраторів, природньо, більше можливостей, ніж у простих користувачів.

В нашому сервісі перед наданням комусь певних додаткових функцій проходить перевірка ролі користувача.

У табл. 3.11 описано основні можливості нашого додатку та кому вони доступні.

Табл. 3.11: *Можливості користувачів різних рівнів*

Можливість	Користувач	Адміністратор
Створювати групи та списки	-	+
Створення персональних сторінок у списках	+	+
Перегляд всіх доступних персональних сторінок	-	+
Редагування та видалення особистих сторінок	-	+
Пошук окремих особистих сторінок	-	+
Завантаження та видалення документів	+	+
Додавання коментарів	-	+
Читання коментарів	+	+
Додавання персональних сторінок у CSV форматі	-	+
Позначення особистих сторінок як опрацьованих	-	+
Проводити сентиментальний аналіз документів	-	+
Встановлювати пріоритет особистих сторінок	-	+

3.4.5. Тестування роботи веб-сервісу

Домашня сторінка нашого веб-додатку зображена на рис. 3.26.

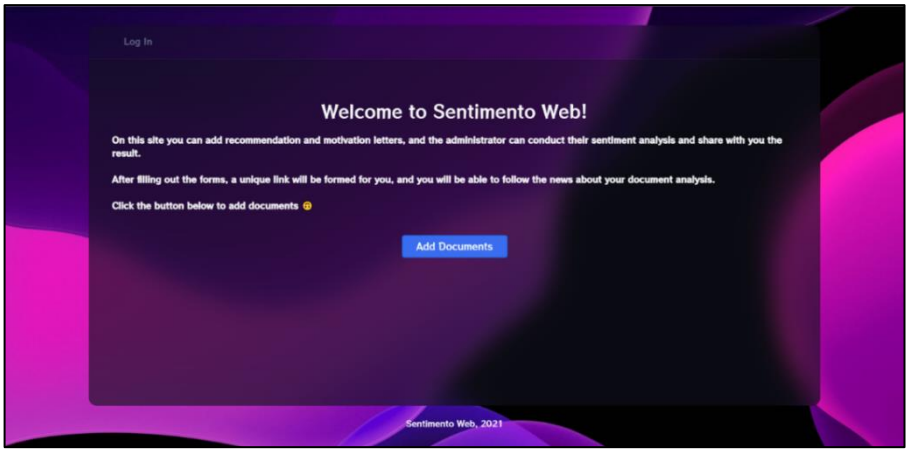


Рис. 3.26: *Домашня сторінка веб-сервісу для сентимент-аналізу*

Після натискання на кнопку «Додати документи» користувач отримає форму, де зможе ввести своє ім'я, пошту, кому призначено завдання, а також завантажити необхідні документи для перевірки (рис. 3.27).

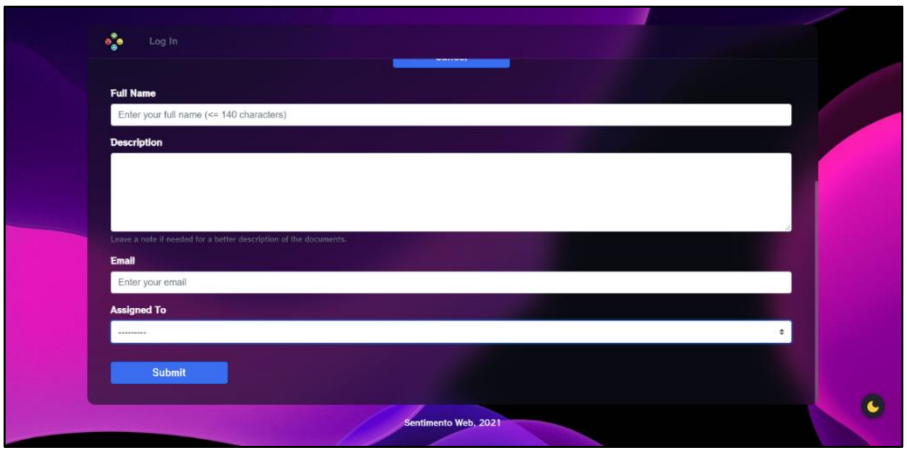


Рис. 3.27: *Сторінка для додачі задачі для перевірки*

Оцінювач, в свою чергу, після входження в систему, зможе побачити усі доступні йому списки завдань в групах, до яких він має доступ (рис. 3.28).

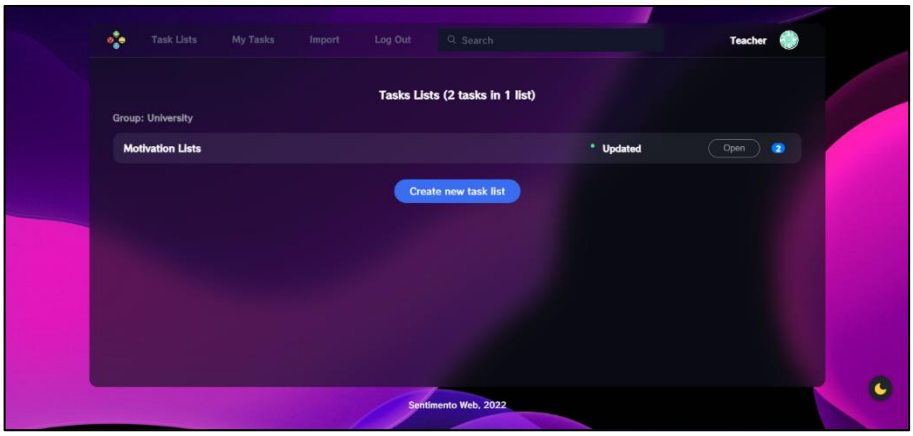


Рис. 3.28: *Приклад списків завдань у групі «Університет»*

В кожному списку наявні завдання, адресовані конкретному оцінювачу від користувачів (рис. 3.29).

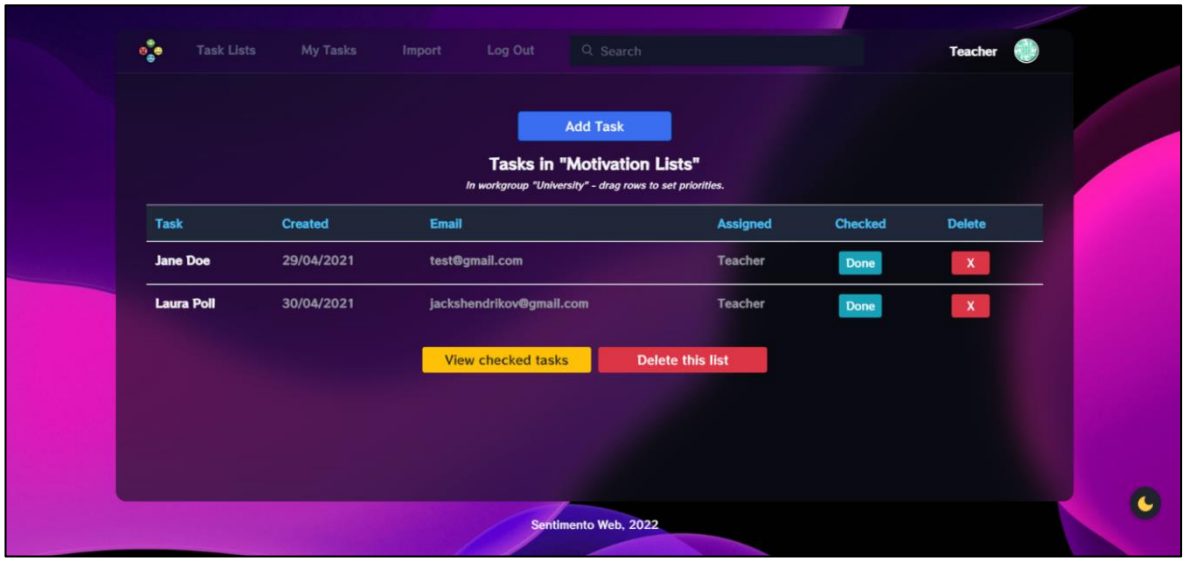


Рис. 3.29: *Приклад завдань у списку «Мотиваційні листи» групи «Університет»*

Оцінювач може обрати завдання і побачити детальну інформацію по ньому (рис. 3.30).

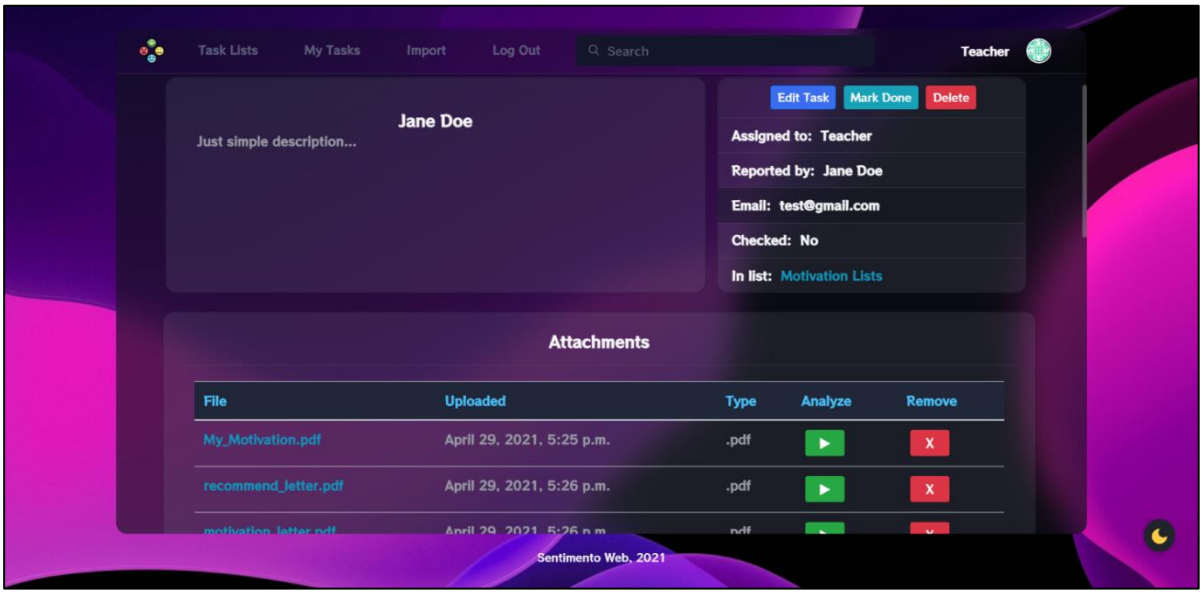


Рис. 3.30: *Вигляд детальної інформації по задачі від користувача*

Після чого оцінювач може запустити аналіз настрою обраного документи і отримати результат класифікації (рис. 3.31).

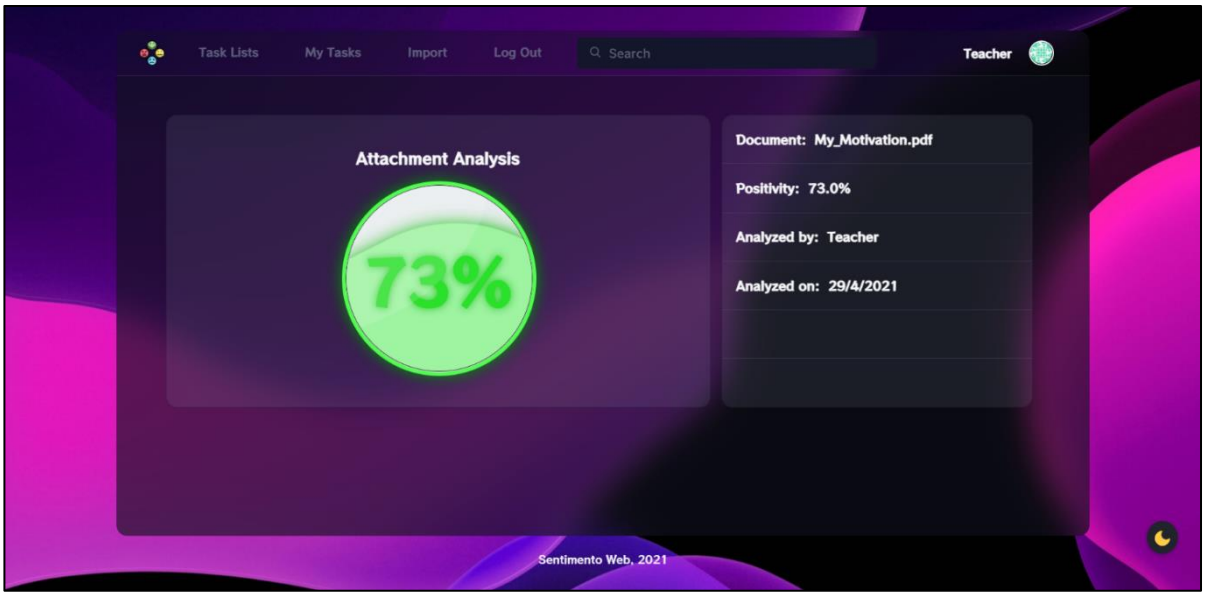


Рис. 3.31: *Приклад оцінки аналізу настрою для обраного документу*

За необхідності оцінювач може також завантажувати одразу декілька завдань в CSV форматі через вкладку «Імпорт CSV» (рис. 3.32).

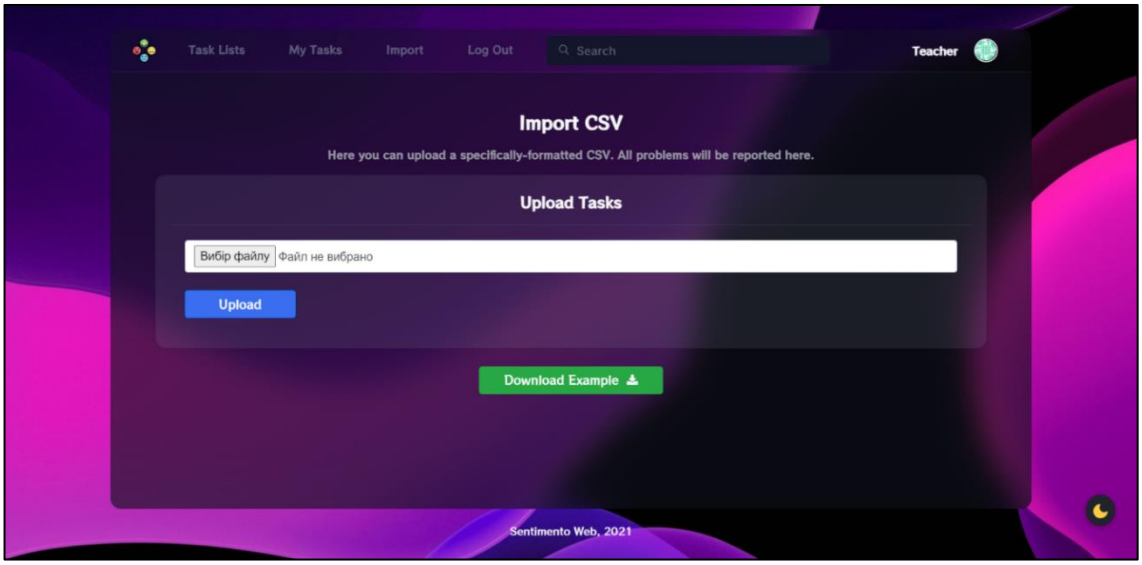


Рис. 3.32: *Сторінка імпорту завдання*

ВИСНОВОК ДО ТРЕТЬОГО РОЗДІЛУ

У цьому розділі ми здійснили опис використаного програмного забезпечення, а також реалізували, представили оціночні експерименти та зробили порівняльний аналіз для класифікаторів машинного навчання та моделей глибинного навчання, описаних у 2 розділі. Спочатку ми очистили наші дані, щоб вони були готові до обробки, використовуючи операції регулярного виразу, токенізацію, видалення стоп-слів та лематизацію.

Після чого набір даних був поділений на набори навчальних і тестових даних, та була застосована техніка перехресної перевірки, щоб випадковим чином розділити навчальний набір на k підрозділів, щоб запобігти перенавчанню.

Результати навчання показали, що найкращим класифікатором машинного навчання для українських та англійських даних є модель логістичної регресії. А найкращими моделями глибинного навчання стали комбінована модель ЗНМ-ДКЧП та двонаправлені ДКЧП. Такі результати близькі до найкращих, представлених у літературі.

Ми також розробили відповідний веб-сервіс для сентимент-аналізу на базі Django, де використали наші найкращі моделі. Був зроблений повний опис сервісу та протестовано весь його функціонал. Сервіс працює коректно, як і планувалося, тому було здійснено його розгортання на платформі Heroku.

Наступний розділ завершує даний дипломний проєкт представленням висновків щодо зробленої роботи, а також описом деяких напрямків щодо майбутньої роботи.

ВИСНОВКИ

В.1. Загальний висновок

На основі результатів тестів і навчання нейронних мереж було показано, що майже усі моделі глибинного навчання, що були використані в нашому проєкті можна використовувати для аналізу настроїв для українських та англійських текстів.

Класифікатором машинного навчання, що дав найліпші результати стала модель логістичної регресії, яка показала точність 80.58% для даних англійською та 77.7% для даних українською.

В свою чергу, комбінована модель ЗНМ-ДКЧП та двонаправлені ДКЧП стали найкращими моделями глибинного навчання, і дали нам точність в приблизно 95% для англійської та 91% для української моделі.

Оскільки моделі показують дійсно хороший результат як для тренувальних, так і для тестових даних, це вказує на те, що мережі чудово навчилися з початкового набору даних і можуть бути використані для аналізу нових даних в реальному часі.

Тому ми використали наші моделі при створенні веб-сервісу для аналізу настроїв мотиваційних листів та зворотного зв'язку, що має покращити та полегшити роботу викладачів, оскільки вони зможуть отримувати необхідний відгук щодо їх роботи та настрою студентів зручно та швидко.

Великий огляд літератури не виявив жодного додатку чи дослідження, яке б вивчало аналіз настроїв для мотиваційних листів та зворотного зв'язку студентів у реальному часі для української мови, тому наш сервіс цілком може стати прикладом для наслідування.

Крім того, відомо дуже мало українських моделей для сентимент-аналізу, тому наші наробітки, які показали чудові результати і тренувались на даних різної тематики можуть бути застосовані майже до будь-чого.

В.2. Обмеження та подальша робота

У моделях передбачення неможливо охопити всі можливості, однак існує кілька способів компенсації цього. Розширення наборів даних шляхом включення більшої кількості негативних та позитивних оглядів є одним з таких способів і має ще більше покращити результати класифікації.

Якість запропонованої системи можна також підвищити за рахунок виконання більш детального процесу аналізу настроїв на рівні аспектів. Цей підхід зосереджується на аналізі настроїв у конкретних аспектах, а не для цілого речення.

Створення індивідуальних вкладень слів специфічних для кожної предметної області повинно також мати позитивний вплив на процес класифікації, однак це доволі складний і тривалий процес.

Інша річ, яка може вплинути на результат, – це розробка іншої архітектури мереж. Наприклад, додавання випадаючих шарів в більшій кількості моделей теоретично може збільшити її точність на деякий відсоток.

Ще одна проблема, яка може виникнути в майбутньому, – це нові слова в українській мові та збільшення використання української. Щоб підтримувати моделі в актуальному стані, їх потрібно перенавчати новим даним у майбутньому, щоб не відставати від суспільства та використання мови.

На завершення, якби було більше часу, було б корисно додати в наші моделі можливість розпізнавати сарказм, стійкість думки, а в більшій перспективі – аналізувати емоції, а не тільки настрої.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

[1] Asur, S., Huberman, B. A. (2010). Predicting the Future with Social Media. [Електронний ресурс] – Режим доступу: https://arxiv.org/pdf/1003.5699.pdf

[2] Gesenhues A. (2013). 90% Of Customers Say Buying Decisions Are Influenced By Online Reviews [Електронний ресурс] – Режим доступу: https://martech.org/survey-customers-more-frustrated-by-how-long-it-takes-to-resolve-a-customer-service-issue-than-the-resolution/

[3] Blorm, C. (2017). 84 Percent of People Trust Online Reviews As Much As Friends. [Електронний ресурс] – Режим доступу: https://www.inc.com/craig-bloem/84-percent-of-people-trust-online-reviews-as-much-.html

[4] Mohammad S.M., Salameh M. (2016). How Translation Alters Sentiment. [Електронний ресурс] – Режим доступу: https://www.researchgate.net/publication/294090330_How_Translation_Alters_Sentiment

[5] «Sentiment» Merriam-Webster.com Dictionary, Merriam-Webster (2022). [Електронний ресурс] – Режим доступу: https://www.merriam-webster.com/dictionary/sentiment

[6] Liu B. (2012). Sentiment Analysis and Opinion Mining. [Електронний ресурс] – Режим доступу: https://www.cs.uic.edu/~liub/FBS/SentimentAnalysis-and-OpinionMining.pdf

[7] Farzindar A., Inkpen D. (2015). Natural Language Processing for Social Media. [Електронний ресурс] – Режим доступу: https://www.morganclaypoolpublishers.com/catalog_Orig/samples/9781681736136_sample.pdf

[8] Times of India (2019). Why Steve Jobs thought Microsoft was like McDonald's. [Електронний ресурс] – Режим доступу: https://bit.ly/3PPutiy

[9] Відгуки про компанію KindGeek. [Електронний ресурс] – Режим доступу: https://jobs.dou.ua/companies/kindgeek/reviews/

[10] Turney P. D. (2002). Thumbs Up or Thumbs Down? Semantic Orientation Applied to Unsupervised Classification of Reviews. [Електронний ресурс] – Режим доступу: https://arxiv.org/ftp/cs/papers/0212/0212032.pdf

[11] Yang B., Cardie C. (2014). Context-aware Learning for Sentence-level Sentiment Analysis with Posterior Regularization. [Електронний ресурс] – Режим доступу: https://aclanthology.org/P14-1031.pdf

[12] Liu B., Hu M. (2004). Mining and Summarizing Customer Reviews. [Електронний ресурс] – Режим доступу: https://www.cs.uic.edu/~liub/publications/kdd04-revSummary.pdf

[13] Wilson T., Wiebe J., Hoffmann P. (2005). Recognizing Contextual Polarity in Phrase-Level Sentiment Analysis. [Електронний ресурс] – Режим доступу: https://dl.acm.org/doi/pdf/10.3115/1220575.1220619

[14] Wilson T., Wiebe J., Hwa R. (2004). Just how mad are you? Finding strong and weak opinion clauses. [Електронний ресурс] – Режим доступу: https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.5.2078&rep=rep1&type=pdf

[15] Chen M., Wang S, Baltrušaitis T., Zadeh A., (2019). Multimodal Sentiment Analysis with Word-Level Fusion and Reinforcement Learning. [Електронний ресурс] – Режим доступу: https://arxiv.org/pdf/1802.00924.pdf

[16] D'Andrea A., Ferri F., Grifoni P., Guzzo T. (2015). Approaches, Tools and Applications for Sentiment Analysis Implementation. [Електронний ресурс] – Режим доступу: https://www.ijcaonline.org/research/volume125/number3/dandrea-2015-ijca-905866.pdf

[17] King T. (2019). 80 Percent of Your Data Will Be Unstructured in Five Years. [Електронний ресурс] – Режим доступу: https://solutionsreview.com/data-management/80-percent-of-your-data-will-be-unstructured-in-five-years/

[18] Jianqiang, Z., Xiaolin, G. (2017). Comparison Research on Text Pre-processing Methods on Twitter Sentiment Analysis. [Електронний ресурс] –

Режим доступу: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7862202

[19] Toman M., Tesar R., Jezek K. (2014). Influence of Word Normalization on Text Classification. [Електронний ресурс] – Режим доступу: http://textmining.zcu.cz/publications/inscit20060710.pdf

[20] Esuli A., Sebastiani F. (2006). Sentiwordnet: a publicly available lexical resource for opinion mining. [Електронний ресурс] – Режим доступу: https://www.esuli.it/publications/LREC2006.pdf

[21] Hutto C., Gilbert E. (2014). VADER: A Parsimonious Rule-Based Model for Sentiment Analysis of Social Media Text. [Електронний ресурс] – Режим доступу: https://ojs.aaai.org/index.php/ICWSM/article/view/14550/14399

[22] Hatzivassiloglou V., McKeown K. R. (1997). Predicting the semantic orientation of adjectives. [Електронний ресурс] – Режим доступу: https://dl.acm.org/doi/pdf/10.3115/976909.979640

[23] Kanayama H., Nasukawa T. (2006). Fully automatic lexicon expansion for domain-oriented sentiment analysis. [Електронний ресурс] – Режим доступу: https://dl.acm.org/doi/pdf/10.5555/1610075.1610125

[24] Augustyniak L., Szymański P., Kajdanowicz T., Tuligłowicz W. (2015). Comprehensive Study on Lexicon-based Ensemble Classification Sentiment Analysis. [Електронний ресурс] – Режим доступу: https://www.mdpi.com/1099-4300/18/1/4/htm

[25] Pang B., Lee L., Vaithyanathan S. (2002). Thumbs up? Sentiment Classification using Machine Learning Techniques. [Електронний ресурс] – Режим доступу: https://aclanthology.org/W02-1011.pdf

[26] Tan S., Cheng X., Wang Y., Xu H. (2009). Adapting Naive Bayes to Domain Adaptation for Sentiment Analysis. [Електронний ресурс] – Режим доступу: https://www.researchgate.net/publication/221397247_Adapting_Naive_Bayes_to_Domain_Adaptation_for_Sentiment_Analysis

[27] Mullen T., Collier N. (2004). Sentiment analysis using support vector machines with diverse information sources. [Електронний ресурс] – Режим доступу: https://aclanthology.org/W04-3253.pdf

[28] Mehra N., Khandelwal S., Patel P. (2002). Sentiment Identification Using Maximum Entropy Analysis of Movie Reviews [Електронний ресурс] – Режим доступу: https://web.stanford.edu/class/cs276a/projects/reports/nmehra-kshashi-priyank9.pdf

[29] Liu B., Li X., Lee W.S., Yu P.S. (2004). Text classification by labeling words. [Електронний ресурс] – Режим доступу: https://www.cs.uic.edu/~liub/publications/aaai04-labelWords.pdf

[30] Prabowo R., Thelwall M. (2009). Sentiment Analysis: A combined Approach. [Електронний ресурс] – Режим доступу: http://www.cyberemotions.eu/rudy-sentiment-preprint.pdf

[31] Sebastiani F. (2002). Machine Learning in Automated Text Categorization. [Електронний ресурс] – Режим доступу: https://arxiv.org/pdf/cs/0110053.pdf

[32] Wang L. (2005). Support Vector Machines: Theory and Applications [Електронний ресурс] – Режим доступу: https://personal.ntu.edu.sg/elpwang/PDF_web/05_SVM_basic.pdf

[33] Joachims T. (1998) Text Categorization with Support Vector Machines: Learning with many relevant features. [Електронний ресурс] – Режим доступу: https://www.cs.cornell.edu/people/tj/publications/joachims_98a.pdf

[34] Platt J. (1998). Sequential Minimal Optimization: A Fast Algorithm for Training Support Vector Machines. [Електронний ресурс] – Режим доступу: https://web.iitd.ac.in/~sumeet/tr-98-14.pdf

[35] Cunningham P., Delany S.J. (2007). K-Nearest Neighbour Classifiers. [Електронний ресурс] – Режим доступу: https://arxiv.org/pdf/2004.04523.pdf

[36] Imandoust S.B., Bolandraftar M. (2013). Application of K-Nearest Neighbor (KNN) Approach for Predicting Economic Events: Theoretical

Background. [Електронний ресурс] – Режим доступу: https://www.ijera.com/papers/Vol3_issue5/DI35605610.pdf

[37] Park H.A. (2013). An Introduction to Logistic Regression: From Basic Concepts to Interpretation with Particular Attention to Nursing Domain. [Електронний ресурс] – Режим доступу: https://synapse.koreamed.org/upload/synapsedata/pdfdata/0006jkan/jkan-43-154.pdf

[38] Kleinbaum D.G., Klein M. (2010). Logistic Regression: A Self-Learning Text. [Електронний ресурс] – Режим доступу: https://www.pdfdrive.com/logistic-regression-a-self-learning-text-third-edition-e19557975.html

[39] Breiman L. (2001). Random Forest. [Електронний ресурс] – Режим доступу: https://www.stat.berkeley.edu/~breiman/randomforest2001.pdf

[40] Biau G., Scornet E. (2016). A Random Forest Guided Tour. [Електронний ресурс] – Режим доступу: https://www.normalesup.org/~scornet/paper/test.pdf

[41] Liaw A., Wiener M. (2002). Classification and Regression by Random Forest. [Електронний ресурс] – Режим доступу: https://cogns.northwestern.edu/cbmg/LiawAndWiener2002.pdf

[42] Goldberg Y., Hirst G. (2017). Neural Network Methods in Natural Language Processing. [Електронний ресурс] – Режим доступу: https://bit.ly/3MRVD5Q

[43] Pascanu R., Mikolov T., Bengio Y. (2012). On the difficulty of training Recurrent Neural Networks. [Електронний ресурс] – Режим доступу: https://arxiv.org/pdf/1211.5063.pdf

[44] Werbos P. J. (1998). Generalization of backpropagation with application to a recurrent gas market model. [Електронний ресурс] – Режим доступу: https://www.academia.edu/48833987/Generalization_of_backpropagation_with_application_to_a_recurrent_gas_market_model

[45] Hochreiter S., Schmidhuber J. (1997). Long Short-term Memory [Електронний ресурс] – Режим доступу: http://www.bioinf.jku.at/publications/older/2604.pdf

[46] Albawi S., Mohammed T.A, Al-Zawi S. (2017). Understanding of a Convolutional Neural Network. [Електронний ресурс] – Режим доступу: https://www.researchgate.net/publication/319253577_Understanding_of_a_Convolutional_Neural_Network

[47] Wu J., Hu W., Xiong H., Huan J., Braverman V., Zhu Z. (2019). On the Noisy Gradient Descent that Generalizes as SGD. [Електронний ресурс] – Режим доступу: https://arxiv.org/pdf/1906.07405.pdf

[48] Kandel I., Castelli M. (2020). The effect of batch size on the generalizability of the convolutional neural networks on a histopathology dataset. [Електронний ресурс] – Режим доступу: https://www.researchgate.net/publication/341173194_The_effect_of_batch_size_on_the_generalizability_of_the_convolutional_neural_networks_on_a_histopathology_dataset

[49] Opitz D., Maclin R. (1999). Popular ensemble methods: An empirical study. [Електронний ресурс] – Режим доступу: https://arxiv.org/abs/1106.0257

[50] Sainath T.N., Vinyals O., Senior A., Sak H. (2015). Convolutional, Long Short-Term Memory, fully connected Deep Neural Networks. [Електронний ресурс] – Режим доступу: https://static.googleusercontent.com/media/research.google.com/en//pubs/archive/43455.pdf

[51] Blitzer J., Dredze M., Pereira F. (2007). Biographies, Bollywood, Boom-boxes and Blenders: Domain Adaptation for Sentiment Classification. [Електронний ресурс] – Режим доступу: https://aclanthology.org/P07-1056.pdf

[52] Колекція стоп-слів для української мови. [Електронний ресурс] – Режим доступу: https://github.com/stopwords-iso/stopwords-uk

[53] Колекція попередньо навчених векторів Word2Vec. [Електронний ресурс] – Режим доступу: https://code.google.com/archive/p/word2vec/

[54] Колекція попередньо навчених векторів Word2Vec для української мови. [Електронний ресурс] – Режим доступу: https://lang.org.ua/uk/models/

[55] Колекція попередньо навчених векторів GloVe. [Електронний ресурс] – Режим доступу: https://www.kaggle.com/datasets/anindya2906/glove6b

[56] Trains a Bidirectional LSTM on the IMDB sentiment classification task. [Електронний ресурс] – Режим доступу: https://keras.io/zh/examples/imdb_bidirectional_lstm/

[57] Cui Z., Ke R., Pu Z., Wang Y. (2019). Deep Bidirectional and Unidirectional LSTM Recurrent Neural Network for Network-wide Traffic Speed Prediction. [Електронний ресурс] – Режим доступу: https://arxiv.org/abs/1801.02143.

[58] Chiu J.P.C., Nichols E. (2016). Named Entity Recognition with Bidirectional LSTM-CNNs. [Електронний ресурс] – Режим доступу: https://aclanthology.org/Q16-1026.pdf

ДОДАТОК 1

Сервіс аналізу емоційного забарвлення текстових повідомлень

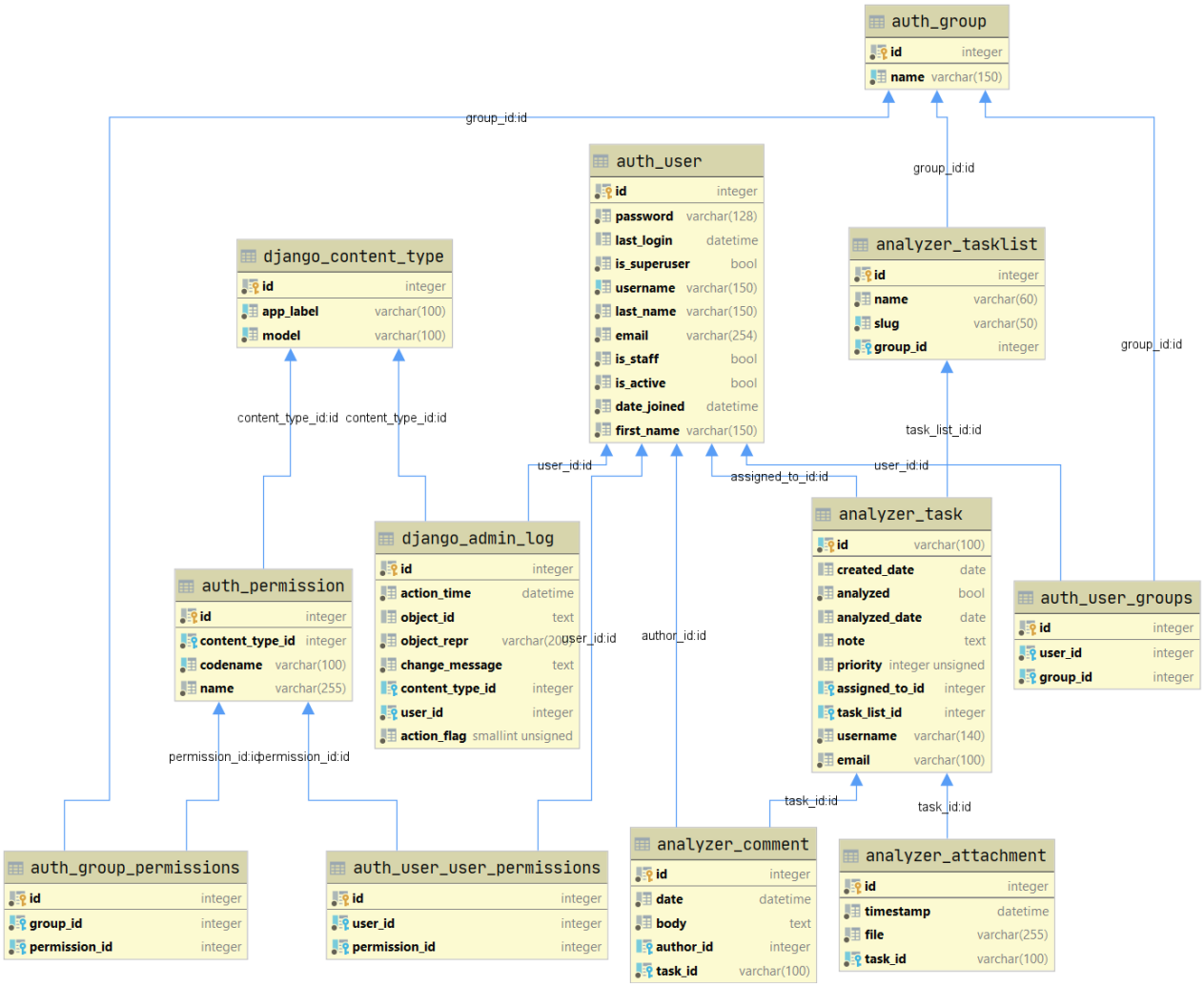
Структурна схема

Діаграма бази даних

ІАЛЦ.467800.004 Д1

Аркушів 1

Київ – 2022 року



Зм.	Арк.	№ докум.	Підпис	Дата	ІАЛЦ.467800.004 Д1		
Розробив		Шендріков Є.О.			Сервіс аналізу емоційного забарвлення текстових повідомлень	Літ.	Аркуш 1 / Аркушів 1
Перевірив		Болдак А.О.					
Реценз.					Діаграма бази даних (Структурна схема)	«КПІ імені Ігоря Сікорського» ФІОТ, гр. ІО-82	
Н. Контр.		Сімоненко В.П.					
Затв.		Стіренко С.Г.					

ДОДАТОК 2

Сервіс аналізу емоційного забарвлення текстових повідомлень

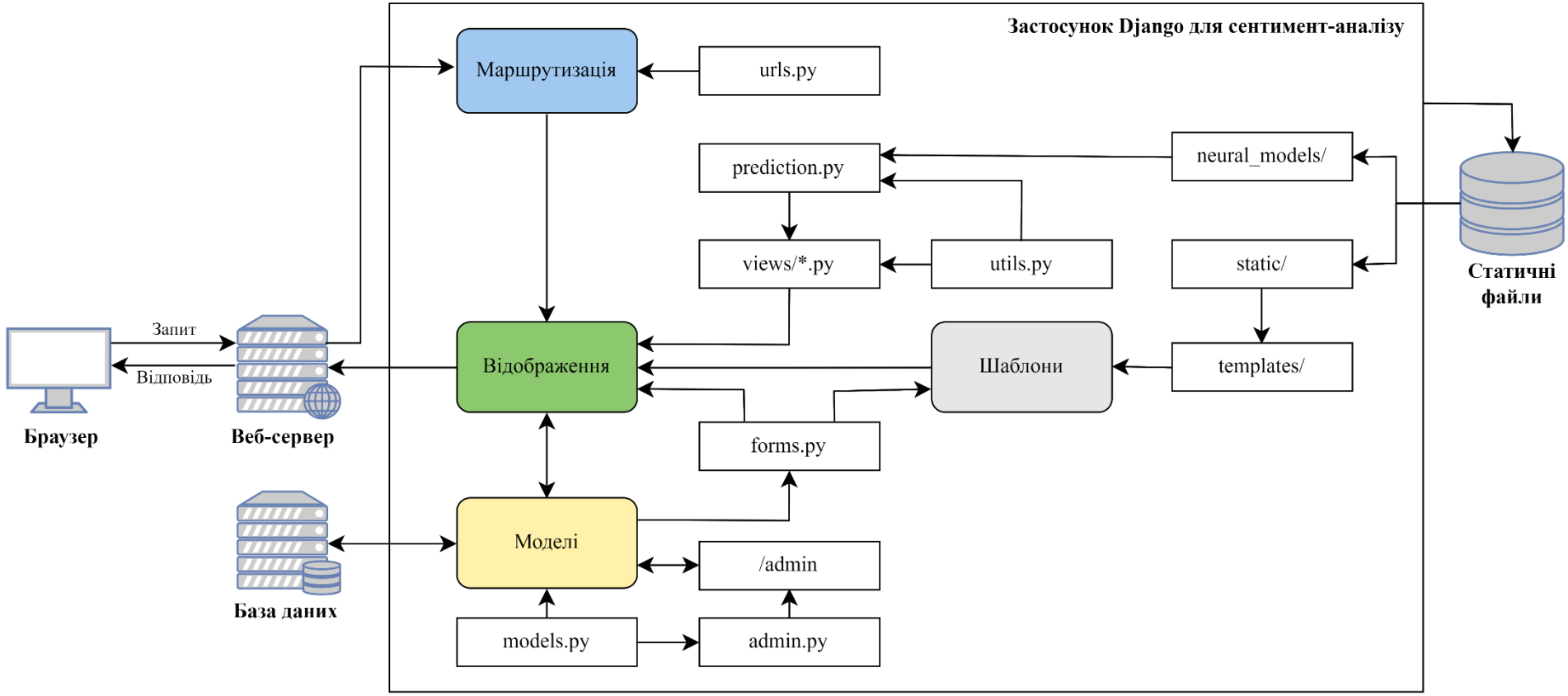
Функціональна схема

Архітектура веб-сервісу

ІАЛЦ.467800.005 Д2

Аркушів 1

Київ – 2022 року



					ІАЛЦ.467800.005 Д2	
Змн.	Арк.	№ докум.	Підпис	Дата	Сервіс аналізу емоційного забарвлення текстових повідомлень	Літ. / Арк. 1 / Аркушів 1
Розроб.		Шендріков Є.О.				
Перевір.		Болдак А.О.				
Реценз.						
Н.контр.		Сімоненко В.П.			Архітектура веб-сервісу (Функціональна схема)	НТУУ «КПІ імені Ігоря Сікорського», ФІОТ, гр. ІО-82
Затверд.		Стіренко С.Г.				

ДОДАТОК З

Сервіс аналізу емоційного забарвлення текстових повідомлень

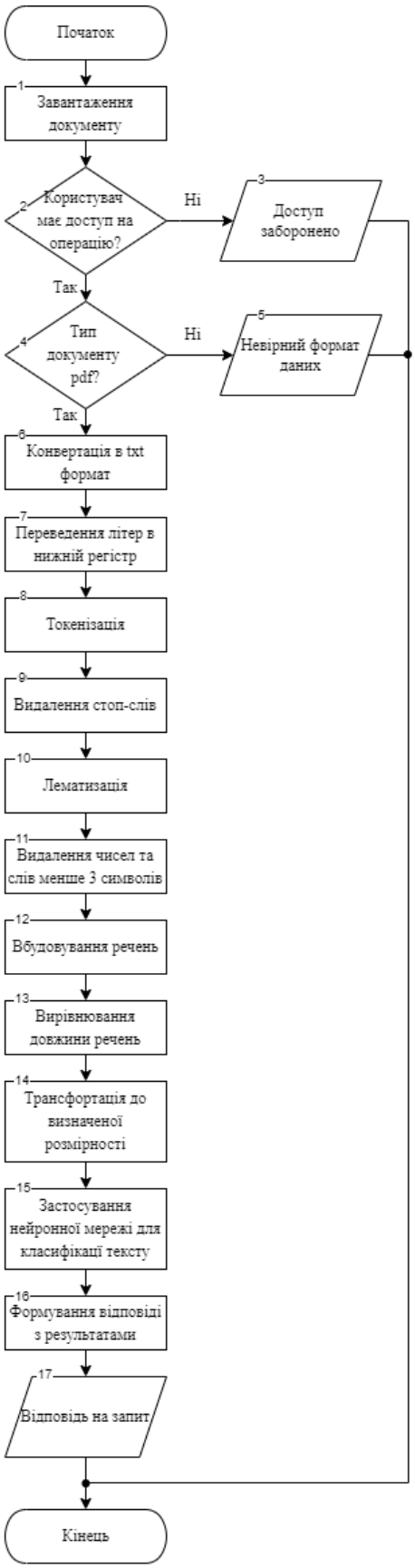
Принципова схема

Послідовність дій роботи сервісу

ІАЛЦ.467800.006 ДЗ

Аркушів 1

Київ – 2022 року



					ІАЛЦ.467800.006 ДЗ		
Зм.	*Арк.*	*№ докум.*	*Підпис*	*Дата*			
Розробив		*Шендріков Є.О.*			*Сервіс аналізу емоційного забарвлення текстових повідомлень*	*Літ.*	*Аркуш* 1 / *Аркушів* 1
Перевірив		*Болдак А.О.*					
Реценз.					*Послідовність дій роботи сервісу (Принципова схема)*	*«КПІ імені Ігоря Сікорського» ФІОТ, гр. ІО-82*	
Н. Контр.		*Сімоненко В.П.*					
Затв.		*Стіренко С.Г.*					