

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Інститут інформатики та радіoeлектроніки,
Факультет комп'ютерних наук і технологій
(повне найменування інституту, назва факультету)

Кафедра програмних засобів
(повне найменування кафедри)

ПОЯСНЮВАЛЬНА ЗАПИСКА

до дипломного проєкту (роботи)
бакалавр
(ступінь вищої освіти)

на тему РОЗРОБЛЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ
ЕМОЦІЙНОГО АНАЛІЗУ УКРАЇНОМОВНИХ ДОКУМЕНТІВ. SOFTWARE
DEVELOPMENT FOR SENTIMENT ANALYSIS IN THE UKRAINIAN
LANGUAGE DOCUMENTS

Виконав: студент(ка) 4 курсу, групи КНТз-118сп
Спеціальності 121 Інженерія програмного
забезпечення

(код

і найменування спеціальності)

Освітня програма (спеціалізація) Інженерія
програмного забезпечення

Котелевець О.А.

(прізвище та ініціали)

Керівник Шитікова О.В.

(прізвище та ініціали)

Рецензент Морщавка С.В.

(прізвище та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
 (повне найменування закладу вищої освіти)

Інститут, факультет ІРЕ, ФКНТ
 Кафедра програмних засобів
 Ступінь вищої освіти бакалавр
 Спеціальність 121 Інженерія програмного забезпечення
 (код і найменування)

Освітня програма (спеціалізація) Інженерія програмного забезпечення
 (назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
С.О. Субботін
 “ ” 2021 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЕКТ (РОБОТУ) СТУДЕНТА(КИ)

Котелевець Олександра Андрійовича
 (прізвище, ім'я, по батькові)

1. Тема проекту (роботи) Розроблення програмного забезпечення для емоційного аналізу україномовних документів. Software development for sentiment analysis in the ukrainian language documents

керівник проекту (роботи) Шитікова Олена Вікторівна, к.т.н.
 (прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом закладу вищої освіти від “30”березня 2021 року № 103

2. Строк подання студентом проекту (роботи) 28 травня 2021 року

3. Вихідні дані до проекту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2. Розробка архітектури програми. 3. Основні рішення щодо реалізації компонентів системи. 4. Експериментальне дослідження методів класифікації текстів.

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

6. Консультанти розділів проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4 Основна частина	Шитікова О.В., доцент		
Нормоконтроль	Камінська Ж.К., асистент		

7. Дата видачі завдання 29 лютого 2020 р.**КАЛЕНДАРНИЙ ПЛАН**

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту (роботи)	Примітка
1	Постановка завдання роботи.	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області.	2–3 тижні	Розділ 1
3	Розробка архітектури програми.	4 тиждень	Розділ 2
4	Розробка програми.	5–6 тижні	Розділ 3
5	Тестування та експериментальне дослідження програми.	7 тиждень	Розділ 4
6	Оформлення пояснювальної записки та документів до неї. Нормоконтроль та рецензування	8 тиждень	Додатки
7	Захист роботи.	9 тиждень	

Студент (ка) _____ Котелевець О.А.
(підпис) (прізвище та ініціали)Керівник проекту (роботи) _____ Шитікова О.В.
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра:
83 с., 2 табл., 34 рис., 2 дод., 18 джерел.

ЕМОЦІЙНИЙ АНАЛІЗ, НЕЙРОННІ МЕРЕЖІ, ДОКУМЕНТИ,
УКРАЇНСЬКА МОВА, PYTHON, ЗАСОБИ ОБРОБЛЕННЯ ПРИРОДНОЇ
МОВИ.

Об'єкт дослідження – процес визначення емоцій в документах.

Предмет дослідження – програмні засоби для емоційного аналізу
україномовних документів.

Мета роботи – покращити розуміння україномовних електронних
ресурсів за рахунок використання програмного забезпечення для емоційного
аналізу тексту.

Матеріали, методи та технічні засоби: рекурентні нейронні мережі,
мережі з довготривалою і короткочасною пам'яттю, згорткові нейронні мережі,
методи та засоби обробки природної мови, python, keras.

Результати. Розроблено веб-застосування, що розпознає тональність
україномовних документів з використанням різноманітних моделей обробки
природної мови.

Висновки. Розроблено програмне забезпечення для навчання та
тестування моделей обробки природної мови для визначення тональності
україномовних текстів. Розроблено веб-застосування для розпізнавання емоцій
україномовних документів, що використовує п'ять різних моделей.

Галузь використання – соціальні мережі, конкурси студентських робіт,
відгуки на процеси та продукти.

ABSTRACT

Explanatory note to the final diploma work of the bachelor: 83 pages, 34 figures, 2 tables, 2 appendixes, 18 sources.

SENTIMENT ANALYSIS, NEURAL NETWORKS, DOCUMENTS, UKRAINIAN LANGUAGE, PYTHON, NATURAL LANGUAGE PROCESING TOOLKIT.

Object of study is a process of sentiment identification in the documents.

Subject of study are software for sentiment analysis of ukrainian documents.

The purpose of the work is to improve the understanding the digital Ukrainian resources by means of software for the sentiment analysis.

Materials, methods, and tools: convolutional neural networks, recurrent neural networks, natural language processing toolkit, python, keras.

Results. Software with web interface for Ukrainian documents sentiment analysis were developed, based on the usage of the variety of natural language processing models.

Conclusions. Software for data preprocessing, training and validation of the models for the natural language processing for the sentiment analysis of the Ukrainian documents were developed. Software with web-interface for the Ukrainian documents sentiment analysis with 5 models were developed.

The field - social media, students contests, processes and products review.

ЗМІСТ

С.

Зміст	6
Перелік посилань та умовних познач	7
<u>Вступ</u>	8
<u>1 Аналіз предметної області</u>	9
<u>1.1 Емоційний аналіз текстів</u>	9
<u>1.2 Нейроні мережі для аналізу текстів</u>	13
<u>1.2.1 Рекурентні нейроні мережі для аналізу текстів</u>	13
<u>1.2.2 Згорткові нейронні мережі для аналізу текстів</u>	14
<u>1.3 Висновки за розділом та постановка завдання</u>	16
<u>2 Розробка архітектури програми</u>	17
<u>2.1 Аналіз вимог до системи</u>	17
<u>2.2 Оброблення природної мови з використанням NLTK</u>	19
<u>3 Основні рішення щодо реалізації компонентів системи</u>	22
<u>3.1 Реалізація розпізнавання емоцій україномовних текстів</u>	26
<u>3.2 Реалізація веб-застосування</u>	28
<u>4 Експериментальне дослідження методів класифікації текстів</u>	36
<u>4.1 Попередня обробка даних</u>	36
<u>4.2 Класифікація тональності україномовних документів</u>	40
<u>4.3 Створення моделей з використанням LSTM</u>	43
Висновки	53
Перелік джерел посилань	54
<u>Додаток А Текст програми</u>	56
<u>Додаток Б Слайди презентації</u>	79

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

LSTM	Long short-term memory
NLTK	Natural Language Toolkit
ROC	Receiver operating characteristic
XML	Extensible Markup Language
ЗНМ	згорткові нейроні мережі
Keras	бібліотека, заснована на Python
PHM	рекурентні нейрони мережі

ВСТУП

Обробка природної мови (Natural Language Processing, NLP) -загальне напрямок штучного інтелекту і математичної лінгвістики, який вивчає проблеми комп'ютерного аналізу і синтезу природних мов.

До основних напрямів обробки природної мови відносять такі, як витяг фактів, аналіз тональності тексту, відповіді на питання, інформаційний пошук, генерація тексту, переклад і т.д.

Важливим та перспективним контент-аналіз є в управлінні інформаційними ресурсами. Без відповідної актуальної інформації зробити це важко. Управління є важливою частиною будь-якого виду аналізу і проводиться на основі поточних даних, а також динаміки змін попередніх даних. Базова система аналізу контенту передбачає такі можливості: швидке поновлення контенту, пошук контенту на певному ресурсі, збирання контенту про постійних та потенційних клієнтів, аналіз цільової та постійної аудиторії, формування та редагування опитувань, аналіз відвідування ресурсу [1].

Аналіз контенту на англійській мові присвячено багато робіт, але для української мови не існує достатньо вихідних даних для якісного аналізу.

Тому метою роботи було розроблення вебзастосунку для аналізу україномовних текстів.

Для досягнення поставленої мети було визначено наступні задачі:

1. дослідження методів, що використовуються для аналізу тональності текстів;
2. виконати проектування системи;
3. провести дослідження методів класифікації текстів для україномовних документів;
4. розробити вебзастосування для аналізу україномовних текстів.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Емоційний аналіз текстів

Емоційно- смисловий аналіз або аналіз тональності це галузь науки, яка займається аналізом людських думок, емоцій, відношення стосовно певних подій , продукції, послуг, організацій, тощо [1].

Завдяки тональності окремих слів можна визначити тональність всього повідомленн язагалом. Емоційне забарвлення може бути позначене як позитивне, негативне чи нейтральне. Наприклад, слова чудовий, щастя, доброта можуть бути позначені як позитивні, а от такі слова як поганий, потворний, сум позначаються як негативні. Тональність – це засіб для визначення, класифікації та вилучення емоційно-смислової лексики з речення або більших частин тексту.

Аналіз тональності - клас методів контент-аналізу в комп'ютерній лінгвістиці, основне завдання якого полягає в класифікації тексту з його настрою. У простих випадках завдання аналізу тональності зводяться до бінарної класифікації текстів на позитивні і негативні. У деяких випадках додають ще клас нейтральних текстів. Більш просунуті підходи намагаються визначати емоційні стани, асоційовані з якимось текстом, наприклад, страх, злість, смуток або щастя. У ряді підходів текстам присвоюються значення заздалегідь певної шкали: наприклад, від -2 для негативних до 2 для позитивних; таким чином аналіз зводиться до задачі регресії. Аспектний аналіз тональності (aspect-based sentiment analysis) - це різновид аналізу тональності, чиє завдання полягає у визначенні відношення до конкретного аспекту основного предмета обговорення. Всі підходи до аналізу тональності можна розділити на три групи.

Аналіз тональності став потужним інструментом для масштабної обробки думок, які висловлюються в будь-яких текстових джерелах.

Аналіз тональності текстів знаходить своє застосування в соціології - для обробки даних з соціальних мереж, медицині і психології – для виявлення

особливостей поведінки користувачів, маркетингу – для оцінювання якості товарів та послуг у відгуках, політології для збору даних щодо політичних поглядів та інш.

На рис. 1.1 наведено галузі застосування визначення тональності тексту.



Рисунок 1.1 – Галузі застосування визначення тональності тексту

Можна розділити усі підходи до класифікації тональності тексту на такі категорії [4]:

- підходи, основані на правилах;
- підходи, що ґрунтуються на словниках (тональні словники);
- підходи, основані на шаблонах;
- частотні методи;
- машинне навчання (навчання без вчителя).

Перша - підходи на основі правил (rule-based) [5]. Найчастіше в них використовуються вручну задані правила класифікації і емоційно розмічені словники. Ці правила зазвичай на основі емоційних ключових слів і їх спільного використання з іншими ключовими словами розраховують клас тексту. Незважаючи на прекрасну ефективність в текстах з якоїсь певної тематики, методи на основі правил погано здатні узагальнювати. Крім того, вони вкрай трудомісткі у створенні, особливо коли немає доступу до

потрібного словником настроїв. Останнє особливо характерно для російської мови, тому що на ньому не так багато джерел, як англійською, особливо в сфері аналізу тональності.

Підходи, основані на тональних словниках (affective lexicons) використовують список слів зі значенням тональності (валентності) кожного слова.

Найпоширеніший спосіб подання документів в задачах комп'ютерної лінгвістики і пошуку, це подання у вигляді:

- набору слів (bag-of-words);
- набору N-грам;
- вектору.

Основним алгоритмом для цих методів є те, що на першому етапі навчається машинний класифікатор (наприклад, байесовський) на попередньо розмічених текстах, а потім використовують отриману модель при аналізі нових документів.

Підходи на основі машинного навчання використовують автоматичне вилучення ознак з тексту і застосовують алгоритми машинного навчання. Класичними алгоритмами класифікації полярності є наївний байесовський класифікатор (Naive Bayes Classifier) [6], дерево рішень (Decision Tree) [7], логістична регресія (Logistic Regression) [8] і метод опорних векторів (Support Vector Machine) [9]. В останні роки увагу дослідників привертають методи глибокого навчання, які значно перевершують традиційні методи в аналізі тональності [4]. Одна з головних особливостей систем на основі машинного навчання - автоматичне вилучення ознак з тексту. У простих підходах для представлення тексту у векторному просторі зазвичай використовується модель «мішок слів» (bag of words).

Гібридні підходи об'єднують в собі підходи двох попередніх видів. Наприклад, розробили гібридний фреймворк для аналізу тональності перської мови, в якому поєднуються лінгвістичні правила, а також модулі згорткових

нейромереж і LSTM для класифікації настроїв [4]. З одного боку, комбінація методів на основі правил і машинного навчання зазвичай дозволяє домогтися більш точних результатів. А з іншого - гібридні підходи успадковують труднощі і обмеження складових їх алгоритмів.

Точність і якість системи аналізу тональності тексту оцінюється тим, наскільки добре вона узгоджується з думкою людини щодо емоційної оцінки досліджуваного тексту. Для цього можуть використовуватися такі показники як точність і повнота. Формула для знаходження повноти:

$$R = CEO/TNO, \quad (1.1)$$

де *CEO*— правильно розпізнані думки, *TNO*— загальна кількість думок (як знайдених системою, так і не знайдених).

Точність обчислюється за формулою:

$$P = CEO/TNOFS,$$

де *TNOFS*— загальна кількість думок знайдених системою.

Таким чином, точність виражає кількість досліджуваних текстів, речень або документів, в оцінці яких думка системи аналізу тональності збіглось з думкою експерта. При цьому, згідно з дослідженням, експерти зазвичай погоджуються в оцінках тональності конкретного тексту в 79 % випадків. Тобто, програма, яка визначає тональність тексту з точністю 70 % робить це майже так само добре, як і людина.

1.2 Нейронні мережі для аналізу текстів

Нещодавно програми на основі нейронних мереж показали перспективні результати для класифікації тексту. Одним із методів глибокого навчання є рекурентні нейронні мережі (RNN) з архітектурою Long Short-Term Memory (LSTM). Перевага LSTM полягає в тому, що цей метод може захоплювати інформацію в будь-якій частині документа. Також ця модель дозволяє враховувати конкретний порядок слів.

Головна перевага рекурентних нейронних (РНМ) мереж у тому, що вони можуть ефективно використовувати дані з попередніх кроків.

РНМ називаються рекурентними, бо вони виконують одну і ту ж операцію для кожного елемента послідовності, при цьому значення на виході залежать від попередніх обчислень.

Іншим популярним підходом є використання згорткових нейронних мереж (ЗНМ). Оскільки ЗНМ є ефективним сучасним методом розпізнавання зображень було показано, що вони можуть бути успішними в різних завданнях обробки природної мови. Так було доведено ефективність при озрізнаванні символів з використаннями 29 згорткових шарів, також було доведено ефективність при вирішенні завданні з мітками.

Комбінації рекурентних та згорткових нейронних мереж показали добрі результати при класифікації тексту декількома класами.

1.2.1 Рекурентні нейронні мережі для аналізу текстів

Рекурентні нейронні мережі (РНМ, англ. Recurrent нейронна мережа; RNN) - вид нейронних мереж, де зв'язки між елементами формують створену послідовність. Завдяки цьому з'являється можливість обрати серію подій у часі або подальші просторові цепочки. На відміну від багатошарових перцептронів, рекурентні мережі можуть використовувати свою внутрішню пам'ять для

бробки послідовностей довільної довжини. Тому мережі РНН застосовні в таких завданнях, де щось цілісне розбите на частини, наприклад: розпізнавання рукописного тексту [1] або розпізнавання мови [2],[3].

Останнім часом найбільшого поширення набули мережі з довготривалою і короткочасною пам'яттю (LSTM) і керований рекурентний блок (КРБ).

Двонаправлені LSTM є розширенням традиційних LSTM, які можуть поліпшити продуктивність моделі в задачах класифікації послідовностей.

У завданнях, де доступні всі тимчасові кроки вхідної послідовності, двонаправлені LSTM навчають два замість одного LSTM у вхідній послідовності. Перший на вхідній послідовності як є, а другий на зворотній копії вхідної послідовності. Це може забезпечити додатковий контекст для мережі і привести до більш швидкого і навіть більш повного вивчення проблеми.

Вихідна модель LSTM складається з одного прихованого шару LSTM, за яким слід стандартний вихідний шар з прямим зв'язком.

Stacked LSTM - це розширення цієї моделі, яке має кілька прихованих шарів LSTM, де кожен шар містить кілька осередків пам'яті [10].

Укладання прихованих шарів LSTM робить модель глибше. Саме глибина нейронних мереж зазвичай пояснюється успіхом підходу в широкому спектрі складних завдань прогнозування.

1.2.2 Згорткові нейронні мережі для аналізу текстів

Згорткова нейронна мережа, СНС, CNN – основний інструмент для класифікації та розпізнавання об'єктів, облич на фотографіях, розпізнавання мови.

Згорткові нейронні мережі можна застосовувати не тільки для вирішення завдань комп'ютерного зору. Наприклад, недавно Facebook AI

Research виклала у відкритий доступ wav2letter ++ – свою технологію розпізнавання мови, засновану на CNN.

Вхідними даними CNN (рис. 1.2) є матриця з фіксованою висотою n , де кожен рядок являє собою векторне відображення токена в просторі ознак розмірності k . Для формування простору ознак часто використовують інструменти дистрибутивної семантики, такі як Word2Vec, Glove, FastText і т.д. [11].

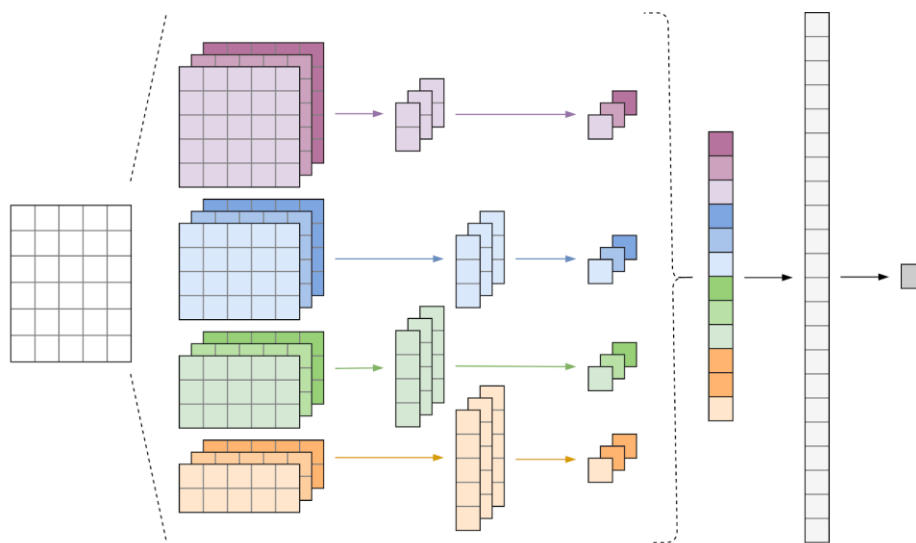


Рисунок 1.2 – Згорткові нейронні мережі [11]

На першому етапі вхідні матриця обробляється шарами згортки. Як правило, фільтри мають фіксовану ширину, рівну розмірності простору ознак, а для підбору розмірів у фільтрів налаштовується тільки один параметр - висота h . Виходить, що h - це висота суміжних рядків, що розглядаються фільтром спільно. Відповідно, розмірність вихідний матриці ознак для кожного фільтра варіюється в залежності від висоти цього фільтра h і висоти вихідної матриці n .

Далі карта ознак, отримана на виході кожного фільтра, обробляється шаром Субдискретизація з певною функцією ущільнення (на зображенні - 1-тах pooling), тобто зменшує розмірність сформованої карти ознак. Таким чином

витягується найбільш важлива інформація для кожної згортки незалежно від її положення в тексті. Іншими словами, для використовуваного векторного відображення комбінація шарів згортки і верств Субдіскретизація дозволяє витягувати з тексту найбільш значущі n-грами.

Після цього карти ознак, розраховані на виході кожного шару Субдіскретизація, об'єднуються в один загальний вектор ознак. Він подається на вхід прихованого повнозв'язну шару, а потім надходить на вихідний шар нейронної мережі, де і розраховуються підсумкові мітки класів.

1.3 Висновки за розділом та постановка завдання

Аналіз контенту електронних ресурсів на сьогодні є дуже актуальним завданням, що використовується для різноманіття цифрових ресурсів. Були розглянуті основи аналізу тональності тексту, визначено, що підхід який для цього використовується відповідає підходу до машинного навчання. Розглянуто рекурентні та згорткові нейронні мережі.

Наступним кроком є визначення фреймворків, проектування архітектури застосування, навчання моделей та використання їх для визначення тональності україномовних текстів.

2 РОЗРОБКА АРХІТЕКТУРИ ПРОГРАМИ

2.1 Аналіз вимог до системи

Розроблення системи складається з розроблення двох підсистем. Підсистема для розроблення моделі для класифікації текста та розроблення інтерфейсу користувача для визначення емоцій в українських текстах.

На рисунках 2.1 та 2.2 наведені два основних прецеденти, що визначають рамки системи.

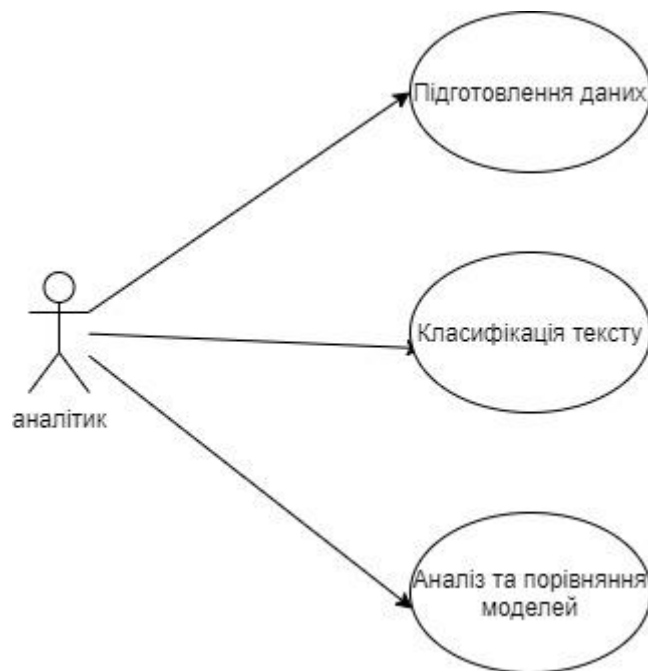


Рисунок 2.1 – Прецедент з розроблення моделі визначення емоцій в тексті

Прецедент 1. Визначення тональності тексту.

Основний виконавець. Користувач.

Передумови. Завантажені дані для навчання.

Постумови. Отримані та збережені моделі.

Основний позитивний сценарій.

Крок 1. Користувач переводить текст англійською мовою на українську для формування бази даних для навчання.

Крок 2. Користувач проводить попереднє оброблення даних.

Крок 3. Користувач виконує навчання моделей для класифікації україномовних текстів.

Крок 4. Користувач аналізує та перевіряє метрики.

Прецедент2. Робота користувача з системою.

Основний виконавець. Користувач.

Передумови. Користувач зареєстрований в системі.

Постумови. Результати аналізу завантажені до системи.

Основний позитивний сценарій.

Користувач проходить авторизацію.

Користувач переглядає завантажені документи.

Користувач завантажує новий документ.

Користувач запускає аналіз тексту.

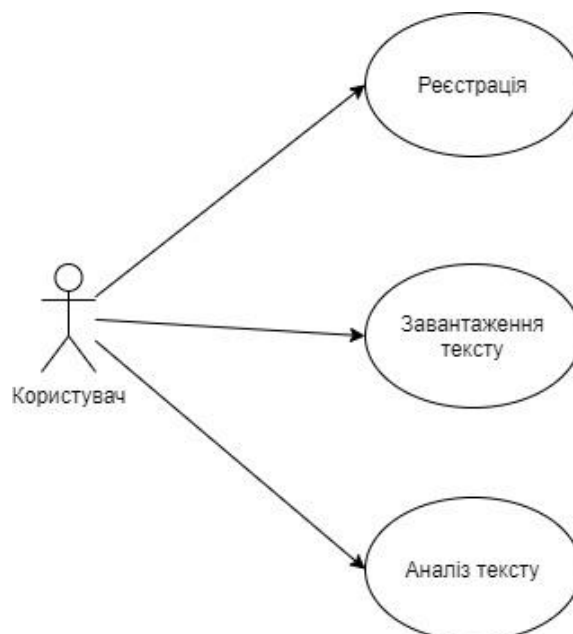


Рисунок 2.2 – Прецедент функціонування системи

2.2 Оброблення природної мови з використанням NLTK

Python представляється простим і в той самий час потужним мовним програмуванням із великою кількістю бібліотек для обробки лінгвістичної інформації. Python це інтерпретатор, що сприяє інтерактивним дослідженням. Як об'єктно-орієнтована мова Python дозволяє легко інкапсулювати та повторно використовувати дані та методи. Python іде з загальною стандартною бібліотекою, яка містить інструментарій для графічного програмування та обробки кількісних даних, що означає, що він може бути використаний для широкого спектра нетривіальних додатків. Python широко застосовується у промисловості, наукових дослідженнях та навчанні по всьому світу.

Natural Language Toolkit, NLTK – це безкоштовна бібліотека для мови програмування Python, що надає великий набір інструментів, для обробки текстів. Вона містить графічні уявлення та приклади даних.

Вона містить функції токенизації за реченнями, токенизацію за словами, лематизацію і стемінг тексту, створення стоп-слів, регулярних виразів, мішок слів, TF-IDF.

Токенизація (іноді - сегментація) за реченнями - це процес поділу писемної мови на речення-компоненти. В англійській і деяких інших мовах ми можемо виокремлювати речення кожен раз, коли знаходимо певний знак пунктуації – точку.

Токенизація (іноді - сегментація) за словами - це процес поділу пропозицій на слова-компоненти. В англійській і багатьох інших мовах, що використовують ту чи іншу версію латинського алфавіту, пробіл - це непоганий роздільник слів.

Зазвичай тексти містять різні граматичні форми одного і того ж слова, а також можуть зустрічатися однокореневі слова. Лематизації і стемінг мають на меті привести все зустрічаються словоформи до однієї, нормальної словникової форми.

Лематизації і стемінг - це окремі випадки нормалізації і вони відрізняються.

Стемінг - це грубий евристичний процес, який відрізає «зайве» від кореня слів, часто це призводить до втрати словотворчих суфіксів.

Лематизації - це більш тонкий процес, який використовує словник і морфологічний аналіз, щоб в результаті привести слово до його канонічної форми - лемме.

Відмінність в тому, що стеммер діє без знання контексту і, відповідно, не розуміє різницю між словами, які мають різний зміст в залежності від частини мови. Однак у стеммер є і свої переваги: їх простіше впровадити і вони працюють швидше. Плюс, більш низька «акуратність» може не мати значення в деяких випадках.

Стоп-слова - це слова, які викидаються з тексту до/після обробки тексту. Коли ми застосовуємо машинне навчання до текстів, такі слова можуть додати багато шуму, тому необхідно позбавлятися від нерелевантних слів.

Стоп-слова це зазвичай розуміють артиклі, вигуки, сполучники і т.д., які не несуть смислового навантаження. При цьому треба розуміти, що не існує універсального списку стоп-слів, все залежить від конкретного випадку.

Регулярний вираз (регулярка, `regex`, `regex`) - це послідовність символів, яка визначає шаблон пошуку. наприклад:

- будь-який символ, крім перекладу рядки;
- `\ W` - один символ;
- `\ D` - одна цифра;
- `\ S` - один пробіл;
- `\ W` - один НЕсимвол;
- `\ D` - одна НЕцифра;
- `\ S` - один НЕпробел;
- `[Abc]` - знаходить будь-який з зазначених символів `match any of a, b, or c`;

- [[^] Abc] - знаходить будь-який символ, крім зазначених;
- [A-g] - знаходить символ в проміжку від a до g.

У NLTK є попередньо встановлений список стоп-слів. Перед першим використанням вам знадобиться його завантажити: `nltk.download ("stopwords")`. Після скачування можна імпортувати пакет `stopwords` і подивитися на самі слова:

Алгоритми машинного навчання не можуть безпосередньо працювати з сирих текстом, тому необхідно конвертувати текст в набори цифр (вектори). Це називається витяганням ознак.

Мішок слів - це популярна і проста техніка вилучення ознак, яка використовується при роботі з текстом. Вона описує входження кожного слова в текст.

Щоб використовувати модель, нам потрібно:

- визначити словник відомих слів (токенов).
- вибрати ступінь присутності відомих слів.

Будь-яка інформація про порядок або структуру слів ігнорується. Ось чому це називається мішки слів. Ця модель намагається зрозуміти, чи зустрічається знайоме слово в документі, але не знає, де саме воно зустрічається.

Scikit learn. Хоча ця бібліотека не має ніяких специфічних інструментів для обробки природного мови, в ній є величезна кількість класифікаторів, заснованих на різних алгоритмах; моделей нейронних мереж та інших спільних інструментів для машинного навчання. Використовуючи її разом з іншими, вузьконаправленими інструментами, можна створювати дуже складні і якісні системи для обробки, аналізу, класифікації і навіть генерації природного тексту.

3. Основні рішення щодо реалізації компонентів системи

TensorFlow [12]-[14], найпопулярніший інструмент для навчання нейромереж. Google активно використовує власний фреймворк глибокого навчання для таких великомасштабних сервісів як Gmail і Google Translate.

Але вона також може бути використана для аналізу і генерації природного тексту. Вищеописаний seq2seq, як і Word2Vec, є частиною даної бібліотеки, постійно допрацьовуються і вже успішно використовується для різних досліджень і створення програмних продуктів.

TensorFlow оперує статичним обчислювальним графом. Тобто спочатку ми визначаємо граф, далі запускаємо обчислення і, якщо необхідно внести зміни в архітектуру, заново навчаємо модель. Такий підхід обраний заради ефективності, але багато сучасних нейромережеві інструменти вміють враховувати уточнення в процесі навчання без істотної втрати швидкості навчання. В цьому плані основним конкурентом TensorFlow є фреймворк глибокого навчання PyTorch.

PyTorch [15] - це середовище машинного навчання на мові Python з відкритим вихідним кодом, що забезпечує тензорні обчислення з GPU-прискоренням. Вона була розроблена компанією Facebook і представлена в жовтні 2016 року, а відкрита для сторонніх розробників - в січні 2017 року. Фреймворк підходить для швидкого прототипування в дослідженнях, а також для любителів і невеликих проектів.

Фреймворк пропонує динамічні графи обчислень, які дозволяють обробляти введення і виведення змінної довжини, що корисно, наприклад, при роботі з рекурентними нейронними мережами. Якщо коротко, то за рахунок цього інженери і дослідники можуть змінювати поведінку мережі «нальоту».

За рахунок глибокої інтеграції фреймворка з кодом C ++ розробники можуть програмувати на C і C ++ за допомогою API-розширення на основі FFI для Python.

Keras - це мінімалістичний бібліотека, заснована на Python, яка може запускатися поверх TensorFlow, Theano або CNTK. Вона була розроблена інженером компанії Google, Франсуа Шолль, з метою прискорення експериментів. Keras підтримує широкий спектр шарів нейронних мереж [16].

MXNet - це фреймворк для глибокого навчання створений Apache, який підтримує достаток мов, наприклад, Python, Julia, C ++, R або JavaScript. Він застосовується в Microsoft, Intel і вебзастосунок Amazon [17].

Фреймворк MXNet відомий своєю високою масштабованістю, тому він використовується великими компаніями в основному для розпізнавання мови і почерку, NLP і прогнозування.

Microsoft Cognitive Toolkit - він представляє з себе відкритий фреймворк для глибокого навчання, створений для роботи з дійсно великими датасета з підтримкою Python, C ++, C # і Java.

CNTK забезпечує ефективне навчання моделей для розпізнавання голосу, рукописного тексту і картинок, також підтримує CNN і RNN. Вона використовується в Skype, Xbox і Cortana[18].

Таблиця 2.1 – Фреймворки для глибиного навчання [10] - [11]

N	Фреймворк	Переваги	Недоліки
1	2	3	4
1	TensorFlow	Обширна документація. Потужні засоби моніторингу процесу навчання моделей і візуалізації (Tensorboard). Підтримується великою спільнотою розробників і технічними компаніями. Забезпечує обслуговування моделей; Вона підтримує розподілене навчання. Низька затримка для мобільних пристроїв.	Програє по швидкості роботи в еталонних тестах. Має більш високий вхідний поріг для початківців. Досить низкоуровневої і вимагає багато шаблонного коду. Python єдина мова, що підтримується.
2	PyTorch	Безліч модульних елементів, які легко комбінувати. Дозволяє писати власні типи шарів і працювати з GPU. Має широкий вибір попередньо навчених моделей	Відсутні засоби моніторингу процесу навчання моделей і візуалізації

Продовження таблиці 2.1

1	2	3	4
3	Keras	Швидке і просте прототипування. Має повністю гнучки модулі. Має простий і інтуїтивно-зрозумілий інтерфейс, відповідно, хороший для новачків. Має вбудовану підтримку для навчання на декількох GPU. Може бути налаштований в якості оцінювачів для TensorFlow і навчений на кластерах GPU на платформі Google Cloud; Запускається на Spark; Підтримує GPU від NVIDIA, TPU від Google, GPU з Open-CL, такі як AMD.	не завжди легко кастомізується; Він обмежений бекендом Tensorflow, CNTK і Theano.

Продовження таблиці 2.1

1	2	3	4
4	MXNet	Швидкий, гнучкий і ефективний. Забезпечує просунуту підтримку GPU. Він може запускатися на будь-якому пристрої. Має високопродуктивне імперативне API; Забезпечує легку підтримку моделей; Доюра масштабований; Забезпечує підтримку безлічі мов програмування, таких як Python, R, Scala, JavaScript і C ++ і інших.	Обмежена підтримка
5	CNTK	Показує хорошу продуктивність і масштабованість; Має добре оптимізовані компоненти; Пропонує підтримку Apache Spark; Ефективний при використанні ресурсів; Має просту інтеграцію з Azure Cloud.	Обмежена підтримка

3.1 Реалізація розпізнавання емоцій українських текстів

Для реалізації підсистеми був використані наступні бібліотеки:

- Keras (v2.4+);
- Gensim (v3.8+);
- Pandas (v1.2+);
- NumPy (v1.19.5+);
- NLTK (v3.5+);
- python-decouple (v3.4+);

- pymorphy2-dicts-uk (v2.4.1+);
- pymorphy2 (v0.9+);
- scikit-learn (v0.24.1);
- SciPy (v0.19.1+);
- Matplotlib (v2.1.1+);
- Jupyter.

Вихідні дані для підсистеми розпізнавання емоцій зберігаються в xml файлах:

- negative.review - позитивні дані;
- positive.review – негативні дані;
- stopwords_ua – українські стоп слова.

В файлі dataset_analysis.py реалізовані наступні методи для попереднього аналізу тексту:

- - parcelXML(xml_reviews) - створення словника;
- - dict_to_dataframe(reviews_dict) – перетворення вихідних даних до

типу Pandas DataFrame;

- word_sent_tokenize(sent)
- regex_tokenizer(sent)
- stopwords_elimination(stop_words, sent)- видаляє зайві слова у

переліку стоа-слів;

- load_mask(mask_url) – візуалізація стоп-слів
- stem_words(words) – метод для стемінгу;
- lemmatize_words(morph, words) – метод для лематизації
- create_doc(reviews, words_list) – створення словника у вигляді tuple
-

– find_matches(doc, all_words, num_freq=50) – повертає словник з найбільш використовуваними словами

– create_matches_set(docs, all_words, num_freq=100) – створення карти зучтрічі найбільш використовуваних слів;

Для навчання був використаний модуль `scikit-learn` or `sklearn`.

Для створення вектора слів використовувалась попередньо навчена модель `Word2Vec`.

Розроблені методи для визначення емоцій реалізовані у файлі `dataset_analysis.py`:

- `sent_embed(words, docs)` – визначення слів, що не зустрічаються у словнику;
- `cosine_similarity(u, v)` = метод, що визначає подібність векторів;
- `plot_classification_report(classification_report, title='Classification Report', cmap='RdBu')` – метод для візуалізації результатів;
- `heatmap(AUC, title, xlabel, ylabel, xticklabels, yticklabels, figure_width=40, figure_height=20, correct_orientation=False, cmap='RdBu')`
- створення `heatmap` для тексту
- `show_val(pc, fmt="%.2f", **kw)` - метод додає комірки до `heatmap`
- `cm_to_inch(*dim)` – редагує параметри графіку

3.2 Реалізація вебзастосування

Для функціонування програми необхідно додати модулі:

- `Keras (v2.4+)`;
- `Gensim (v3.8+)`;
- `Pandas (v1.2+)`;
- `NumPy (v1.19.5+)`;
- `NLTK (v3.5+)`;
- `python-decouple (v3.4+)`;
- `pymorphy2-dicts-uk (v2.4.1+)`;
- `pymorphy2 (v0.9+)`;
- `scikit-learn (v0.24.1)`;

- SciPy (v0.19.1+);
- Matplotlib (v2.1.1+);
- Jupyter.

Структура проєкту наведена на рис. 3.1.

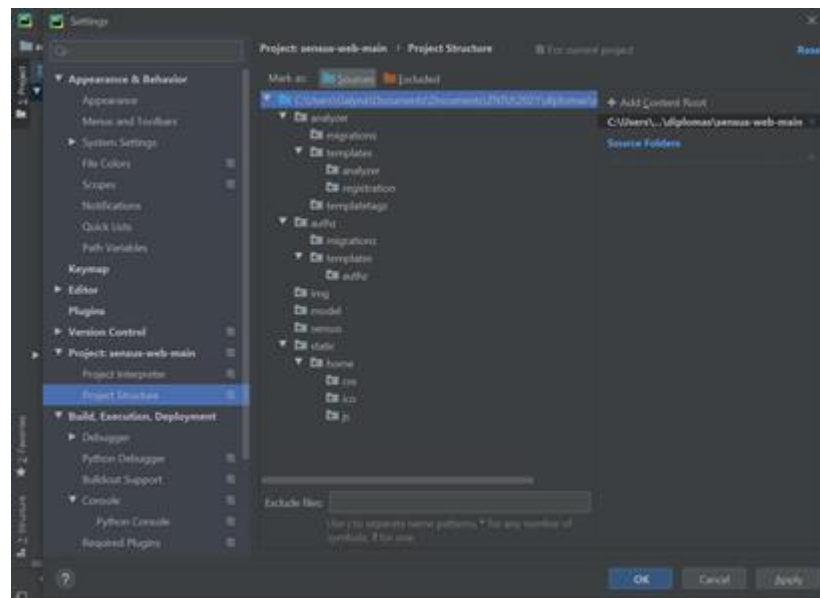


Рисунок 3.1 Структура проєкту

Реалізовані моделі (Models):

- class Document(models.Model);
- class Sentence(models.Model).

Реалізовані форми:

- class DocumentForm(forms.Form).

Реалізовані предствалення (Views):

- home(request);
- upload(request);
- convert_pdf_to_txt(path);
- prediction(request);

Та реалізовані шаблони сторнікок:

- home.html

- predict.html
- upload.html
- login.html.

Для авторизації релізовані предствлення:

- class OpenView(View) :
- class ApereoView(View) :
- class ManualProtect(View)
- class ProtectView(LoginRequiredMixin, View)
- class DumpPython(View)

Та реалізовний шаблон (template) main.html.

Для запуску проєкта необхідно виконати наступні кроки.

1. Клонувати проєкт з github
2. Перевірити наступні вимоги:
 - директорія де знаходиться файл requirements.txt локалізована;
 - віртуальне середовище активоване;
3. Виконати команду


```
pip install -r requirements.txt.
```
4. Завантажити модель Word2Vec
5. Переіменувати файл до model-word2vec-300d.txt
6. Перемістити модель до папки model/ застосування
7. Виконати наступні команди:


```
>> python manage.py makemigrations
>> python manage.py migrate
>> python manage.py createsuperuser
>> python manage.py runserver
```
8. Після запуску сервера можна почати роботу з системою.

Робота з системою починається з вікна авторизації (рис. 3.2), в якому користувачу необхідно ввести логін та пароль.

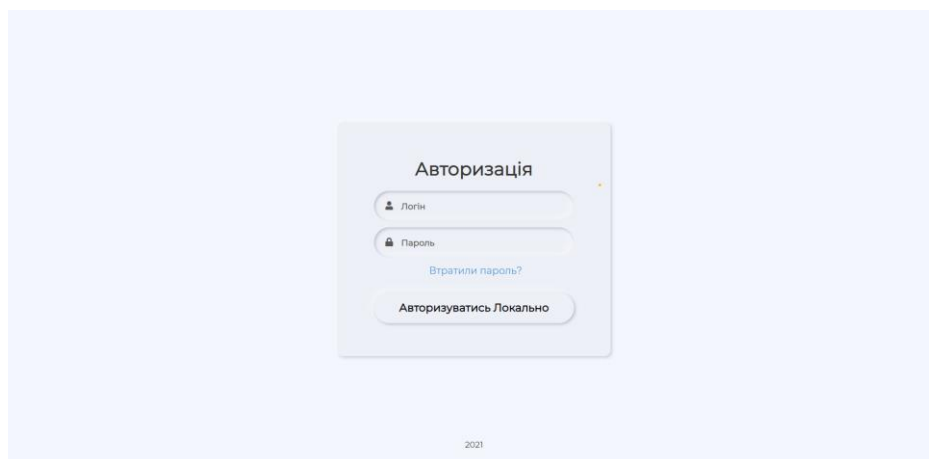


Рисунок 3.2 - Інтерфейс вебзастосування. Авторизація.

Якщо логін чи пароль не знайдено то виводиться відповідна інформація користувачу (рис.3.3).

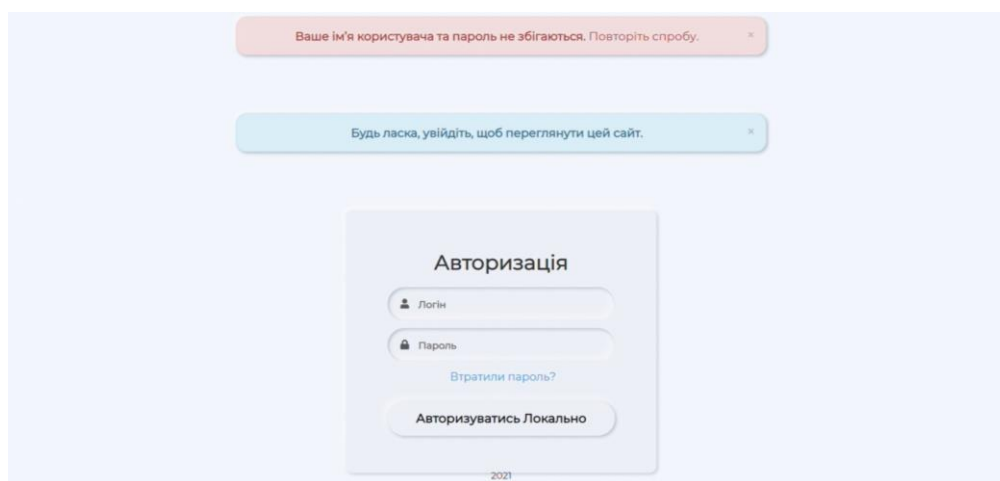


Рисунок 3.3 – Приклад обробки помилки

Користувач може завантажити документи для аналізу (рис. 3.3).

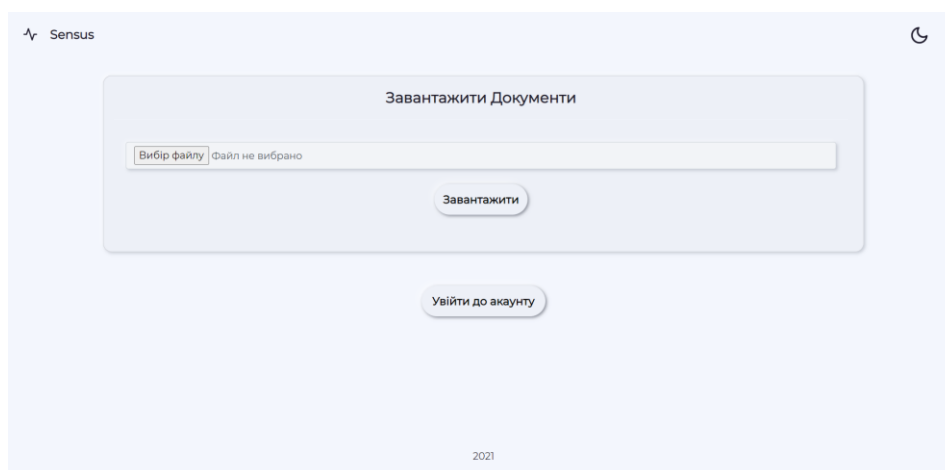


Рисунок 3.3 Інтерфейс веб-застосування.Завантаження документів

Застосування підтримує роботу з 3 основними типами файлів .pdf, .doc, .odt.

В системі реалізовано 2 режими – дневний та ночний (рис. 3.4), який можна переключити на панелі адміністратора.

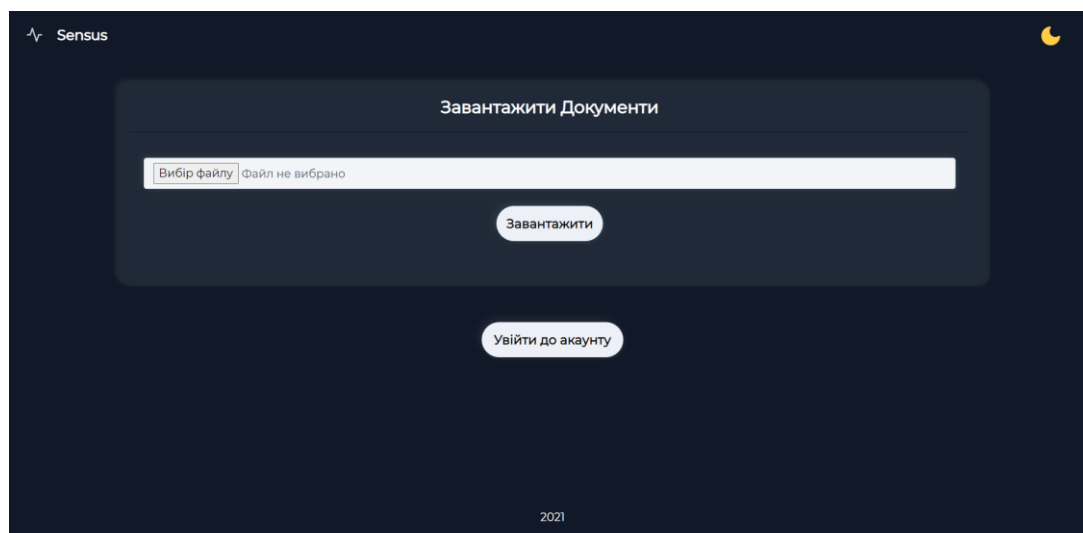


Рисунок 3.4 Інтерфейс веб-застосування. Ночний режим.

Користувач може переглядати завантажені документи в звикористанням двох різних шаблонів перегляду – рис. 3.5 – рис. 3.6.

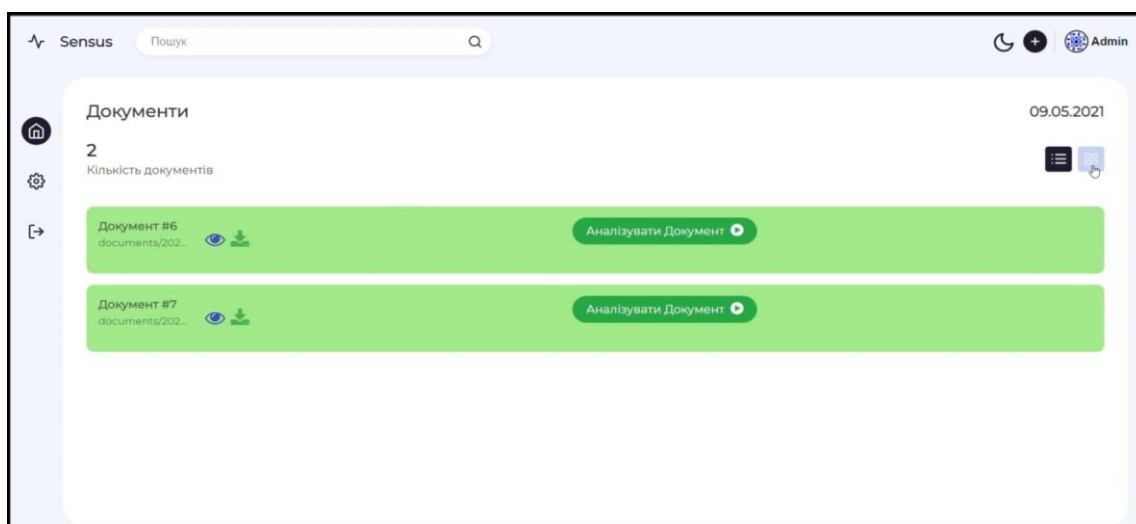


Рисунок 3.5 – Перелік документів

Користувач може закрити можливість перегляду документу та/або завантажити результати аналізу.

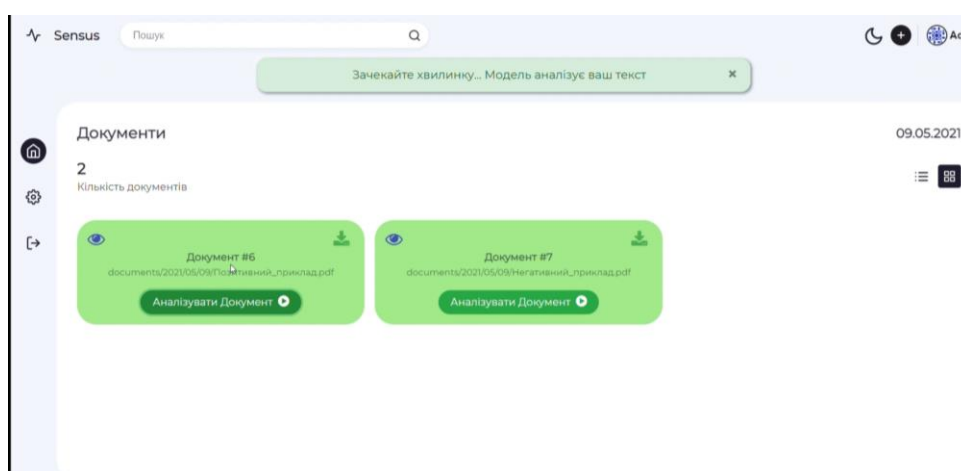


Рисунок 3.6 – Зміна вигляду панелі адміністратора

На рис. 3.7 та 3.8 можна побачити результати аналізу документів з позитивною та негативною тональністю відповідно.

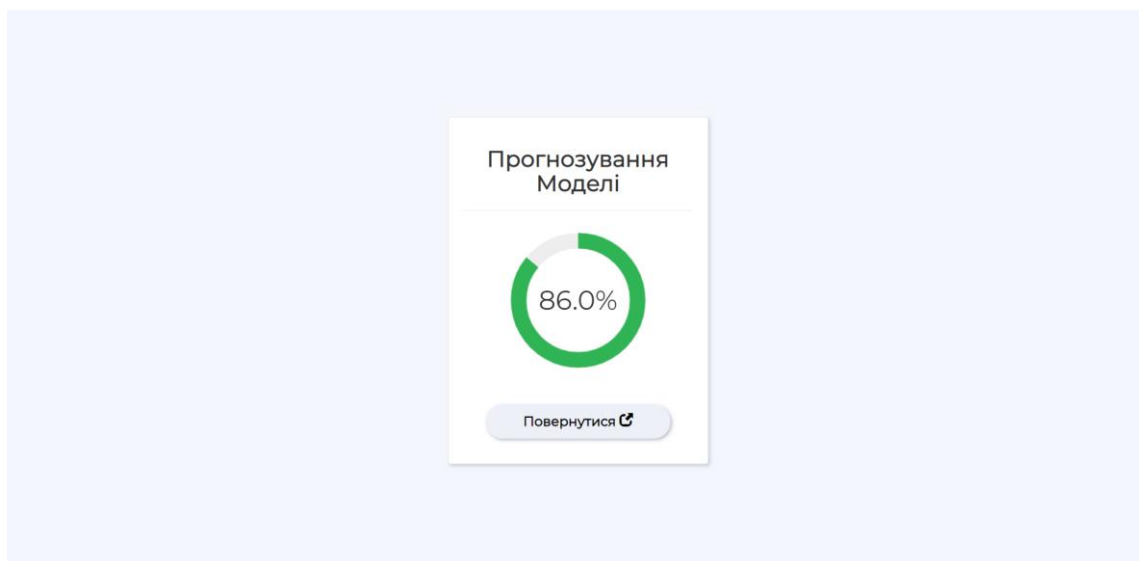


Рисунок 3.7 Інтерфейс вебзастосування. Приклад негативного результату аналізу тексту

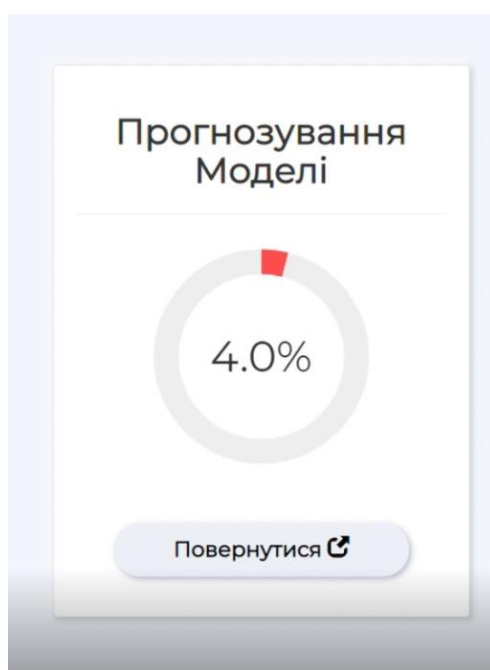


Рисунок 3.8 Інтерфейс вебзастосування. Приклад негативного результату аналізу тексту

4. ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ МЕТОДІВ КЛАСИФІКАЦІЇ ТЕКСТІВ

Набори навчальних даних містять 1000 позитивних та 1000 негативних відгуків про книги. Оригінальні тексти були отримані з Amazon, переведені на українську мову з використанням Google Translator.

Оскільки в бібліотеці NLTC немає підтримки української мови, ми підемо іншим шляхом. Був знайдений найбільш повний перелік українських стоп-слів, які були використані в цьому проекті [<https://github.com/stopwords-iso/stopwords-uk>].

Також на етапі обробки для порівняння був розроблений авторський стеммер для Python.

Для простоти ми будемо використовувати попередньо навчену модель Word2Vec з українськими словами-векторами, кожна з яких має розмірність 300. Ми обрали лематизовану версію цієї моделі, оскільки ми вже маємо наші дані.

4.1. Попередня обробка даних

Метою даного етапу було зробити семантичний аналіз українських текстів, тому збірка позитивних та негативних відгуків була перекладена українською мовою за допомогою Google Translate.

Перший етап проаналізувати документи та вилучили xml теги. В результаті в нашому довіднику з навчальними наборами отримуємо є два файли - з позитивними та негативними відгуками, кожен з яких містить 1000 необроблених відгуків.

Ось ми спробуємо проаналізувати наш XML-файл на базовому рівні. Ми створимо словник зі списків. Кожен список міститиме певні теги та їх вміст для окремого перегляду. У нас вже є дерево із позитивними та негативними відгуками, тому залишається лише проаналізувати його. Але перед цим

розробимо метод `def parseXML(xml_reviews)` який перерахує всі теги у наших файлах `xml`, щоб нам не довелося повторювати. Наступним кроком створюємо словарі та конвертуємо їх до типу `DataFrame`.

На рис. 4.1 приклад отриманих унікальних позитивних джерел, а на 4.2 негативних.

Out[9]:

	unique_id	asin	product_name	helpful	rating	title	date	reviewer	reviewer_location
0	0375416811: owner_of_chick_lit: erin_oakes_ "...	0375416811	Щоденник Бріджит Джонс: Книги: Хелен Філдинг...		5.0	Засновник Чик-літ	24 серпня 2006 р	Ерін Оукс "завзятий читач"	Центральна ПА
1	188105201X: відмінний ресурс для принципалів: ...	188105201X	Лідерство та нова наука: вивчення організації...	3 з 3	5.0	Чудовий ресурс для директорів!	6 липня 1999 р	ONICKRE@mail.milwaukee.k12.wi.us	Мілуокі, штат Вісконсін
2	0634079190: fun_to_play: michael_p_nolan	0634079190	Avenue Q - мюзикл (аранжування на фортепіано ...		5.0	Забавно грати	15 вересня 2005 р	Майкл П. Нолан	Таллахассі, Флорида, США

In [10]: neg_books.head(n=3)

Out[10]:

	unique_id	asin	product_name	helpful	rating	title	date	reviewer	reviewer_location	review
--	-----------	------	--------------	---------	--------	-------	------	----------	-------------------	--------

Рисунок 4.1 – Приклад позитивних джерел

Out[10]:

	unique_id	asin	product_name	helpful	rating	title	date	rev
0	0743518144: ok ~ _but_i_think_the_keirse...y_tem...	0743518144	Тепер відкрийте свої сильні сторони: Книги: М...	7 з 8	2.0	Гаразд ~ але я думаю, що тест на спокій Керсі...	10 вересня 2006 р	Дж "пр.
1	0895261715: відпустка_продажу_консервативізму...	0895261715	Консервативний розум: від Берка до Еліота: Кн...	32 із 95	2.0	Відштовхувальний продаж консерватизму	18 серпня 2005 р	Е. І Свс
2	0787978949: i_had_a_bad_feeling_about_this! _...	0787978949	Християнська мудрість майстрів-джедайв: Книги...	6 з 12	1.0	У мене було погане відчуття з цього приводу! ...	4 грудня 2005 р	3. d

Рисунок 4.2 – Приклад негативних джерел

Давайте сформуємо описову статистику. Описна статистика `Pandas DataFrame` включає статистику, яка узагальнює центральну тенденцію, дисперсію та форму розподілу набору даних, виключаючи значення `NaN`. Ми також перевіримо, чи збережено кількість позитивних та негативних відгуків (рис. 4.3).

<pre>pos_books.describe() neg_books.describe()</pre>									
Number of Positive Reviews (Dataframe): 1000 Number of Negative Reviews (Dataframe): 1000									
Out[11]:		unique_id	asin	product_name	helpful	rating	title	date	reviewer
count	1000	1000	1000	1000	1000	1000	1000	1000	1
unique	997	990	986	343	2	941	695	893	5
top	0910791481: have_not_received_de_item_yet_ x...		0226869024	Жіноча помилка: ми занадто багато здаємось ?....		1.0	Розчарований	5 квітня 2007 р	
freq	2	2	2	82	544	7	8	75	2

Рисунок 4.3 – Описова статистика

Тому що NLTK не містить української мови використаємо стоп слова з цього джерела <https://github.com/stopwords-iso/stopwords-uk>. Для видалення непотрібних слів, було написано власний метод `stopwords_elimination`.

Також було розроблено власний метод для стемінгу:

```
from stemmer_ua import UAStemmer
```

```
def stem_words(words):
```

```
    return [UAStemmer(word).stem_word() for word in words]
```

```
stemmed_words = stem_words(filtered_words)
```

```
print(stemmed_words)
```

Python NLTK пропонує WordNet Lemmatizer, який використовує базу даних WordNet для пошуку лем слів. Але української мови немає, після деяких пошуків було виявлено, що в `pymorphy2` є підтримка української, фактично, на експериментальному рівні. Для встановлення бібліотеки необхідно виконати команду:

```
pip install pymorphy2 та pip install -U pymorphy2-dicts-uk
```

Після чого було обрлено всі позитивні та негативні відгуки.

З використання бібліотеки seaborn візуалізуємо найбільш вживані слова (рис. 4.4 та рис. 4.5).

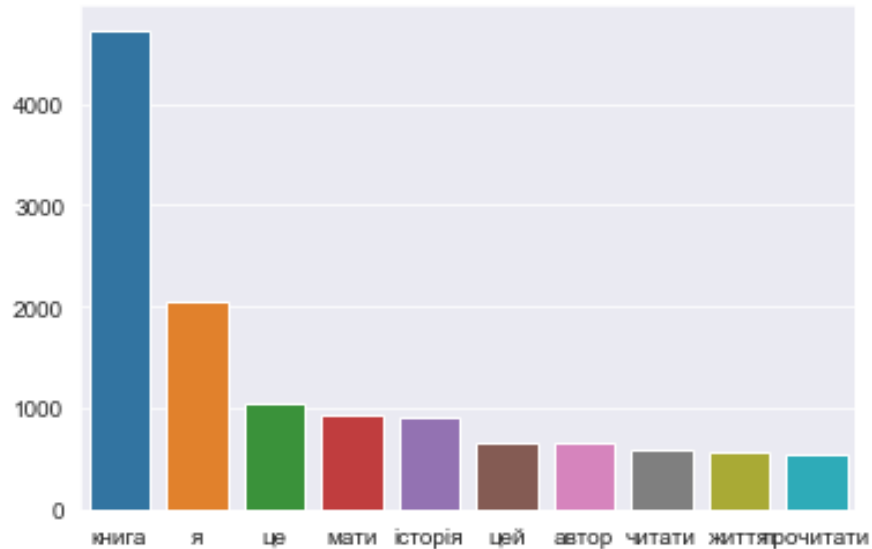


Рисунок 4.4 - Візуалізація найбільш використовуваних слів



Рисунок 4.5 Візуалізація найбільш використовуваних слів після обробки

Для перевірки репрезентативності даних скористались класичним байєсівським класифікатором. Результат на рис. 4.6.

```

In [44]: import nltk.classify.util
         from nltk.classify import NaiveBayesClassifier

         def naive_bayes_model(train_set, test_set):
             classifier = NaiveBayesClassifier.train(train_set)
             informative = classifier.show_most_informative_features(10)
             accuracy = nltk.classify.accuracy(classifier, test_set) * 100
             return classifier, informative, accuracy

In [45]: classifier, informative, accuracy = naive_bayes_model(train_set, test_set)
         print(informative)
         print("\nClassifier Accuracy:", str(round(accuracy, 2)) + "%") # predict

Most Informative Features
настійно = True          pos : neg = 11.0 : 1.0
упереджений = True      neg : pos = 11.0 : 1.0
дратувати = True        neg : pos = 10.3 : 1.0
запитати = True          neg : pos = 10.3 : 1.0
передуюча = True         neg : pos = 9.6 : 1.0
відмінний = True         pos : neg = 9.0 : 1.0
кшталт = True            neg : pos = 9.0 : 1.0
помилковий = True        neg : pos = 9.0 : 1.0
натомість = True         neg : pos = 8.7 : 1.0
вплинути = True          pos : neg = 7.7 : 1.0

None

Classifier Accuracy: 75.3%

```

Рисунок 4.6 – Використання наївного баєсівського класифікатора

Отримана точність 75% мотивує використовувати більш складні моделі для отримання кращих результатів.

4.2. Класифікація тональності українських документів

По-перше, нам потрібно створити вектори слів. Для простоти ми будемо використовувати попередньо навчену модель Word2Vec з українськими словами-векторами, кожна з яких має розмірність 300 [<https://lang.org.ua/uk/models/>].

Цей зразок векторів слів був створений на основі художньої літератури. Ми вибрали лематизовану версію цієї моделі, оскільки вже маємо наш зразок, який обробили в попередній частині, і який тут ідеально вписується.

Після завантаження моделі та нашого вектора слів було проаналізовано наш вектор. Було використано підхід average embedding model [<https://towardsdatascience.com/document-embedding-techniques-fed3e7a6a25d>].

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

Було отримано, що 2986 слів не знайдено у словнику. Далі було проаналізовано подібність, що склала 70%, що ми розглядаємо як допустиму.

Порівняємо результати класифікації з використанням методів k-ближчих сусідів(рис.4.7-4.8), логістичної регресії (рис. 4.9) та випадкового ліса (рис.4.10).

Алгоритм К-найближчих сусідів (KNN) - це тип керованих алгоритмів машинного навчання. KNN надзвичайно простий у впровадженні в самому базовому вигляді, і при цьому виконує досить складні класифікаційні завдання. Алгоритм KNN просто обчислює відстань нової точки даних до всіх інших точок даних навчання. Відстань може бути будь-якого типу, наприклад, Евклідова, Манхеттенська тощо. Потім вона вибирає К-найближчі точки даних, де К може бути будь-яким цілим числом. Нарешті, він призначає точку даних класу, до якого належить більшість точок даних К.

Спочатку обчислимо середнє значення похибки для всіх передбачуваних значень, де сусід коливається від 1 до 25.

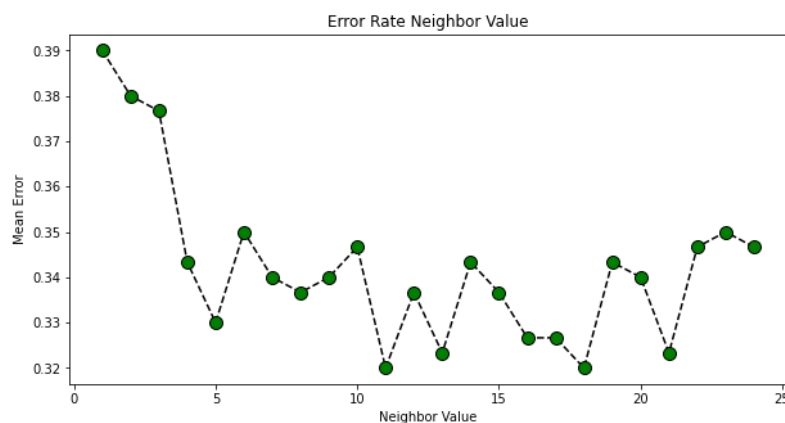


Рисунок 4.7 – Середньо квадратчина помилка

Для $k=5$ отримаємо наступні результати:

Accuracy Score ($k=5$): 67.0%

Accuracy (x_{train}, y_{train}): 100.0%.

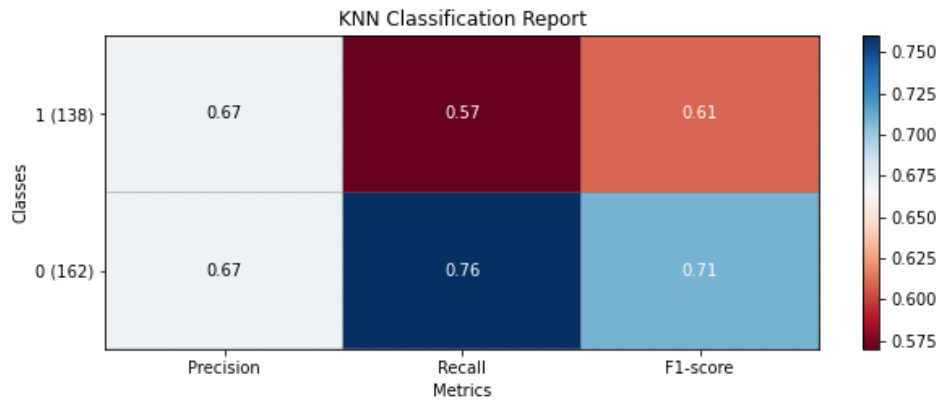


Рисунок 4.8 -Метод k-ближчих сусідів

Точність з використанням логістичної регресії 75.67% (рис. 4.9).

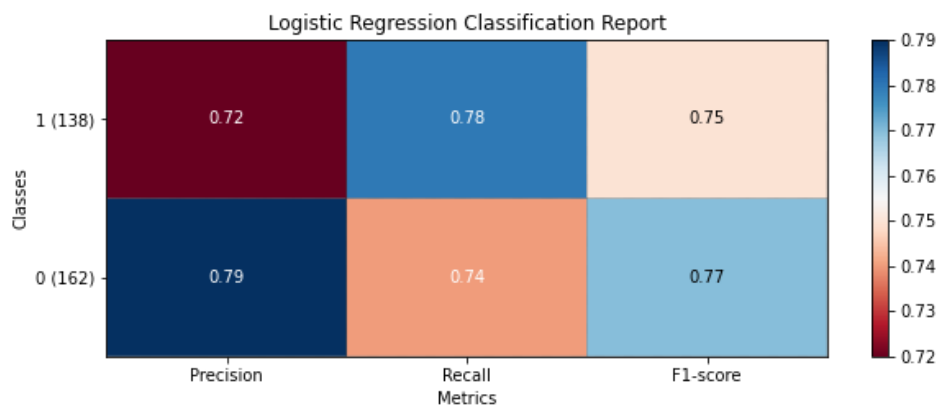


Рисунок 4.9 – Логістично регресія

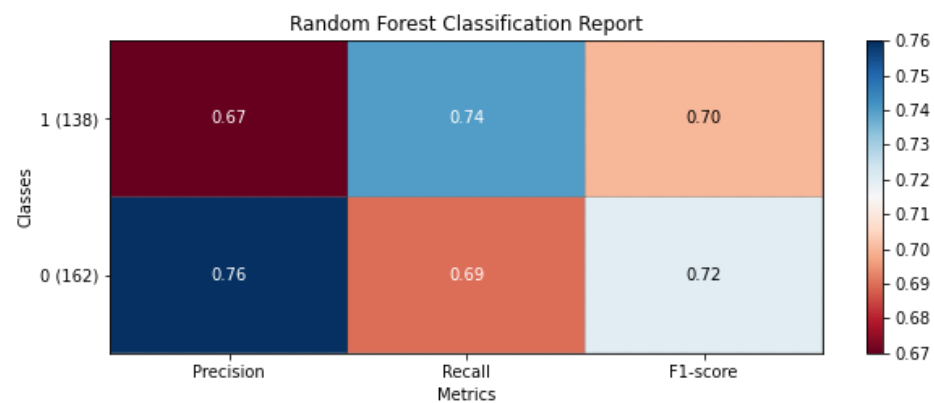


Рисунок 4.10 – Випадковий ліс

Для випадкового ліса отримали точність 71%.

Крива помилок або ROC-крива - графіческая характеристика якості бінарного класифікатора, залежність частки вірних позитивних класифікацій

від частки помилкових позитивних класифікацій при варіюванні порога вирішального правила. Перевагою ROC-кривої є її інваріантність щодо ставлення ціни помилки I і II роду [http://www.machinelearning.ru/wiki/index.php?title=%D0%9A%D1%80%D0%B8%D0%B2%D0%B0%D1%8F_%D0%BE%D1%88%D0%B8%D0%B1%D0%BE%D0%BA].

З графіку на рис. 4.11 можна зробити висновок, що найкращою моделлю є логістична регресія AUC=0.84 та точністю 75%.

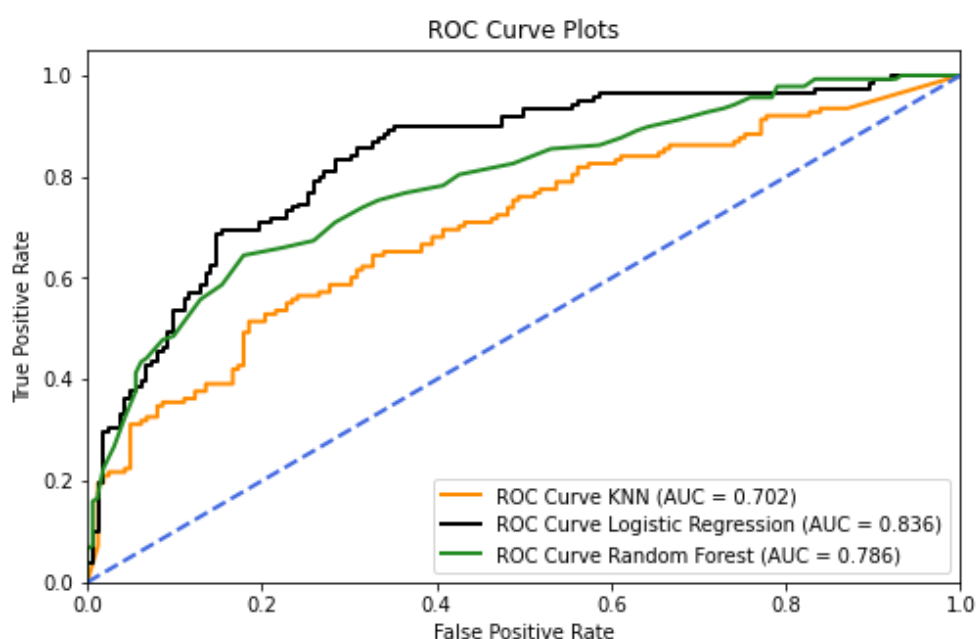


Рисунок 4.11 – Крива помилок

4.3 Створення моделей з використанням LSTM

Для створення моделей ми будемо використовувати нашу отриману навчану модель word2vec на попередньому етапі. Наша модель містить 400 000 слів, з розмірністю 50 для кожного слова.

На рис.4.12 можна побачити результат розподілу кількості слів в наших документах.

Для оптимізації нашої вибірки ми фіксуємо наші статті до фіксованого розміру, щоб передати їх до нейронної мережі. Менші за розміром ми доповнимо нулями, більші – усічені.

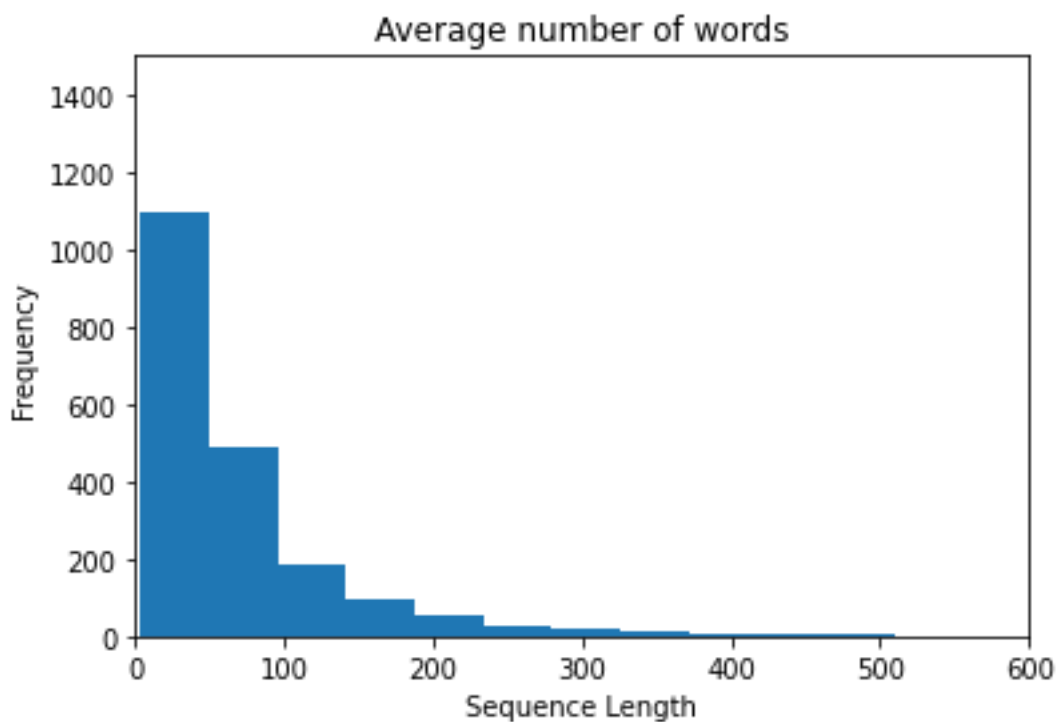


Рисунок 4.12 – Частотний аналіз слів

Як можна побачити з гістограми (рис. 4.12), більшість статей містить менше ніж 100 слів, що відповідає нашим вимогам.

Перевага використання Keras полягає в тому, що він побудований поверх символічних математичних бібліотек, таких як TensorFlow і Theano, для швидких і ефективних обчислень. Це необхідно для великих нейронних мереж.

Недоліком використання цих ефективних бібліотек є те, що ви повинні визначити обсяг ваших даних заздалегідь і назавжди. Зокрема, розмір партії.

Розмір партії обмежує кількість вибірок, які повинні бути показані в мережі, перш ніж може бути виконано оновлення ваги. Це те ж саме обмеження тоді накладено, роблячи прогнози з моделлю відповідності.

Зокрема, розмір партії, який використовується при підгонці вашої моделі, визначає, скільки прогнозів ви повинні робити за один раз.

Це часто не є проблемою, якщо ви хочете робити одне і те ж число прогнозів за раз, як розмір партії, який використовується під час навчання.

Це стає проблемою, коли ви хочете зробити менше прогнозів, ніж розмір пакету. Наприклад, ви можете отримати найкращі результати при великому розмірі пакета, але при цьому вам необхідно робити прогнози для одного спостереження за раз, наприклад, в часі ряду або в послідовності.

Ось чому може бути бажано мати інший розмір пакета при підгонці мережі до навчальних даних, ніж при прогнозуванні на тестових даних або нових вхідних даних

Реалізована мережа має один вхід, прихований шар з 100 одиницями і вихідний шар з 1 одиницею. Функції активації за замовчуванням використовуються в одиницях LSTM, а функція лінійної активації - в вихідному шарі.

Функція оптимізації середньоквадратичної помилки використовується з ефективним алгоритмом оптимізації ADAM.

Точність моделі 68.44%.

Функція втрат допомагає оптимізувати параметри нейронних мереж. Наша мета - мінімізувати втрати для нейронної мережі шляхом оптимізації її параметрів (ваг). Втрати розраховуються з використанням функції втрат шляхом зіставлення цільового (фактичного) значення і прогнозованого значення нейронною мережею. Потім ми використовуємо метод градієнтного спуску для поновлення ваг нейронної мережі таким чином, щоб втрати були зведені до мінімуму. Так ми навчаємо нейронну мережу.

Функція втрат зображена на рис. 4.13.



Рисунок 4.13 – Функція втрат для LSTM-Batch моделі

Зберігаємо модель:

```
from keras.models import load_model
model_batch.save('models/LSTM-Batch-Model.h5')
```

Mini-batch градієнтний спуск - це різновид алгоритму градієнтного спуска, який розділяє навчальний набір даних на невеликі партії, які використовуються для обчислення похибки моделі та оновлення коефіцієнтів моделі.

Mini-batch алгоритм є найбільш сприятливим і широко використовуваним алгоритмом, який дає точні та швидші результати, використовуючи пакет прикладів навчання. В цьому алгоритмі замість використання повного набору даних, у кожній ітерації ми використовуємо набір з m навчальних прикладів, що називається batch, для обчислення

градієнта функції витрат. Загальні розміри міні-партій коливаються від 32 до 256.

Будемо використовувати мережу LSTM, придатну для 50 епох і з розміром партії = 16, оскільки наш набір даних дуже малий.

Також використовуємо алгоритм оптимізації ADAM. Функція втрат на рис.4.14. Точність отриманої моделі 70.31%.



Рисунок 4.14 - Функція втрат для LSTM-Mini- Batch моделі

Зберігаємо отриману модель як LSTM-Mini-Batch-Model.h5.

Наступна модель це складена Stacked LSTM з 3 шарами (рис. 4.15).

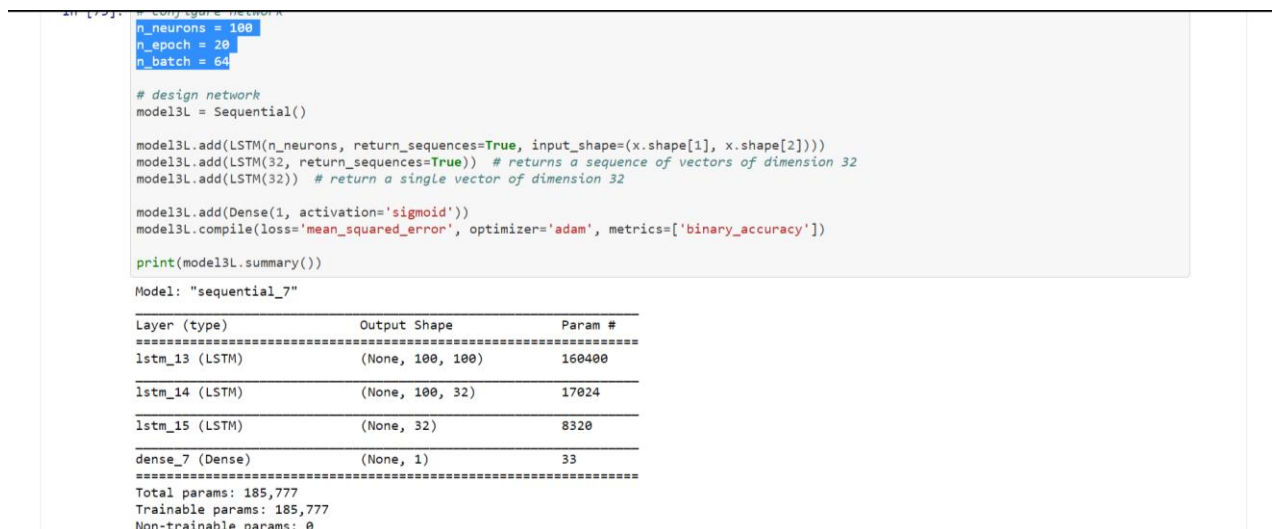


Рисунок 4.15 - Stacked LSTM з 3 шарами

Отримана точність 73,44%. Функція втрат представлена на рис.4.15. Модель збережена як LSTM-3L-Model.h5.



Рисунок 4.16 - Функція втрат для LSTM-3L моделі

Для останньої моделі будемо використовувати наступний алгоритм: словесні вектори словникових слів навчаються з великого корпусу за допомогою інструментарію word2vec. Для кожного даного тексту регіональна

модель CNN використовує речення як область для розділення тексту на R-домени, тобто $r_1, \dots, r_i, r_j, r_k, \dots, r_R$. У кожній області корисні афективні функції можуть бути видалені, коли вектори слів проходять послідовно через згортковий шар та шар максимального сполучення. Потім такі локальні (регіональні) особливості послідовно інтегруються між регіонами, використовуючи LSTM для побудови текстового вектора для прогнозування VA.

Функція втрат на рис. 4.17. Отримана точність 73.75%. Модель збережена як LSTM-CNN-Model.h5.

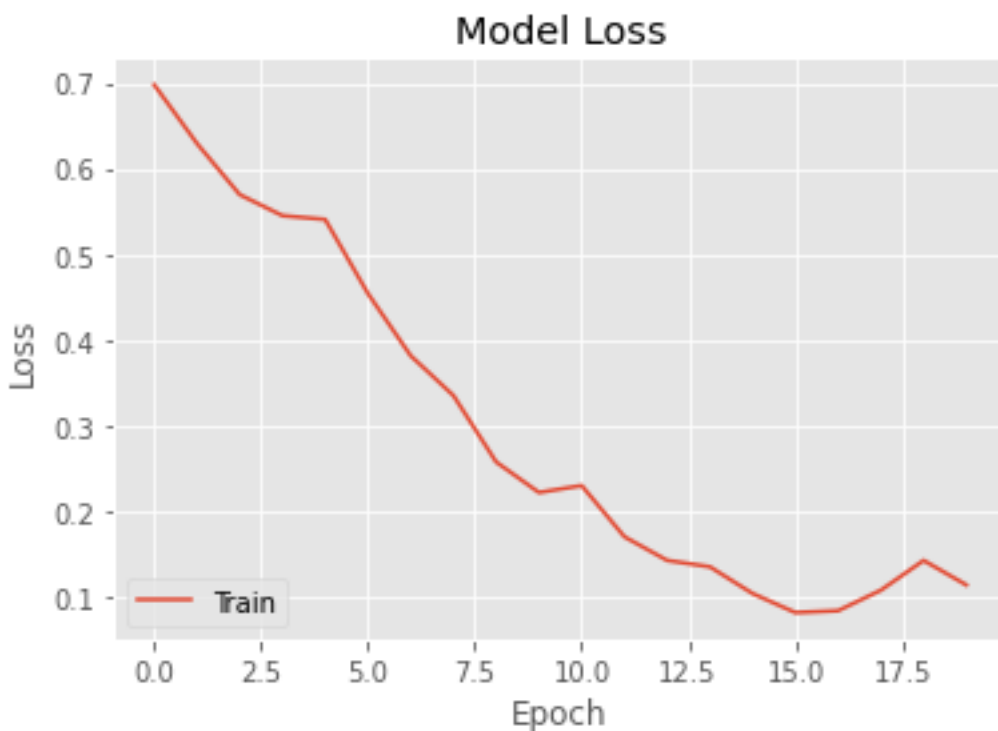


Рисунок 4.17 - Функція втрат для LSTM-CNN моделі

Для порівняння моделей використовуємо матриці помилок класифікації (рис. 4.18 – 4.21) та ROC криву (рис. 4.22)

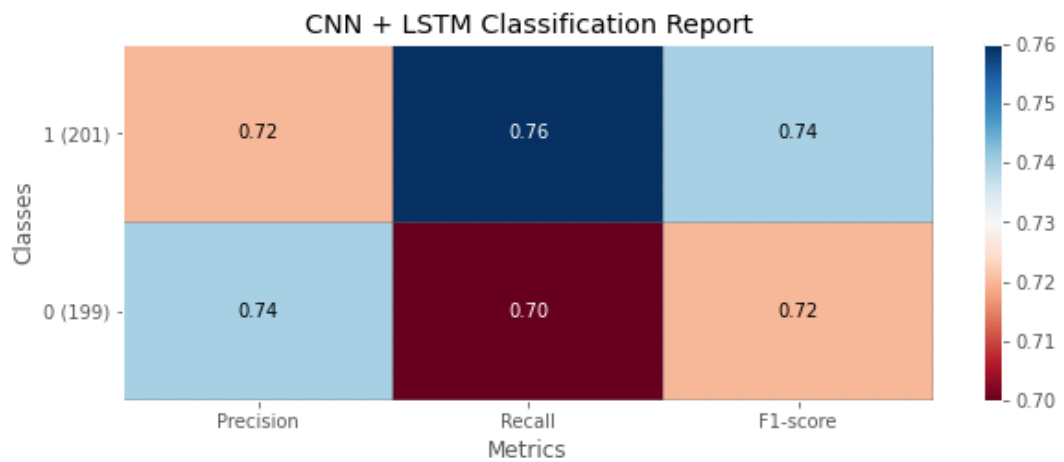


Рисунок 4.18 – Результати класифікації для LSTM-CNN моделі

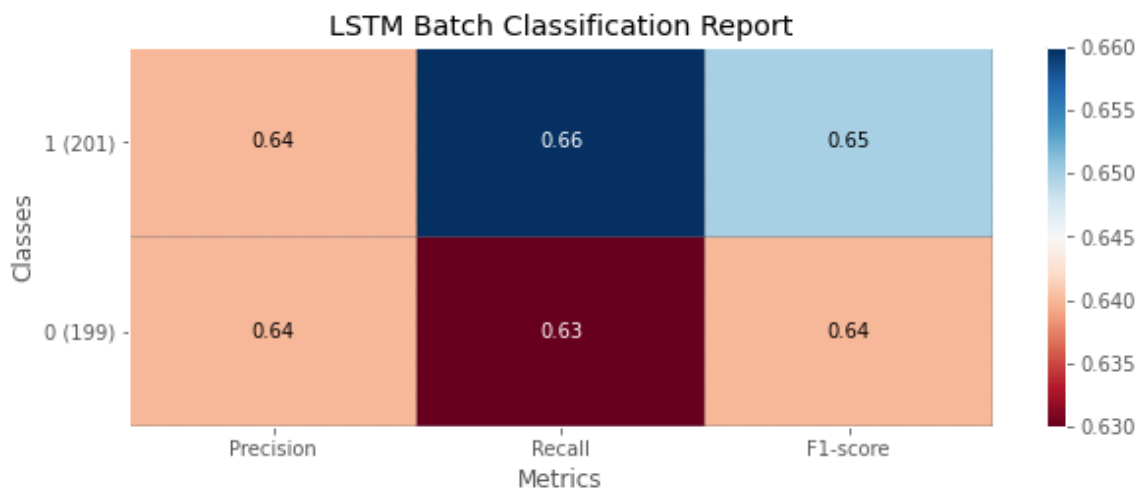


Рисунок 4.19 – Результати класифікації для LSTM-BATCH моделі

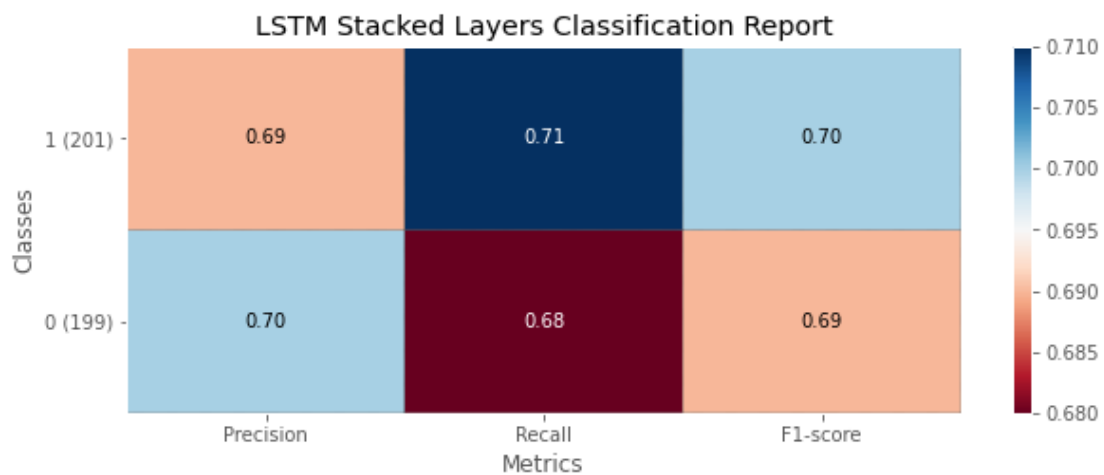


Рисунок 4.20 – Результати класифікації для LSTM-3L моделі

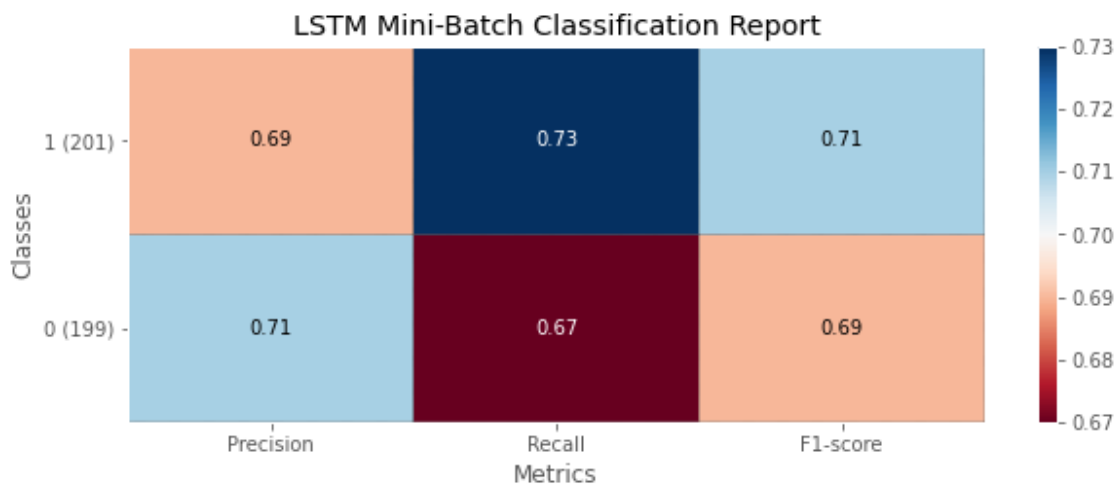


Рисунок 4.21 – Результати класифікації для LSTM-Mini-Batch моделі

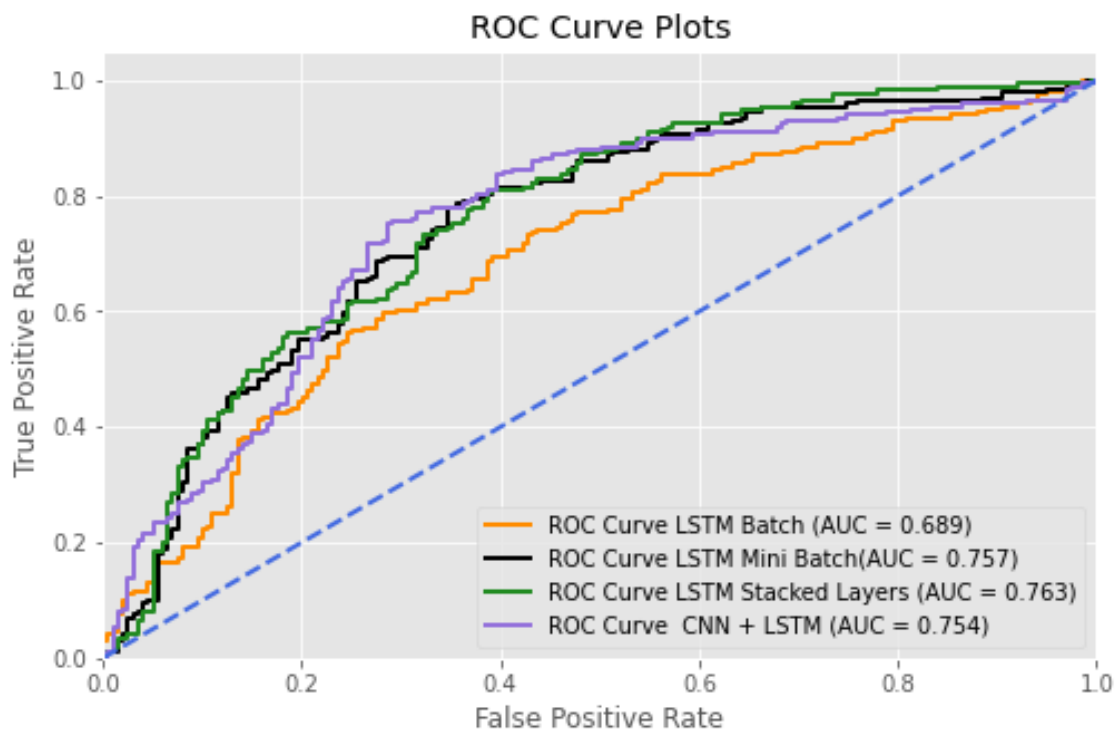


Рисунок 4.22 – ROC крива для досліджуваних моделей

Як бачимо з рисунка 4.22, усі моделі показують приблизно однаковий результат, але поки що найкращою є модель з 3 шарами, хоча під час тестування можна було побачити, що найбільш стабільною є модель Mini-Batch та CNN + LSTM.

ВИСНОВКИ

В роботі було розроблене вебзастосування, що дозволяє класифікувати емоції україномовних документів.

Розроблене застосування дозволяє користувачу завантажувати документи, маніпулювати зовнішнім виглядом системи, використовувати різні моделі для аналізу тональності.

Розроблене та протестоване програмне забезпечення для навчання неймереж. Було навчено та протестовано 5 моделей, які можна використовувати в розробленому веб-застосуванні. Найкращу точність отримано для моделі Stacked LSTM з 3 шарами.

Для навчання та тестування моделей було використано наступні пакети python: Для реалізації підсистеми був використано наступні бібліотеки:

- Keras (v2.4+);
- Gensim (v3.8+);
- Pandas (v1.2+);
- NumPy (v1.19.5+);
- NLTK (v3.5+);
- python-decouple (v3.4+);
- pymorphy2-dicts-uk (v2.4.1+);
- pymorphy2 (v0.9+);
- scikit-learn (v0.24.1);
- SciPy (v0.19.1+);
- Matplotlib (v2.1.1+);
- Jupyter.

Для створення вебзастосування використувався фреймворк Django.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Фольтович В. Метод контент-аналізу текстової інформації інтернет-газети / В. Фольтович, М. Коробчинський, Л. Чирун, В. Висоцька // Вісник Національного університету «Львівська політехніка». Серія: Комп'ютерні науки та інформаційні технології. — Львів : Видавництво Львівської політехніки, 2017. - № 864. - С. 7–19.
2. Пальчук В. Сучасні особливості розвитку методів контент-моніторингу і контент-аналізу інформаційних потоків / В. Пальчук // Наук. пр. Нац. б-ки України ім. В. І. Вернадського : зб. наук. пр. / НАН України, Нац. б-ка України ім. В. І. Вернадського, Асоц. б-к України. — Київ, 2017. — Вип. 48. — С. 360–374.
3. Немеш О. Аналіз тональності тексту: основні поняття та приклади застосування / Оксана Немеш, Андрій Романюк, Василь Теслюк // Людина. Комп'ютер. Комунікація : збірник наукових праць / Національний університет "Львівська політехніка", Інститут комп'ютерних наук та інформаційних технологій, Кафедра прикладної лінгвістики. — Львів : Видавництво Львівської політехніки, 2015. — С. 47–49.
4. Шаховська Н.Б. Шкалювання емоційно забарвлених слів для використання у методах класифікації тональності / Н. Б. Шаховська, Х. Ю. Гірак // Вісник Національного університету "Львівська політехніка". Серія : Інформаційні системи та мережі. - 2017. - № 872. - С. 195-203. - Режим доступу: http://nbuv.gov.ua/UJRN/VNULPICM_2017_872_23
5. S. Smetanin, "The Applications of Sentiment Analysis for Russian Language Texts: Current Challenges and Future Perspectives," in IEEE Access, vol. 8, pp. 110693-110719, 2020, doi: 10.1109/ACCESS.2020.3002215.
6. . Schütze, C. D. Manning and P. Raghavan, Introduction to Information Retrieval, Cambridge, U.K.:Cambridge Univ. Press, vol. 39, 2008.
7. J. R. Quinlan, "Induction of decision trees", Mach. Learn., vol. 1, no. 1,

pp. 81-106, 1986.

8. D. G. Kleinbaum, K. Dietz, M. Gail, M. Klein and M. Klein, Logistic Regression, Atlanta, GA, USA:Springer, 2002.

9. S. R. Gunn, "Support vector machines for classification and regression", vol. 14, no. 1, pp. 5-16, 1998, [online] Available: <http://ce.sharif.ir/courses/85-86/2/ce725/resources/root/LECTURES/SVM.pdf>.

10. Сложенные длинные сети краткосрочной памяти. [Электрон. ресурс]. – Режим доступа до ресурсу <https://www.machinelearningmastery.ru/stacked-long-short-term-memory-networks/>

11. Анализ тональности текстов с помощью сверточных нейронных сетей [Электрон. ресурс]. – Режим доступа до ресурсу <https://habr.com/ru/company/mailru/blog/417767/>

12. Топ-10 фреймворков для искусственного интеллекта: часть первая [Электрон. ресурс]. – Режим доступа до ресурсу: <https://vc.ru/ml/80391-top-10-freymvorkov-dlya-iskusstvennogo-intellekta-chast-pervaya>

13. Сравнение фреймворков для глубокого обучения: TensorFlow, PyTorch, Keras, MXNet, Microsoft Cognitive Toolkit, Caffe, etc [Электрон. ресурс]. – Режим доступа до ресурсу: <https://habr.com/ru/company/otus/blog/443874/>

14. An end-to-end open source machine learning platform [Electronic resource]. – Access mode: <https://www.tensorflow.org/>.

15. From research to production a flexible and efficient [Electronic resource]. – Access mode: <https://pytorch.org/>.

16. Simple. Flexible. Powerful. [Electronic resource]. – Access mode: <https://keras.io/>.

17. Library for deep learning. [Electronic resource]. – Access mode: <https://mxnet.apache.org/versions/1.8.0/>.

18. The Microsoft Cognitive Toolkit. [Electronic resource]. – Access mode: <https://docs.microsoft.com/en-us/cognitive-toolkit/>

ДОДАТОК А

Текст програми

dataset_analysis.py

```

import pickle
import pymorphy2
import numpy as np
import pandas as pd
import nltk.classify.util
import xml.etree.ElementTree as ET

from re import match
from nltk import FreqDist
from random import shuffle
from stemmer_ua import UAStemmer
from nltk.classify import NaiveBayesClassifier
from sklearn.model_selection import train_test_split
from nltk.tokenize import word_tokenize, sent_tokenize, RegexpTokenizer

testRead      =      open('data/positive.review',      'r',      encoding='utf8',
errors='ignore').readlines()
print('=====TEXT EXAMPLE=====')
for i in range(1, 44):
    print(testRead[i])
print('=====')

pos_reviews    =      open('data/positive.review',      'r',      encoding='utf8',
errors='ignore').read()
neg_reviews    =      open('data/negative.review',      'r',      encoding='utf8',
errors='ignore').read()

pos_rev_tree = ET.fromstring(pos_reviews)
neg_rev_tree = ET.fromstring(neg_reviews)

pos_tags = pos_rev_tree.findall('review')
neg_tags = neg_rev_tree.findall('review')

print('\nNumber of Positive Reviews:', len(pos_tags),
      '\nNumber of Positive Reviews:', len(neg_tags))

```

```
REVIEW_TAGS = ['unique_id', 'asin', 'product_name', 'helpful', 'rating', 'title',
               'date', 'reviewer', 'reviewer_location', 'review_text']
```

```
def parseXML(xml_reviews):
    reviews = {}
    count = 0

    for item in xml_reviews:
        count += 1
        rev_name = 'review' + str(count)
        reviews[rev_name] = [
            REVIEW_TAGS[0] + ' | ' +
            item.find(REVIEW_TAGS[0]).text.strip(),
            REVIEW_TAGS[1] + ' | ' +
            item.find(REVIEW_TAGS[1]).text.strip(),
            REVIEW_TAGS[2] + ' | ' +
            item.find(REVIEW_TAGS[2]).text.strip(),
            REVIEW_TAGS[3] + ' | ' +
            item.find(REVIEW_TAGS[3]).text.strip(),
            REVIEW_TAGS[4] + ' | ' +
            item.find(REVIEW_TAGS[4]).text.strip(),
            REVIEW_TAGS[5] + ' | ' +
            item.find(REVIEW_TAGS[5]).text.strip(),
            REVIEW_TAGS[6] + ' | ' +
            item.find(REVIEW_TAGS[6]).text.strip(),
            REVIEW_TAGS[7] + ' | ' +
            item.find(REVIEW_TAGS[7]).text.strip(),
            REVIEW_TAGS[8] + ' | ' +
            item.find(REVIEW_TAGS[8]).text.strip(),
            REVIEW_TAGS[9] + ' | ' + item.find(REVIEW_TAGS[9]).text.strip()
        ]

    return reviews
```

```
pos_reviews_dict = parseXML(pos_tags)
neg_reviews_dict = parseXML(neg_tags)
```

```
def dict_to_dataframe(reviews_dict):
    # prepare our dataframe for the data
```

```

df = pd.DataFrame(columns=REVIEW_TAGS)
count = 0
for val in reviews_dict.values():
    df.loc[count] = [
        val[0].split("|")[1], val[1].split("|")[1],
        val[2].split("|")[1], val[3].split("|")[1],
        val[4].split("|")[1], val[5].split("|")[1],
        val[6].split("|")[1], val[7].split("|")[1],
        val[8].split("|")[1], val[9].split("|")[1]
    ]

    count = count + 1

return df

pos_books = dict_to_dataframe(pos_reviews_dict)
neg_books = dict_to_dataframe(neg_reviews_dict)

print('\nNumber of Positive Reviews (Dataframe):', len(pos_books),
      '\nNumber of Negative Reviews (Dataframe):', len(neg_books))

pos_books.drop('unique_id', axis=1, inplace=True)
neg_books.drop('unique_id', axis=1, inplace=True)

pos_books['Class'] = "pos" # positive reviews
neg_books['Class'] = "neg" # negative reviews

reviews = pd.concat([pos_books, neg_books])

print('\n=====REVIEW TEXT EXAMPLE=====\\n' +
      reviews.iloc[1, 8] +
      '\\n=====\\n')

def word_sent_tokenize(sent):
    return word_tokenize(sent), sent_tokenize(sent)

def regex_tokenizer(sent):
    return RegexpTokenizer(r'\w+').tokenize(sent)

```

```

words, sents = word_sent_tokenize(reviews.iloc[1, 8])
print("\tReview ID1 - Words:\n", words, "\n\n\tReview ID1 - Sentences:\n",
sents)

```

```

STOP_WORDS = set(line.strip() for line in open('data/stopwords_ua.txt',
mode="r", encoding="utf-8"))

```

```

def stopwords_elimination(stop_words, sent):
    return [w for w in sent if w not in stop_words]

```

```

filtered_words = stopwords_elimination(STOP_WORDS, words)
print("\n===== WORDS FROM REVIEW 1
=====
\n\n===== FILTERED WORDS REVIEW
1=====
\n\n===== STOP WORDS
=====
\n", STOP_WORDS)

```

```

def stem_words(words):
    return [UAStemmer(word).stem_word() for word in words]

```

```

stemmed_words = stem_words(filtered_words)
print(stemmed_words)

```

```

morph = pymorphy2.MorphAnalyzer(lang='uk')
print(morph.parse('відчувається'))

```

```

def lemmatize_words(morph, words):
    return [morph.parse(word)[0].normal_form for word in words]

```

```

lemmatized_words = lemmatize_words(morph, filtered_words)
print(lemmatized_words)

```

```

# Use regex_tokenizer to get rid of punctuation
words_list = [regex_tokenizer(item) for item in list(reviews['review_text'])]
print("Review without punctuation: ", words_list[1])

```

```

# Eliminate stopwords
words_list = [stopwords_elimination(STOP_WORDS, word) for word in
words_list]
print("\n\nReview after stopwords elimination: ", words_list[1])

# Lemmatizing
words_list = [lemmatize_words(morph, word) for word in words_list]
print("\n\nReview after lemmatizing: ", words_list[1])

all_words = np.concatenate(words_list)
all_words_freq = FreqDist(all_words) # count the number of times that each
outcome of an experiment occurs

print("Most popular words: ", all_words_freq.most_common(100))
print("\nTotal number of words: ", len(all_words_freq.keys()))

def irr_words_elim(words):
    for word in words:
        if bool(match(r"[a-zA-Z0-9]", word)):
            words.remove(word)
        elif len(word) <= 3:
            words.remove(word)

    return words

for review in words_list:
    irr_words_elim(review)

all_words = np.concatenate(words_list)
all_words_freq = FreqDist(all_words)

print("\nTotal number of words: ", len(all_words_freq))

def create_doc(reviews, words_list):
    category = list(reviews['Class'])
    docs = [(words_list[word], category[word]) for word in
range(len(words_list))]
    shuffle(docs)
    return docs

```

```

docs = create_doc(reviews, words_list)
print(docs[1], "\n\nNumber of documents:", len(docs))

with open("output/documents.pql", "wb") as pqld: # Pickling
    pickle.dump(docs, pqld)

def find_matches(doc, all_words, num_freq=50):
    word_matches = [word[0] for word in
all_words.most_common(num_freq)]
    words = regex_tokenizer(doc)

    matches = {}
    for word in word_matches:
        matches[word] = (word in words)

    return matches

print(find_matches("У мене є книга, яку я люблю читати коли
вечоріє...", all_words_freq))

def create_matches_set(docs, all_words, num_freq=100):
    matches = []
    for doc, state in docs:
        match = find_matches(' '.join(doc), all_words, num_freq)
        matches.append((match, state))
    return matches

matches_set = create_matches_set(docs[:10], all_words_freq)
print(matches_set[1])

train_set, test_set = train_test_split(matches_set, random_state=42)
print(train_set[5])

with open("output/matches_set.pql", "wb") as ms: # Pickling
    pickle.dump(matches_set, ms)

```

```

reviews.to_csv("output/reviews.csv", sep='\t', encoding='utf-8')

with open("output/matches_set.pkl", "rb") as ms: # Unpickling
    r = pickle.load(ms)
    print(r[2])

matches_set = create_matches_set(docs, all_words_freq, 2500)

train_set, test_set = train_test_split(matches_set, test_size=0.33,
random_state=42)
    print("Length of Matches Set:", len(matches_set), "\nLength of Train Set:",
len(train_set))

def naive_bayes_model(train_set, test_set):
    classifier = NaiveBayesClassifier.train(train_set)
    informative = classifier.show_most_informative_features(10)
    accuracy = nltk.classify.accuracy(classifier, test_set) * 100
    return classifier, informative, accuracy

classifier, informative, accuracy = naive_bayes_model(train_set, test_set)
print(informative)
print("\nClassifier Accuracy:", str(round(accuracy, 2)) + "%") # predict

def words_preparation(sent):
    words = regex_tokenizer(sent) # remove punctuation
    words = stopwords_elimination(STOP_WORDS, words) # eliminate
stopwords
    words = lemmatize_words(morph, words) # lemmatizing
    words = irr_words_elim(words) # irrelevant words elimination
    print("Sentence after processing:", words)
    return " ".join(words)

test_sent = "Це найкращий фільм, який я бачив!"
print("Test sentence:", test_sent)

test_sent = find_matches(words_preparation(test_sent), all_words_freq, 2500)
print(classifier.classify(test_sent))

test_sent = "Ця книга є достатньо читабельною.."

```

```

print("Test sentence:", test_sent)

test_sent = find_matches(words_preparation(test_sent), all_words_freq, 2500)
print(classifier.classify(test_sent))

test_sent = "Мене дратує, що знову ввели локдаун, це неправильний
підхід."
print("Test sentence:", test_sent)

test_sent = find_matches(words_preparation(test_sent), all_words_freq, 2500)
print(classifier.classify(test_sent))

```

sa_classifications.py

```

import numpy as np
import pickle
import gensim.models.keyedvectors as word2vec

from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report

with open('data/documents.pql', 'rb') as f:
    docs = pickle.load(f)
    print("Number of documents:", len(docs))

model = word2vec.KeyedVectors.load_word2vec_format('uber corpus.lowercased.lemmatized.
word2vec.300d.txt', binary=False)

words = list(model.vocab)
print(words[:50], "\n\nTotal words:", len(words), "\n\nWord-Vectors shape:",
model.vectors.shape)
print(model['гарний'])

def sent_embed(words, docs):
    x_sent_embed, y_sent_embed = [], []
    count_words, count_non_words = 0, 0

```

recover the embedding of each sentence with the average of the vector that composes it

sent - sentence, state - state of the sentence (pos/neg)

for sent, state in docs:

average embedding of all words in a sentence

sent_embed = []

for word in sent:

try:

if word is present in the dictionary - add its vector representation

count_words += 1

sent_embed.append(model[word])

except KeyError:

if word is not in the dictionary - add a zero vector

count_non_words += 1

sent_embed.append([0] * 300)

add a sentence vector to the list

x_sent_embed.append(np.mean(sent_embed, axis=0).tolist())

add a label to y_sent_embed

if state == 'pos':

y_sent_embed.append(1)

elif state == 'neg':

y_sent_embed.append(0)

print(count_non_words, "out of", count_words, "words were not found in the vocabulary.")

return x_sent_embed, y_sent_embed

x, y = sent_embed(words, docs)

def cosine_similarity(u, v):

"""

Cosine similarity reflects the degree of similarity between u and v

Arguments:

u -- a word vector of shape (n,)

v -- a word vector of shape (n,)

Returns:

cosine_similarity -- the cosine similarity between u and v defined by the formula above.

```

"""

distance = 0.0

# compute the dot product between u and v
dot = np.dot(u, v)

# compute the L2 norm of u
norm_u = np.sqrt(sum(u ** 2))

# Compute the L2 norm of v
norm_v = np.sqrt(sum(v ** 2))

# Compute the cosine similarity defined by formula above
cosine_similarity = dot / (norm_u * norm_v)

return cosine_similarity

print("Sentence #6: ", docs[6], "\n\nSentence #7: ", docs[7])
print("\nSentence Embedding #6: ", x[6], "\n\nSentence Embedding #7: ",
x[7])
print("cosine_similarity = ", cosine_similarity(np.array(x[6]), np.array(x[7])))

print("Sentence #0: ", docs[0], "\n\nSentence #3: ", docs[3])
print("\nSentence Embedding #0: ", x[0], "\n\nSentence Embedding #3: ",
x[3])
print("cosine_similarity = ", cosine_similarity(np.array(x[0]), np.array(x[3])))

print("Sentence #1: ", docs[1], "\n\nSentence #2: ", docs[2])
print("cosine_similarity = ", cosine_similarity(np.array(x[1]), np.array(x[2])))

# train test
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.15,
random_state=1)

# train dev
x_train, x_val, y_train, y_val = train_test_split(x_train, y_train,
test_size=0.15, random_state=1)

print('Length of x_train:', len(x_train), '| Length of y_train:', len(y_train))
print('Length of x_test: ', len(x_test), '| Length of y_test: ', len(y_test))
print('Length of x_val: ', len(x_val), '| Length of y_val: ', len(y_val))

```

```

print("Shape of x_train set:", np.array(x_train).shape)

error = []

# calculating error for neighbor values between 1 and 25
for i in range(1, 25):
    knn = KNeighborsClassifier(n_neighbors=i)
    knn.fit(x_train, y_train)
    pred_i = knn.predict(x_test)
    error.append(np.mean(pred_i != y_test))

# create KNN Classifier
knn = KNeighborsClassifier(n_neighbors=5, weights='distance')

# train the classifier using the training sets
knn.fit(x_train, y_train)

# predict the response for test dataset
y_pred = knn.predict(x_test)

print("Nearest Neighbors Result (k=5):\n" + '-' * 35)
print("Accuracy Score (k=5):", str(round(knn.score(x_test, y_test), 4) * 100)
+ '%')
print("Accuracy (x_train, y_train):", str(round(knn.score(x_train, y_train), 4)
* 100) + '%')
print('\nClassification          KNN:\n',          classification_report(y_test,
knn.predict(x_test)))

logit = LogisticRegression(solver='liblinear', multi_class='ovr', n_jobs=1)
logit.fit(x_train, y_train)
print("Accuracy Score:", str(round(logit.score(x_test, y_test), 4) * 100) + '%')
print('\nClassification Logistic Regression:\n', classification_report(y_test,
logit.predict(x_test)))

clf = RandomForestClassifier(n_estimators=100)
clf.fit(x_train, y_train)
print("Accuracy Score:", str(round(clf.score(x_test, y_test), 4) * 100) + '%')
print('\nClassification Random Forest:\n', classification_report(y_test,
clf.predict(x_test)))

with open('LogitModel.pickle', 'wb') as m:
    pickle.dump(logit, m)

```

```

with open('LogitModel.pickle', 'rb') as m:
    logit = pickle.load(m)
    print("Logistic Regression Accuracy Score:", str(round(logit.score(x_test,
y_test), 4) * 100) + '%')

```

home.html

```

{% load static %}
{% load app_tags %}

<!DOCTYPE html>
<html lang="en" >
<head>
    <meta charset="UTF-8">
    <title>Management Page</title>

    <meta property="og:title" content="Sensus Web" />
    <meta property="og:description" content="My site with lots of my old and
new works on JavaScript and Django" />
    <meta property="og:type" content="website" />

    <link rel="icon" href="{% static 'home/ico/logo.ico' %}" type="image/x-
icon">
    <link rel='stylesheet' href='https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/4.7.0/css/font-awesome.min.css'>
    <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@4.6.0/dist/css/bootstrap.min.css"
integrity="sha384-
B0vP5xmATw1+K9KRQjQERJvTumQW0nPEzvf6L/Z6nronJ3oUOFUFpCjEUQou
q2+l" crossorigin="anonymous">
    <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/normalize/5.0.0/normalize.min.css">
    <link rel="stylesheet" href="{% static 'home/css/analyze.css' %}">
</head>
<body>

<div class="app-container">
    <div class="app-header">
        <div class="app-header-left">
            <span class="app-icon">
                <svg xmlns="http://www.w3.org/2000/svg" width="24"
height="24" viewBox="0 0 24 24" fill="none" stroke="currentColor" stroke-

```

```

width="2" stroke-linecap="round" stroke-linejoin="round" class="feather feather-
activity">
    <polyline points="22 12 18 12 15 21 9 3 6 12 2 12" />
</svg>
</span>
<p class="app-name" style="margin: 0 32px 0 20px;">Sensus</p>
<div class="search-wrapper">
    <input class="search-input" type="text" placeholder="Поиск">
    <svg      xmlns="http://www.w3.org/2000/svg"      width="20"
height="20" fill="none" stroke="currentColor" stroke-linecap="round" stroke-
linejoin="round" stroke-width="2" class="feather feather-search" viewBox="0 0 24
24">
        <defs></defs>
        <circle cx="11" cy="11" r="8"></circle>
        <path d="M21 21l-4.35-4.35"></path>
    </svg>
</div>
</div>

<div class="app-header-right">
    <button class="mode-switch" title="Switch Theme">
        <svg class="moon" fill="none" stroke="currentColor" stroke-
linecap="round" stroke-linejoin="round" stroke-width="2" width="32" height="32"
viewBox="0 0 24 24">
            <defs></defs>
            <path d="M21 12.79A9 9 0 1111.21 3 7 7 0 0021
12.79z"></path>
        </svg>
    </button>

    <button class="add-btn" title="Add New File"
onclick="window.location.href='{% url "upload" %}'">
        <svg      xmlns="http://www.w3.org/2000/svg"      width="16"
height="16" viewBox="0 0 24 24" fill="none" stroke="currentColor" stroke-
width="3" stroke-linecap="round" stroke-linejoin="round" class="btn-icon feather
feather-plus">
            <line x1="12" y1="5" x2="12" y2="19" />
            <line x1="5" y1="12" x2="19" y2="12" />
        </svg>
    </button>

    <button class="profile-btn">
        {% if user.is_authenticated %}

```

```

        
        <span>{{ user.username }}</span>
    {% endif %}
</button>
</div>
</div>

<div class="app-content" id="container">
    <div class="app-sidebar">
        <a href="" class="app-sidebar-link active">
            <svg      xmlns="http://www.w3.org/2000/svg"      width="24"
height="24" viewBox="0 0 24 24" fill="none" stroke="currentColor" stroke-
width="2" stroke-linecap="round" stroke-linejoin="round" class="feather feather-
home">
                <path d="M3 9l9-7 9 7v11a2 2 0 0 1-2 2H5a2 2 0 0 1-2-2z" />
                <polyline points="9 22 9 12 15 12 15 22" />
            </svg>
        </a>

        <a href="" class="app-sidebar-link">
            <svg      xmlns="http://www.w3.org/2000/svg"      width="24"
height="24" fill="none" stroke="currentColor" stroke-linecap="round" stroke-
linejoin="round" stroke-width="2" class="link-icon feather feather-settings"
viewBox="0 0 24 24">
                <defs></defs>
                <circle cx="12" cy="12" r="3" />
                <path d="M19.4 15a1.65 1.65 0 00.33 1.82l.06.06a2 2 0 010 2.83
2 2 0 01-2.83 0l-.06-.06a1.65 1.65 0 00-1.82-.33 1.65 1.65 0 00-1 1.51V21a2 2 0 01-
2 2 2 2 0 01-2-2v-.09a1.65 1.65 0 009 19.4a1.65 1.65 0 00-1.82.33l-.06.06a2 2 0 01-
2.83 0 2 2 0 010-2.83l.06-.06a1.65 1.65 0 00.33-1.82 1.65 1.65 0 00-1.51-1H3a2 2 0
01-2-2 2 2 0 012-2h.09a1.65 1.65 0 004.6 9a1.65 1.65 0 00-.33-1.82l-.06-.06a2 2 0
010-2.83 2 2 0 012.83 0l.06.06a1.65 1.65 0 001.82.33H9a1.65 1.65 0 001-1.51V3a2
2 0 012-2 2 2 0 012 2v.09a1.65 1.65 0 001 1.51 1.65 1.65 0 001.82-.33l.06-.06a2 2 0
012.83 0 2 2 0 010 2.83l-.06.06a1.65 1.65 0 00-.33 1.82V9a1.65 1.65 0 001.51
1H21a2 2 0 012 2 2 2 0 01-2 2h-.09a1.65 1.65 0 00-1.51 1z" />
            </svg>
        </a>

        {% if user.is_authenticated %}
        <a href="{% url 'logout' %}" class="app-sidebar-link">
            <svg      xmlns="http://www.w3.org/2000/svg"      width="24"
height="24" fill="none" stroke="currentColor" stroke-linecap="round" stroke-

```

```

linejoin="round" stroke-width="2" class="link-icon feather feather-log-out"
viewBox="0 0 24 24">
    <path d="M9 21H5a2 2 0 0 1-2-2V5a2 2 0 0 1 2-2h4" />
    <polyline points="16 17 21 12 16 7" />
    <line x1="21" y1="12" x2="9" y2="12" />
</svg>
</a>
{% else %}
<a href="{% url 'login' %}" class="app-sidebar-link">
    <svg xmlns="http://www.w3.org/2000/svg" width="24"
height="24" fill="none" stroke="currentColor" stroke-linecap="round" stroke-
linejoin="round" stroke-width="2" class="feather feather-log-in" viewBox="0 0 24
24">
        <path d="M15 3h4a2 2 0 0 1 2 2v14a2 2 0 0 1-2 2h-4" />
        <polyline points="10 17 15 12 10 7" />
        <line x1="15" y1="12" x2="3" y2="12" />
    </svg>
</a>
{% endif %}
</div>

<div class="projects-section">
    <div class="projects-section-header">
        <p>Документи</p>
        <p class="time">{% now "d.m.Y" %}</p>
    </div>

    <div class="projects-section-line">
        <div class="projects-status">
            <div class="item-status">
                <span class="status-number">{{ documents|length }}</span>
                <span class="status-type">Кількість документів</span>
            </div>
        </div>

        <div class="view-actions">
            <button class="view-btn list-view" title="List View">
                <svg xmlns="http://www.w3.org/2000/svg" width="24"
height="24" viewBox="0 0 24 24" fill="none" stroke="currentColor" stroke-
width="2" stroke-linecap="round" stroke-linejoin="round" class="feather feather-
list">
                    <line x1="8" y1="6" x2="21" y2="6" />
                    <line x1="8" y1="12" x2="21" y2="12" />

```

```

        <line x1="8" y1="18" x2="21" y2="18" />
        <line x1="3" y1="6" x2="3.01" y2="6" />
        <line x1="3" y1="12" x2="3.01" y2="12" />
        <line x1="3" y1="18" x2="3.01" y2="18" />
    </svg>
</button>

<button class="view-btn grid-view active" title="Grid View">
    <svg xmlns="http://www.w3.org/2000/svg" width="20"
height="20" viewBox="0 0 24 24" fill="none" stroke="currentColor" stroke-
width="2" stroke-linecap="round" stroke-linejoin="round" class="feather feather-
grid">
        <rect x="3" y="3" width="7" height="7" />
        <rect x="14" y="3" width="7" height="7" />
        <rect x="14" y="14" width="7" height="7" />
        <rect x="3" y="14" width="7" height="7" />
    </svg>
</button>
</div>
</div>

<div class="project-boxes jsGridView">
    {% if documents %}
    {% for document in documents %}
    <div class="project-box-wrapper">
        <div class="project-box" style="background-color: #a0e989;">
            <div class="project-box-header" style="margin: 0;">
                <a href="{{ document.docfile.url }}">
                    <i class="fa fa-eye" style="color: #3f51b5; font-size:
1.7em;"></i>
                </a>

                <a href="{{ document.docfile.url }}" download
class="sidebar-link">
                    <i class="fa fa-download" style="color: #4caf50; font-
size: 1.7em;"></i>
                </a>
            </div>

            <div class="project-box-content-header">
                <p class="box-content-header">Документ #{{
document.id|stringformat:"s" }}</p>

```

```

        <p                                class="box-content-subheader">{{
document.docfile.name }}</p>
        </div>

        <div        class="box-progress-wrapper"        style="text-align:
center;">
                <button title="Analyze Document" class="btn btn-success"
style="width: 250px;" onclick="processData('{{ document.docfile }}')">
                        Аналізувати Документ &nbsp;<i class="fa fa-play-
circle" style="font-size: 1.25em;"></i>
                </button>
        </div>
</div>
</div>
        {% endfor %}
        {% else %}
        <p style="text-align: center;">Документів ще немає.</p>
        {% endif %}
</div>
</div>
</div>
</div>

<script src='{% static "home/js/analyzer.js" %}'></script>
<script    src="https://code.jquery.com/jquery-3.5.1.js"    integrity="sha256-
QWo7LDvxbWT2tbbQ97B53yJnYU3WhH/C8ycbRAkjPDc="
crossorigin="anonymous"></script>
<script>
    function getCookie(c_name)
    {
        if (document.cookie.length > 0)
        {
            c_start = document.cookie.indexOf(c_name + "=");
            if (c_start !== -1)
            {
                c_start = c_start + c_name.length + 1;
                c_end = document.cookie.indexOf(";", c_start);
                if (c_end === -1) c_end = document.cookie.length;
                return unescape(document.cookie.substring(c_start,c_end));
            }
        }
        return "";
    }

```

```

function processData(link) {
    let newDiv = document.createElement("div");
    newDiv.className = "alert alert-success";
    newDiv.role = "alert";
    newDiv.innerHTML = "<button type=\"button\" class=\"close\" data-
dismiss=\"alert\"          aria-label=\"Close\"><span          aria-
hidden=\"true\">&times;</span></button>\n" +
        "Зачекайте хвилинку... Модель аналізує ваш текст";

    $("#container").before(newDiv);

    $.ajax({
        headers: { "X-CSRFToken": getCookie("csrftoken") },
        type: "POST",
        url: '/predict/',
        data: {Link: link},

        success: function() {
            window.location = "{% url 'prediction' %}";
        },

        error: function () {
            let newDiv = document.createElement("div");
            newDiv.className = "alert alert-danger";
            newDiv.role = "alert";
            newDiv.innerHTML = "<button type=\"button\" class=\"close\"
data-dismiss=\"alert\"          aria-label=\"Close\"><span          aria-
hidden=\"true\">&times;</span></button>\n" +
                "Під час спроби передбачити результат сталася помилка...";

            $("#container").before(newDiv);
        }
    });
}
</script>
<script
src="https://cdn.jsdelivr.net/npm/popper.js@1.16.1/dist/umd/popper.min.js"
integrity="sha384-
9/reFTGAW83EW2RDu2S0VKA1Zap3H66lZ81PoYlFhbGU+6BZp6G7niu735Sk7l
N" crossorigin="anonymous"></script>
<script
src="https://cdn.jsdelivr.net/npm/bootstrap@4.6.0/dist/js/bootstrap.min.js"

```

```

integrity="sha384-
+YQ4JLhJyBLPDQt//I+STsc9iw4uQqACwlvpslubQzn4u2UU2UFM80nGisd026JF"
crossorigin="anonymous"></script>
</body>
</html>

```

predict.html

```

{% load static %}

<!DOCTYPE html>
<html lang="en" >
<head>
  <meta charset="UTF-8">
  <title>Sentiment Prediction</title>
  <link rel="icon" href="{% static 'home/ico/logo.ico' %}" type="image/x-
icon">
  <link rel='stylesheet'
href='https://maxcdn.bootstrapcdn.com/bootstrap/3.2.0/css/bootstrap.min.css'>
  <link rel="stylesheet" href="{% static 'home/css/predict.css' %}">
</head>
<body>

<div class="well" style="background: #fff;">
  <h2 style="font-weight: bold;">Прогнозування Моделі</h2>
  <hr>
  <div class="row">
    {% if prediction == None %}
    <div id="prediction" data-dimension="200" data-text="0%" data-
fontsize="36" data-percent="0" data-fgcolor="#30B455" data-bgcolor="#eee" data-
width="20" data-bordersize="20" data-animationstep="2"></div>
    {% else %}
    {% if prediction < 50 %}
    <div id="prediction" data-dimension="200" data-text="{{ prediction
 }}" data-fontsize="36" data-percent="{{ prediction }}" data-fgcolor="#ff4c4c"
data-bgcolor="#eee" data-width="20" data-bordersize="20" data-
animationstep="2"></div>
    {% else %}
    <div id="prediction" data-dimension="200" data-text="{{ prediction
 }}" data-fontsize="36" data-percent="{{ prediction }}" data-fgcolor="#30B455"
data-bgcolor="#eee" data-width="20" data-bordersize="20" data-
animationstep="2"></div>

```

```

        {% endif %}
    {% endif %}
</div>

<br>
<a class="btn btn-default" href="{% url 'home_page' %}" target="_self">
    Повернуться <i class="glyphicon glyphicon-new-window"></i>
</a>
</div>

<script src="https://code.jquery.com/jquery-3.5.1.js" integrity="sha256-
QWo7LDvxbWT2tbbQ97B53yJnYU3WhH/C8ycbRAkjPDc="
crossorigin="anonymous"></script>
<script src='{% static "home/js/predict.js" %}'></script>

</body>
</html>

```

upload.html

```

{% load static %}
{% load app_tags %}

<!DOCTYPE html>
<html lang="en" >
<head>
    <meta charset="UTF-8">
    <title>Management Page</title>
    <link rel="icon" href="{% static 'home/ico/logo.ico' %}" type="image/x-
icon">
    <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@4.6.0/dist/css/bootstrap.min.css"
integrity="sha384-
B0vP5xmATw1+K9KQRQjQERJvTumQW0nPEzvF6L/Z6nronJ3oUOFUFpCjEUQou
q2+l" crossorigin="anonymous">
    <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/normalize/5.0.0/normalize.min.css">
    <link rel="stylesheet" href="{% static 'home/css/analyze.css' %}">
</head>
<body>

<div class="app-container">

```

```

<div class="app-header">
  <div class="app-header-left">
    <span class="app-icon">
      <svg      xmlns="http://www.w3.org/2000/svg"      width="24"
height="24" viewBox="0 0 24 24" fill="none" stroke="currentColor" stroke-
width="2" stroke-linecap="round" stroke-linejoin="round" class="feather feather-
activity">
        <polyline points="22 12 18 12 15 21 9 3 6 12 2 12" />
      </svg>
    </span>
    <p class="app-name">Sensus</p>
  </div>

  <div class="app-header-right">
    <button class="mode-switch" title="Switch Theme">
      <svg class="moon" fill="none" stroke="currentColor" stroke-
linecap="round" stroke-linejoin="round" stroke-width="2" width="32" height="32"
viewBox="0 0 24 24">
        <defs></defs>
        <path d="M21 12.79A9 9 0 1111.21 3 7 7 0 0021
12.79z"></path>
      </svg>
    </button>

    <button class="profile-btn">
      {% if user.is_authenticated %}
      
      <span>{{ user.username }}</span>
      {% endif %}
    </button>
  </div>
</div>

<div class="card" style="width: 80%;">
  <div class="item-text" style="color: var(--main-color); font-size: 24px;
font-weight: bold;">
    Завантажити Документи
  </div>

  <div class="card-body">
    <form action="{% url "upload" %}" method="post"
enctype="multipart/form-data">

```

```

        {% csrf_token %}
        <p>{{ form.non_field_errors }}</p>
        <div class="upload" style="color: var(--app-container); text-align:center;">
            <input type="file" class="form-control" name="docfile"
            accept=".pdf" style="height: 100%;">
            <button type="submit" class="btn btn-primary mt-4"
            style="width: 150px;">Завантажити</button>
        </div>
        {{ message }}
    </form>

</div>
</div>
{% if user.is_authenticated %}
<span style="text-align: center;">
    <form action="{% url "home_page" %}" method="post"
    enctype="multipart/form-data" style="display: inline-block; margin: 0 10px;">
        {% csrf_token %}
        <div class="upload" style="color: var(--app-container); text-align:center;">
            <button type="submit" class="btn btn-primary mt-4" style="width:
            300px;">Перейти на Головну (Адмін)</button>
        </div>
    </form>

    <form action="{% url "logout" %}" method="post"
    enctype="multipart/form-data" style="display: inline-block;">
        {% csrf_token %}
        <div class="upload" style="color: var(--app-container); text-align:center;">
            <button type="submit" class="btn btn-primary mt-4" style="width:
            200px;">Вийти з акаунту</button>
        </div>
    </form>
</span>
{% else %}
<form action="{% url "login" %}" method="post"
enctype="multipart/form-data">
    {% csrf_token %}
    <div class="upload" style="color: var(--app-container); text-align:center;">

```

```

        <button type="submit" class="btn btn-primary mt-4" style="width:
200px;">Увійти до акаунту</button>
    </div>
</form>
{% endif %}

{% if messages %}
{% for message in messages %}
    <div class="alert {% if message.tags %}alert-{{ message.tags }}{% endif
%}" role="alert">
        <button type="button" class="close" data-dismiss="alert" aria-
label="Close"><span aria-hidden="true">&times;</span></button>
        {{ message }}
    </div>
{% endfor %}
{% endif %}

<span class="footer-title" style="font-size: 16px;">{% now "Y" %}</span>

<script>
    document.addEventListener('DOMContentLoaded', function () {
        let modeSwitch = document.querySelector('.mode-switch');
        modeSwitch.addEventListener('click', function () {
            document.documentElement.classList.toggle('dark');
            modeSwitch.classList.toggle('active');
        });
    });
</script>
<script src="https://code.jquery.com/jquery-3.5.1.js" integrity="sha256-
QWo7LDvxbWT2tbbQ97B53yJnYU3WhH/C8ycbRAkjPDc="
crossorigin="anonymous"></script>
<script
src="https://cdn.jsdelivr.net/npm/popper.js@1.16.1/dist/umd/popper.min.js"
integrity="sha384-
9/reFTGAW83EW2RDu2S0VKA1Zap3H66lZ81PoYlFhbGU+6BZp6G7niu735Sk7l
N" crossorigin="anonymous"></script>
<script
src="https://cdn.jsdelivr.net/npm/bootstrap@4.6.0/dist/js/bootstrap.min.js"
integrity="sha384-
+YQ4JLhgyBLPDQt/I+STsc9iw4uQqACwlvpslubQzn4u2UU2UFM80nGisd026JF"
crossorigin="anonymous"></script>
</body>
</html>

```

ДОДАТОК Б

СЛАЙДИ ПРЕЗЕНТАЦІЇ

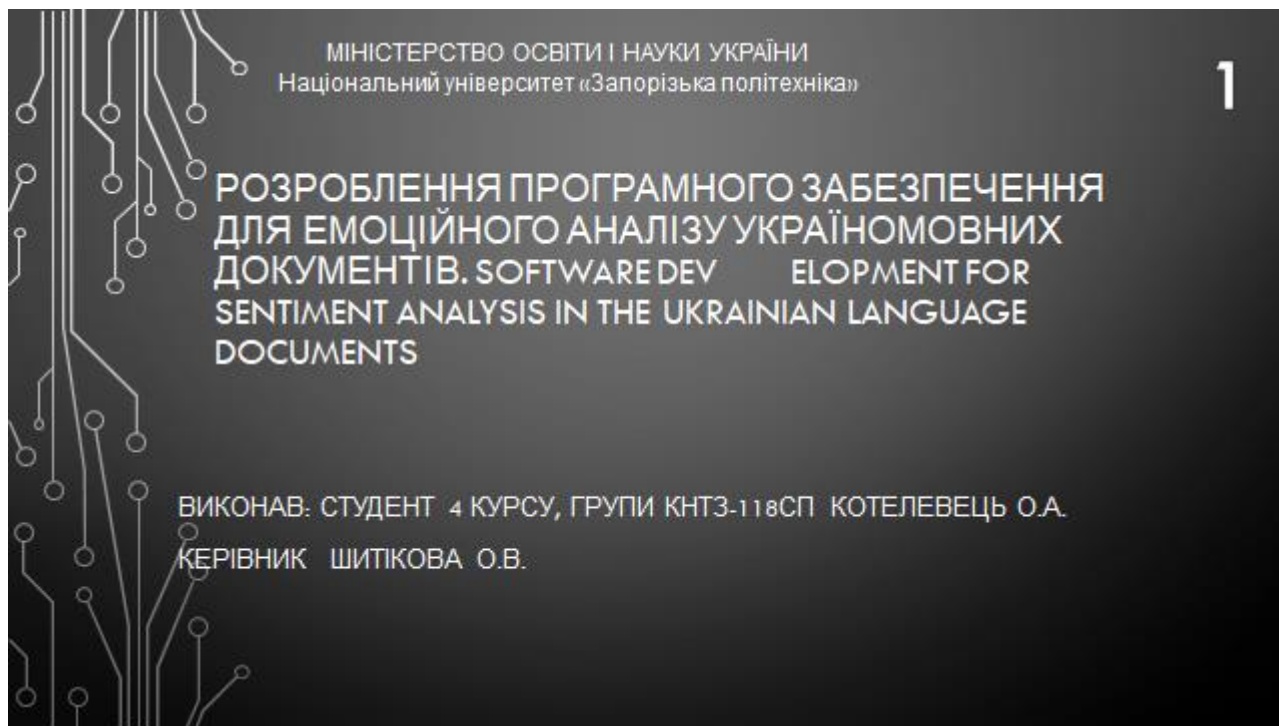


Рисунок Б.1 – Слайд 1

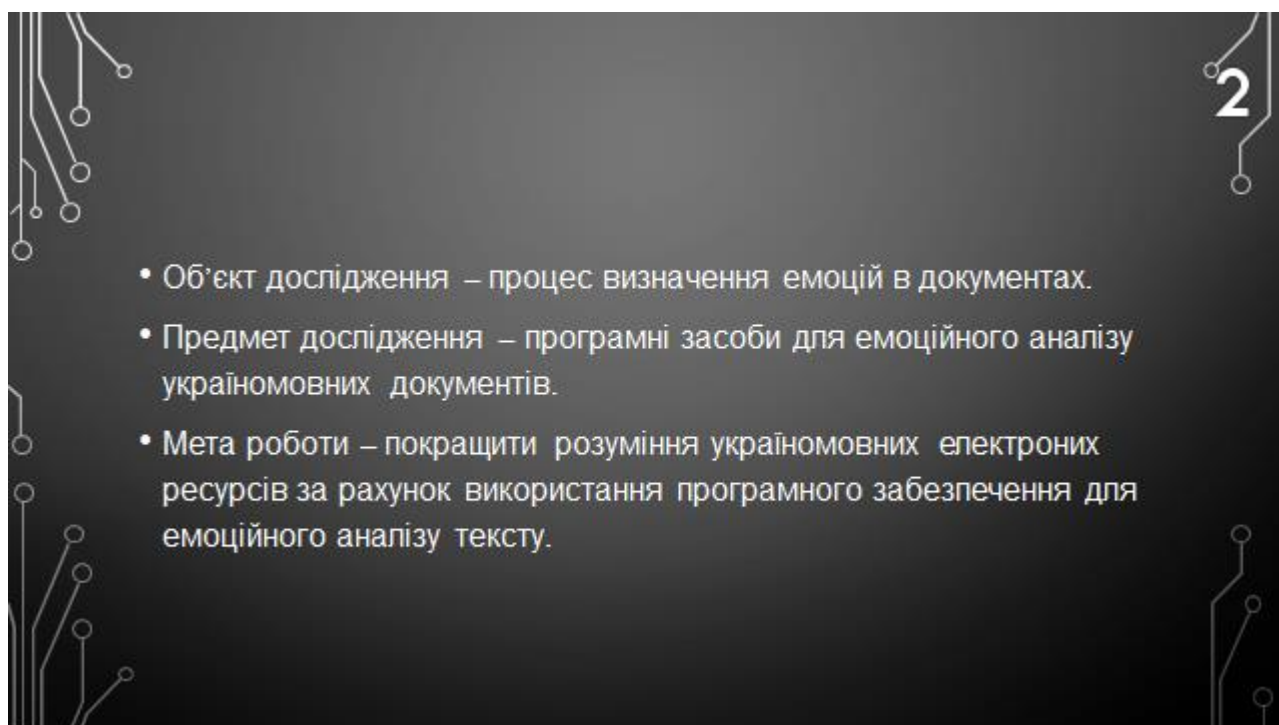


Рисунок Б.2 – Слайд 2



Рисунок Б.3 – Слайд 3

4

ПОРІВНЯННЯ ФРЕЙМВОРКІВ

	Фреймворк	Переваги	Недоліки
1	TensorFlow	Обширна документація. Потужні засоби моніторингу процесу навчання моделей і візуалізації (Tensorboard). Підтримується великою спільнотою розробників і технічними компаніями. Забезпечує обслуговування моделей; Вона підтримує розподілене навчання. Низька затримка для мобільних пристроїв.	Програє по швидкості роботи в еталонних тестах. Має більш високий вхідний поріг для початківця. Досить низькоуровнева і вимагає багато шаблонного коду.
2	PyTorch	Безліч модульних елементів, які легко комбінувати. Дозволяє писати власні типи шарів і працювати з GPU. Має широкий вибір попередньо навчених моделей. Швидко і просто прототипування.	Python єдина мова, що підтримується. Відсутні засоби моніторингу процесу навчання моделей і візуалізації
3	Keras	Має повністю гнучкий модуль. Має простий і інтуїтивно-розумілий інтерфейс, відповідно, хороший для навчання. Має вбудовану підтримку для навчання на декількох GPU. Може бути налаштований в якості оцінювача для TensorFlow і навчений на кластерах GPU на платформі Google Cloud; Запускається на Spark; Підтримує GPU від NVIDIA, TPU від Google, GPU з OpenCL, такі як AMD.	не завжди легко кастомізується; Він обмежений бекендом Tensorflow, CNTK і Theano.
4	MXNet	Швидкий, гнучкий і ефективний. Забезпечує просунуту підтримку GPU. Він може запускатися на будь-якому пристрої. Є високопродуктивне імперативне API; Забезпечує легку підтримку моделей; Він вкрай масштабований; Він забезпечує хорошу підтримку безлічі мов програмування, таких як Python, R, Scala, JavaScript і C++ і багато інших.	Обмежена підтримка
5	CNTK	Показує хорошу продуктивність і масштабованість; Надає безліч добре оптимізованих компонентів; Пропонує підтримку Apache Spark; Ефективний при використанні ресурсів; Має просту інтеграцію з Azure Cloud;	Обмежена підтримка

Рисунок Б.4 – Слайд 4



Рисунок Б.5 – Слайд 5

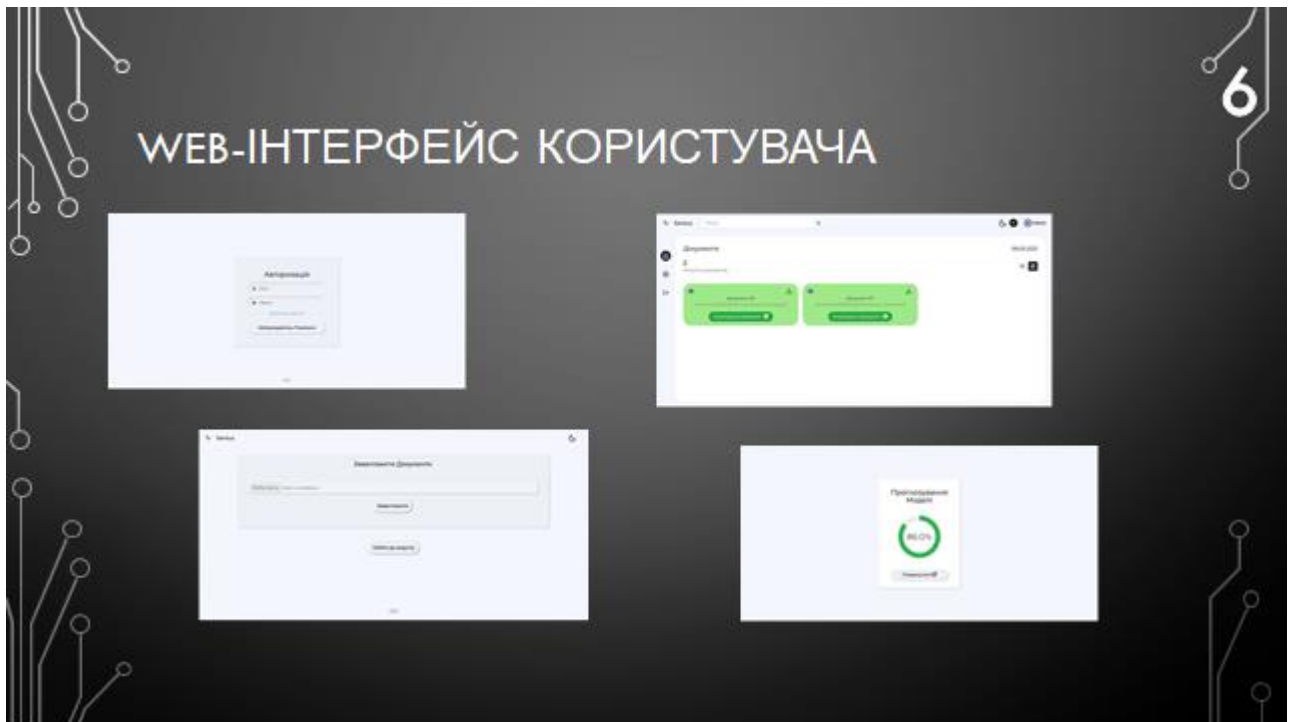


Рисунок Б.6 – Слайд 6

СТВОРЕННЯ НАБОРУ ДАНИХ

Рисунок Б.7 – Слайд 7

8

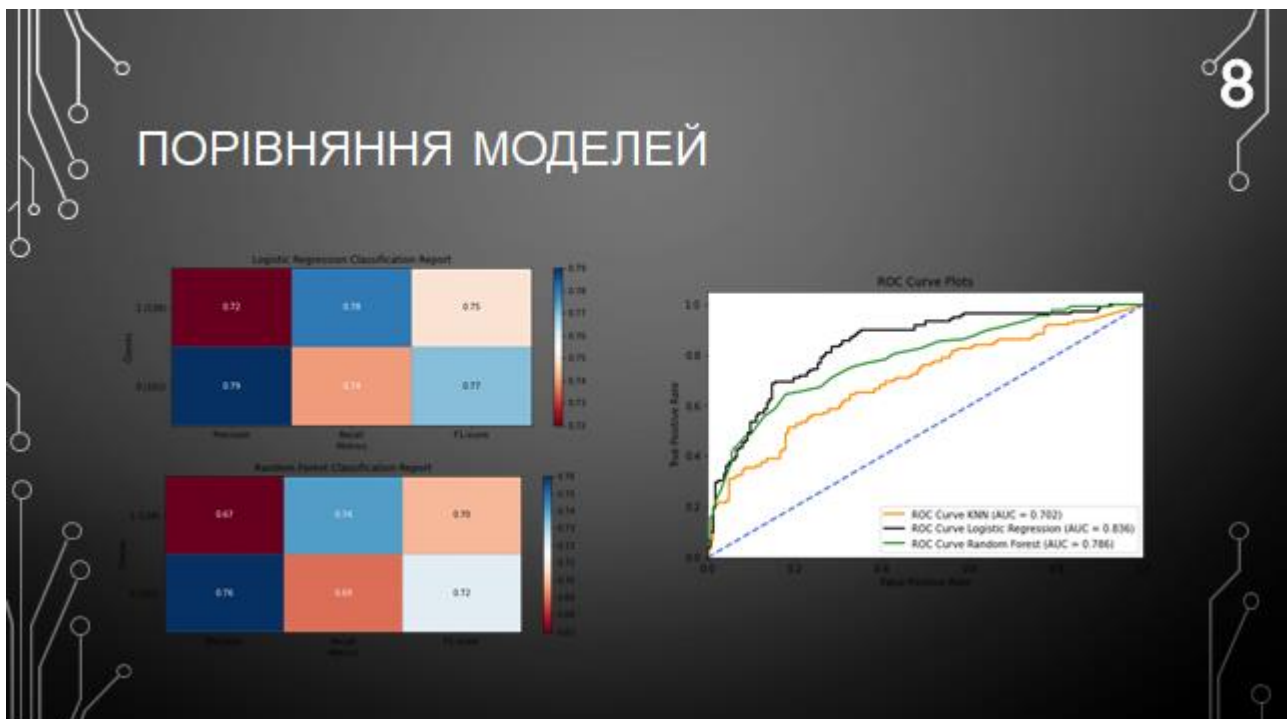


Рисунок Б.8 – Слайд 8



Рисунок Б.9 – Слайд 9

ВИСНОВКИ

- В роботі було розроблене вебзастосування, що дозволяє класифікувати емоції українськомовних документів.
- Розроблене застосування дозволяє користувачу завантажувати документи, маніпулювати зовнішнім виглядом системи, використовувати різні моделі для аналізу тональності.
- Розроблене та протестоване програмне забезпечення для навчання нейромереж. Було навчено та протестовано 5 моделей, які можна використовувати в розробленому веб-застосуванні. Найкращу точність отримано для моделі Stacked LSTM з 3 шарами.

Рисунок Б.10 – Слайд 10