

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 ЗАСТОСУНКУ ДЛЯ ВИЗНАЧЕННЯ ЕМОЦІЙ
У ДОКУМЕНТАХ ЗДОБУВАЧА МІЖНАРОДНОЇ МОБІЛЬНОСТІ
APPLICATION DEVELOPMENT FOR EMOTION RECOGNITION
IN THE DOCUMENTS OF THE APPLICANTS FOR INTERNATIONAL
MOBILITY PROGRAMMES

Виконав: студент(ка) 4 курсу, групи КНТ-218сп

Спеціальності 122 Комп'ютерні науки

(код і найменування спеціальності)

Освітня програма (спеціалізація)

Комп'ютерні науки

Цапін А.Д.

(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 122 Комп'ютерні науки
(код і найменування)
Освітня програма (спеціалізація) Комп'ютерні науки
(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 ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

Цапін Антона Дмитровича

(прізвище, ім'я, по батькові)

1. Тема проєкту (роботи) Розроблення застосунку для визначення емоцій у документах здобувача міжнародної мобільності. Application Development for Emotion Recognition in the Documents of the Applicants for International Mobility Programmes

керівник проєкту (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 проєкту (роботи) травень 2021 року
3. Вихідні дані до проєкту (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 проєкту (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 “28” лютого 2021 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (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)

(підпис)

Цапін А.Д.
(прізвище та ініціали)

Керівник проєкту (роботи)

(підпис)

Шитікова О.В.
(прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка до випускної кваліфікаційної роботи бакалавра: 84 с., 47 рис., 2 табл., 2 дод., 17 джерел.

ЕМОЦІЙНИЙ АНАЛІЗ, ДОКУМЕНТИ, МІЖНАРОДНА МОБІЛЬНІСТЬ, PYTHON, ВЕБЗАСТОСУНОК.

Об'єкт дослідження – процес автоматичного визначення емоцій в документах.

Предмет дослідження – програмні засоби для емоційного аналізу документів здобувачів міжнародної мобільності.

Мета роботи – розроблення програмного забезпечення для емоційного аналізу документів здобувачів міжнародної мобільності.

Матеріали, методи та технічні засоби: глибинні нейронні мережі, рекурентні нейронні мережі з довгою короткочасною пам'ятю, згортові нейронні мережі, Django, Python, Keras.

Результати. Розроблено вебзастосунок, що дозволяє завантажувати документи здобувачам міжнародної мобільності та визначати їх емоційний стан з використанням.

Висновки. Досліджені моделі для визначення емоцій в англійських текстах. Розроблено програмне рішення з використанням вебінтерфейсу, що дозволяє здобувачам міжнародної мобільності завантажувати відповідні документи, а співробітникам міжнародного відділу аналізувати їх емоційний стан.

Галузь використання – міжнародні відділи університетів.

ABSTRACT

Explanatory note to the final qualifying work of the bachelor: 84 pages, 47 figures, 2 tables, 2 appendixes, 17 sources.

SENTIMENT ANALYSIS, DOCUMENTS, INTERNATIONAL MOBILITY, PYTHON, WEB APPLICATION.

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

Subject of study are software for the emotion analysis of the documents of the international mobility applicants.

The purpose of the work is software development for the emotion analysis of the documents of the applicants of the international mobility.

Materials, methods, and tools: neural networks, deep learning, recurrent neural networks, convolutional neural networks, Django, Python, Keras.

Results. There was developed a web application which allows students to upload the application for the international mobility and define their emotional status.

Conclusions. Investigated the models for the emotions identification in the texts. The software with web interface were developed, which allows students who apply for the international mobility to upload their documents and for the staff from the international department to analyse its emotional status.

The field of use are universities international department.

ЗМІСТ

	С.
Перелік скорочень та умовних позначок.....	7
Вступ.....	8
1 Аналіз предметної області	9
1.1 Міжнародна мобільність студентів.....	9
1.2 Емоційний аналіз документів	10
1.3 Висновки та постановка завдання дослідження	12
2 Розробка архітектури програми.....	13
2.1 Аналіз вимог до системи.....	13
2.2 Обґрунтування вибору мови програмування	15
2.3 Обґрунтування вибору програмних засобів для обробки природної мови	20
3 Основні рішення щодо реалізації компонентів системи.....	22
3.1 Керівництво програміста.....	22
3.2 Керівництво користувача	27
4 Експериментальне дослідження методів класифікації текстів	31
4.1. Попередня обробка даних	31
4.2 Порівняльний аналіз методів для визначення емоцій.....	35
4.3 Тестування моделей довгої короткочасної пам'ятю.....	40
Висновки	50
Перелік джерел посилання.....	51
Додаток А Текст програми.....	53
Додаток Б Слайди презентації	76

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

- PHM – рекурентні нейронні мережі;
LSTM – Long short-term memory;
MTV – Model View Template;
NLTK – Natural Language Toolkit
AM – академічна мобільність;
ЄКТС – Європейська кредитно трансферна-накопичувальна система;
ЗНМ – згорткові нейронні мережі.

ВСТУП

Одним із основних вимірів сучасної освіти є глобалізація освітніх процесів. І однією з програм, що дозволяє університетам відповідати сучасним вимогам є програма Erasmus+, що дозволяє студентам всього світу проходити навчання у партнерських університетах протягом семестру.

При збільшенні кількості студентів, що приймають участь у академічній мобільності необхідна автоматизація процесів подання документів та їх аналізу. Одним із методів аналізу може бути емоційний аналіз документів.

Емоції є об'єктом аналізу для різних наукових напрямів – психології, нейропсихології, психіатрії, фізіології, педагогіки, лінгвістики.

У зв'язку зі збільшенням кількості електронних документів, що можуть бути оброблені для виявлення різноманітної суб'єктивної інформації все більшої актуальності набуває завдання аналізу природної мови.

Аналіз природної мови є підрозділом комп'ютерної лінгвістики. Автоматизованому аналізу тональності текстів приділяється все більше уваги, так як він дозволяє виділити емоційне відношення авторів тексту. Він використовується для автоматичного аналізу відгуків в мережі, для систем перекладу та автоматичного генерування текстів.

Останнім часом розроблено велику кількість алгоритмів емоційного аналізу текстів. Тому актуальним завданням є дослідження цих методів та використання їх для попереднього аналізу документів здобувачів міжнародної мобільності.

Завданнями роботи є:

- дослідження методів та засобів для визначення емоцій тексту;
- розроблення вебзастосунку для аналізу тональності документів здобувачів міжнародної мобільності;
- оформлення пояснювальної записки.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Міжнародна мобільність студентів

Академічна мобільність (АМ) – це можливість учасників освітнього процесу (студентів, викладачів та ін.) навчатися, викладати, стажуватися чи проводити наукову діяльність в іншому закладі вищої освіти або науковій установі на території України чи поза її межами (ст.1 Закону України «Про вищу освіту») [1].

АМ може бути внутрішньою, коли студент тимчасово переходить на навчання, стажування або здійснення наукової роботи в інший український ЗВО (наукову установу), та зовнішньою, коли обраний студентом університет чи наукова установа знаходяться за кордоном.

Окрім цього, академічна мобільність може бути ступеневою, якщо студент навчається у іншому закладі вищої освіти для того, щоб здобути освітній ступінь (наприклад, бакалавра або магістра) та диплом про вищу освіту, або кредитною, якщо в результаті навчання студент здобуває не освітній ступінь, а вивчає певні навчальні дисципліни (оцінені в кредитах ЄКТС) та/або здобуває компетентності чи інші результати (які не оцінюються в кредитах ЄКТС, але можуть бути визнані закладом вищої освіти постійного місця навчання).

В Національному Університеті Запорізька Політехніка студенти можуть прийняти участь в мобільності з 9 університетами партнерами з 5 країн Австрії, Бельгії, Німеччини, Румунії, Португалії. Також є можливість віртуальної мобільності з Холонським університетом (Ізраїль) [2].

Основною тенденцією є спільні магістерські програми, вікна мобільності та індивідуальні траєкторії навчання

При реєстрації на участь у програмі міжнародної мобільності студент подає наступні документи англійською мовою:

- рекомендаційний лист;
- мотиваційний лист;

- автобіографію.

Відповідно до національної стратегії понад 20% студентів вищого навчального закладу щорічно можуть приймати участь у міжнародної мобільності. Для середнього університету в 5000 студентів цу 100 студентів щорічно. Обробка такої кількості документів вимагає збільшення штату співробітників міжнародних відділів, або їх сверж навантаження. Тому необхідна попередня автоматизована обробка документів. Одним із засобів є емоційний аналіз рекомендаційного та мотиваційного листа.

1.2 Емоційний аналіз документів

Інтелектуальний аналіз тексту, це напрям інтелектуального аналізу даних та штучного інтелекту. Ключовими завданнями інтелектуального аналізу тексту є категоризація текстів, пошук інформації, обробка змін у колекціях текстів, а також розроблення засобів представлення інформації для користувача

Попередня обробка неструктурованого тексту складається з наступних етапів, що можуть змінюватися відповідно до конкретного завдання:

- сегментування тексту;
- токенізація. Це найперший крок при обробці тексту. Полягає в розбитті (поділі) довгих рядків тексту в більш дрібні: абзаци ділимо на пропозиції, пропозиції на слова;
- визначення частин речення;
- стемізація або стемінг. Це усунення придатків до кореня, тобто відділення суфікса, приставки, закінчення;
- лематизація. Це більш тонкий спосіб нормалізації текстових наданих, який використовує словник та морфологічний аналіз, що знайти правильну форму – лему;

- видалення стоп слів. Стоп-слова - слова, які не несуть ніякого смислового навантаження. Їх ще називають шумовими словами. Наприклад, в англійській мові - це артиклі, в українській - междомеття, сполучники і т.п.

- парсінг залежностей;

- визначення груп;

- векторизація. Даний процес називається вилученням ознак. Торба слів (Bag – of –Words Model) [3] – це метод вилучення ознак, який використовується при роботі з текстовими даними. В результаті отримуються вектори входження кожного слова в текст.

Виражена в тексті емоційна оцінка називається тональністю або сентиментом (від англ. Sentiment - почуття; думка, настрій) тексту.

Традиційний підхід до сентимент аналізу являє собою задачу класифікації тексту (частини тексту) на дві-три категорії (негативний, позитивний, нейтральний або просто: негативний або позитивний).

Існує два основні методи вирішення цього завдання автоматичного визначення тональності:

Статистичний метод. Для нього потрібні заздалегідь розмічені по тональності колекції (корпусу) текстів, на яких відбувається навчання моделі, за допомогою якої і відбувається визначення тональності тексту або фрази.

Метод, заснований на словниках і правилах. Для цього заздалегідь складаються словники позитивних і негативних слів і виразів. Цей метод може використовувати як списки шаблонів, так і правила з'єднання тональної лексики всередині пропозиції, засновані на граматичному і синтаксичному розборі.

Процес створення системи аналізу тональності дуже схожий на процес створення інших систем із застосуванням машинного навчання і містить наступні кроки.

На першому етапі необхідно зібрати колекцію документів для навчання класифікатора.

Кожен документ з навчальної колекції потрібно представити у вигляді вектора ознак.

Для кожного документа потрібно вказати «правильну відповідь», тобто тип тональності (наприклад, позитивна чи негативна), за цими відповідями і буде навчатися класифікатор.

Вибір алгоритму класифікації та навчання класифікатора.

Використання отриманої моделі.

1.3 Висновки та постановка завдання дослідження

В результаті потійного збільшення кількості студентів, що приймають участь у міжнародній мобільності зростає кількість електронних документів, що необхідно обробляти. В якості попереднього аналізу документів, що студент подає для участі в мобільності було обрано емоційний аналіз, або аналіз тональності тексту. Тому завданнями роботи є проаналізувати вимоги до системи, проаналізувати бібліотеки, що реалізують методи та алгоритми для аналізу природної мови, та релізувати вебзастосунок для визначення тональності документів.

2 РОЗРОБКА АРХІТЕКТУРИ ПРОГРАМИ

Ключовими елементами при розробленні архітектури програми є аналіз вимог користувачів та обґрунтування вибору програмних засобів для реалізації компонент системи.

2.1 Аналіз вимог до системи

Функціональні вимоги до системи:

- здобувач міжнародної мобільності може подати документи;
- здобувач міжнародної мобільності отримує реєстраційний номер;
- здобувач міжнародної мобільності завантажує документи до системи;
- співробітник проходить авторизацію в системі;
- співробітник може відновити пароль;
- співробітник може переглядати всі документи завантажені здобувачем;
- співробітник може проаналізувати обраний документ;
- співробітник може переглянути всіх здобувачів.

Прецедент 1. Завантаження документів здобувачем міжнародної мобільності.

Основний виконавець: Здобувач міжнародної мобільності.

Передумови: доступ до сервера.

Постумови: здобувач зареєстрован, отримав номер, документи завантажені.

Основний позитивний сценарій:

Здобувач обирає сторінку завантаження документів.

Здобувач вводить основні дані. Система перевіряє коректність та присвоює здобувачу унікальний номер.

Здобувач завантажує мотиваційний лист, рекомендаційний лист, додатковий файл. Та підтверджує кінець введення.

Данні завантажуються на сервер.

Альтернативний сценарій 1.

Якщо e-mail введено некоректно. Виводить повідомлення та повторне завантаження форми.

Альтернативний сценарій 2.

Якщо завантажено більше ніж 3 файли система виводить повідомлення та очікує зміну кількості завантажених файлів.

Діаграма прецедентів для користувача міжнародної мобільності відображена на рис. 2.1.

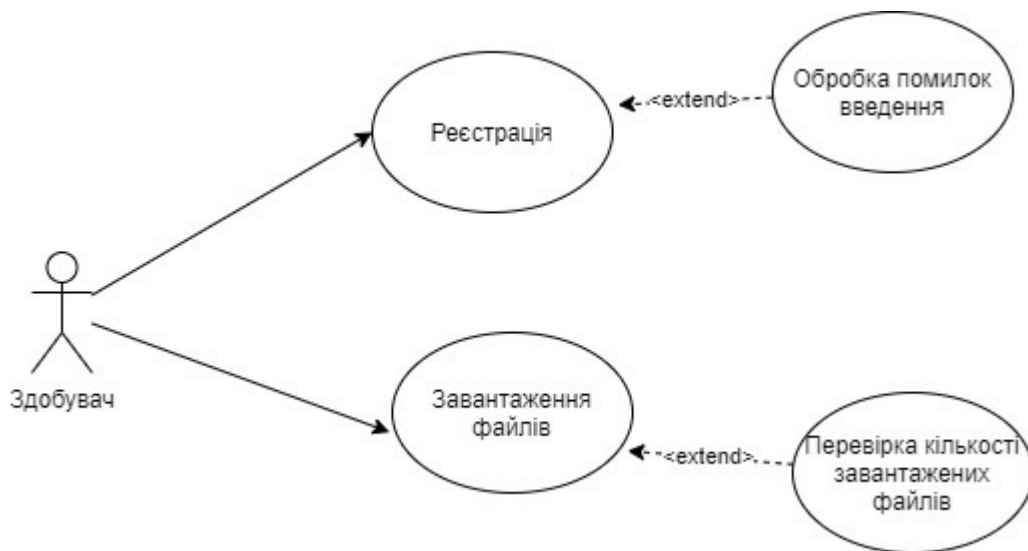


Рисунок 2.1 – Прецедент завантаження документів здобувачем міжнародної мобільності

Прецент 2. Робота співробітника міжнародного відділу з системою роботи співробітника міжнародного відділу з системою.

Основний виконавець. Співробітник міжнародного відділу.

Передумови. Співробітник зареєстрований в системі.

Постумови: Файли проаналізовані і результати аналізу збережені в системі.

Основний позитивний сценарій.

Співробітник авторизується в системі. Система виводить перелік зареєстрованих здобувачів претендентів на міжнародну мобільність.

Співробітнику обирає здобувача. Система виводить перелік документів завантажених здобувачем.

Співробітник обирає документ. Запускається аналізатор тексту. Система виводить результат аналізу. Результат зберігається у системі.

Діаграма прецедентів для співробітника міжнародного відділу відображена на рис. 2.2.

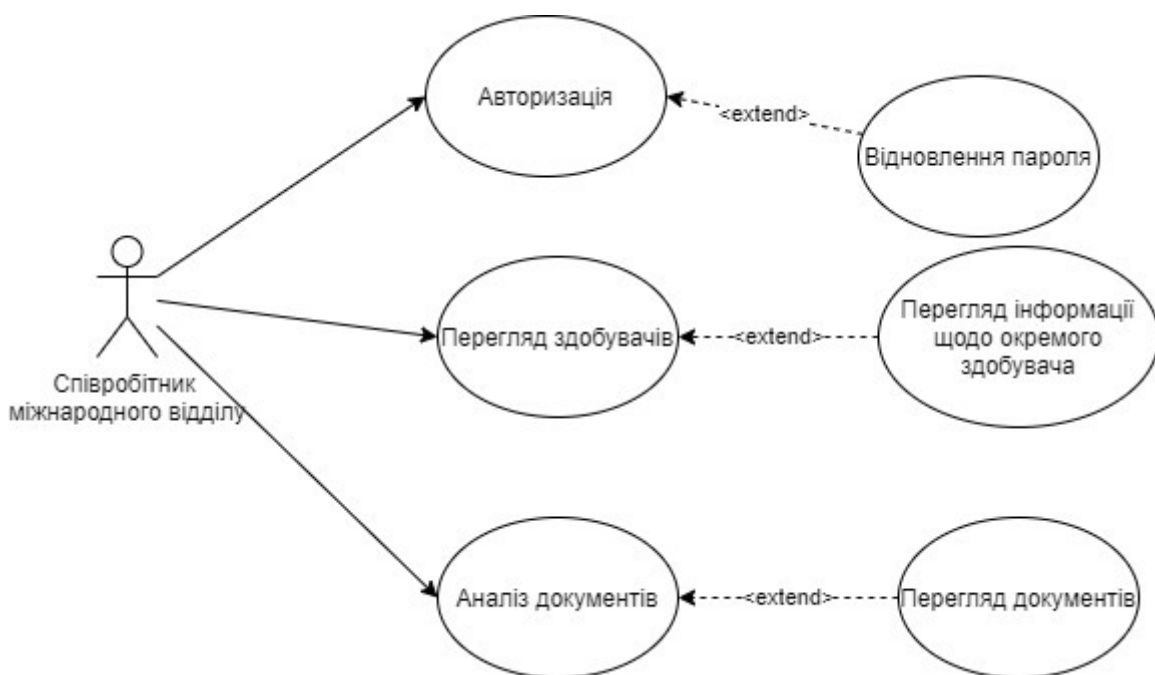


Рисунок 2.2. Прецедент роботи співробітника міжнародного відділу з системою

2.2 Обґрунтування вибору мови програмування

Python - високорівнева мова програмування. Синтаксис ядра Python мінімалістичний. Стандартна бібліотека містить значний обсяг корисних функцій. Python підтримує декілька парадигм програмування, в тому числі структурне, функціональне, об'єктно-орієнтоване, імперативне і аспектно-орієнтоване. До основних властивостей мови належить динамічна типізація,

автоматичне звільнення пам'яті, зручні стандартні типи даних, обробка виключень. Код в Python зазвичай збирається у функції та класи, які потім можна об'єднати в модулі, які в свою чергу вже при бажанні об'єднуються в пакети.

2.2.1 Бібліотеки Python

Самий фундаментальний пакет – NumPy [4]. Він дозволяє виконувати основні операції над n-масивами і матрицями: додавання віднімання, ділення, множення, транспонування, обчислення визначника і т. Д. Завдяки механізму векторизації, NumPy підвищує продуктивність і, відповідно, прискорює виконання операцій.

SciPy [5] - бібліотека в складі Python SciPy Stack, що не заплутатися. Вона містить модулі для лінійної алгебри, оптимізації, інтеграції та статистики. SciPy працює спільно з NumPy, що дозволяє їй значно розширити функціональність. Два аргументи, чому ця бібліотека дійсно хороша:

Pandas - це пакет, призначений для простий і інтуїтивно зрозумілою роботи з «поміченими» і «реляційними» даними. Теж працює в зв'язці з NumPy, і крім математичних обчислень забезпечує їх агрегацію і візуалізацію.

Matplotlib є основним інструментом для візуалізації даних на мові Python, підтримується різними платформами і IDE (iPython, Jupyter та ін.)

Саме можливості Matplotlib дозволяють розглядати Python, як повноправного конкурента MATLAB або Mathematica. З її допомогою можна створювати: лінійні графіки; графіки розсіювання; гістограми; кругові діаграми; стовбурові ділянки; контурні ділянки; спектрограми.

Бібліотека низкоуровневая, що означає великий обсяг коду для розширеної візуалізації. Але продуктивність і робота зі звичним мовою дозволяють закрити очі на цей недолік.

Seaborn базується на Matplotlib, але оптимізований для візуалізації статистичних моделей: теплові карти, розподілу, результати математичних операцій. Незважаючи на зазначені можливості, більшість розробників використовує бібліотеку для відображення простих тимчасових розподілів.

На відміну від Seaborn, Bokeh не залежить від Matplotlib. Основна область його застосування - веб, де візуалізація даних відбувається відповідно до стилю d3.js (Data-Driven Documents). Візуально графіки, побудовані за допомогою Bokeh, виглядають привабливіше, ніж в попередніх двох випадках.

TensorFlow [6] - найпопулярніший інструмент для машинного навчання нейромереж.

Принципи роботи з tensorflow досить прості. Ми повинні скласти граф операцій, потім передати в цей граф дані і дати команду зробити обчислення.

У нього є два елементи:

- набір `tf.Operation`, який представляє одиниці обчислень;
- набір `tf.Tensor`, який представляє одиниці даних.

2.2.2 Фреймворки для розроблення вебзастосувань

Найбільш популярними фреймворками для розроблення вебзастосувань на мові Python є Django, Pyramid, Flask.

У таблиці 2.1 наведено порівняння фреймворків [7].

Django [8] є надзвичайно популярним і повнофункціональним серверним вебфреймворком, написаним на Python.

Flask [9] - легкий і гнучкий фреймворк. Він дозволяє вибирати модулі для вирішення конкретних завдань і встановлювати їх у міру необхідності.

Pyramid [10] - наймолодший фреймворк для синхронного Веба серед популярних Python фреймворків. Дизайн Pyramid заснований на наступних принципах: простота, мінімалізм, документованість, швидкість, надійність, відкритість.

В своє дослідження за швидкістю фреймворків автори додали ще aiohttp, Sanic, Tornado. Але видно, що Django впевнено утримує лідуєчий результат серед фул-стек фреймворків.

Sanic [11] - це вебсервер та фреймворк Python 3.7+, створений для швидкого використання. Це дозволяє використовувати синтаксис `async / await`, доданий в Python 3.5, що робить ваш код неблокуючим і швидким. Sanic також відповідає ASGI, тому ви можете розгорнути його за допомогою альтернативного вебсервера ASGI.

Tornado [12] - це вебфреймворк Python та асинхронна мережева бібліотека, спочатку розроблена на FriendFeed. Використовуючи неблокуючий мережевий ввід-вивід, Tornado може масштабуватися до десятків тисяч відкритих з'єднань, що робить його ідеальним для тривалого опитування, WebSockets та інших додатків, які вимагають довгого з'єднання з кожним користувачем.

Таблиця 2.1 – Порівняння фреймворків [7]

Фреймворк	Середнє		Максимальне		Запит, с.	Пере- дача sec (KB)
	затримка (мс)	зап./ сек	затрим- ка (мс)	зап./ сек		
Django	147,99	120,91	1,98	640	425,71	78,99
Flask	186,8	65,65	1,75	280	546,2	127,48
aiohttp	165,07	247,16	1,98	585	1966,28	439,72
Sanic	194,8	170,85	1,98	470	1520,9	292,6
Tornado	279,3	101,71	1,74	272	778,75	222,07

Django - безкоштовний і відкритий фреймворк для створення вебдодатків, написаний мовою програмування Python.

Проект Django відповідає концепції MTV в якій:

- М означає «Model» - модель, рівень доступу до даних. Так само як все, що пов'язано з даними - як отримати до них доступ, як перевірити їх, яке у них поведінка і ставлення один з одним;
- Т означає «Template» - шаблони. Цей рівень містить в собі все, що пов'язано з відображенням: як саме щось повинно бути відображено на вебсторінці або будь-якому іншому документі;
- V означає «View» - уявлення. Цей рівень пов'язаний з усією логікою, що зв'язує модель і необхідні шаблони. Ви можете собі уявляти це рівень як міст між моделями і шаблонами.

Вебзастосування Django отримують доступ і управляють даними через об'єкти Python, що називаються моделями. Моделі визначають структуру даних, що зберігаються, замовчуванням, параметри списку вибору, текст довідки для документації, текст міток для форм і т.п.

Взаємодія елементів в проєкті Django наведено на рис. 2.3 [8].

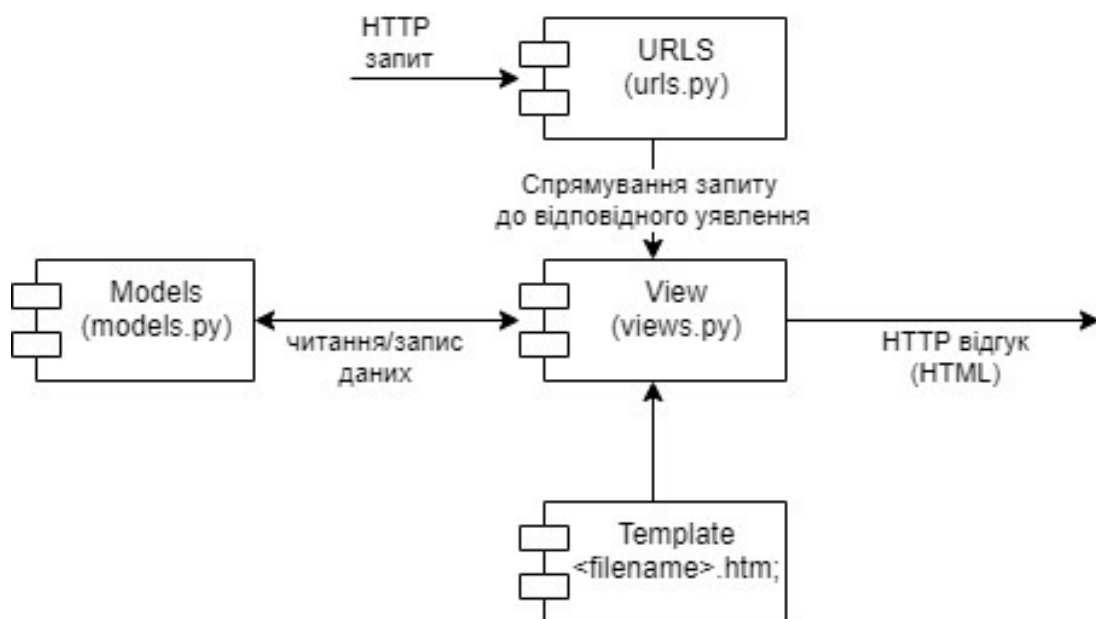


Рисунок 2.3 – Взаємодія файлів в проєкті Django

2.3 Обґрунтування вибору програмних засобів для обробки природної мови

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 токенізації за реченнями, токенізацію за словами, лематизацію і стемінг тексту, створення стоп-слів, регулярних виразів, мішок слів, TF-IDF.

NLTK доступний для Windows, Mac OS X та Linux. NLTK - це безкоштовний проєкт із відкритим кодом, який керується спільнотою.

Іншою популярною бібліотекою є spaCy. Між NLTK та spaCy є суттєві відмінності, які полягають у наступному.

NLTK надає безліч алгоритмів для вибору для конкретної проблеми, яка є корисною для дослідника, але неприємною для розробника. У той час як spaCy зберігає найкращий алгоритм проблеми у своєму наборі інструментів та постійно оновлює його у міру вдосконалення.

NLTK підтримує різні мови, тоді як spaCy має статистичні моделі для 7 мов (англійської, німецької, іспанської, французької, португальської, італійської та голландської). spaCy також підтримує іменовані сутності для багатомовності.

NLTK - це бібліотека обробки рядків. Вона приймає рядки як вхідні дані і повертає рядки або списки рядків як вихідні. Тоді як spaCy використовує об'єктно-орієнтований підхід. Коли ми аналізуємо текст, spaCy повертає об'єкт документа, слова та речення якого є самими об'єктами.

spaCy має підтримку векторів слів, тоді як NLTK - ні.

В таблиці 2.2 наведено порівняльну характеристику бібліотек python для обробки природної мови.

Таблиця 2.2 – Бібліотеки обробки природної мови [13]

Бібліотека	Мова	Позитивні характеристики	Негативні характеристики
NLTK (Natural Language ToolKit)	Python	Найбільш відома і багатофункціональна бібліотека для NLP. Велика кількість сторонніх розширень. Швидка токенизація. Підтримується безліч мов.	Повільна. Працює з рядками. Відсутні вбудованих векторів слів.
sprasy	Cython	Найбільш швидка Працює з об'єктами Є вбудовані вектори слів Використовує нейронні мережі для навчання моделей	Менш гнучка ніж NLTK Токенизація повільніша ніж у NLTK Підтримує невелику кількість мов
scikit-learn	Python	Містить велику кількість алгоритмів для побудови моделей. Містить функції для роботи з Bag-of-Words. Гарна документація.	Поганий препроцесінг, що змушує використовувати її в зв'язці з іншою бібліотекою.
gensim	Python	Працює з великими наборами даних; Підтримує глибоке навчання; word2vec, tf-idf vectorization, document2vec.	Реалізовані тільки моделі без вчителя. Не містить достатнього функціоналу, необхідного для NLP, що змушує використовувати її разом з іншими бібліотеками.

3 ОСНОВНІ РІШЕННЯ ЩОДО РЕАЛІЗАЦІЇ КОМПОНЕНТІВ СИСТЕМИ

3.1 Керівництво програміста

Розроблене застосування призначено для попереднього аналізу документів, що подані здобувачем міжнародної мобільності. Система надає можливість студенту завантажувати документи для участі в міжнародній мобільності. Для представників міжнародних відділів система надає інтерфейс перегляду завантажених документів та аналізу тональності завантажених документів.

Загалом, ця програма багатокористувацька та багатогрупова. Користувачі, які не зареєстровані, можуть заповнити форму (повне ім'я, електронна адреса, примітка (необов'язково), assign_to (необов'язково)), після чого сформується унікальне посилання, і користувач зможе завантажити свої документи (максимум 3, це змінюється в налаштуваннях) і подивіться на коментарі викладача та результати аналізу його документів.

Перед запуском потрібно створити принаймні одну групу (налаштовану адміністратором Django), і ця група повинна мати принаймні одного користувача (адміністратора) як члена. У проєкті була створена університетська група, а в цій групі - список мотиваційних листів. Посилання на цей список було використано на головній сторінці, не забудьте змінити його при початковій конфігурації.

Щоб створити групу, перейдіть на вкладку адміністратора, виберіть вкладку «Групи», а потім натисніть «Створити». Щоб додати користувача до групи, перейдіть на вкладку «Користувачі», перейдіть до свого користувача та призначте йому групу.

У цій програмі лише адміністратори (або користувачі персоналу) можуть додавати або видаляти списки. Завдання в списках може створювати кожен, якщо незареєстрованому користувачеві надається посилання на

список від адміністратора. Завдання можуть бути призначені конкретному користувачеві. Непризначені завдання відображаються як належні комусь.

Адміністратори можуть модифікувати отримані сторінки, проводити аналіз настроїв документів (можна лише завантажувати PDF - чим вищий прогнозований відсоток результату, тим позитивніший текст), писати коментарі, створювати електронні листи тощо.

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.

На рис. 3.1 відображена діаграма компонент для розробленої системи.

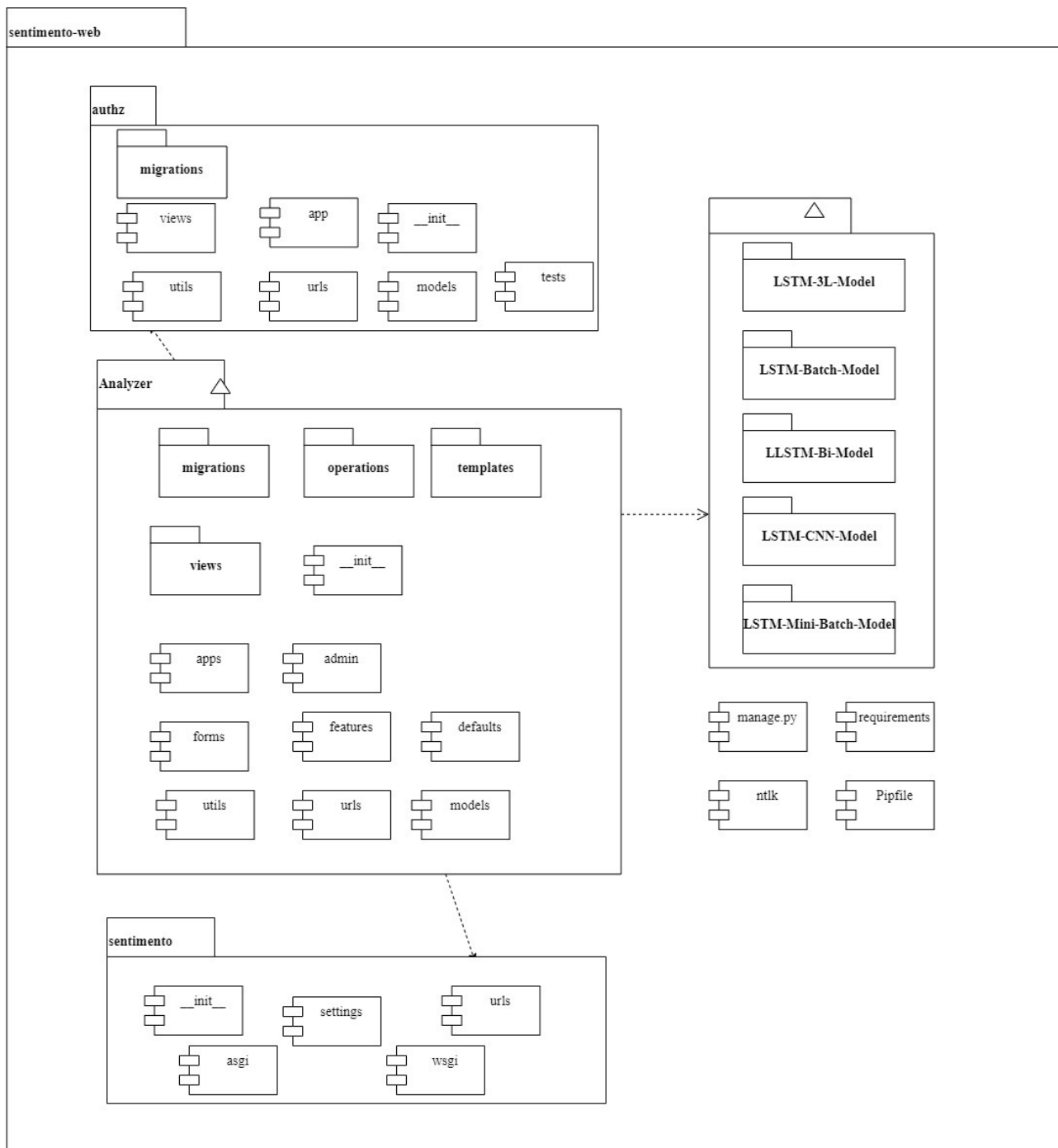


Рисунок 3.1 – Діаграма компонент системи

Пакет analyzer містить класи застосування.

Файл models.py:

- class LockedAtomicTransaction(Atomic) – здійснює атомну транзакцію, але також блокує всю таблицю для будь-яких транзакцій на час транзакції, що виконується;
- class TaskList(models.Model) – перелік завдань;

- `class Task(models.Model)` – відповідає за опис завдання;
- `class Comment(models.Model)` – створює коментарії не вбудованими засобами django;
- `class Attachment(models.Model)` – відповідає за загальне завантаження файлу до завдання.

Файл `forms.py` містить класи:

- `class AddTaskListForm(ModelForm)` – створює перелів вибору, що відображає допустимі групи, до яких можна додати новий список, визначає, до яких груп належить користувач;
- `class AddEditTaskForm(ModelForm)` - потрібний для безпечного злиття. Перелік вибору, що відображає користувачів, яким може бути призначено нове завдання;
- `class SearchForm(forms.Form)` – форма для пошуку.

Файл `admin.py` містить наступні класи, що описують модель даних:

- `class TaskAdmin(admin.ModelAdmin);`
- `class CommentAdmin(admin.ModelAdmin);`
- `class AttachmentAdmin(admin.ModelAdmin).`

Для уявлень були розроблені наступні html шаблони:

- `add_list.html;`
- `base.html;`
- `del_list.html;`
- `import_csv.html;`
- `list_detail.html;`
- `list_lists.html;`
- `predict.html;`
- `search_result.html;`
- `task_detail.html.`

Пакет `authz` містить реалізацію авторизації. Реалізовані наступні класи для уявлення, що знаходяться в файлі `views.py`:

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

Моделі для емоційного аналізу містяться в пакеті models:

- LSTM-3L-Model.h5;
- LSTM-Batch-Model.h5;
- LSTM-Bi-Model.h5;
- LSTM-CNN-Model.h5;
- LSTM-Mini-Batch-Model.h5.

Для запуску проєкта необхідно виконати наступні кроки.

Крок 1. клонувати репозиторій з github.

Крок 2. скопіювати вимоги до бібліотек та активувати віртуальне середовище:

- cd to the directory where requirements.txt is located;
- activate your virtualenv;
- run: pip install -r requirements.txt in your shell.

Крок 3. виконати наступні консольні команди:

- python manage.py makemigrations;
- python manage.py migrate;
- python manage.py createsuperuser;
- python manage.py runserver.

Крок 4. якщо сервер стартував успішно перейти до використання програми.

3.2 Керівництво користувача

Розроблена система призначена для використання двома типами користувачів – студентам, здобувачам міжнародної мобільності та співробітникам міжнародного відділу.

Незареєстрований користувач попадає на головну сторінки програми рис. 3.2 та перейти на заповнення форми.

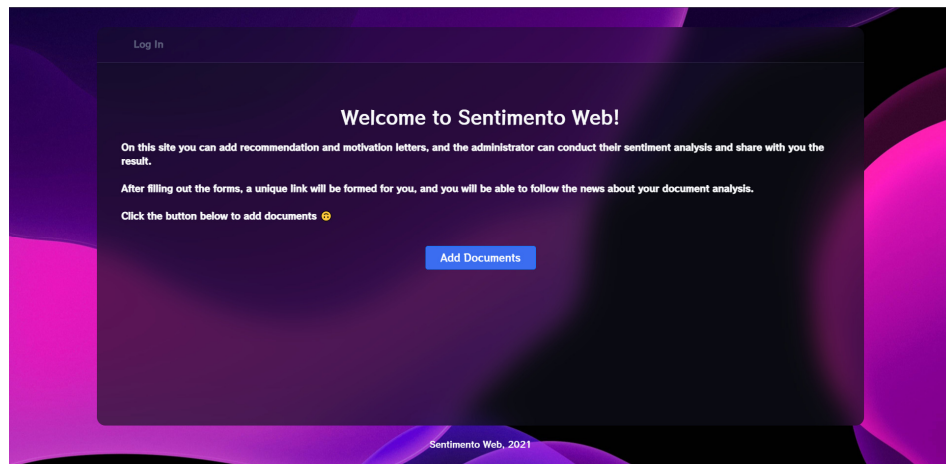


Рисунок 3.2 – Головна сторінка програми

Студент може заповнити форму (рис. 3.3) де він може ввести повне ім'я, електронну адресу, примітки (необов'язково), assign_to (необов'язково), після чого сформується унікальне посилання.

Рисунок 3.3 – Реєстрація учасника міжнародної мобільності

Після чого студент зможе завантажити свої документи (рис. 3.4) (максимум 3, це змінюється в налаштуваннях) і подивіться на коментарі викладача та результати аналізу його документів.

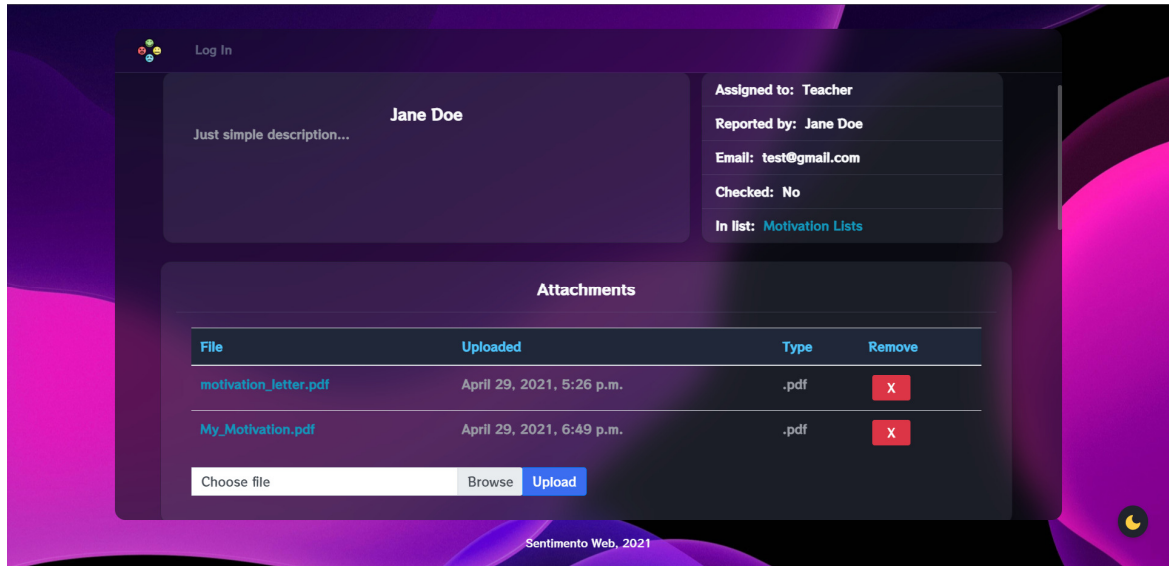


Рисунок 3.4 – Приклад завантажених документів

Співробітник повинний поперше пройти авторизацію (рис. 3.5).

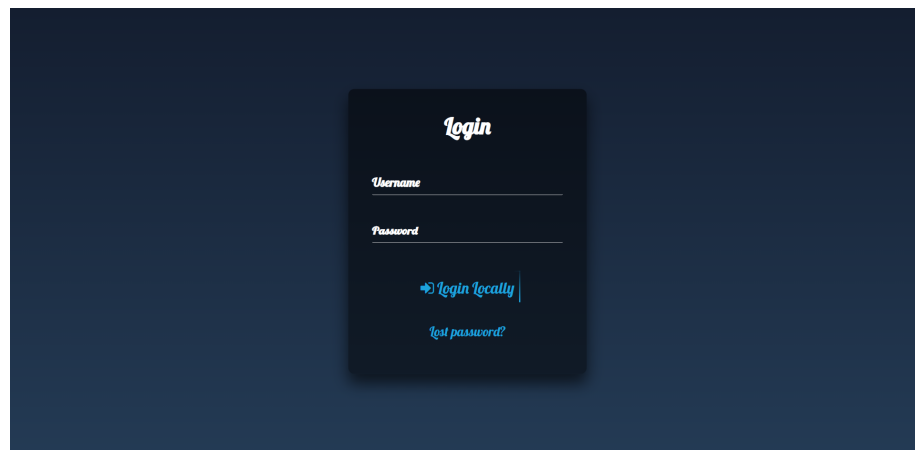


Рисунок 3.5 – Аутентифікація співробітника

Після чого перед користувачем з'являється перелік завантажених заяв (рис. 3.6).

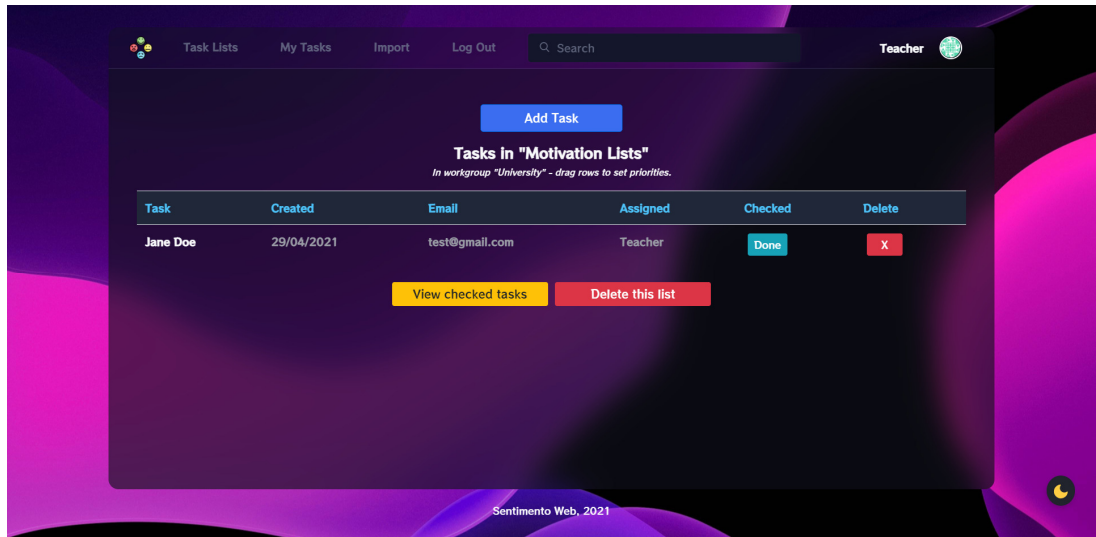


Рисунок 3.6 – Перелік зареєстрованих учасників мобільності

Користувач може передивитись інформацію для кожного здобувача окремо (рис. 3.7). Виконати пошук відповідного здобувача. Обрати документ для аналізу.

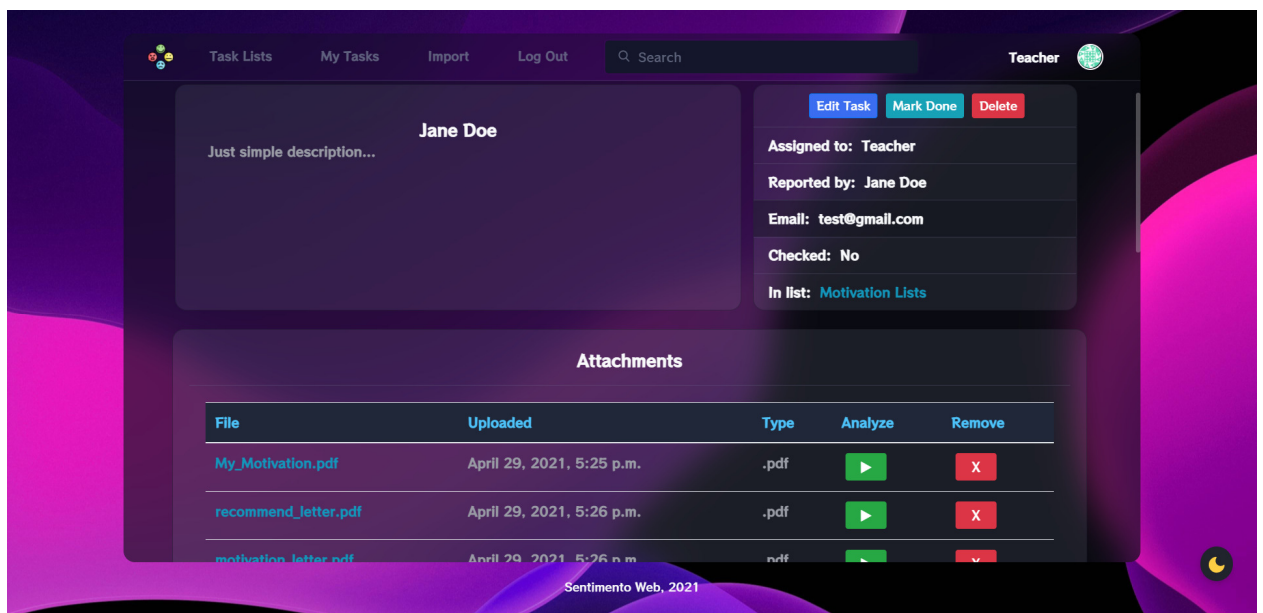


Рисунок 3.7 Перелік документів завантажених студентом

Приклад результатів аналізу наведено на рис. 3.8.

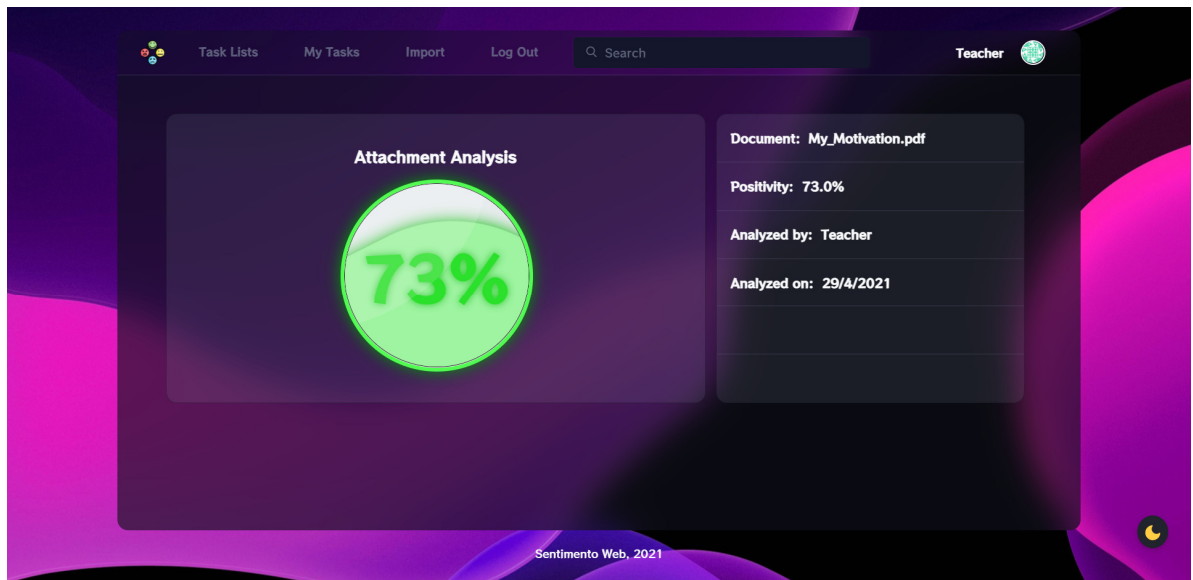


Рисунок 3.8 – Приклад результатів аналізу

4 ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ МЕТОДІВ КЛАСИФІКАЦІЇ ТЕКСТІВ

4.1 Попередня обробка даних

В якості бази даних позитивних та негативних відгуків було обрано базу Амазон [17].

Оскільки нам потрібні лише файли з позитивними та негативними відгуками, то інші були видалені. А для наших негативних та позитивних відгуків ми додали батьківський тег, в якому будуть всі відгуки, тому буде простіше зробити дерево xml.

В результаті у нашому каталозі з тестовими навчальними наборами ми отримали два файли - з позитивними та негативними відгуками, кожен із цих файлів містить 1000 необроблених відгуків.

Наступний крок - проаналізувати наш XML-файл на базовому рівні. Було створено словник зі списків. Кожен список містить певні теги та їх вміст для окремого перегляду. Так як у нас вже є дерево із позитивними та негативними відгуками, тому залишається лише проаналізувати його.

Наступним кроком ми проаналізували позитивні та негативні відгуки для 25 різних категорій. Наступним кроком ми перетворили наші словники оглядів у кожній категорії на Pandas DataFrame.

Тепер ми перетворимо наші словники оглядів у кожній категорії на Pandas DataFrame:

```
positive_df, negative_df = pd.DataFrame(columns=REVIEW_TAGS),  
pd.DataFrame(columns=REVIEW_TAGS)
```

Ми також додали новий стовпець Class, який матиме значення pos або nega, необхідно розрізняти позитивні та негативні відгуки після конкатенації:

```
positive_df['Class'] = "pos" # positive reviews  
negative_df['Class'] = "neg" # negative reviews  
reviews = pd.concat([positive_df, negative_df])
```

Тепер наша робота з формуванням наборів даних завершена.

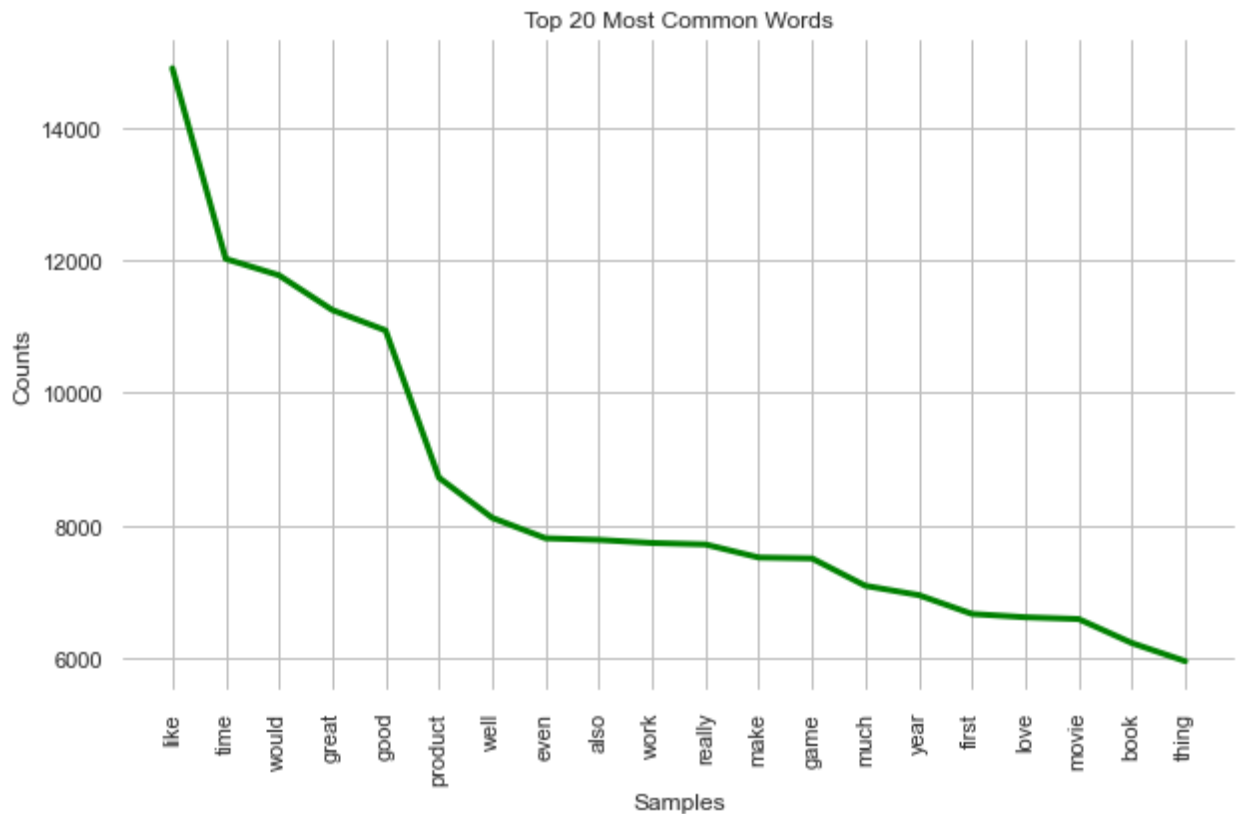


Рисунок 4.4 – 20 найбільш використаних слів

Наступним кроком ми створимо наші навчальні документи у формі кортежу. Тому що кортежи є незмінними, тобто вони мають фіксований розмір за своєю суттю, тоді як списки динамічні, їх можна змінювати. Тепер нам не потрібно додавати або видаляти елементи, а кортежі швидші за списки.

У наступній частині нам також потрібно працювати з нашими даними, які ми обробили тут, тож давайте їх обробим з використанням модулю `pickle`.

`pickle` - модуль, який реалізує потужний алгоритм серіалізації та десеріалізації об'єктів Python. «Протравлення» - це процес перетворення об'єкта Python у потік байтів, а «вилучення» - операція зворотного дії, яка перетворює потік байтів назад у об'єкт Python. Оскільки потік байтів можна легко записати у файл, модуль `pickle` широко використовується для зберігання та завантаження складних об'єктів у Python.

Далі ми створили метод `find_matches`, який повертає словник зі словами як пара «ключ» та «наявність» у нашому словнику найпоширеніших слів

(all_words_freq) як значення True або False, додатковий параметр буде представляти ряд найбільш часто повторюваних слів, які нам потрібно використовувати (за замовчуванням - 50).

Тепер нам потрібно розділити набір даних для того, щоб виконати неупереджену оцінку моделі та виявити наявність оверфітінгу.

Для цього ми будемо використали модуль scikit-learn або sklearn.

Використання наївного баєсівського класифікатора показало точність 81.51% (рис. 4.5). Ми будемо приймати точність достатньою якщо значення знаходиться в діапазоні [80%,100%].

```
Most Informative Features
      refund = True          neg : pos    =    23.1 : 1.0
    refreshing = True       pos : neg    =    21.9 : 1.0
    delicious = True       pos : neg    =    15.7 : 1.0
      garbage = True       neg : pos    =    11.8 : 1.0
      worst = True         neg : pos    =    11.7 : 1.0
      waste = True         neg : pos    =    11.1 : 1.0
    contacted = True       neg : pos    =    10.8 : 1.0
    worthless = True       neg : pos    =    10.2 : 1.0
      cheaply = True       neg : pos    =    10.1 : 1.0
      poorly = True       neg : pos    =     9.7 : 1.0
None
Classifier Accuracy: 81.51%
```

Рисунок 4.5 – Використання наївного баєсівського класифікатора

Щоб правильно протестувати наше вхідне речення, ми протестували методи для всіх етапів обробки, що дозволяє токенизувати, лематизувати та зупинити_слова, усунути нерелевантні_слова.

4.2 Порівняльний аналіз методів для визначення емоцій

На цьому етапі ми будемо використовувати модель word2vec. Алгоритми, що містяться в цієї моделі створюють векторні уявлення слів. Вектори слів лежать в основі багатьох систем обробки природної мови (NLP), які захлеснули сучасний світ (Amazon Alexa, Google translate і т.п.).

Слова-вектори (word vectors) - це чисельні уявлення слів, що зберігають семантичну зв'язок між ними.

Щоб навчити вибірку слів без заздалегідь розмічених даних, спочатку нам потрібно вирішити кілька завдань:

- створити кортежі даних в форматі [вхідний слово, вихідне слово], кожне слово представлено у вигляді довічного вектора довжини n , де i -е значення кодується одиницею на i -ій позиції і нулями на всіх інших (one-hot кодування);
- створити модель, яка на вхід і вихід отримує one-hot вектори;
- визначити функцію втрат, пророкує вірне слово, щоб оптимізувати модель;
- визначити якість моделі, переконавшись, що схожі слова мають схожі векторні уявлення.

Першим етапом беремо середнє значення всіх векторів слів із речення, щоб отримати один 50-мірний вектор, який представляє тон усього речення, яким ми подаємо модель, і намагаємось отримати швидкий результат.

Результати перевірки нам дали, що 30709 слів з 1802696 не були знайдені в словнику.

Для вимірювання подібності 2 слів, ми вимірювали ступінь подібності (cosine_similarity) між двома векторами вбудовування для цих 2 слів.

Ця подібність залежить від кута між u та v . Якщо u і v дуже схожі, їх косинусова подібність буде близькою до 1; якщо вони несхожі, подібність косинусів прийматиме менше значення.

Перевірка позитивних речень дала результат:

$$\text{cosine_similarity} = 0.9481159093219256.$$

Перевірка негативних речень дала результат:

$$\text{cosine_similarity} = 0.8968743967161681.$$

Наступним кроком ми порівнюємо результати отримані наступними методами: логістичної регресією (рис. 4.6), метод k-найближчих сусідів (рис. 4.7), випадковий ліс (рис. 4.8), метод опорних векторів (рис.4.9).

З метою спрощення інтерпретації та виявлення проблем, звіт інтегрує числові оцінки з кольоровою тепловою картою. Всі теплові карти знаходяться в діапазоні (0,0, 1,0), щоб полегшити легке порівняння моделей класифікації в різних звітах про класифікацію.

Звіт про класифікацію показує представлення основних показників класифікації на основі кожного класу. Візуальні класифікаційні звіти використовуються для порівняння моделей класифікації для вибору моделей, які є «червонішими», наприклад мають сильніші показники класифікації або більш збалансовані.

Метрики визначаються з точки зору істинних та помилкових позитивних даних, а також істинних та помилкових негативних результатів. Позитивні та негативні в даному випадку є загальними назвами класів двійкової класифікаційної задачі. Показники визначаються наступним чином:

- точність (precision) - можна розглядати як міру точності класифікатора. Для кожного класу воно визначається як відношення справжніх позитивів до суми справжніх і помилкових позитивів;

- відкликання(recall) - є мірою повноти класифікатора; здатність класифікатора правильно знаходити всі позитивні випадки. Для кожного класу воно визначається як відношення справжніх позитивів до суми справжніх позитивних і помилково негативних;

- оцінка F1 є зваженим гармонічним середнім значенням точності та відкликання, така що найкраща оцінка становить 1,0, а найгірша - 0,0. Взагалі кажучи, показники F1 нижчі за показники точності, оскільки вони вбудовують точність і запам'ятовують у свої обчислення. Як правило, середнє зважене значення F1 слід використовувати для порівняння моделей класифікаторів, а не глобальної точності.

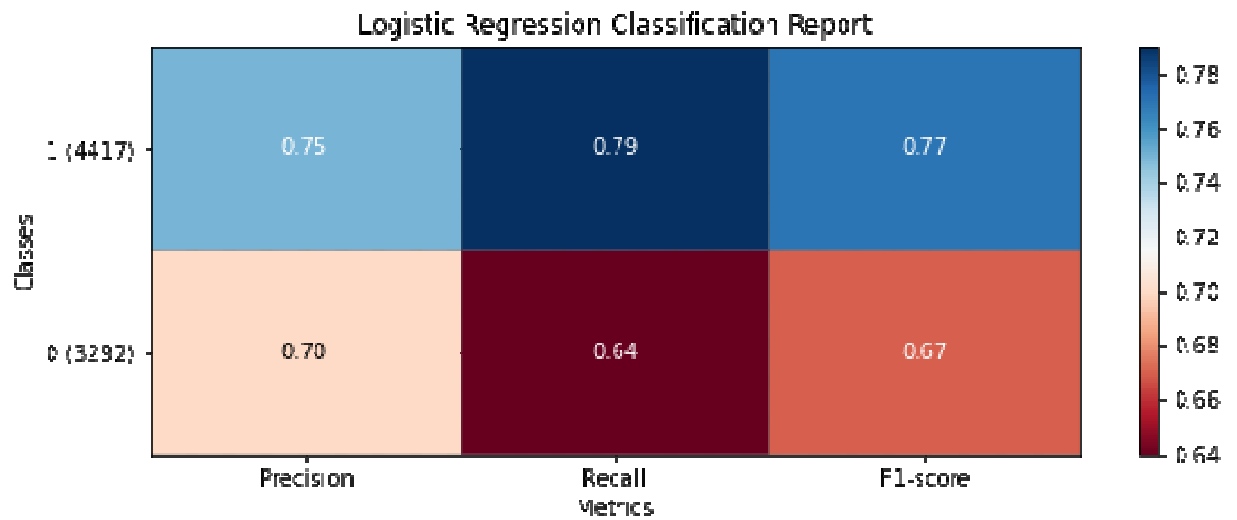


Рисунок 4.6 – Логістична регресія

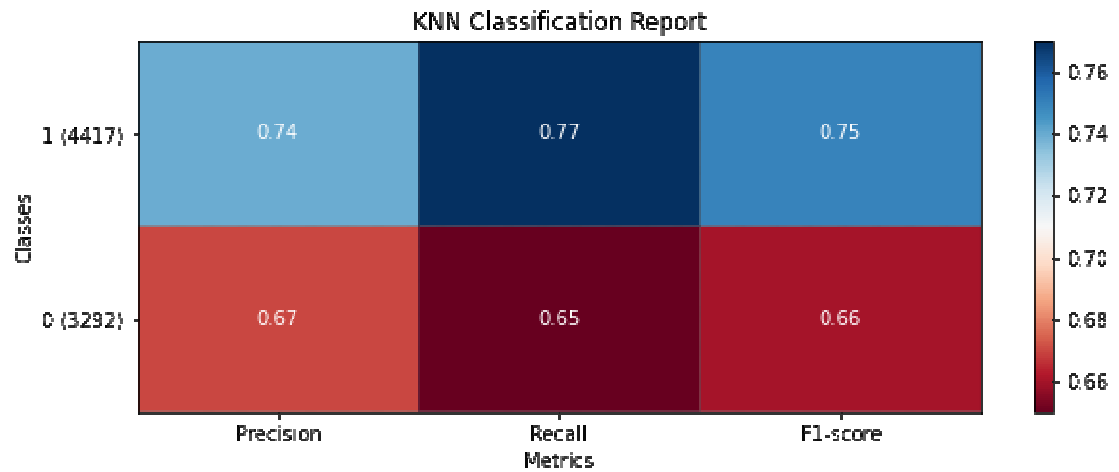


Рисунок 4.7 – Метод k-найближчих сусідів

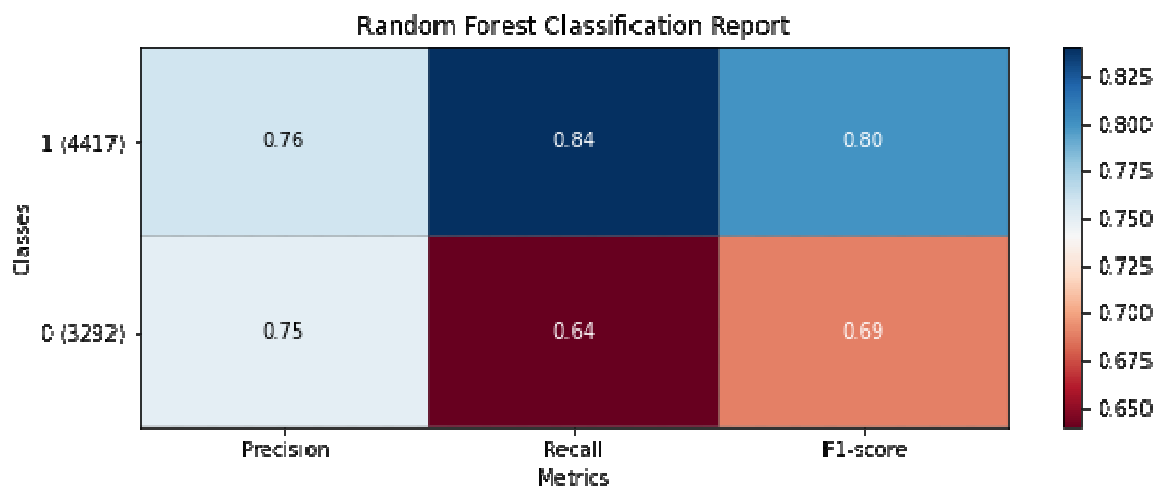


Рисунок 4.8 – Випадковий ліс

На рисунку 4.9 наведено метод опорних векторів.

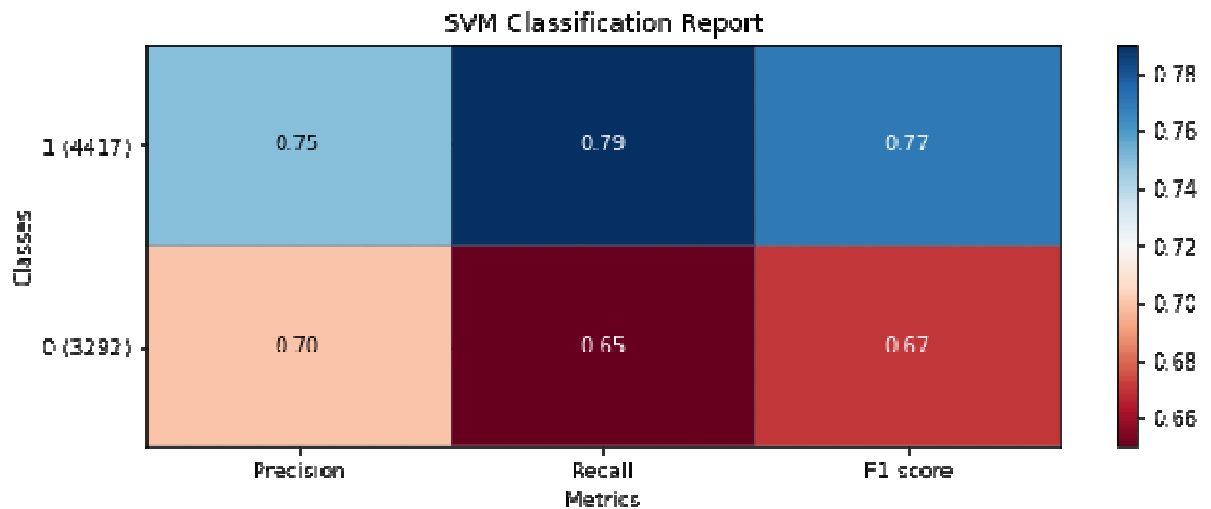


Рисунок 4.9 – Метод опорних векторів

Для порівняння методів використаємо візуальний графік результатів класифікації та ROC-криву (рис. 4.10).

Справжній позитивний коефіцієнт обчислюється як кількість дійсних позитивів, поділена на суму кількості справжніх позитивів та кількості помилково негативних. Він описує, наскільки добре модель прогнозує позитивний клас, коли фактичний результат позитивний.

Показник хибнопозитивних результатів обчислюється як кількість помилкових спрацьовувань, поділене на суму кількості хибнопозитивних результатів та кількості справжніх негативів.

Його також називають швидкістю помилкової тривоги, оскільки вона підсумовує, як часто прогнозується позитивний клас, коли фактичний результат є негативним.

Менші значення на осі x графіку вказують на нижчі помилкові спрацьовування та вищі справжні негативні.

Більші значення на осі y графіку вказують на вищі істинні позитиви і нижчі помилкові негативні.

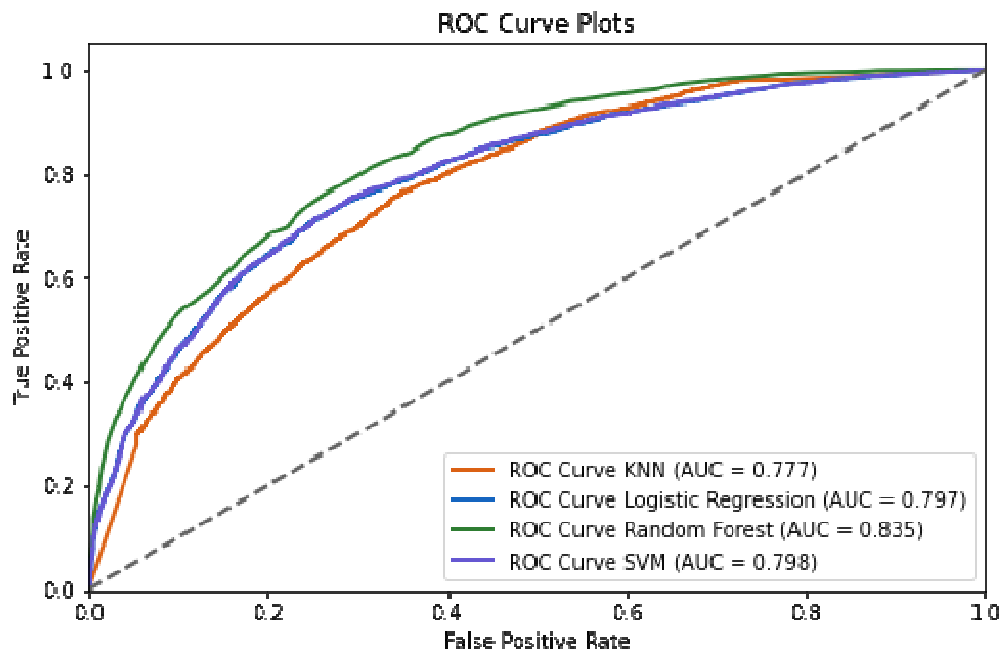


Рисунок 4.10 – Порівняння класифікаторів

Відповідно до отриманого результату використання випадкового лісу надає найбільшу поверхню під ROC та точність 75%. Тому зберігаємо модель для подальшої роботи.

4.3 Тестування моделей довгою короткочасною пам'яттю

Перший етап ми завантажуємо отримані на попередніх етапах словник та отриману модель. Модель містить 400 000 слів, тому ми отримаємо матрицю вбудовування 400 000 x 50, яка містить усі значення векторів слів.

Візуалізація результатів аналізу кількості слів в словнику наведено на рис. 4.11.

Виходячи з даних гістограми, а також середньої кількості слів у файлах, ми можемо впевнено сказати, що більшість оглядів матимуть менше 100 слів, що є максимальним значенням довжини послідовності, яку ми встановили.

Перевага Кераса полягає в тому, що він побудований на базі символічних математичних бібліотек, таких як TensorFlow та Theano, для

швидких та ефективних обчислень. Це потрібно для великих нейронних мереж.

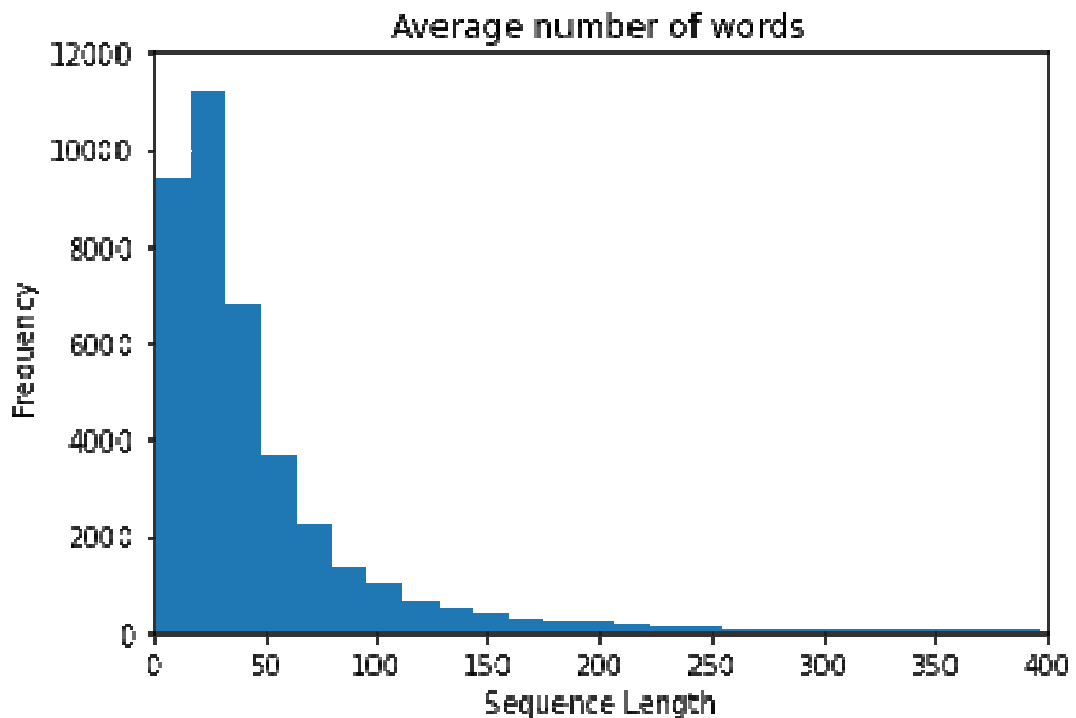


Рисунок 4.11 – Частотний аналіз слів

Недоліком використання цих ефективних бібліотек є те, що ви повинні визначити обсяг своїх даних заздалегідь і на весь час. Зокрема, розмір партії.

Розмір партії обмежує кількість зразків, які слід показати мережі перед тим, як можна виконати оновлення ваги. Потім це саме обмеження накладається при прогнозуванні за допомогою моделі підгонки.

Зокрема, розмір партії, який використовується під час встановлення вашої моделі, визначає, скільки прогнозів ви повинні робити за раз.

Це часто не є проблемою, коли потрібно зробити прогнози того самого числа за один раз, що і розмір партії, що використовується під час навчання.

Це стає проблемою, коли ви хочете зробити менше прогнозів, ніж розмір партії. Наприклад, ви можете отримати найкращі результати при

великому обсязі партії, але вам потрібно робити прогнози для одного спостереження за раз на щось на зразок часового ряду або задачі послідовності.

Ось чому може бути бажаним мати інший розмір партії при пристосуванні мережі до навчальних даних, ніж при прогнозуванні тестових даних або нових вхідних даних.

Пакетний градієнтний спуск - це різновид алгоритму градієнтного спуска, який обчислює похибку для кожного прикладу в наборі даних навчального матеріалу, але оновлює модель лише після того, як всі навчальні приклади були оцінені.

У цьому розділі ми запроєкуємо LSTM для нашого завдання.

Для експерименту ми створюємо мережу з одним входом та виходом, прихований рівень на 100 нейронів.

Функція оптимізації середньоквадратичної помилки використовується для цієї задачі регресії з ефективним алгоритмом оптимізації ADAM (adaptive moment estimation). Функція втрат наведена на рис. 4.12. Отримана точність (Model Accuracy) - 82.54%.

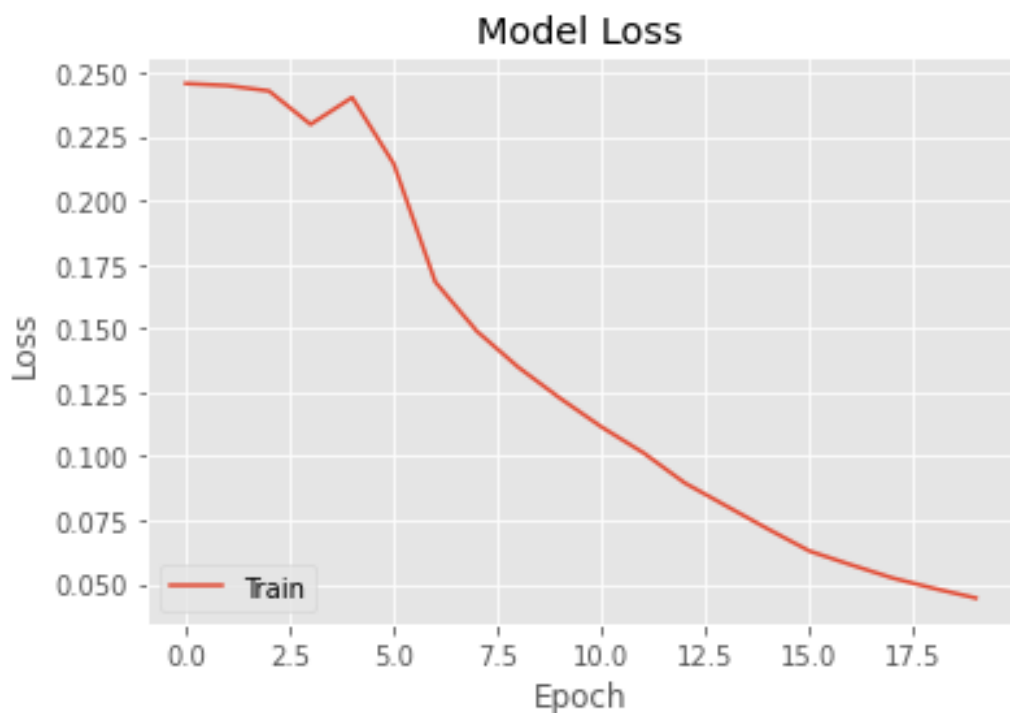


Рисунок 4.12 – Функція втрат для LSTM (розмір партії - 5)

Mini-batch gradient descent - це різновид алгоритму градієнтного спуску, який розділяє навчальний набір даних на невеликі партії, які використовуються для обчислення помилок моделі та оновлення коефіцієнтів моделі.

Mini-batch gradient descent є найбільш сприятливим та 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.

Ми будемо використовувати мережу LSTM, придатну для 100 епох, з розміром партії = 128.

Функція оптимізації середньоквадратичної помилки використовується для цієї задачі регресії з ефективним алгоритмом оптимізації ADAM.

Ми отримали точність (Model Accuracy) 83.23%. Функція втрат на рис. 4.13.

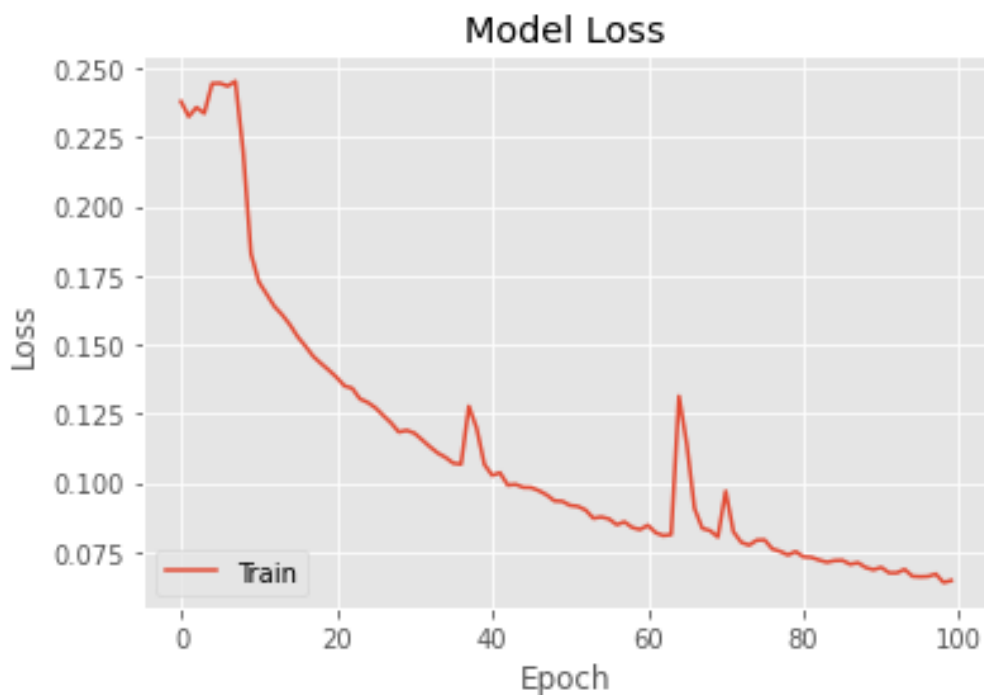


Рисунок 4.13 – Функція втрат для LSTM для Mini-batch gradient descent алгоритма

Отриману модель зберігаємо як STM-Mini-Batch-Model.h5

Stacked LSTM - це розширення цих моделей, що містить кілька прихованих слоїв LSTM, де кожен шар містить кілька ячеек пам'яті.

Розглянемо мережу з 3 прихованими шарами, розмір партії 120, 20 епох.

Отримана точність (Model Accuracy) 82.12%. Функція втрат на рис. 4.14.

Отриману модель зберігаємо для подальшого використання як:

```
model3L.save('models/LSTM-3L-Model.h5') # save model
```

```
del model3L # delete existing model
```

```
# returns a compiled model, identical to the previous one
```

```
model3L = load_model('models/LSTM-3L-Model.h5')
```

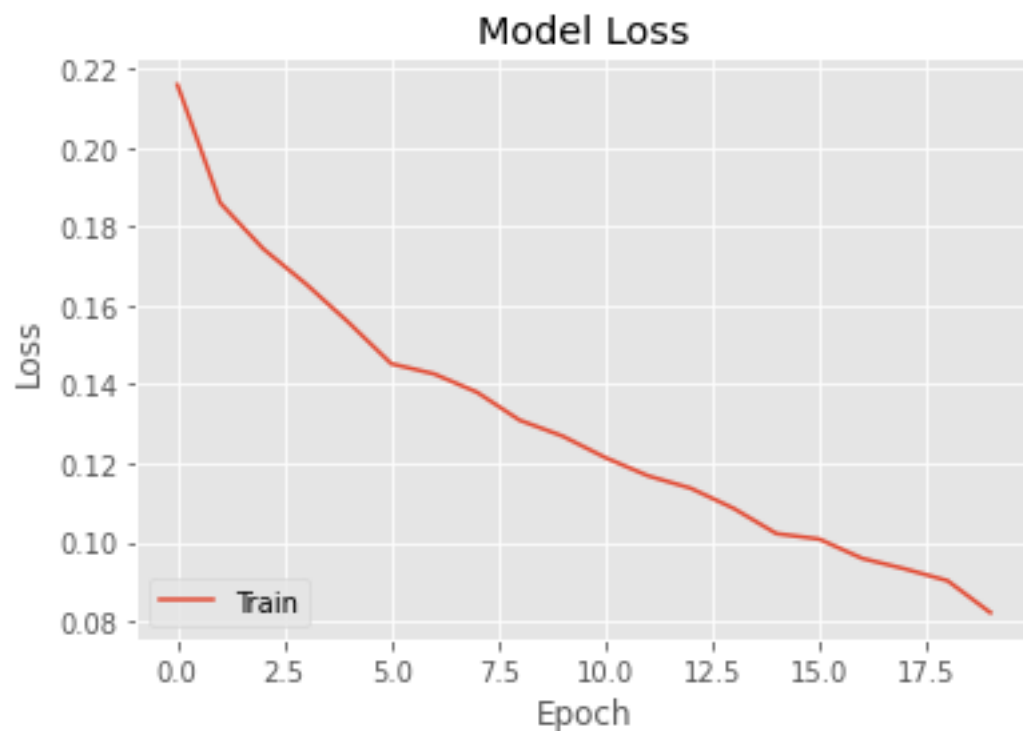


Рисунок 4.14 – Функція втрат для Stacked LSTM з 3 шарами

Двонаправлені LSTM - це продовження традиційних LSTM, які можуть покращити ефективність моделі при проблемах класифікації послідовностей.

У випадках, коли доступні всі кроки часу вхідної послідовності, двонаправлені LSTM тренують два, а не один LSTM на вхідній послідовності. Перший на вхідній послідовності як є, а другий на зворотній копії вхідної послідовності. Це може надати додатковий контекст мережі та призвести до швидшого та ще більш повного вивчення проблеми.

Ідея двонаправлених рекурентних нейронних мереж (RNN) є простою.

Він передбачає дублювання першого повторюваного шару в мережі так, що тепер є два шари поруч, потім надання вхідної послідовності такою, якою є як вхід для першого шару, і надання зворотної копії вхідної послідовності другому.

Отримана точність (Model Accuracy) 80.94%, функція втрат на рис.4.15.

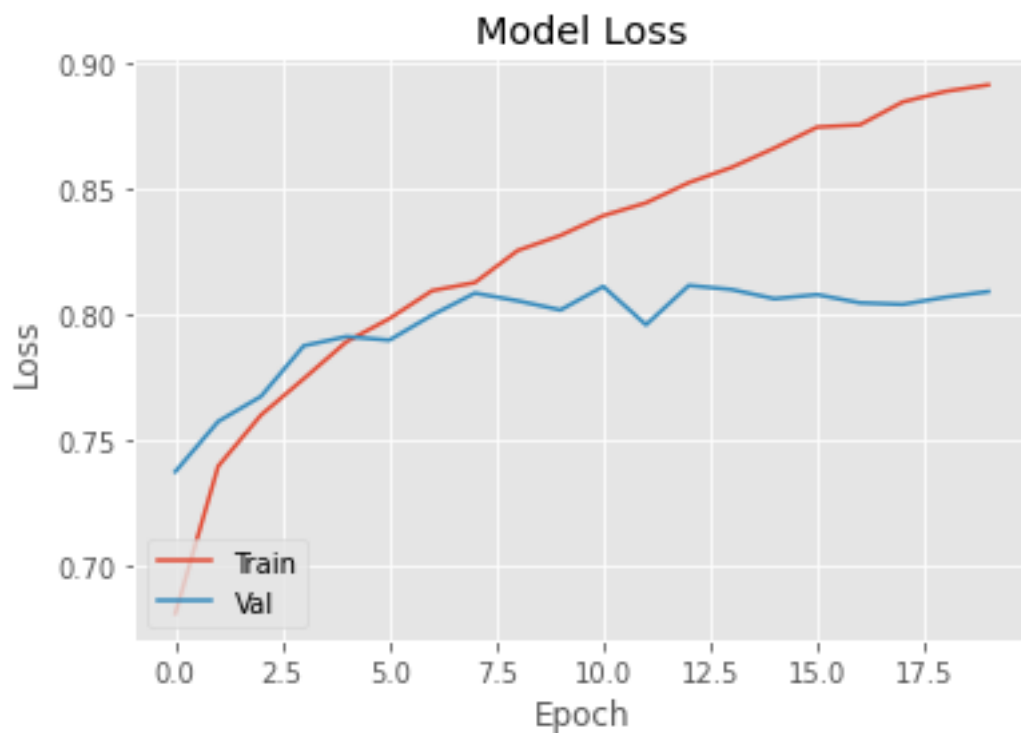


Рисунок 4.15 – Функція втрат для двонаправлених LSTM

Зберігаємо отриману модель:

```
model_Bi_LSTM.save('models/LSTM-Bi-Model.h5') # save model
```

```
del model_Bi_LSTM # delete existing model
```

returns a compiled model, identical to the previous one

```
model_Bi_LSTM = load_model('models/LSTM-Bi-Model.h5')
```

Отримаємо модель розпізнавання безперервних числових значень у багатьох вимірах, таких як простір валентності (VA). Будемо випробувати модель CNN-LSTM, що складається з двох частин: регіональної CNN та LSTM для прогнозування рейтингів VA для текстів.

Вектори слів із словникових слів навчаються з великого корпусу за допомогою набору інструментів 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 послідовно проходять 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.

Отримана точність (Model Accuracy) 81.08%. Функція втрат на рис. 4.16.

Зберігаємо отриману модель:

```
model_CNN_LSTM.save('models/LSTM-CNN-Model.h5') # save model
del model_CNN_LSTM # delete existing model
```

returns a compiled model, identical to the previous one

```
model_CNN_LSTM = load_model('models/LSTM-CNN-Model.h5')
```

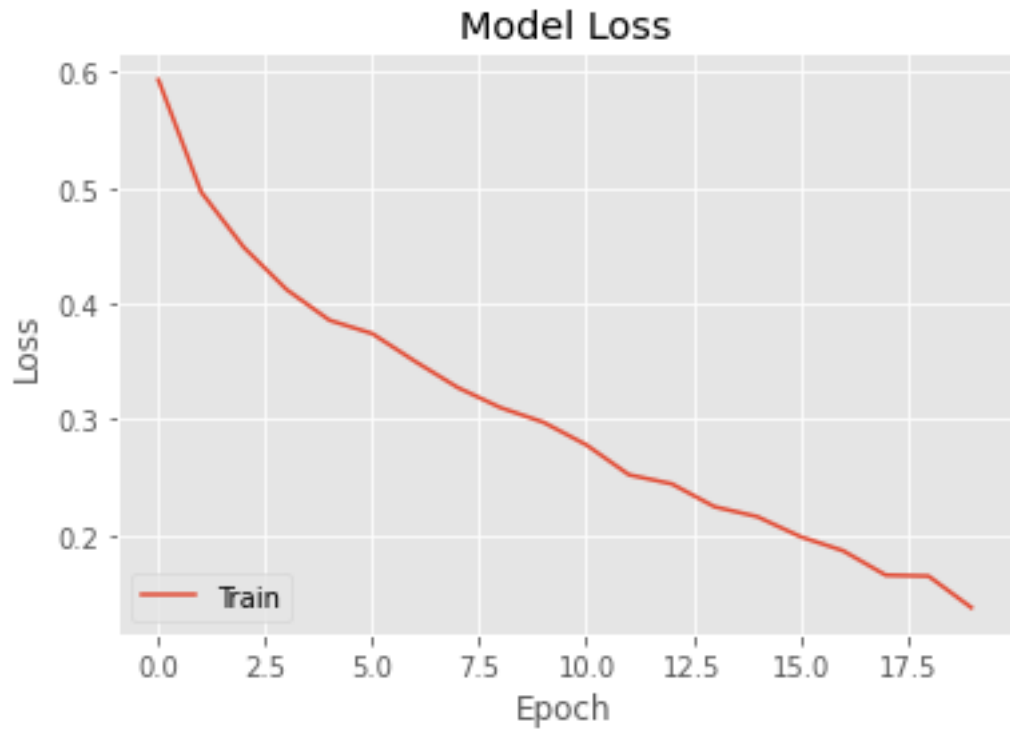


Рисунок 4.16 – Функція втрат для моделі CNN-LSTM

Для порівняння моделей використовуємо графіки (рис. 4.17-4.20) та ROC криву рис. 4.21.

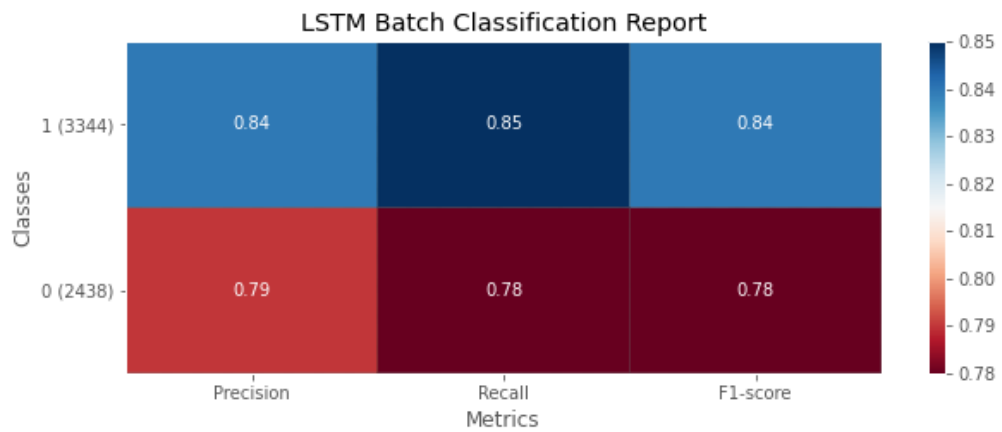


Рисунок 4.17 – Графік результатів класифікації для LSTM

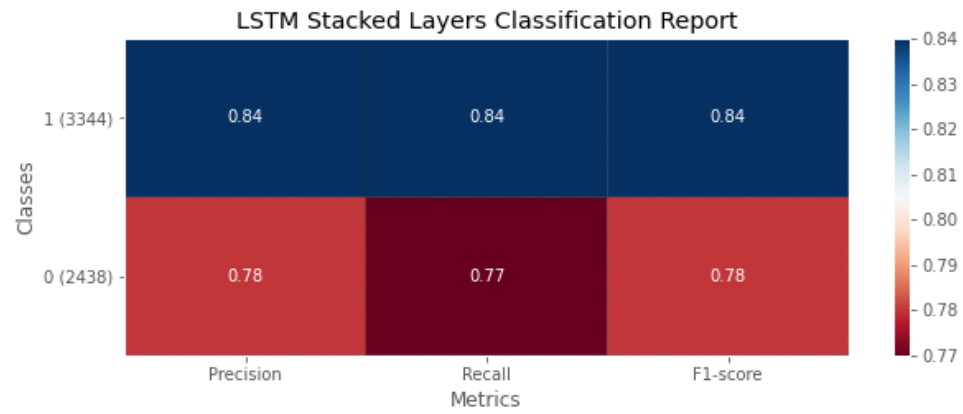


Рисунок 4.18 – Графік результатів класифікації для Stacked LSTM

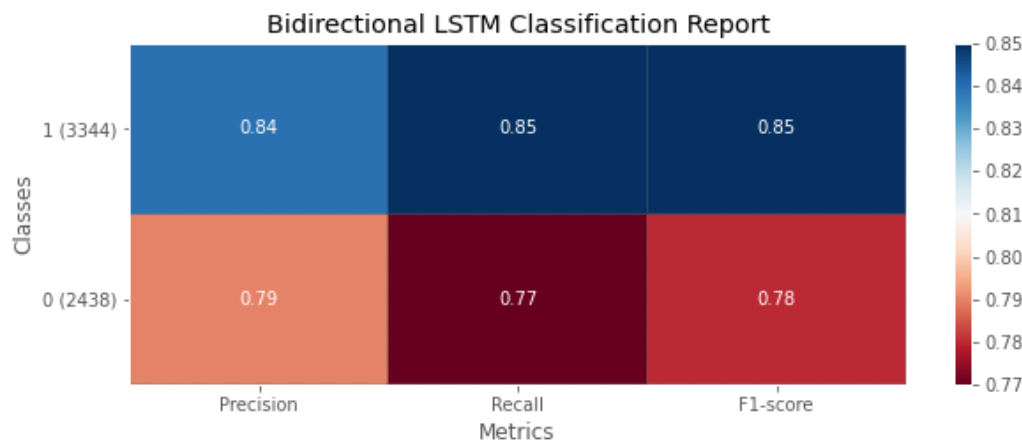


Рисунок 4.19 – Графік результатів класифікації для двонаправленого LSTM

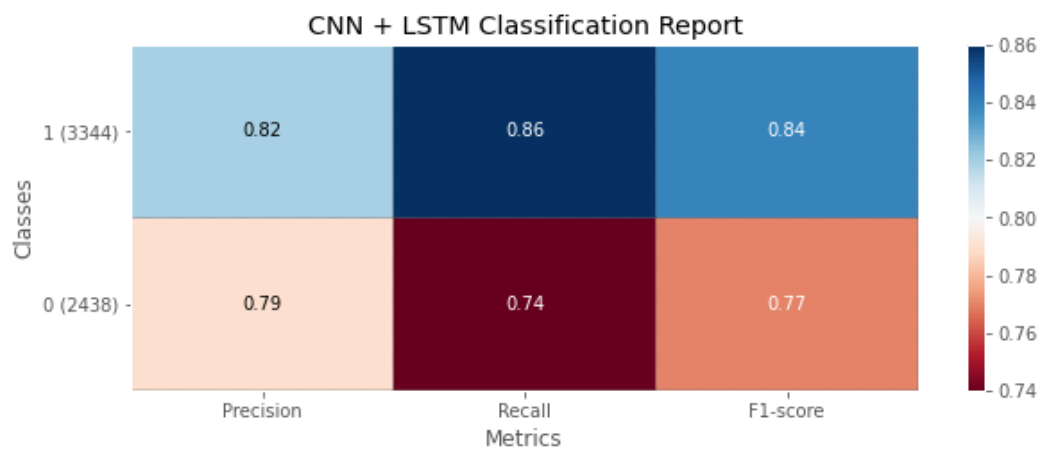


Рисунок 4.20 – Графік результатів класифікації для ЗНМ та LSTM

ROC-крива (англ. *receiver operating characteristic*, *робоча характеристика приймача*) — графік, що дозволяє оцінити якість бінарної

класифікації, відображає співвідношення між часткою об'єктів від загальної кількості носіїв ознаки, правильно класифікованих до загальної кількості об'єктів, що не несуть ознаки, помилково класифікованих, як такі, що мають ознаку. Також відома як крива похибок. Аналіз класифікацій із застосуванням ROC-кривих називається *ROC-аналізом*. На рисунку 4.21 наведена ROC крива.

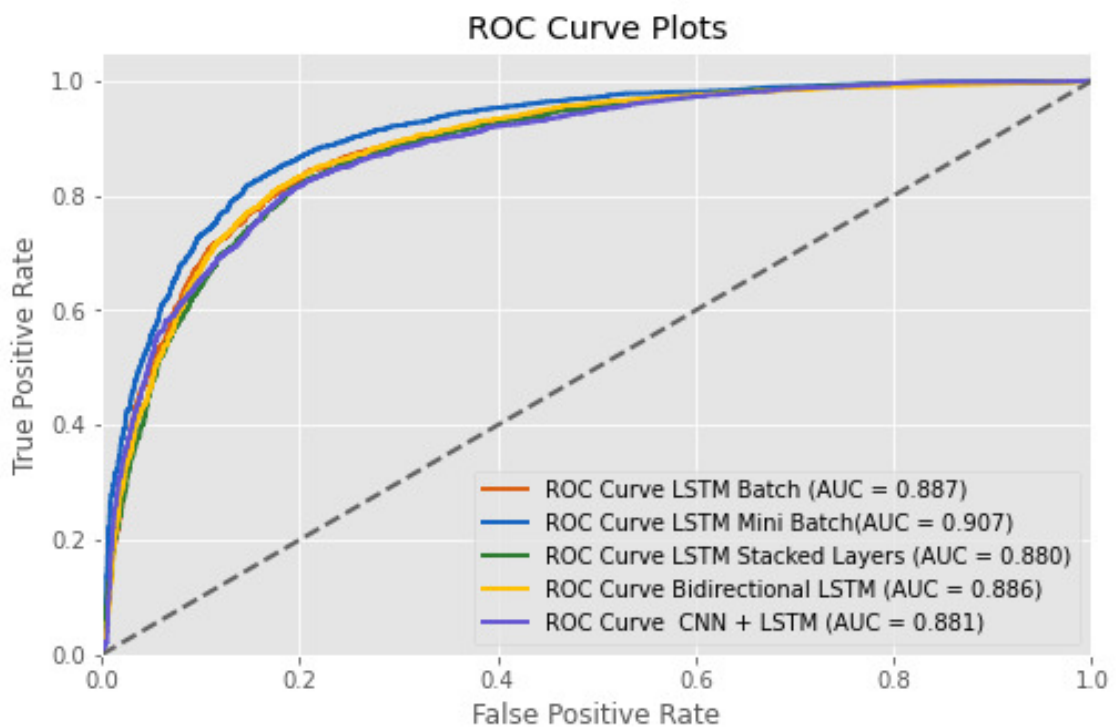


Рисунок 4.21 – ROC крива

Як можна побачити, усі моделі показують приблизно однаковий результат, але поки що найкращою є модель з Mini Batch (90%), тому вони і була використана для вебзастосування.

ВИСНОВКИ

В роботі вирішено завдання розроблення застосунку для визначення емоцій у документах здобувача міжнародної мобільності.

Для вирішення цього завдання були досліджені моделі для визначення емоцій в англomовних текстах з використанням бібліотек мови Python. Було отримано 5 моделей для визначення тональності текстів. Всі отримані моделі показали приблизно однаковий результат, але поки що найкращою є модель з Mini Batch (90%), тому вона і була використана для вебзастосування.

Розроблено програмне рішення з використанням вебінтерфейсу, що дозволяє здобувачам міжнародної мобільності завантажувати відповідні документи, а співробітникам міжнародного відділу аналізувати їх емоційний стан.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Реалізація студентами права на академічну мобільність. [Електрон. ресурс]. – Режим доступу: https://osvita.ua/vnz/high_school/71057/.
2. Національний університет Запорізька політехніка. Академічна мобільність. [Електрон. ресурс]. – Режим доступу: <https://zp.edu.ua/akademichna-mobilnist>.
3. Jocelyn D'Souza: An Introduction to Bag-of-Words in NLP [Electronic Resource] – Access mode: <https://medium.com/greyatom/an-introduction-to-bag-of-words-in-nlp-ac967d43b428>.
4. Numpy. [Electronic resource]. – Access mode: <https://numpy.org/>.
5. Scipy.org. [Electronic resource]. – Access mode: <https://www.scipy.org/>.
6. Introduction to TensorFlow. [Electronic resource]. – Access mode: <https://www.tensorflow.org/learn>.
7. Самые быстрые Python веб-фреймворки в 2019. [Електрон. ресурс]. – Режим доступу до ресурсу: <https://habr.com/ru/post/440282/>.
8. Django. [Electronic resource]. – Access mode: <https://www.djangoproject.com/>.
9. Flask. [Electronic resource]. – Access mode: <https://flask.palletsprojects.com/en/2.0.x/>.
10. Pyramid. [Electronic resource]. – Access mode: <https://trypyramid.com/>.
11. Sanic Framework. [Electronic resource]. – Access mode: <https://sanic.readthedocs.io/en/stable/#:~:text=Sanic%20is%20a%20Python%203.7,with%20an%20alternative%20ASGI%20webserver>.
12. Tornado. [Electronic resource]. – Access mode: <https://www.tornadoweb.org/en/stable/>.
13. Обработка естественного языка . [Електрон. ресурс]. – Режим доступу до ресурсу:

http://neerc.ifmo.ru/wiki/index.php?title=%D0%9E%D0%B1%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B0_%D0%B5%D1%81%D1%82%D0%B5%D1%81%D1%82%D0%B2%D0%B5%D0%BD%D0%BD%D0%BE%D0%B3%D0%BE_%D1%8F%D0%B7%D1%8B%D0%BA%D0%B0.

14. Natural Language Toolkit. [Electronic resource]. – Access mode: <https://www.nltk.org/>.

15. spaCy 101: Everything you need to know. [Electronic resource]. – Access mode: <https://spacy.io/usage/spacy-101>.

16. Gensim Tutorial – A Complete Beginners Guide. [Electronic resource]. – Access mode: <https://www.machinelearningplus.com/nlp/gensim-tutorial/>.

17. Multi-Domain Sentiment Dataset. [Electronic resource]. – Access mode: <http://www.cs.jhu.edu/~mdredze/datasets/sentiment/>.

ДОДАТОК А
Текст програми

A.1 Код файла settings.py

Django settings for sentimento project.

Generated by 'django-admin startproject' using Django 3.1.

For more information on this file, see
<https://docs.djangoproject.com/en/3.1/topics/settings/>

For the full list of settings and their values, see
<https://docs.djangoproject.com/en/3.1/ref/settings/>

```

"""
import os
import django_heroku
import dj_database_url
import tensorflow as tf
import gensim.models.keyedvectors as word2vec
from decouple import config, Csv

# Build paths inside the project like this: BASE_DIR / 'subdir'.
BASE_DIR = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))

# Quick-start development settings - unsuitable for production
# See https://docs.djangoproject.com/en/3.1/howto/deployment/checklist/

# Used for a default title
APP_NAME = 'SENTIMENTO WEB'

# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = config('SECRET_KEY')

# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = config('DEBUG', default=False, cast=bool)

ALLOWED_HOSTS = config('ALLOWED_HOSTS', cast=Csv())

# Document
ALLOW_FILE_ATTACHMENTS = True
ALLOWED_FILE_ATTACHMENTS = [".pdf"]
MAXIMUM_ATTACHMENT_SIZE = 5000000 # In bytes
MAXIMUM_ATTACHMENT_FILES = 3

```

```

COMMENT_CLASSES = []
STAFF_ONLY = config('STAFF_ONLY', default=False, cast=bool)

# Without this, uploaded files > 4MB end up with perm 0600, unreadable
# by web server process
FILE_UPLOAD_PERMISSIONS = 0o644

SITE_ID = 1

WordVec = word2vec.KeyedVectors.load_word2vec_format('models/glove.6B.50d.w2vformat
.txt', binary=False, encoding="utf-8")
graph = tf.compat.v1.get_default_graph()

# Application definition

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'django.contrib.sites',

    'analyzer.apps.AnalyzerConfig',
    'authz.apps.AuthzConfig',
    'home.apps.HomeConfig',
]

MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
    'whitenoise.middleware.WhiteNoiseMiddleware',
]

ROOT_URLCONF = 'sentimento.urls'

TEMPLATES = [
    {

```

```

'BACKEND': 'django.template.backends.django.DjangoTemplates',
'DIRS': [],
'APP_DIRS': True,
'OPTIONS': {
    'context_processors': [
        'django.template.context_processors.debug',
        'django.template.context_processors.request',
        'django.template.context_processors.media',
        'django.contrib.auth.context_processors.auth',
        'django.contrib.messages.context_processors.messages',
        'home.context_processors.settings',
    ],
},
],
]

```

```
WSGI_APPLICATION = 'sentimento.wsgi.application'
```

```
# Database
```

```
# https://docs.djangoproject.com/en/3.1/ref/settings/#databases
```

```

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': os.path.join(BASE_DIR, 'db.sqlite3'),
    }
}

```

```
DEFAULT_AUTO_FIELD = 'django.db.models.AutoField'
```

```
# Password validation
```

```
# https://docs.djangoproject.com/en/3.1/ref/settings/#auth-password-  
validators
```

```

AUTH_PASSWORD_VALIDATORS = [
    {
        'NAME':
'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
    },
    {
        'NAME':
'django.contrib.auth.password_validation.MinimumLengthValidator',
    },
    {

```

```

        'NAME':
'django.contrib.auth.password_validation.CommonPasswordValidator',
    },
    {
        'NAME':
'django.contrib.auth.password_validation.NumericPasswordValidator',
    },
]

```

```

# Internationalization
# https://docs.djangoproject.com/en/3.1/topics/i18n/

```

```
LANGUAGE_CODE = 'en-us'
```

```
TIME_ZONE = 'UTC'
```

```
USE_I18N = True
```

```
USE_L10N = True
```

```
USE_TZ = True
```

```

# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/3.1/howto/static-files/
STATICFILES_STORAGE =
'whitenoise.storage.CompressedStaticFilesStorage'

```

```
STATIC_ROOT = os.path.join(BASE_DIR, 'staticfiles')
```

```
STATIC_URL = '/static/'
```

```

# Extra places for collectstatic to find static files.
# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/2.0/howto/static-files/
STATICFILES_DIRS = (
    'static',
    'models',
    os.path.join(BASE_DIR, 'static'),
)

```

```

AUTHENTICATION_BACKENDS = (
    'django.contrib.auth.backends.ModelBackend',
)

```

```

LOGOUT_REDIRECT_URL = '/'
LOGIN_REDIRECT_URL = '/'

```

```

LOGGING = {
    'version': 1,
    'disable_existing_loggers': False,
    'handlers': {
        'console': {
            'class': 'logging.StreamHandler',
        },
    },
    'loggers': {
        '': {
            'handlers': ['console'],
            'level': 'DEBUG',
            'propagate': True,
        },
    },
}

```

```

MEDIA_URL = '/media/'
MEDIA_ROOT = os.path.join(BASE_DIR, 'media')

```

```

# Heroku: Update database configuration from $DATABASE_URL.
db_from_env = dj_database_url.config(conn_max_age=2000)
DATABASES['default'].update(db_from_env)

```

```

django_heroku.settings(locals())

```

A.2 Код файла url.py

```

import os

```

```

from django.contrib import admin
from django.urls import include, path
from django.views.generic import TemplateView
from django.views.static import serve
from django.conf.urls.static import static
from django.conf import settings

```

```

# Up two folders to serve "site" content
BASE_DIR = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))

```

```

urlpatterns = [
    path('admin/', admin.site.urls),

    # Sample applications
    path("", TemplateView.as_view(template_name='home/main.html'),
name='home_page'),
    path('authz/', include('authz.urls')),
    path('accounts/', include('django.contrib.auth.urls')),
    path('analyzer/', include('analyzer.urls', namespace='analyzer')),
]

# Serve the favicon
urlpatterns += [
    path('favicon.ico', serve, {
        'path': 'favicon.ico',
        'document_root': os.path.join(BASE_DIR, 'home/static'),
    }),
]

# Serve the media
urlpatterns += static(settings.MEDIA_URL,
document_root=settings.MEDIA_ROOT)

```

A.3 Код файла `models.py`

```

from __future__ import unicode_literals

import datetime
import os
import textwrap
import uuid

from django.conf import settings
from django.contrib.auth.models import Group
from django.db import DEFAULT_DB_ALIAS, models
from django.db.transaction import Atomic, get_connection
from django.urls import reverse
from django.utils import timezone

def get_attachment_upload_dir(instance, filename):
    """Determine upload dir for task attachment files."""

```

```
return "/" .join(["tasks", "attachments", str(instance.task.id), filename])
```

```
class LockedAtomicTransaction(Atomic):
```

```
    """
```

```
    This is needed for safely merging
```

```
    Does a atomic transaction, but also locks the entire table for any
    transactions, for the duration of this
```

```
    transaction. Although this is the only way to avoid concurrency issues in
    certain situations, it should be used with
```

```
    caution, since it has impacts on performance, for obvious reasons...
```

```
    """
```

```
    def __init__(self, *models, using=None, savepoint=None):
```

```
        if using is None:
```

```
            using = DEFAULT_DB_ALIAS
```

```
        super().__init__(using, savepoint)
```

```
        self.models = models
```

```
    def __enter__(self):
```

```
        super(LockedAtomicTransaction, self).__enter__()
```

```
        # Make sure not to lock, when sqlite is used, or you'll run into
        problems while running tests!!!
```

```
        if settings.DATABASES[self.using]["ENGINE"] !=
        "django.db.backends.sqlite3":
```

```
            cursor = None
```

```
            try:
```

```
                cursor = get_connection(self.using).cursor()
```

```
                for model in self.models:
```

```
                    cursor.execute("LOCK TABLE
```

```
{table_name} ".format(table_name=model._meta.db_table))
```

```
            finally:
```

```
                if cursor and not cursor.closed:
```

```
                    cursor.close()
```

```
class TaskList(models.Model):
```

```
    name = models.CharField(max_length=60)
```

```
    slug = models.SlugField(default="")
```

```
    group = models.ForeignKey(Group, on_delete=models.CASCADE)
```

```
    def __str__(self):
```

```
        return self.name
```

```

class Meta:
    ordering = ["name"]
    verbose_name_plural = "Task Lists"

    # Prevents (at the database level) creation of two lists with the same
    slug in the same group
    unique_together = ("group", "slug")

class Task(models.Model):
    id = models.CharField(max_length=100, blank=True, unique=True,
primary_key=True, default=uuid.uuid4)
    username = models.CharField(max_length=140)
    task_list = models.ForeignKey(TaskList, on_delete=models.CASCADE,
null=True)
    created_date = models.DateField(default=timezone.now, blank=True,
null=True)
    email = models.CharField(max_length=100, blank=False, default=False)
    checked = models.BooleanField(default=False)
    checked_date = models.DateField(blank=True, null=True)
    assigned_to = models.ForeignKey(settings.AUTH_USER_MODEL,
blank=True, null=True, related_name="task_assigned_to",
on_delete=models.CASCADE, )
    note = models.TextField(blank=True, null=True)
    priority = models.PositiveIntegerField(blank=True, null=True)

    def __str__(self):
        return self.username

    def get_absolute_url(self):
        return reverse("analyzer:task_detail", kwargs={"task_id": self.id})

    # Auto-set the Task creation / checked date
    def save(self, **kwargs):
        # If Task is being marked complete, set the checked_date
        if self.checked:
            self.checked_date = datetime.datetime.now()
            super(Task, self).save()

    def merge_into(self, merge_target):
        if merge_target.pk == self.pk:
            raise ValueError("can't merge a task with self")

```

```

# lock the comments to avoid concurrent additions of comments after
the
# update request. these comments would be irremediably lost because
of
# the cascade clause
with LockedAtomicTransaction(Comment):
    Comment.objects.filter(task=self).update(task=merge_target)
    self.delete()

class Meta:
    ordering = ["priority", "created_date"]

class Comment(models.Model):
    """
    Not using Django's built-in comments because we want to be able to
save
    a comment and change task details at the same time. Rolling our own
since it's easy.
    """

    task = models.ForeignKey(Task, on_delete=models.CASCADE)
    author = models.ForeignKey(settings.AUTH_USER_MODEL,
on_delete=models.CASCADE, blank=True, null=True)
    date = models.DateTimeField(default=datetime.datetime.now)

    body = models.TextField(blank=True)

    @property
    def author_text(self):
        if self.author is not None:
            return str(self.author)

    @property
    def snippet(self):
        body_snippet = textwrap.shorten(self.body, width=35,
placeholder="...")
        # Define here rather than in __str__ so we can use it in the admin
list_display
        return "{author} - {snippet}...".format(author=self.author_text,
snippet=body_snippet)

    def __str__(self):
        return self.snippet

```

```

class Attachment(models.Model):
    """ Defines a generic file attachment for use in M2M relation with Task
    """

    task = models.ForeignKey(Task, on_delete=models.CASCADE)
    timestamp = models.DateTimeField(default=datetime.datetime.now)
    file = models.FileField(upload_to=get_attachment_upload_dir,
max_length=255)

    def filename(self):
        return os.path.basename(self.file.name)

    def extension(self):
        name, extension = os.path.splitext(self.file.name)
        return extension

    def __str__(self):
        return f"{self.task.id} - {self.file.name}"

```

A.4 Код файла forms.py

```

from django import forms
from django.contrib.auth.models import Group
from django.forms import ModelForm
from analyzer.models import Task, TaskList

class AddTaskListForm(ModelForm):
    """The picklist showing allowable groups to which a new list can be
added
determines which groups the user belongs to. This queries the form
object
to derive that list."""

    def __init__(self, user, *args, **kwargs):
        super(AddTaskListForm, self).__init__(*args, **kwargs)
        self.fields["group"].queryset = Group.objects.filter(user=user)
        self.fields["group"].widget.attrs = {"id": "id_group", "class": "custom-
select mb-3", "name": "group", }

    class Meta:
        model = TaskList

```

```

exclude = ["created_date", "slug"]

class AddEditTaskForm(ModelForm):
    """The picklist showing the users to which a new task can be assigned
    must find other members of the group this TaskList is attached to."""

    def __init__(self, user, *args, **kwargs):
        super().__init__(*args, **kwargs)
        task_list = kwargs.get("initial").get("task_list")
        members = task_list.group.user_set.all()
        self.fields["assigned_to"].queryset = members
        self.fields["assigned_to"].label_from_instance = lambda obj: "%s
(%s)" % (obj.get_full_name(), obj.username, )
        self.fields["assigned_to"].widget.attrs = {"id": "id_assigned_to",
                                                    "class": "custom-select mb-3",
                                                    "name": "assigned_to", }
        self.fields["task_list"].value = kwargs["initial"]["task_list"].id

    email = forms.CharField(widget=forms.widgets.TextInput())
    username = forms.CharField(widget=forms.widgets.TextInput())
    note = forms.CharField(widget=forms.Textarea(), required=False)
    checked = forms.BooleanField(required=False)

    def clean_created_by(self):
        """Keep the existing created_by regardless of anything coming from
        the submitted form.
        If creating a new task, then created_by will be None, but we set it
        before saving."""
        return self.instance.created_by

    class Meta:
        model = Task
        exclude = []

    class SearchForm(forms.Form):
        """Search."""
        q = forms.CharField(widget=forms.widgets.TextInput(attrs={"size":
35})))

```

A.5 Код файла analyzer/views/__init__.py

```

from analyzer.views.add_list import add_list # noqa: F401
from analyzer.views.del_list import del_list # noqa: F401
from analyzer.views.delete_task import delete_task # noqa: F401
from analyzer.views.import_csv import import_csv # noqa: F401
from analyzer.views.list_detail import list_detail # noqa: F401
from analyzer.views.list_lists import list_lists # noqa: F401
from analyzer.views.remove_attachment import remove_attachment #
noqa: F401
from analyzer.views.prediction import prediction # noqa: F401
from analyzer.views.reorder_tasks import reorder_tasks # noqa: F401
from analyzer.views.search import search # noqa: F401
from analyzer.views.task_detail import task_detail # noqa: F401
from analyzer.views.toggle_done import toggle_done # noqa: F401

```

A.6 Код файла analyzer/views/add_list.py

```

from django.contrib import messages
from django.contrib.auth.decorators import login_required,
user_passes_test
from django.core.exceptions import PermissionDenied
from django.db import IntegrityError
from django.http import HttpResponseRedirect
from django.shortcuts import redirect, render
from django.utils.text import slugify

from analyzer.forms import AddTaskListForm
from analyzer.utils import staff_check

@login_required
@user_passes_test(staff_check)
def add_list(request) -> HttpResponseRedirect:
    """Allow users to add a new list to the group they're in."""

    # Only staffers can add lists, regardless of STAFF_USER setting.
    if not request.user.is_staff:
        raise PermissionDenied

    if request.POST:
        form = AddTaskListForm(request.user, request.POST)

```

```

    if form.is_valid():
        try:
            newlist = form.save(commit=False)
            newlist.slug = slugify(newlist.name, allow_unicode=True)
            newlist.save()
            messages.success(request, "A new list has been added.")
            return redirect("analyzer:lists")

        except IntegrityError:
            messages.warning(request, "There was a problem saving the new
list. "
                                "Most likely a list with the same name in the
same group already exists.", )
        else:
            if request.user.groups.all().count() == 1:
                form = AddTaskListForm(request.user, initial={"group":
request.user.groups.all()[0]})
            else:
                form = AddTaskListForm(request.user)

    context = {"form": form}

    return render(request, "analyzer/add_list.html", context)

```

A.7 Код файла analyzer/views/del_list.py

```

from django.contrib import messages
from django.contrib.auth.decorators import login_required,
user_passes_test
from django.core.exceptions import PermissionDenied
from django.http import HttpResponseRedirect
from django.shortcuts import get_object_or_404, redirect, render

from analyzer.models import Task, TaskList
from analyzer.utils import staff_check

@login_required
@user_passes_test(staff_check)
def del_list(request, list_id: int, list_slug: str) -> HttpResponseRedirect:
    """Delete an entire list. Only staff members should be allowed to access
this view."""
    task_list = get_object_or_404(TaskList, id=list_id)

```

```

        # Ensure user has permission to delete list. Get the group this list belongs
to,
        # and check whether current user is a member of that group AND a
staffer.
        if task_list.group not in request.user.groups.all():
            raise PermissionDenied
        if not request.user.is_staff:
            raise PermissionDenied

        if request.method == "POST":
            TaskList.objects.get(id=task_list.id).delete()
            messages.success(request, "{list_name} is
gone.".format(list_name=task_list.name))
            return redirect("analyzer:lists")
        else:
            task_count_done = Task.objects.filter(task_list=task_list.id,
checked=True).count()
            task_count_undone = Task.objects.filter(task_list=task_list.id,
checked=False).count()
            task_count_total = Task.objects.filter(task_list=task_list.id).count()

            context = {
                "task_list": task_list,
                "task_count_done": task_count_done,
                "task_count_undone": task_count_undone,
                "task_count_total": task_count_total,
            }

            return render(request, "analyzer/del_list.html", context)

```

A.8 Код файла analyzer/views/import_csv.py

```

from django.contrib import messages
from django.contrib.auth.decorators import login_required,
user_passes_test
from django.http import HttpResponseRedirect
from django.shortcuts import redirect, render, reverse

from analyzer.operations.csv_importer import CSVImporter
from analyzer.utils import staff_check

```

```

@login_required
@user_passes_test(staff_check)
def import_csv(request) -> HttpResponse:
    """Import a specifically formatted CSV into stored tasks."""

    ctx = {"results": None}

    if request.method == "POST":
        filepath = request.FILES.get("csvfile")

        if not filepath:
            messages.error(request, "You must supply a CSV file to import.")
            return redirect(reverse("analyzer:import_csv"))

        importer = CSVImporter()
        results = importer.upsert(filepath)

        if results:
            ctx["results"] = results
        else:
            messages.error(request, "Could not parse provided CSV file.")
            return redirect(reverse("analyzer:import_csv"))

    return render(request, "analyzer/import_csv.html", context=ctx)

```

A.9 Код файла analyzer/views/prediction.py

```

import os

from django.contrib.auth.decorators import login_required,
user_passes_test
from django.core.exceptions import PermissionDenied
from django.http import HttpResponse
from django.shortcuts import get_object_or_404, render

from analyzer.utils import staff_check, predict, convert_pdf_to_txt
from analyzer.models import Attachment, Comment

@login_required
@user_passes_test(staff_check)
def prediction(request, attachment_id: int) -> HttpResponse:

```

```

        """ Prediction make sentiment analysis of given pdf attachment and
        return result specific percent of positivity.
        Redirect to the prediction page.
        """
        if request.method == "POST":
            attachment = get_object_or_404(Attachment, pk=attachment_id)

            # Permissions
            if not (attachment.task.task_list.group in request.user.groups.all() or
request.user.is_superuser):
                raise PermissionDenied

            txt = convert_pdf_to_txt(attachment.file.path)

            txt_prediction = predict(txt)
            msg = "Prediction for file " + \
str(os.path.basename(attachment.file.name)) + ": " + \
            str(round(float(txt_prediction), 4) * 100) + '%'
            print(msg)

            # posted comment with result of prediction
            Comment.objects.create(author=request.user, task=attachment.task,
body=msg)

            context = {'prediction': round(float(txt_prediction), 2) * 100,
'attachment': attachment.filename}

            return render(request, 'analyzer/predict.html', context)

```

A.10 Код файла analyzer/views/task_detail.py

```

import datetime
import os

import bleach
from django import forms
from django.contrib import messages
from django.core.exceptions import PermissionDenied
from django.http import HttpResponseRedirect
from django.shortcuts import get_object_or_404, redirect, render
from django.urls import reverse

from analyzer.defaults import defaults

```

```

from analyzer.features import HAS_TASK_MERGE
from analyzer.forms import AddEditTaskForm
from analyzer.models import Attachment, Comment, Task
from analyzer.utils import (toggle_task_checked, user_can_read_task)

if HAS_TASK_MERGE:
    from dal import autocomplete

def handle_add_comment(request, task):
    if not request.POST.get("add_comment"):
        return

    Comment.objects.create(author=request.user, task=task,
body=bleach.clean(request.POST["comment-body"], strip=True))

    messages.success(request, "Comment posted.")

def task_detail(request, task_id: int) -> HttpResponse:
    """View task details. Allow task details to be edited. Process new
comments on task."""

    task = get_object_or_404(Task, pk=task_id)
    comment_list = Comment.objects.filter(task=task_id).order_by("-date")

    # Handle task merging
    if not HAS_TASK_MERGE:
        merge_form = None
    else:

        class MergeForm(forms.Form):
            merge_target = =
forms.ModelChoiceField(queryset=Task.objects.all(),
widget=autocomplete.ModelSelect2(
url=reverse("analyzer:task_autocomplete", kwargs={"task_id":
task_id})), )

    # Handle task merging
    if not request.POST.get("merge_task_into"):
        merge_form = MergeForm()
    else:
        merge_form = MergeForm(request.POST)
        if merge_form.is_valid():

```

```

        merge_target = merge_form.cleaned_data["merge_target"]
        if not user_can_read_task(merge_target, request.user):
            raise PermissionDenied

        task.merge_into(merge_target)
        return redirect(reverse("analyzer:task_detail", kwargs={"task_id":
merge_target.pk})))

    # Save submitted comments
    handle_add_comment(request, task)

    # Save task edits
    if not request.POST.get("add_edit_task"):
        form = AddEditTaskForm(request.user, instance=task,
initial={"task_list": task.task_list})
    else:
        form = AddEditTaskForm(request.user, request.POST, instance=task,
initial={"task_list": task.task_list})

    if form.is_valid():
        item = form.save(commit=False)
        item.note = bleach.clean(form.cleaned_data["note"], strip=True)
        item.username = bleach.clean(form.cleaned_data["username"],
strip=True)
        item.save()
        messages.success(request, "The task has been edited.")
        return redirect("analyzer:list_detail", list_id=task.task_list.id,
list_slug=task.task_list.slug)

    # Mark complete
    if request.POST.get("toggle_done"):
        results_changed = toggle_task_checked(task.id)
        if results_changed:
            messages.success(request, f"Changed completion status for task
{task.id}")

    return redirect("analyzer:task_detail", task_id=task.id)

    # Handle uploaded files
    if request.FILES.get("attachment_file_input"):
        file = request.FILES.get("attachment_file_input")

        if Attachment.objects.filter(task_id=task.id).count() >=
defaults("MAXIMUM_ATTACHMENT_FILES"):

```

```

        messages.error(request, f"You can upload maximum 3 files, if you
want re-upload, then delete something.")
        return redirect("analyzer:task_detail", task_id=task.id)

    if file.size > defaults("MAXIMUM_ATTACHMENT_SIZE"):
        messages.error(request, f"File exceeds maximum attachment size.")
        return redirect("analyzer:task_detail", task_id=task.id)

    name, extension = os.path.splitext(file.name)

    if extension not in defaults("LIMIT_FILE_ATTACHMENTS"):
        messages.error(request, f"This site allows you to upload only `pdf`,
you can't upload {extension} files.")
        return redirect("analyzer:task_detail", task_id=task.id)

    Attachment.objects.create(task=task,
timestamp=datetime.datetime.now(), file=file)
    messages.success(request, f"File attached successfully")
    return redirect("analyzer:task_detail", task_id=task.id)

    context = {
        "task": task,
        "comment_list": comment_list,
        "form": form,
        "merge_form": merge_form,
        "comment_classes": defaults("COMMENT_CLASSES"),
        "attachments_enabled":
defaults("ALLOW_FILE_ATTACHMENTS"),
    }

    return render(request, "analyzer/task_detail.html", context)

```

A.11 Код файла `authz/views.py`

```

from django.shortcuts import render, redirect
from django.views import View
from django.contrib.auth.mixins import LoginRequiredMixin
from django.urls import reverse
from django.utils.http import urlencode

class OpenView(View):
    def get(self, request):
        return render(request, 'authz/main.html')

```

```

class ApereoView(View) :
    def get(self, request):
        return render(request, 'authz/main.html')

class ManualProtect(View) :
    def get(self, request):
        if not request.user.is_authenticated :
            loginurl = reverse('login')+ '?'+urlencode({'next': request.path})
            return redirect(loginurl)
        return render(request, 'authz/main.html')

class ProtectView(LoginRequiredMixin, View) :
    def get(self, request):
        return render(request, 'authz/main.html')

from django.http import HttpResponseRedirect

class DumpPython(View) :
    def get(self, req):
        resp = "<pre>\nUser Data in Python:\n\n"
        resp += "Login url: " + reverse('login') + "\n"
        resp += "Logout url: " + reverse('logout') + "\n\n"
        if req.user.is_authenticated:
            resp += "User: " + req.user.username + "\n"
            resp += "Email: " + req.user.email + "\n"
        else:
            resp += "User is not logged in\n"

        resp += "\n"
        resp += "</pre>\n"
        resp += """"<a href="/authz">Go back</a>""""
        return HttpResponseRedirect(resp)

```

A.12 Код файла authz/urls.py

```

from django.urls import path
from . import views
from django.views.generic import TemplateView

app_name='authz'
urlpatterns = [
    path("", TemplateView.as_view(template_name='authz/main.html')),

```

```

path('open', views.OpenView.as_view(), name='open'),
path('apereo', views.ApereoView.as_view(), name='apereo'),
path('manual', views.ManualProtect.as_view(), name='manual'),
path('protect', views.ProtectView.as_view(), name='protect'),
path('python', views.DumpPython.as_view(), name='python'),
]

```

A.13 Код файла authz/templates/authz/main.html

```

h1>Current request.path {{ request.path }}</h1>
{% if user.is_authenticated %}
<p>Authenticated as
<pre>
Name: {{ user.get_full_name }}
Email: {{ user.email }}
Id: {{ user.id }}
</pre>
</p>
<p>You can <a href="{% url 'logout' %}"?next={% url 'authz:open'
%}">Logout</a></p>
{% else %}
<p>You are not logged in</p>
<p>You can <a href="{% url 'login' %}"?next={{ request.path
}}">Login</a> if you like.</p>
{% endif %}
<ul>
<li>
Go to <a href="{% url 'authz:open' %}">{% url 'authz:open' %}</a>
(no login required)
</li>
<li>
Go to <a href="{% url 'authz:apereo' %}">{% url 'authz:apereo'
%}</a>
(no login required)
</li>
<li>
Go to <a href="{% url 'authz:manual' %}">{% url 'authz:manual'
%}</a>
(protected by user.is_authenticated)
</li>
<li>
Go to <a href="{% url 'authz:protect' %}">{% url 'authz:protect'
%}</a>

```

```
        (protected by LoginRequiredMixin)
    </li>
    <li>
        Go to <a href="{% url 'authz:python' %}">{% url 'authz:python'
    %}</a>
        dump request.user data in python
    </li>
</ul>
```

ДОДАТОК Б
Слайди презентації

РОЗРОБЛЕННЯ ЗАСТОСУНКУ ДЛЯ ВИЗНАЧЕННЯ ЕМОЦІЙ
У ДОКУМЕНТАХ ЗДОБУВАЧА МІЖНАРОДНОЇ
МОБІЛЬНОСТІ APPLICATION DEVELOPMENT FOR EMOTION
RECOGNITION IN THE DOCUMENTS OF THE APPLICANTS
FOR INTERNATIONAL MOBILITY PROGRAMMES

ВИКОНАВ: СТУДЕНТ(КА) 4 КУРСУ, ГРУПИ КНТ-218СП

ЦАПІН А.Д.

КЕРІВНИК: ДОЦЕНТ, К.Т.Н.

ШИТІКОВА О.В.

Рисунок Б.1 – Слайд 1

2

- ▶ Об'єкт дослідження – процес автоматичного визначення емоцій в документах.
- ▶ Предмет дослідження – програмні засоби для емоційного аналізу документів здобувачів міжнародної мобільності.
- ▶ Мета роботи – розроблення програмного забезпечення для емоційного аналізу документів здобувачів міжнародної мобільності.

Рисунок Б.2 – Слайд 2

3

Академічна мобільність

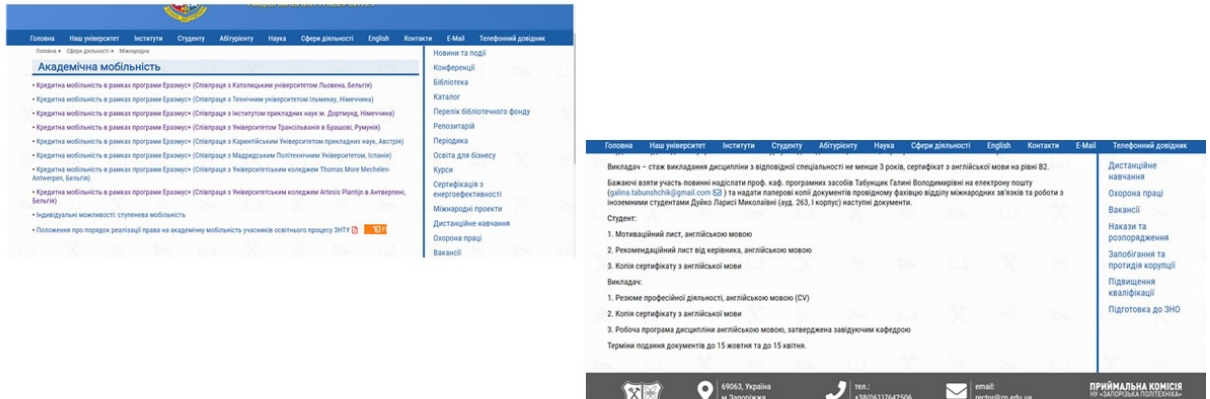


Рисунок Б.3 – Слайд 3

4

Аналіз тональності тексту

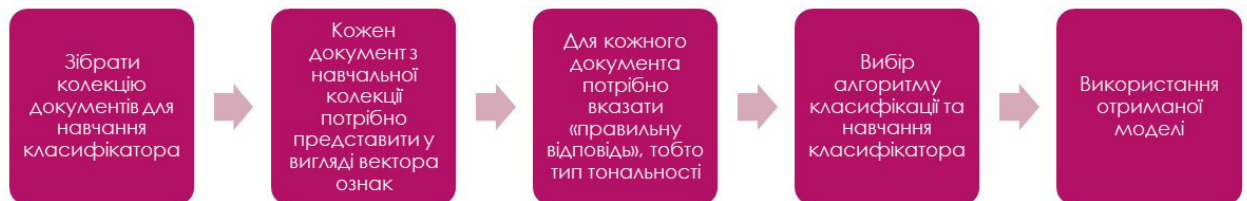


Рисунок Б.4 – Слайд 4

Функціональні вимоги до системи

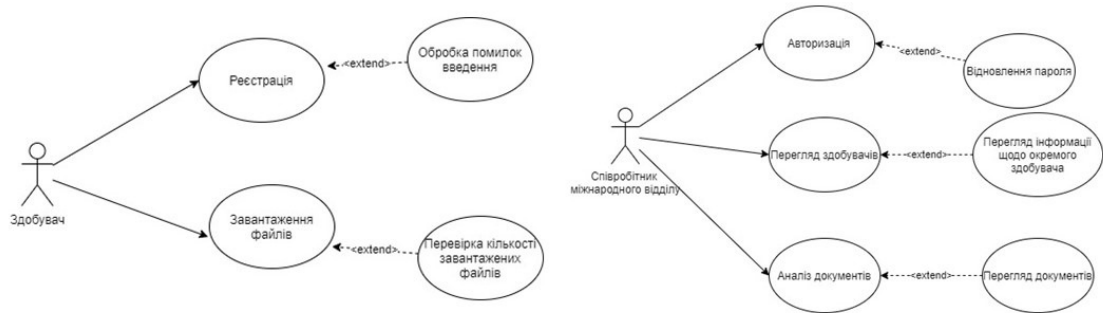


Рисунок Б.5 – Слайд 5

Обґрунтування вибору програмних засобів для розроблення веб застосунку

	Середнє		Максимальне		Запит, с.	Передача сек (KB)
	Затримка (ms)	Зап./Sec	Затримка (ms)	Зап./Sec		
Django	147,99	120,91	1,98	640	425,71	78,99
Flask	186,8	65,65	1,75	280	546,2	127,48
aiohttp	165,07	247,16	1,98	585	1966,28	439,72
Sanic	194,8	170,85	1,98	470	1520,9	292,6
Tornado	279,3	101,71	1,74	272	778,75	222,07

Рисунок Б.6 – Слайд 6

Обґрунтування вибору програмних засобів для обробки природної мови

7

Бібліотека	Мова	Позитивні характеристики	Негативні характеристики
NLTK (Natural Language ToolKit)	Python	Найбільш відома і багатофункціональна бібліотека для NLP. Велика кількість сторонніх розширень. Швидка токенизація пропозицій. Підтримується безліч мов.	Повільна. Складна в вивченні і використанні. Працює з рядками. Чи не використовує нейронні мережі. Ні вбудованих векторів слів.
spaCy	CyPython	Найбільш швидка Проста в навчанні Працює з об'єктами а не строками Є вбудовані вектори слів Використовує нейронні мережі для навчання моделей	Менш гнучка ніж NLTK Токенизація повільніша ніж у NLTK Підтримує невелику кількість мов
scikit-learn	Python	Велика кількість алгоритмів для побудови моделей; Містить функції для роботи з Bag-of-Words моделлю; Гарна документація.	Поганий препроцесинг, що змушує використовувати її в зв'язці з іншою бібліотекою (наприклад, NLTK); Чи не використовує нейронні мережі для препроцесинга тексту.
gensim	Python	Працює з великими наборами даних; Підтримує глибоке навчання; word2vec, tf-idf vectorization, document2vec.	Реалізовані тільки моделі без вчителя; Не містить достатнього функціоналу, необхідного для NLP, що змушує використовувати її разом з іншими бібліотеками.

Рисунок Б.7 – Слайд 7

Реалізація системи

8

- ▶ 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.

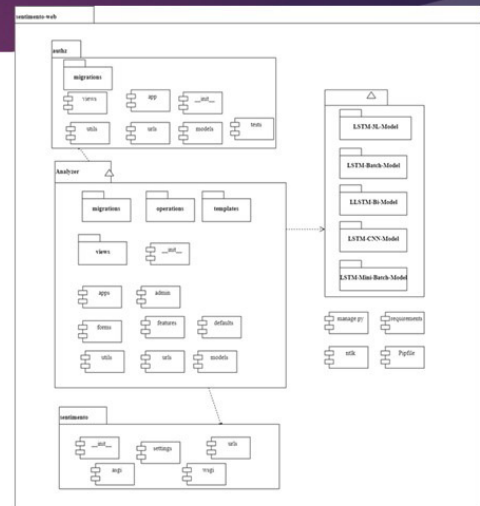


Рисунок Б.8 – Слайд 8

Интерфейс системи

9

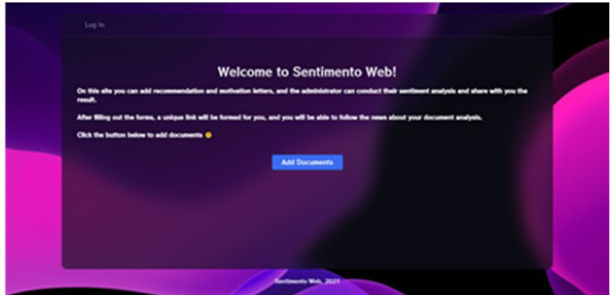


Рисунок 9.1

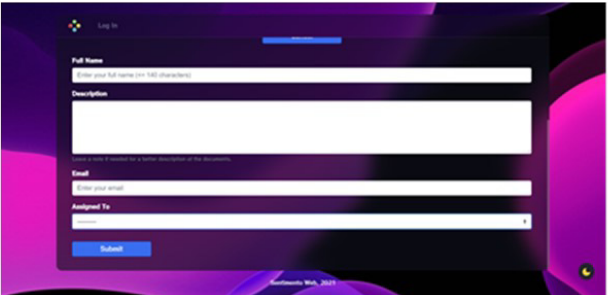


Рисунок 9.2

Рисунок Б.9 – Слайд 9

Интерфейс системи

10

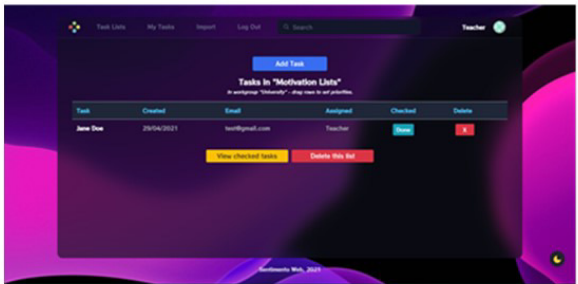


Рисунок 10.1

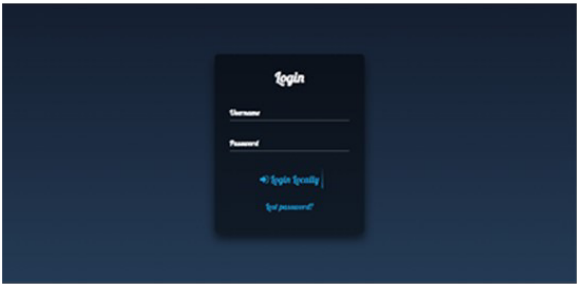


Рисунок 10.2

Рисунок Б.10 – Слайд 10

11

Обробка даних

uniqu e_id	asin	produ ct_na me	helpfu l	rating	title	date	review er	review er_loc ation	review _text	
count	16576	16576	16576	16576	16576	16576	16576	16576	16576	16576
uniqu e	15442	10905	10773	908	2	13330	1664	13427	5736	14726
top	B0000 5OLX1 :sucks _jason	B0009 9QAQ K	TIME [6- month subscr iption] [with \$5 Bonus] : ...		1.0		Disap pointe d	Janua ry 9, 2007	This was my first download of this type and a...	
freq	4	18	20	2748	10443	79	216	452	3572	18

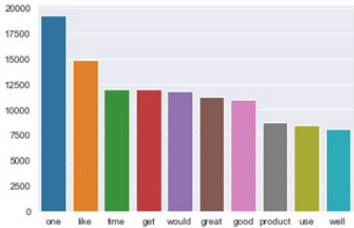


Рисунок 11.1



Рисунок 11.2

Рисунок Б.11 – Слайд 11

12

Класифікація

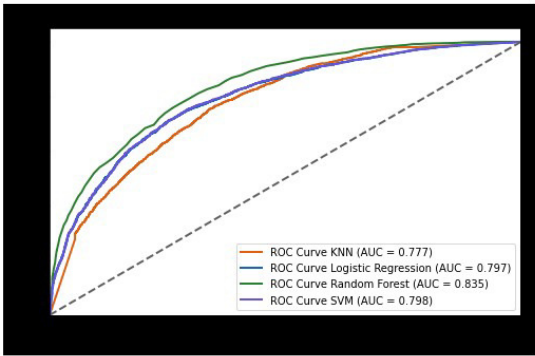
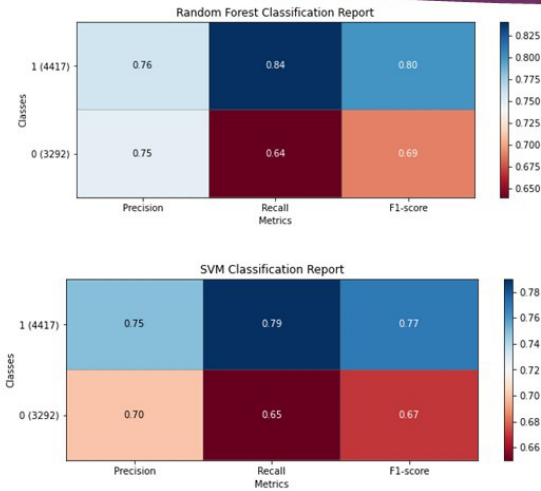


Рисунок Б.12 – Слайд 12

Обробка даних

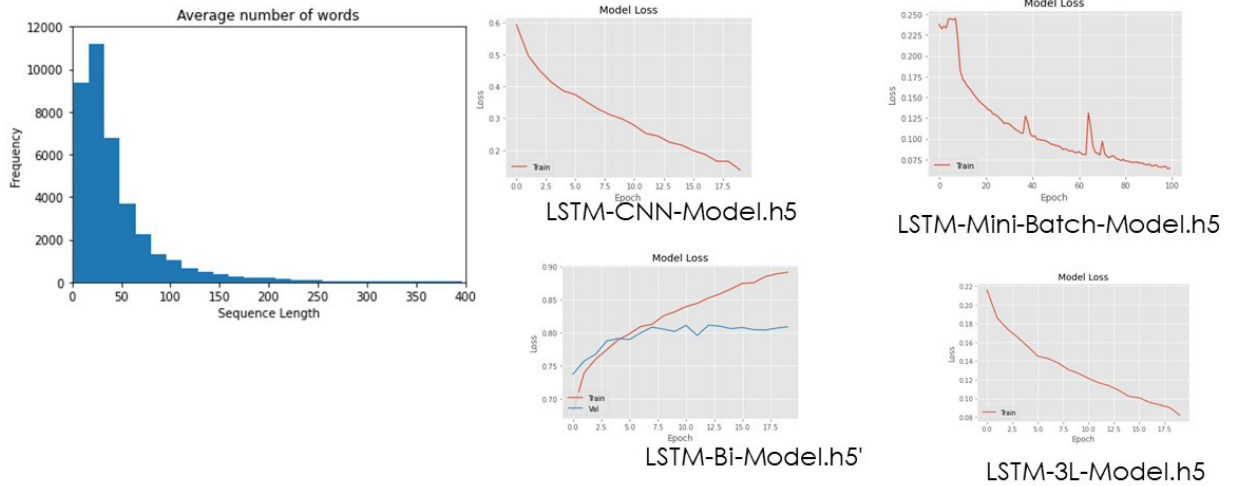


Рисунок Б.13 – Слайд 13

Навчання та створення моделей

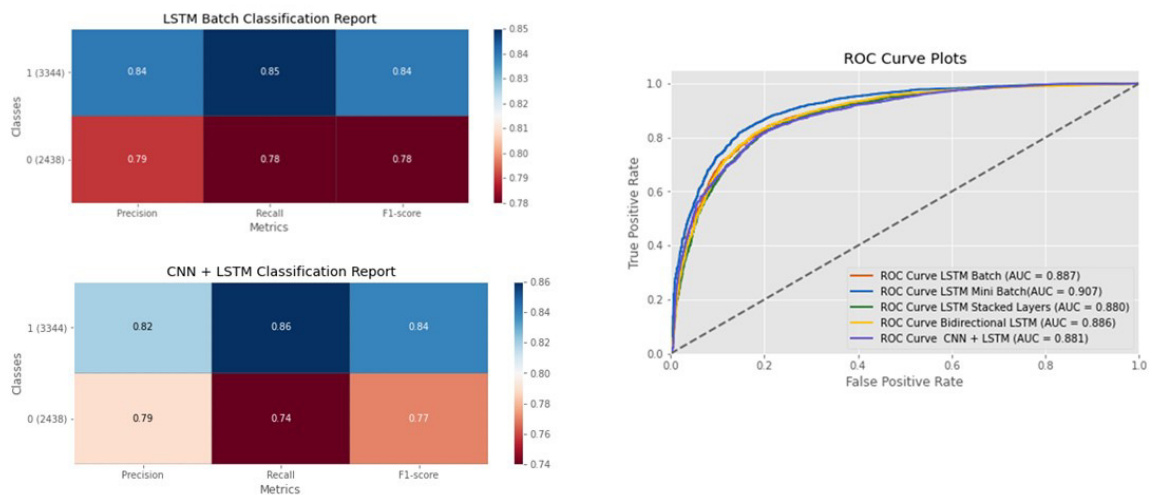


Рисунок Б.14 – Слайд 14

Висновки

- ▶ Були досліджені моделі для визначення емоцій в англomовних текстах з використанням бібліотек мови Python.
- ▶ Було отримано та досліджено 5 моделей для визначення тональності текстів. Всі отримані моделі показали приблизно однаковий результат, але поки що найкращою є модель з Mini Batch (90%), тому вона і була використана для веб застосування.
- ▶ Розроблено програмне рішення з використанням веб інтерфейсу, що дозволяє здобувачам міжнародної мобільності завантажувати відповідні документи, а співробітникам міжнародного відділу аналізувати їх емоційний стан.

Рисунок Б.15 – Слайд 15