Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра захисту інформації

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:
«Метод та засіб розпізнавання емоційного забарвлення повідомлення під час інформаційної війни»

Виконав: студент 2-го курсу, групи 1БС-20м
Спеціальності 125 Кібербезпека
_____Майстренко В. О.

Керівник: к. т. н., доц. каф. ЗІ

_____ Войтович О. П.

Опонент: к. т. н., доц. кафедри ОТ

_____ Войцеховська О.В.

Допущено до захисту
Завідувач кафедри ЗІ
д.т.н., проф.
_____Лужецький В.А.
«__» _____ 2021 р.

Вінниця ВНТУ - 2021 рік

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра захисту інформації
Рівень вищої освіти ІІ-й (магістерський)
Галузь знань 12 Інформаційні технології
Спеціальність 125 Кібербезпека
Освітньо-професійна програма – Безпека інформаційних і комунікаційних систем

ЗАТВЕРДЖУЮ
Завідувач кафедри ЗІ,
д.т.н., проф.
_____ В.А. Лужецький
«___» _____ 2021 року

З А В Д А Н Н Я
НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Майстренко В'ячеславу Олеговичу

1. Тема роботи: «Метод та засіб розпізнавання емоційного забарвлення повідомлення під час інформаційної війни» керівник роботи: Войтович Олеся Петрівна, к.т.н., доц. каф. ЗІ
затверджені наказом ректора ВНТУ від 9 вересня 2021 року №207.
2. Строк подання студентом роботи 21 грудня 2021 р.
3. Вихідні дані до роботи:
   - мова програмування – Python v3.8.5
   - метод розпізнавання емоційного контексту – нейронна мережа
   - мітки класів емоцій відповідно до повідомлень
4. Зміст текстової частини: Вступ. Аналіз методів визначення емоційного контексту. Структура застосунку для визначення емоційного контексту повідомлень. Алгоритм роботи застосунку для визначення емоційного контексту. Розробка та тестування застосунку визначення емоційного контексту. Висновки. Перелік інформаційних джерел. Додатки.
5. Перелік ілюстративного матеріалу: Структура застосунку для визначення емоційного контексту повідомлень (плакат, А4). Алгоритм роботи застосунку для визначення емоційного контексту (плакат, А4). Структура нейронної мережі визначення емоційного контексту (плакат, А4). Аналіз даних (плакат, А4). Аналіз функції витрат (плакат, А4). Результати тестування (плакат А4).

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанти	Підпис, дата	
		завдання видав	виконання прийняв
1	Войтович О. П., к. т. н., доцент каф. ЗІ		
2	Войтович О. П., к. т. н., доцент каф. ЗІ		
3	Войтович О. П., к. т. н., доцент каф. ЗІ		
4	Лесько О. Й., проф., зав. каф. ЕПВМ		

7. Дата видачі завдання 2021 року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз завдання. Вступ	02.09.2021-04.09.2021	
2	Аналіз літературних джерел за напрямком бакалаврської дипломної роботи	04.09.2021-15.09.2021	
3	Розробка технічного завдання	16.09.2021-22.09.2021	
4	Розробка рішень, моделей, алгоритмів	22.09.2021-15.10.2021	
5	Практична реалізація, моделювання, експериментування, результати	15.10.2021-25.10.2021	
6	Аналіз ТЗ, висновки	25.10.2021-30.10.2021	
7	Оформлення пояснювальної записки	30.10.2021-07.11.2021	
8	Попередній захист МКР	03.12.2021	
9	Виправлення зауважень, підготовка ілюстративного матеріалу	04.12.2021-16.12.2021	
10	Представлення МКР до захисту, рецензування	17.12.2021	
11	Захист МКР	21.12.2021-23.12.2021	

Студент _____ Майстренко В.О.

Керівник роботи _____Войтович О.П.

АНОТАЦІЯ

УДК 681.3.069

Майстренко В. О. Метод та засіб розпізнавання емоційного забарвлення повідомлення під час інформаційної війни. Магістерська кваліфікаційна робота зі спеціальності 125 – Кібербезпека, освітня програма – Безпека інформаційних і комунікаційних систем. Вінниця: ВНТУ, 2021. 110 с.

На укр. мові. Бібліогр.: 102 назв; рис.: 21; табл. 5.

Магістерська кваліфікаційна робота присвячена покращенню розпізнавання емоційного забарвлення повідомлення. Розглянуто існуючі підходи до семантичного аналізу, особливо точки зору емоційного контексту. Для подальшої реалізації за основу взято класифікатор BERT моделі, який запропоновано удосконалити шляхом передобробки повідомлення. Також запропоновано збір даних датасету з різних джерел. В результаті створено систему, що може опрацьовувати текстові повідомлення відопівідно вхідним даним, і визначати їх емоційне забарвлення. Проведене тестування показало достатню ефективність роботи створеної системи.

Графічна частина складається з 6 плакатів з демонстрацією результатів моделювання і проведених досліджень.

В економічному розділі оцінено витрати на розробку.

Ключові слова: сентиментальний аналіз, BERT, класифікатор тексту.

ABSTRACT

Dumchykov S. A. Method and means of recognizing the emotional color of the message during the information war. Master’s thesis in specialty 125 – Cybersecurity. Vinnytsia: VNTU, 2021. – 102 p.

In Ukrainian language. Bibliographer: 27 titles; fig.: 21; tabl.: 5.

The master's thesis is devoted to improving the recognition of the emotional color of the message. Existing approaches to semantic analysis, especially in terms of emotional context, are considered. For further implementation, the BERT model classifier is taken as a basis, which is proposed to be improved by pre-processing the message. Dataset data collection from various sources is also proposed. As a result, a system was created that can process text messages according to the input data and determine their emotional color. The conducted testing showed sufficient efficiency of the created system.

The graphic part consists of 6 posters demonstrating the results of modeling and research.

The economic section estimates the development costs.

Key words: sentimental analysis, BERT, text classifier.

ЗМІСТ

ВСТУП

Сьогодні, задачі обробки натуральної мови являються найбільш актуальними, особливо ті, які стосуються контексту емоційного забарвлення. Стрімкий розвиток соціальних мереж призвів до того, що рівень довіри до них став занадто високим і людьми стало значно простіше керувати через агітативні повідомлення.

Аналітика та моніторинг соціальних мереж представляє величезний інтерес для соціологів, лінгвістів, психологів, маркетологів та державних структур. Для вирішення завдань аналізу емоційного забарвлення тексту в комп'ютерній лінгвістиці використовуються методи контент-аналізу, загальна назва яких – Sentiment Analysis (аналіз тональності тексту).

Великі компанії використовують соціальні мережі для дослідження думок про свої продукти. Такий підхід, на відміну від опитувань на сайті виробника та роботи фокус-груп, забезпечує велику широту дослідження думок. В органах, що забезпечують державну безпеку, контент-аналіз використовується для фільтрації та виявлення повідомлень, що містять інформацію про протиправні дії (терористичні загрози, аналіз забороненого контенту на сайтах та ін). У соціологічних дослідженнях методи контент-аналізу дозволяють як показати поточне ставлення населення до об'єкту дослідження, а й прогнозувати зміни ставлення до об'єкту. Відповідно, актуальною є задача машинного аналізу текстових повідомлень, одним із варіантів якого є визначення емоційного контексту.

Будь-які задачі області обробки природньої мови є доволі складними, як в плані закладеного математичного апарату, так і в контексті обчислювальної складності [2].

Об’єктом дослідження є процеси, задіяні при розпізнавання емоційного забарвлення повідомлень.

Предметом дослідження є методи та засоби, що дозволяють здійснювати визначення тональності інформаційного повідомлення та протидії впливу.

Метою магістерської кваліфікаційної роботи є покращення розпізнавання емоційного забарвлення повідомлення під час інформаційної війни. Для досягнення поставленої мети слід виконати наступні задачі:

- провести аналіз існуючих методів розпізнавання емоційного забарвлення повідомлення;
- спроектувати загальну структуру засобу розпізнавання емоційного забарвлення повідомлення;
- розробити алгоритми роботи модулів, а також алгоритм тренування нейронної мережі і обробки даних для неї;
- реалізувати у вигляді програмного засобу описану структуру та алгоритми її модулів, провести тренування створеної мережі
- провести тестування розробленого засобу розпізнавання емоційного забарвлення, оцінити результати його роботи;
- оцінити перспективи розвитку дослідження та зробити висновки по роботі.

Практичне цінність роботи полягає у створенні засобу, який реалізує розроблений метод класифікації емоційного контексту з передобробкою повідомлень та придатний до використання як для задач корпоративного сектору так і для цілей державних структур. Наукова новизна магістерської роботи полягає в тому, що вдосконалено метод класифікації емоційного контексту з передобробкою повідомлень, що дозволяє визначити інформаційні операції під час ведення інформаційної війни.

1 АНАЛІЗ ЛІТЕРАТУРНИХ ДЖЕРЕЛ ЗА ТЕМОЮ РОБОТИ

1.1 Актуальність задачі

Задача визначення емоційного контексту повідомлень сьогодні є актуальною задачею. Разом із загальним розвитком технологій, різні бізнес гіганти почали боротьбу за клієнтів у напрямку клієнтоорієнтовності. Таким чином, такі компанії почали витрачати велику кількість грошей на дослідження технологій штучних мереж та їх застосуванню відповідно до завдань бізнесу.

Однією із таких технологій є обробка природньої мови – різних повідомлень, аудіозаписів, чатів і т.д., оскільки покращення клієнтоорієнтовності полягає у прямій взаємодії із клієнтами і для автоматизації цього сегменту необхідно застосовувати автоматизовані системи. Таким чином визначення емоційного контексту дозволяє з легкістю визначити настрій клієнта, підібрати для нього відповідну тактику діалогу та визначитись з доцільними репліками відовідей.

Однак емоційний контекст також відіграє велику роль і під час ведення інформаційних війн. Особливість інформаційних війн полягає у впливі на громадян іншої країни за допомогою різних провокаційних та інших повідомлень, які несуть в собі певний емоційний контекст. Разом зі зростанням популярності соціальних мереж, процес розповсюдження агітаційних та провокаційних повідомлення став неймовірно легким. Більшість користувачів відповідних соціальних мереж, які розповсюджують різні заклики являють собою автоматизовані системи – боти. Таким чином, виникла необхідність у автоматизованому вивченні контексту повідомлень, його обробки, аналізу і визначені кінцевої емоції та контексту, задля винесення висновку відносно можливого наміру адресата чи вектору атаки.

Задача визначення емоційного забарвлення повідомлення полягає у класифікації вхідних текстових послідовностей відповідно класам емоційного забарвлення. Зазвичай для вирішення цієї задачі, використовують апарат чітких зв’язних правил або апарат нейронних мереж.

1.2 Метод чітких зв’язних правил

Підхід, заснований на правилах, описує основні граматичні та логічні правила, яких слід дотримуватися, щоб виявити емоції з відповідних документів. Правила для невеликої кількості документів можуть бути легко створені, проте для більшого масиву даних вже можуть виникати труднощі. Підхід до побудови правил охоплює розпізнавання ключових слів (КС) та методи лексичної спорідненості.

Метод КС стосується побудови або використання словників емоцій чи лексиконів. Існує чимало словників КС, серед яких можна відзначити WordNet-Affect, EmoSenticNet, DepecheMood та лексикон Національної ради досліджень Канади (NRC) Ці лексикони емоцій містять слова для пошуку емоцій або такі ключові слова, як щасливі, ненависні, злі, сумні, здивовані тощо. Завдання полягає в тому, щоб знайти входження цих пошукових слів у письмовому тексті на рівні речення. Після того, як ключове слово ідентифіковано у реченні, реченню призначається відповідна мітка. Наприклад, якщо побудований словник емоцій містить радість, а письмовий текст, з якого слід визначити емоцію, звучить так: «Я був сповнений такої великої радості, побачивши свого батька вперше за десять років», реченню призначається відповідна мітка по ключовому слову радість. Хоча цей підхід простий і зрозумілий, він стикається з проблемами, включаючи потребу в словнику емоцій, який міститиме розумну кількість категорій емоцій, оскільки обмеженість ключових слів може значно вплинути на ефективність підходу серед неоднозначності ключових слів та відсутності мовної інформації.

Метод лексичної спорідненості (ЛС) доповнює метод КС, тому що окрім ідентифікації ключових слів випадковим емоційним словам призначається ймовірнісна спорідненість. ЛС відповідає за цей другий етап присвоєння ймовірнісних спорідненостей словам випадкових емоцій. Наприклад, слову «добре» може бути присвоєно ймовірне споріднення «позитивний», «сердитий» - спорідненння «негативне», тощо. Недолік, пов'язаний з цим імовірнісним призначенням спорідненості, полягає в тому, що він не повністю представляє

різні категорії емоцій, а навпаки, зводить їх до двох крайніх станів (тобто «позитивних» або «негативних»). Також такий підхід може призвести до неточностей у класифікації емоцій залежно від контексту призначених слів. Наприклад, речення «Зустріч з ним мала б бути гарною ідеєю», можливо, виражало розчарування автора від зустрічі з кимось і повинно бути кваліфіковане як «негативна» емоція, але це, як правило, вважається позитивною емоцією, оскільки слово «гарною» вже отримало імовірнісну спорідненість як "позитивний". Ці недоліки часто викликають необхідність використання інших підходів для виявлення емоцій у текстах.

1.3 Визначення емоційного контексту використовуючи апарат нейронних мереж

Далі, було розпочато аналіз рішень, що базуються на використанні нейронних мереж. Підхід ML вирішує проблему визначення емоцій, класифікуючи тексти за різними категоріями емоцій шляхом реалізації алгоритмів ML. Виявлення часто проводиться з використанням supervised або unsupervised технік машинного навчання. Дослідження Адома Ашемпонга[1] показало, що supervised алгоритми машинного навчання широко застосовуються у текстових задачах визначення емоцій і пропонують порівняно кращі показники виявлення, ніж у задачах, де впроваджувались unsupervised методи машинного навчання. Також у ході аналізу літератури було помічено, що широко досліджувані методи відносились також до класичних методів машинного навчання. Такими традиційними unsupervised методами машинного навчання (TUML) є: метод опорних векторів (SVM), наївний Байєс (NB), умовні випадкові поля (CRF) тощо. Однак такі методи не є достатньо ефективними і не витягують явно семантичну інформацію, що має значення для ефективного виявлення емоцій у текстах. Останнім часом supervised моделі глибокого навчання позиціонуються як ML підходи для виявлення емоцій з текстів. Пояснюється це тим, що методи глибокого навчання є більш ефективними і що

їхні глибокі шари можуть витягувати притаманні/приховані деталі, які можуть містити тексти.

Тріпто та Алі[2] запропонували модель глибокого навчання для виявлення міток настрою та емоцій з бенгальських коментарів YouTube. Після отримання даних вони попередньо обробили текст, щоб видалити стоп слова. Потім вони отримали вектори представлення слова, використовуючи як Skip-Gram, так і безперервний пакет слів (CBOW) у Word2Vec[3]. Потім вихідний сигнал подавався як вхідний сигнал до їх визначеної архітектури короткочасної пам’яті (LSTM) на першому етапі моделі, після чого на другому етапі складалася архітектура згорткової нейронної мережі (CNN). Результати їхнього дослідження показали, що вони застосовують глибокі методи навчання; LSTM і CNN значно перевершують традиційні методи боротьби з відхиленням, такі як SVM і NB, з точністю класифікації емоцій 59,2% та точністю міток настроїв у багатокласних (3 та 5) відповідно 65,97% та 54,24%. Запропоновані авторами архітектури показані на рисунках 1.1 та 1.2.

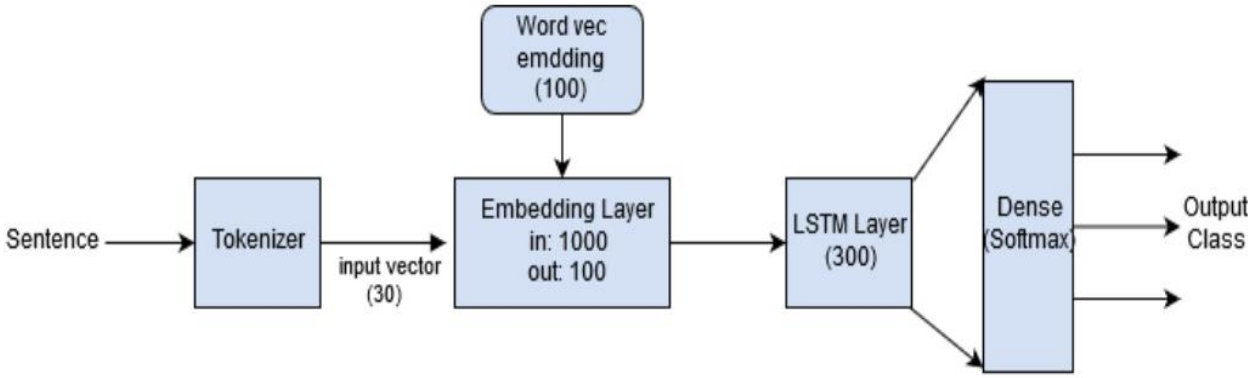


Рисунок 1.1 – LSTM модель для виявлення емоційного контексту повідомлень

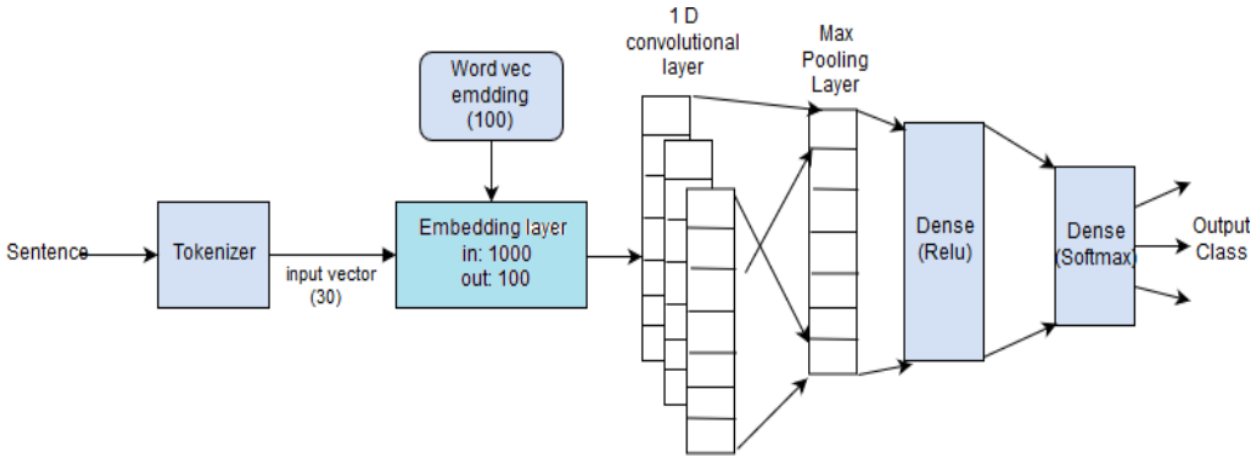


Рисунок 1.2 – CNN модель для виявлення емоційного контексту повідомлень

Абдулла та інші[4] виявили емоції в арабських твітах за допомогою моделі глибокого навчання CNN-LSTM. Вони провели два експерименти. Дослід 1 із застосуванням фази зворотного зв’язку та експеримент 2 із застосуванням фази CNN-LSTM. Вхідні дані для описаної системи були отримані з загальнодоступних наборів даних SemEval-2018 (Перший експеримент). Арабські твіти в розробленій системі використовувалися у двох формах: як оригінальні необроблені арабські твіти (ArTweets) або в перекладі на англійську (TraTweets), оскільки англійська мова має більше методів попередньої обробки та вилучення функцій, ніж арабська. До ArTweets і TraTweets було застосовано кілька кроків попередньої обробки. В першому експерименті вони спочатку подають 4908 розмірний вхідний вектор у мережу передачі вперед з повністю з'єднаними шарами та трьома прихованими шарами, що складаються з 500, 200 та 80 нейронів відповідно, і застосували функцію активації ReLU. На виході була функція Sigmoid для прогнозування настроїв та інтенсивності емоцій. Вони оптимізували цю мережу, використовуючи стохастичний градієнтний спуск (SGD). Другий експеримент передбачав подачу вхідного вектора 300 у модель CNN з 64 фільтрами, розміром ядра 3 та функцією активації ReLU. Потім вектори були передані в модель LSTM після застосування MaxPool розміром = 2. LSTM мав два приховані шари з 200 і 80 нейронами відповідно. Було застосовано функцію активації ReLU та Sigmoid,

як у експерименті 1, та оптимізатор SGD. Їх результати вказували на точність 40% для експерименту 1 та 60% для експерименту 2. Недоліком їхньої системи був невеликий обсяг використаних даних, а також невелика кількість використаних прихованих шарів. Загальна структура запропонованого у статті рішення показана на рисунку 1.3.

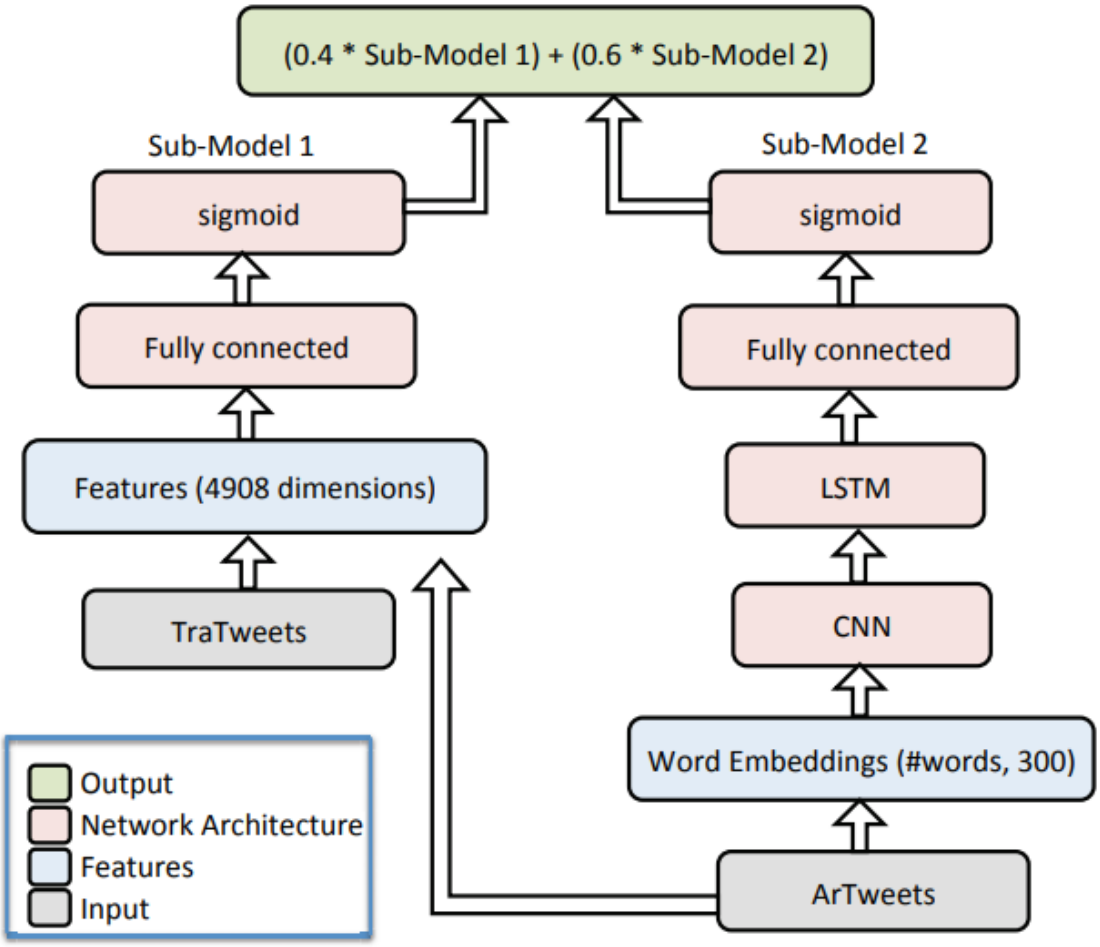


Рисунок 1.3 – Загальна архітектура моделі обробки арабських твітів

Чаттерджі та інші [5] брали участь у конкурсі SemEval-2019 Task 3. Вони виявили контекстуальні емоції з текстових діалогів і класифікували їх в один із чотирьох класів емоцій як щасливі, сумні, злі та інші за допомогою довгострокової короткочасної пам'яті (LSTM) і отримали оцінку F1 0,5861. Далі вони порівняли та обговорили методи, які досягли більшої міри F1, і зрозуміли, що такі системи реалізовані з використанням Bi-LSTM. Cai та Hao78 у своїй

роботі намагалися виявити емоції від Weibo, популярного китайського сайту для мікроблогів, використовуючи мультиперегляд та орієнтовану на увагу Bi-LSTM. Автори підходили до проблеми на трьох рівнях, знаходячи емоції зі слів емоцій, смайлів та витягуючи смислові або приховані вирази емоцій. Спочатку вони отримують тексти Weibo, потім попередньо обробляють тексти за допомогою інструменту сегментації слів Jieba, щоб видалити стоп-слова, після чого слова були позначені та закодовані за допомогою одноразового кодування. Потім слова були перетворені у вектори та навчені за допомогою Word2Vec, використовуючи модель пропуску грамів. Потім векторні уявлення були інтегровані у шар у моделі Bi-LSTM та з'єднані з шаром softmax для отримання вектору розподілу ймовірностей. Навчання проводилося з вхідним виміром 50. З метою підвищення швидкості навчання було використано пакетне навчання з розміром партії 256, а метод наповнення - для підтримки постійного розміру партії. Для навчання використовувалося зворотне розповсюдження, а втрата перехресної ентропії була реалізована як функція втрат. Набір даних оцінки NLP & CC 2013 був використаний для визначення того, чи містить текст емоцію чи інше. Якщо він містив емоцію, її класифікують у визначені категорії емоцій, інакше її відкидають. У своїх результатах вони визначили Bi-LSTM AVE як результати підсумовування виводу текстів, представлених у Bi-LSTM, за яким слідує повний шар з'єднання та функція м'якого максимуму, Bi-LSTM-AVE-MV як Bi -LSTM на основі механізму мультиперегляду та уваги, Bi-LSTM-V1 як кут емоційного слова, Bi-LSTM-V2, кут смайликів, Bi-LSTM-V1-V2 як суміш емоційних слів та смайлів, Bi-LSTM-AVE -V2 як злиття чистої семантики та смайликів. Потім автори дійшли висновку, що оскільки макросередовище та мікросередовище оцінки FI методу Bi-LSTM-AVE-MV були на 7% вищими за середні показники Bi-LSTM-AVE. А макросередовища та мікросередовища балу FI методу Bi-LSTM-AVE-MV на 17% вищі за середні значення Bi-LSTM-V1 та Bi-LSTM-V2, суто семантичний рівень служив представленням весь текст мікроблогів та сприяв глобальному розумінню тексту, що відображає важливість семантики в текстовій ЕД.

Найбільш сучасним методом обробки тексту вважаються моделі трансформери [6]. Трансформери в обробці природньої мови – це нова архітектура, яка має на меті вирішувати завдання послідовність-послідовність (Seq2Seq), з легкістю обробляючи «глибокі» залежності, наприклад коли у реченні перші слова в речені тісно пов'язані за сенсом з останніми. Трансформери – це перша модель трансдукції, яка повністю покладається на власну увагу, щоб обчислити уявлення про свої вхідні та вихідні дані без використання реккурентних послідовностей(як наприклад в RNN або LSTM) або операцій згортки(CNN). Трансдукція означає перетворення вхідних послідовностей у вихідні. Ідея трансформерів полягає в тому, щоб повністю обробляти залежності між входом і виходом з увагою та повторенням.

Більшість звичайних моделей перетворення нейронної послідовності мають структуру енкодер-декодер. Енкодер відображає вхідну послідовність представлень символів$(x_1, \ldots, x_n)$ на послідовність безперервних подань $z = (z_1, \ldots, z_n)$. З огляду на z, декодер потім генерує вихідну послідовність$(y_1, \ldots, y_m)$ символів по одному елементу за раз. На кожному кроці модель має авторегресію, споживаючи раніше створені символи як додатковий вхідний сигнал при генерації наступного. Трансформатор слідує цій загальній архітектурі, використовуючи накладену увагу до себе та точкові, повністю з'єднані шари як для енкодера, так і для декодера, показані на лівій та правій половинах на рисунку 1.4 [6].

Енкодер складається з стека з $N = 6$ однакових шарів. Кожен шар має два підшари. Перший – це багатовхідний механізм самоконтролю, а другий – проста, повністю з'єднана нейронна мережа.

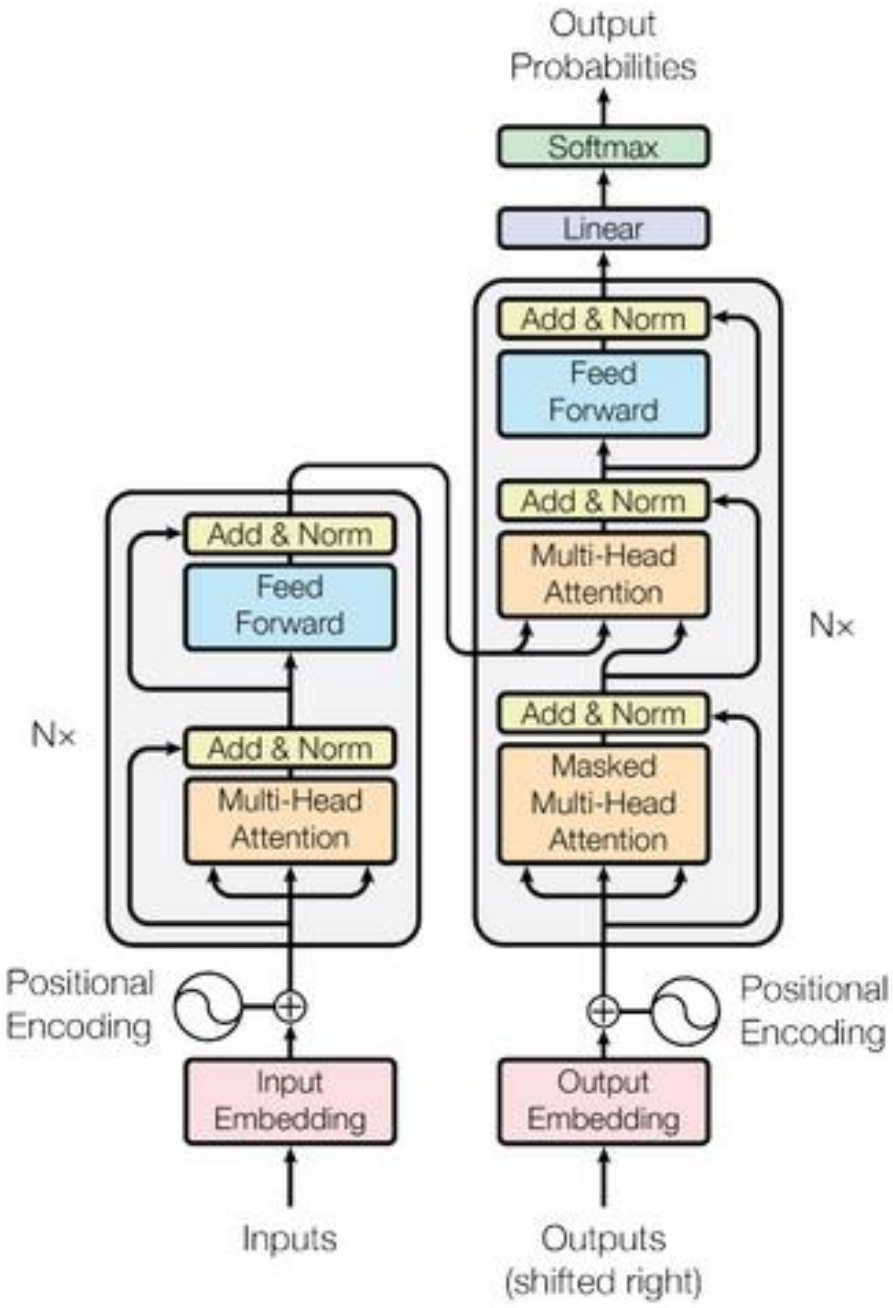


Рисунок 1.4 – Загальний вигляд архітектури трансформера

В описаній архітектурі використовуються залишкові з’єднання(Residual connections) навколо кожного з двох підшарів з наступною нормалізацією шару. Таким чином, результатом кожного підшару є:

$$\text{LayerNorm}\ (x\ +\ \text{Sublayer}\ (x))$$

,де Sublayer (x) – це функція, реалізована самим підшаром.

Для полегшення цих залишкових з'єднань усі підшари в моделі, а також вбудовуючі шари створюють результати виміром $d_{model} = 512$. Декодер

також складається з стека з $N = 6$ однакових шарів. На додаток до двох підшарів у кожному рівні енкодера, декодер вставляє третій підрівень, який виконує багатовхідну увагу над виведенням стека кодера. Подібно до енкодера, залишкові з'єднання використовуються навколо кожного з підшарів з наступною нормалізацією шару. Також, використовується модифікований підшар самоуваги(self-attention) в стеку декодера, щоб запобігти переходу позицій до наступних. Це маскування в поєднанні з тим, що вихідні вбудовування зміщуються на одну позицію, гарантує, що передбачення для позиції і можуть залежати лише від відомих виходів у позиціях, менших за і.

Функція уваги може бути описана як відображення запиту та набору пар ключ-значення до виводу, де запит, ключі, значення та вихід-це всі вектори. Результат обчислюється як зважена сума значень, де вага, призначений кожному значенню, обчислюється функцією сумісності запиту з відповідним ключем.

Функція уваги, описаної у дослідженні має назву "Scaled Dot-Product Attention", а загальний вигляд цієї функції показаний на рисунку 1.5 [6].

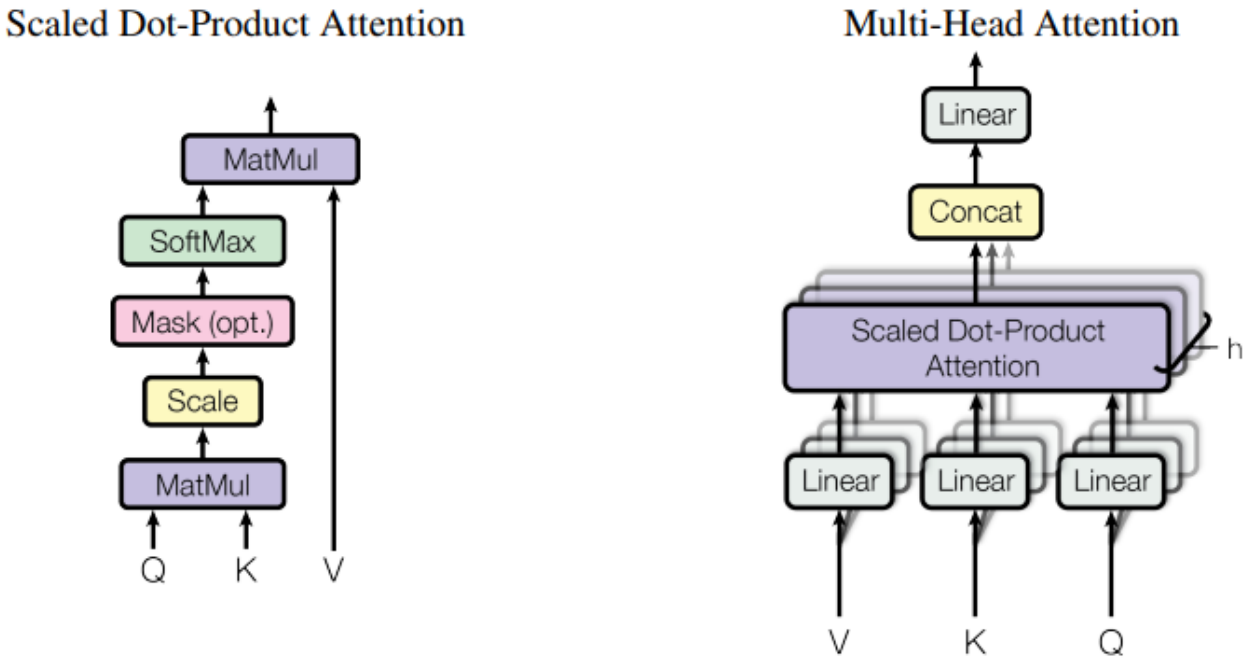


Рисунок 1.5 – функція Scaled Dot-Product Attention (зліва), Багатовхідний шар уваги (справа)

Вхідні дані містять запити та ключі розміру d_k та значення розмірності dv. Спочатку обраховуються добутки матриць запиту з усіма ключами, далі кожен ділиться на $\sqrt{d_k}$ і застосовується функція $softmax$ для отримання ваг значень. На практиці, обчислення функції уваги для набору запитів відбувається одночасно, упакованих разом у матрицю Q. Ключі та значення також упаковуються разом у матриці K та V. Обчислення матриці вихідних даних відбувається за наступною формулою:

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right) * V$$

Дві найбільш часто використовувані функції уваги – це адитивна увага та добуток матриць(мультиплікативна) увага. Увага добутку матриць ідентична дослідженому алгоритму, за винятком коефіцієнта масштабування $\frac{1}{\sqrt{d_k}}$. Додаткова увага обчислює функцію сумісності, використовуючи мережу передачі вперед з одним прихованим шаром. Хоча вони схожі за теоретичною складністю, увага до точкового продукту на практиці набагато швидша та більш простора, оскільки її можна реалізувати за допомогою високооптимізованого коду множення матриць. Хоча для малих значень d_k два механізми діють однаково, адитивна увага перевершує увагу продуктової точки без масштабування для більших значень d_k.

Найбільш відомою і ефективною моделлю, яка використовує архітектуру трансформерів є модель BERT(Bidirectional Encoder Representations from Transformers) [7]. Вона призначена для попередньої підготовки глибоких двонаправлених представлень тексту без міток шляхом спільного кондиціонування як лівого, так і правого контексту. В результаті попередньо навчена модель BERT може бути точно налаштована лише одним додатковим вихідним шаром для створення найсучасніших моделей для широкого спектра завдань НЛП.

Існує два варіанти архітектури BERT:

1) BERT Base: 12 шарів (блоків трансформерів), 12 attention heads та110 мільйонів параметрів;
2) BERT Large: 24 шари, 16 attention heads та 340 мільйонів параметрів.

Обидві моделі використовують тільки трансформери-енкодери. Модель BERT також вимагає специфічну передобробку даних під час тренування та валідації. Кожен вхідний вектор слів повинен є комбінацією з трьох векторів:

1) Position Embeddings: BERT вивчає та використовує позиційні вбудовування для вираження позиції слів у реченні. Вони додаються для подолання обмежень Трансформатора, який, на відміну від RNN, не здатний вловлювати інформацію про "послідовність" або "замовлення";
2) Segment Embeddings: BERT також може приймати пари речень як вхідні дані для завдань (питання-відповідь). Ось чому він вивчає унікальний вектор для першого та другого речень, щоб допомогти моделі розрізняти їх;
3) Token Embeddings: це вектори для конкретної лексеми зі словника токенів WordPiece.

Для даного входу його вхідне подання будується шляхом підсумовування відповідних вбудованих лексем, сегментів та позицій. Така схема вбудовування містить багато корисної інформації для моделі. Ці комбінації кроків попередньої обробки роблять BERT таким універсальним. Це означає, що без істотних змін в архітектурі моделі ми можемо легко навчити її виконанню декількох видів завдань НЛП. Загальних вигляд передоброблених даних, що подаються до моделі BERT під час тренування показано на рисунку 1.6.

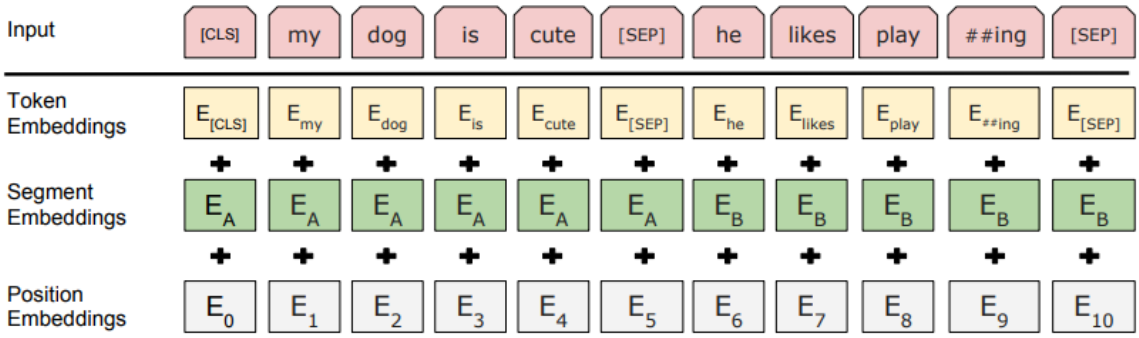


Рисунок 1.6 – Передобробка вхідного речення для моделі BERT

Андреа Кьоріні та інші[8] у своєму досліджені розробили модель для емоційного та семантичного аналізу речень використовуючи BERT як основу. Модель побудована шляхом точного налаштування BERT на конкретних наборах даних твітів, розроблених для подібних завдач. Оскільки твіти зазвичай містять слова, які не мають значення для класифікації тексту, було проведено попередню обробку тексту для того, щоб видалити:

1) Згадування інших користувачів: користувачі часто цитують інші імена користувачів Twitter у своїх твітах через символ «@», щоб направляти свої повідомлення;
2) URL адреси: URL адреси дуже поширені у твітах, як для засобів масової інформації (тобто зображень та відео), так і для посилань на інші веб -сторінки;
3) Ретвіти: користувачі часто надсилають твіти, які вважають актуальними для своїх послідовників. Ретвіти зазвичай позначаються префіксом "RT" і тому їх легко ідентифікувати.

Після фази попередньої обробки дані можуть бути використані як вхідні дані для навчання моделей на основі BERT для конкретних завдань. Архітектура загальної моделі BERT складається з серії двонаправлених багатошарових трансформаторів на основі кодера. В даний час доступно кілька попередньо навчених моделей BERT. Менші моделі призначені для середовищ з обмеженими обчислювальними ресурсами, оскільки більші моделі мають велику кількість доступні для навчання параметри: модель середнього розміру,

така як BERT-Base, має приблизно 110 мільйонів параметрів, що піддаються навчанню, тоді як BERT-Велика має більше 340 мільйонів параметрів. Зокрема, еталонною моделлю, використаною у цій роботі, є BERTBase, як у версії без літери, так і у верхній лінії. Версія без літер передбачає, що текст перетворюється на малі перед процесом токенізації слова, а наголоси ігноруються. Архітектура моделі BERT-Base складається з 12 енкодерів, кожен з яких складається з 8 шарів: 4 багатошарові шари самоконтролю та 4 шари подачі вперед. В досліджені було розширено таку модель, додавши повністю з'єднаний шар та шар softmax для класифікації. Архітектура є спільною як для класифікаторів настроїв, так і для емоцій: єдина відмінність між двома моделями представлена останнім шаром softmax, в якому кількість нейронів дорівнює кількості класів (тобто 3 для аналізу настроїв і 4 для розпізнавання емоцій). Загальна архітектура розробленої дослідниками моделі показана на рисунку 1.7.

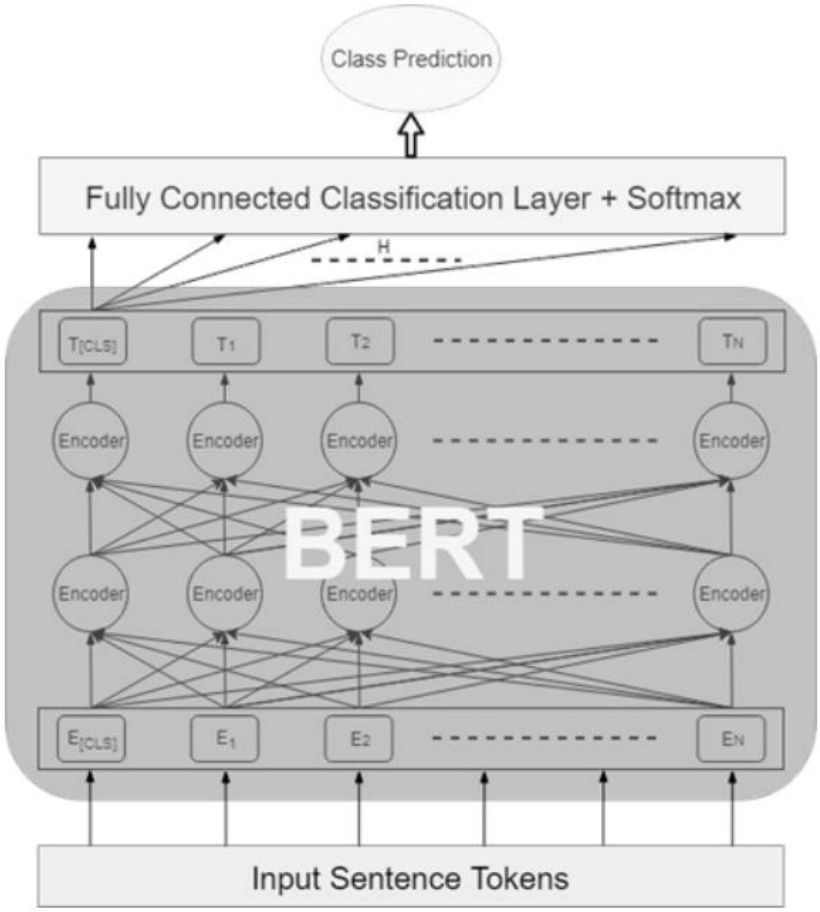


Рисунок 1.7 – Архітектура описаної у досліджені Кьоріні моделі

Продуктивність описаних класифікаторів була виміряна на двох наборах даних твітів, отримана точність склала 92% для аналізу настроїв та 90% для аналізу емоцій, з чого було зроблено висновок, що потужність моделювання BERT значно сприяє досягненню хорошої точності класифікації тексту.

1.4 Постановка завдання

Ключовими теоретичними питаннями під час побудови засобу розпізнавання емоційного засобу є збір даних, аналіз даних, підготовка даних для навчання моделі, розробка архітектури нейромережевої моделі для вирішення задачі, розробка алгоритму тренування для створеної моделі, отримання та валідація результатів роботи нейромережі. Також, оскільки досліджувана задача розпізнавання емоційного забарвлення поставлена в контексті роботи в умовах інформаційної війни, є доцільним розробка моделі саме для російської мови.

В даному розділі було проведено аналіз існуючих методів покращення якості відбитку, а також сформовано задачі, необхідні для реалізації мети, які будуть виконуватись у наступних розділах.

2 ПРОЕКТУВАННЯ ЗАСОБУ РОЗПІЗНАВАННЯ ЕМОЦІЙНОГО ЗАБАРВЛЕННЯ

Метою магістерської роботи є розробка засобу розпізнавання емоційного забарвлення повідомлення. Задача передбачає різну структуру та довжину повідомлень, також різну граматику та пунктуацію написання, а отже вимагає нестандартного підходу до розробки кінцевого рішення. Таким чином, розроблений засіб включає в себе як алгоритми на основі чітких правил, математичні алгоритми та нейромережеві рішення. Також, у зв'язку з тим, що задача передбачає роботу з текстовими даними, які є досить об'ємними для оперативної пам'яті комп'ютера, необхідно передбачити можливість паралелізації даних та використання графічних процесорів.

2.1 Опис використовуваних технологій

2.1.1 Міра TF-IDF

TF-IDF(TF – term frequency, IDF – inverse document frequency) – статистична міра, яка використовується для оцінки важливості слова в контексті документу відносно усього набору документів.[9] Вага певного слова являється пропорційною частоті використання цього слова в документі і є зворотньо пропорційним до частоти використання слова серед усього набору документів. Таким чином, міра TF-IDF часто використовується в задачах аналізу тексту та при розрахунку міри схожості документів при кластеризації.

Формула включає в себе TF та IDF частини. TF – відношення числа входження слова до загальної кількості слів документу, іншими словами обчислюється важливість слова в рамках одного документу. Формула розрахунку значення TF наведена нижче[10]:

$$tf(t,d) = \frac{n_t}{\sum_k n_k},$$

де n_t – число входження слова t в документ, знаменник – загальна кількість слів в даному документі.

IDF – інверсія частоти з якою слово зустрічається в документах колекції. Врахування даного показника дозволяє зменшити вагу слів, які широко використовуються. Для кожного унікального слова з колекції документів існує лише один показник IDF. Формула розрахунку IDF наведена нижче [10].

$$idf(t, D) = \log\left(\frac{D}{\{d_i \in D \mid t \in d_i\}}\right),$$

де D – число документів в колекції, $\{d_i \in D \mid t \in d_i\}$ – число документів із колекції D, яких зустрічається слово t.

Таким чином, міра TF-IDF є добутком двох множників [10]:

$$tfidf(t, d, D) = tf(t, d) \times idf(t, D).$$

Слова з високою частотою в межах конкретного документу і з маленькою частотою використання в інших документах, відповідно до формули, отримають більшу вагу.

2.1.2 Алгоритм K-means

K-means – алгоритм машинного навчання без вчителя, який надає можливість виявити схожі групи (кластери) в даних [11]. Вхідними даними алгоритму повинна бути таблиця з числовим форматом, а також кількість кластерів; вихідними даними будуть номери кластерів відповідно до кожного рядку вхідної таблиці. Робота алгоритму полягає у мінімізації суми квадратичного відхилення точок даних від центрів цих кластерів [11]:

$$V = \sum_{i=1}^{k} \sum_{x \in S_i} (x - \mu_i)^2,$$

де k – число кластерів, S_i – отримані кластери, $i = 1,2, \dots, k$, μ_i – центри мас усіх векторів x із кластера S_i.

Загальний алгоритм роботи K-means, наведено нижче в рисунку 2.1.

Вхідні дані:
X = $\{n_1, n_2, \dots, n_i\}$: набір елементів, вхідна таблиця, де t – рядок даних;
K: кількість кластерів;

Вихідні дані:
S: набір кластерів

1 Випадково вибрати K елементів з X, в якості стартових центроїдів;
2 **Повторити:**
3 Присвоїти кожному елементу $n_i \in X$ номер найближчого по відстані кластеру;
4 Обчислити нове середнє значення для кожного кластеру;
5 Присвоїти центроїдам відповідні нові середні значення;
6 **Допоки** критерій збігу не задоволено

Рисунок 2.1 – Алгоритм роботи K-means

Критерій збігу визначається функцією оптимізації. Найбільш часто використовуваною є Евклідова відстань. Відповідно до даної функції, елементу вхідних даних присвоюються номер кластеру, який знаходиться найближче до нього, іншими словами, кластеру з мінімальною Евклідовою відстанню. Формула Евклідової відстані наведена нижче [12]:

$$J = \sum_{i=1}^{N} \sum_{k=1}^{K} r_{ik} \|x_i - \mu_k\|^2,$$

де J – сума квадратів відстаней між кожний елементом $x \in X$ та присвоєним кластером; r – функція, значення якої рівне 1, якщо елемент знаходиться в кластері $k \in K$.

У випадку текстових даних, їх необхідно спочатку перетворити в числові вектори, скільки алгоритм приймає на вхід лише дані числового типу. Таким чином, необхідно використати TF-IDF для векторизації тексту.

Іншим важливим параметром є кількість кластерів. Для визначення найкращої кількості кластерів існує «метод ліктів» (Elbow method) [13]. Суть полягає в обрахунку точності кластеризації для різної кількості кластерів. Після цього будується графік залежності точності та кількості кластерів і на ньому можна побачити точку зі збалансованим значенням – саме вона і є ідеальною кількістю кластерів. Графік цього методу показано на рисунку 2.2, на ньому точка 3 – є оптимальним числом кластерів.

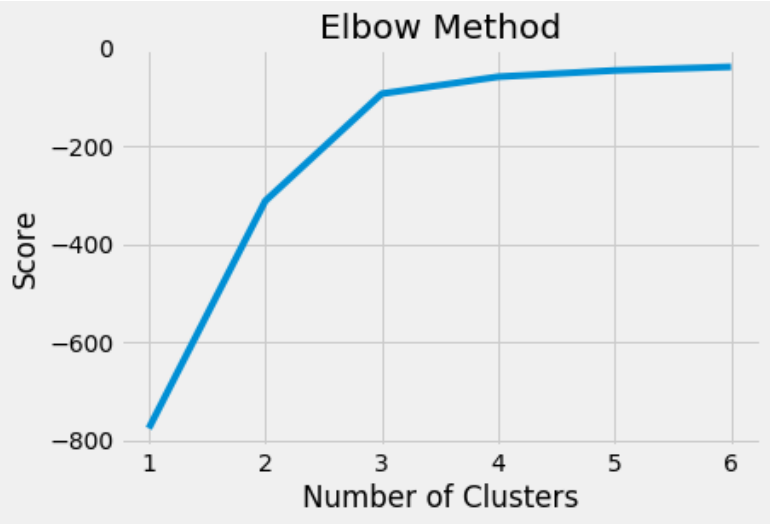


Рисунок 2.2 – Метод ліктів для вибору числа кластерів

Таким чином, даний метод дає змогу розбити неорганізовані вхідні дані на чіткі кластери, які містять схожі між собою дані.

2.1.3 Прихований розподіл Діріхле

Прихований розподіл Діріхле (LDA) – алгоритм, який використовується для задач тематичного моделювання (Topic Modelling), під такими задачами розуміються кластеризація та класифікація – таким чином, що кожен клас або кластер містить в собі тексти зі схожими темами [14]. Отже, використовуючи

даний алгоритм можна розподілити нерозмічені дані за спільними по сенсу категоріями.

Для того, щоб застосувати до датасету текстів (корпусу текстів) алгоритм LDA необхідно перетворити корпус в терм-документну матрицю (term-document matrix). Терм-документна матриця – це матриця розміру $N \times W$, де N – кількість документів в корпусі, а W – розмір словника корпусу, тобто кількість унікальних слів, які зустрічаються в нашому корпусі. В i-му рядку, j-му стовпцю матриці знаходиться число, яке являється кількістю того, скільки разів в i-му тексті зустрічалось j-те слово[15].

2.1.4 Метод t-SNE

T-розподілене вкладення стохастичної близькості(t-SNE) – це алгоритм машинного навчання для візуалізації даних і являє собою метод нелінійного зниження розмірності шляхом вкладення багатовимірних даних у двох або трьох вимірний простір для подальшої візуалізації [16]. Даний алгоритм відображає кожну точку багатовимірного простору в двох або трьох вимірну точку евклідового простору таким чином, щоб розташувати поруч схожі між собою об’єкти, а інші, несхожі об’єкти, відповідають віддаленим точкам з високою імовірністю.

Для початку необхідно описати задачу та алгоритм роботи класичного методу SNE. Припустимо, що у нас є набір даних з точками, які описуються багатовимірною змінною з розмірністю простору більше трьох. Необхідно отримати нову змінну, існуючу в двомірному або тривимірному просторі, яка б в максимальній мірі зберігала структуру та взаємозв’язок в даних. Робота SNE розпочинається з перетворення багатовимірної евклідової відстані між точками в умовні вірогідності, які відображаються схожість точок. Математичну формулу цього етапу наведено нижче [17]:

$$p_{j|i} = \frac{exp(- \parallel x_i - x_j \parallel^2 / 2\sigma_i^2)}{\sum_{k \neq i} exp(- \parallel x_i - x_k \parallel^2 / 2\sigma_i^2)}$$

Дана формула показує наскільки точка X_j близька до точки X_i при гаусовому розподілі навколо X_i із заданим відхиленням σ. Параметр сігма буде мати різні значення для кожної точки, величина вибирається таким чином, щоб точки з більшою щільністю мали меншу дисперсію. Для цього використовується оцінка перплексії [17]:

$$Perp(P_i) = 2^{H(P_i)},$$
$$H(P_i) = -\sum_j p_{j|i} \log_2 p_{j|i}$$

Для двомірних або тривирніих пар значень Y_i та Y_j , відповідним парам X_i та X_j, умовну вірогідність можна оцінити за допомогою ідентичної формули[17]:

$$q_{j|i} = \frac{exp(- \parallel y_i - y_j \parallel^2)}{\sum_{k \neq i} exp(- \parallel y_i - y_k \parallel^2)}$$

У випадку, якщо точки відображення Y_i та Y_j коректно моделюють схожість між початковими точками високої розмірності X_i та X_j, то відповідні їм умові вірогідності $p_{i|j}$ та $q_{i|j}$ будуть еквівалентні. В якості оцінки якості з якою $p_{i|j}$ відображає $q_{i|j}$ використовують дивергенцію або відстань Кульбака-Лейблера. SNE мінімізує суму таких відстаней для всіх точок відображення за допомогою градієнтного спуску. Функцією витрат в такому випадку, визначається наступною формулою [17]:

$$Cost = \sum_i KL(P_i \parallel Q_i) = \sum_i \sum_j p_{j|i} log \frac{p_{j|i}}{q_{j|i}}$$

При цьому, градієнт даної функції витрат має наступний вигляд [17]:

$$\frac{\partial Cost}{\partial y_i} = 2\sum_j (p_{j|i} - q_{j|i} + p_{i|j} - q_{i|j})(y_i - y_j)$$

Таким чином, загальна алгоритмічна формула для пошуку рівноваги враховує моменти і має наступний вигляд [17]:

$$Y^{(t)} = Y^{(t-1)} + \eta\frac{\partial Cost}{\partial Y} + \alpha(t)\left(Y^{(t-1)} - Y^{(t-2)}\right),$$

де η – параметр, визначаючий швидкість навчання (довжину кроку);

α– коефіцієнт інерції.

Результат роботи SNE дає непогані результати, однак можуть виникати проблеми, пов'язані зі складністю оптимізації функції витрат та проблемою сукупності (crowding problem). Алгоритм t-SNE значно полегшує вирішення даних проблем, функція витрат t-SNE має дві суттєві відмінності. По-перше, t-SNE має симетричну форму збіжності в багатовимірному просторі та більш простий варіант градієнта. По-друге, замість гаусового розподілу для точок використовується t-розподіл (Стьюдента).

В якості альтернативи мінімізації суми дивергенцій Кульбака-Лейблера, між умовними вірогідностями $p_{i|j}$ та $q_{i|j}$ використовується одинарна дивергенція між спільною вірогідністю P в багатовимірному просторі та спільною вірогідністю Q в просторі відображення [17]:

$$Cost = KL(P \parallel Q) = \sum_i \sum_j p_{ij} log\frac{p_{ij}}{q_{ij}},$$

де p_{ij} та q_{ij}= 0, p_{ij} = q_{ij},

$q_{ij} = q_{ij}$для будь-яких i та j,

p_{ij} визначається за допомогою формули [17]:

$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2n},$$

де n – кількість точок в наборі даних.

Градієнт для симетричного SNE в такому випадку буде мати наступний вигляд [17]:

$$\frac{\partial Cost}{\partial y_i} = 4\sum_j (p_{ij} - q_{ij})(y_i - y_j)$$

Проблема скупченості полягає в тому, що відстань між двома точками у просторі відображення, що відповідають двом середньовіддаленим точкам у багатовимірному просторі, має бути істотно більшою, ніж відстань, яка дозволяє отримати розподіл гауса. Проблему вирішують хвости Стьюдента. У t-SNE використовується t-розподіл із одним ступенем свободи. Спільна ймовірність простору відображення, і відповідно градієнт, в цьому випадку визначатиметься за наступними формулами [17]:

$$q_{ij} = \frac{(1+\| y_i - y_j \|^2)^{-1}}{\sum_{k \neq l}(1+\| y_k - y_l \|^2)^{-1}}$$

$$\frac{\partial Cost}{\partial y_i} = 4\sum_j (p_{ij} - q_{ij})(y_i - y_j)(1+\| y_i - y_j \|^2)^{-1}$$

Загальний алгоритм роботи методу t-SNE, можна представити псевдокодом показаним на рисунку 2.3

Вхідні дані:

X = $\{x_1, x_2, \dots, x_n\}$: набір елементів;
Loss: функція витрат, перплексія Perp;
T, nu, alpha: параметри оптимізації;

Вихідні дані:

Y(T): представлення даних $\{y_1, y_2, \dots, y_n\}$

---

1 обрахувати попарну схожість p_{ij} з Perp

2 присвоїти $p_{ij} = \left(p_{j|i} + p_{i|j}\right)/2n$

ініціалізувати Y(0) = $\{y_1, y_2, \dots, y_n\}$ (mean=0, sd=1e-4)

2 **Поки t = 1 до T:**

3	обрахувати схожість точок в просторі q_{ij}
4	обрахувати градієнт $\delta Loss/\delta y$
5	присвоїти $Y(t) = Y(t-1) + \eta * \delta Loss/\delta y + \alpha(Y(t-1) - Y(t-2)$
6	**Повторити**

Рисунок 2.3 – Загальний алгоритм роботи t-SNE

2.1.5 Метрики класифікації

Точність – це базова класифікаційна метрика. Найпростіша метрика, яка підходить як для бінарної, так і для багатокласової задачі класифікації. Результат метрики розраховується за формулою [18]:

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN},$$

де TP – кількість істинно позитивних передбачень, TN – кількість істинно негативних передбачень, FP – кількість хибно позитивних передбачень, FN – кількість хибно негативних передбачень.

Однак, простота є причиною частої критики і також причиною чому вона може абсолютно не підійти під вирішуване завдання. Вона не враховує співвідношення помилкових спрацьовувань моделі, що може бути критичним, особливо в медичній сфері, коли постає завдання розпізнати всі справжні випадки діагнозу. Таким чином, як приклад у випадку класифікації захворювань, якщо точність дорівнює 80%, то можна сказати, що в середньому зі 100 осіб вона правильно визначить наявність або відсутність діагнозу лише у 80 осіб, тоді як ще 20 будуть або помилково негативними, або помилково позитивними.

Наступна метрика дослівно перекладається як «акуратність» (Precision), однак останнім часом, у спільноті ML-інженерів, її називають точністю. Дана метрика точності показує кількість істинно позитивних результатів з усього набору позитивних міток і виражається наступною формулою [18]:

$$Precision = \frac{TP}{TP + FP}$$

Важливість цієї метрики визначається тим, наскільки великою для поставленого завдання є «ціна» помилково позитивного результату.

Метрика «повноти» або «чутливість» (Recall) визначає кількість істинно позитивних міток серед усіх інших які були класифіковані як позитивні та виражається наступною формулою [18]:

$$Recall = \frac{TP}{TP + FN}$$

Максимальну увагу даній метриці приділяються у випадку, коли в поставленій задачі помилка нерозпізнання позитивного класу повинна бути високою, наприклад, при виставленні діагнозу смертельної хвороби.

У тому випадку, якщо Precision та Recall є однаково важливими для задачі, можна використовувати їхнє середнє гармонічне (F1-score) для отримання оцінки результатів [18]:

$$F_1 = \frac{2 * Precision * Recall}{Precision + Recall}$$

Усі розглянуті вище метрики відносилися лише до задач бінарної класифікації. Тому одним з можливих способів розрахувати усі метрики у випадку мультикласифікаційної задачі, є обчислення середньої метрики по всіх класах. Тоді як «позитивний» клас береться обчислюваний, а решта — як «негативний».

2.2 Створення структури засобу

Розроблюваний метод розпізнавання емоційного забарвлення базується на використанні нейромереж в якості основного засобу для досягнення поставленої цілі. Процес розробки такого засобу включає в себе роботу над вирішенням двох задач: тренування та висновку. Задача тренування (training) полягає у налаштуванні ваг нейромережі на основі тренувальної вибірки даних. В свою чергу задача висновку (inference) полягає у обрахуванні вихідної мітки класу, базуючись на вхідних даних з тестової вибірки, а також валідація отриманих результатів.

2.2.1 Структура засобу тренуванні нейромережі

Структура засобу для тренування моделі розпізнавання емоційного забарвлення повідомлення складається з п’яти модулів: колектор даних, передобробка повідомлення, аналіз та маркування даних, тренування мережі, взаємодія з базою даних. Структурну схему засобу для тренування моделі показано на рисунку 2.4.

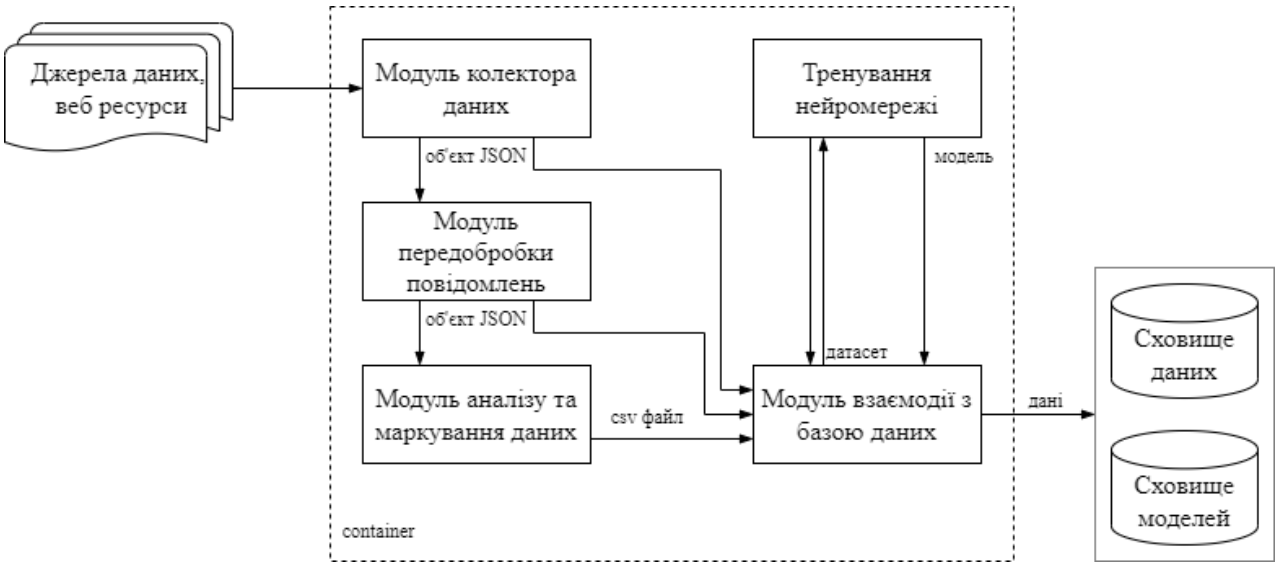


Рисунок 2.4. – Структурна схема засобу тренування

Першим етапом роботи засобу є збір даних з відкритих джерел, таких як: форуми, дошки та соціальні мережі. На цьому етапі працює модуль колектору даних, основною задачею якого є збір та зберігання усіх текстових даних з заданих йому джерел. Для того, щоб забезпечити коректну логіку обробки зібраних даних та їх маркування в подальшому, необхідно сформувати вимоги до вихідного формату даних. Результатом роботи блоку є набір JSON файлів з зібраними повідомленнями.

Далі, отримані дані подаються у модуль передобробки повідомлення. На цьому етапі, вхідні повідомлення піддаються обробці з метою видалення непотрібних символів (емодзі, системні символи, інший шум), а також з метою фільтрації вхідних даних відповідно до критерію довжини повідомлення або його вмісту. Результатом роботи даного модулю є масив очищених повідомлень.

Наступним блоком є модуль аналізу та маркування даних. В даному модулі реалізується використання людського ресурсу для розмітки даних. Перевагою використання людського ресурсу є точність розмітки, оскільки алгоритми не завжди можуть задовольнити вимоги, наприклад, урахування контексту, що є важливим для поставленої задачі. Відповідно до цього, було розроблено web додаток з графічним інтерфейсом для маркування даних. Вихідними даними цього блоку є масив повідомлень та масив відповідних їм міток.

Після того, як усі попередні блоки відпрацювали, розпочинає роботу останній модуль – тренування мережі. Даний модуль відповідає за виконання експериментів (сесія тренування мережі) і зберігання отриманих моделей штучного інтелекту в відповідному сховищі моделей. Модуль передбачає використання моделей різних архітектур, тому налаштовується з відповідного файлу конфігурації. Результатом роботи даного модулю є набір моделей, які представлені у вигляді файлів вагів та файлів конфігурації.

Останній модуль взаємодії з БД використовується під час роботи іншими модулями. Він відповідає за зберігання та зчитування інформації з локального або віддаленого сховища. Даний модуль реалізує різну логіку обробки вхідних та вихідних даних в залежності від того, який блок його викликає.

2.2.2 Структура засобу створення висновку

Структура засобу для створення висновку моделі з трьох модулів: обробки запитів, створення висновку та взаємодія з базою даних. Структурну схему засобу для тренування моделі показано на рисунку 2.5.

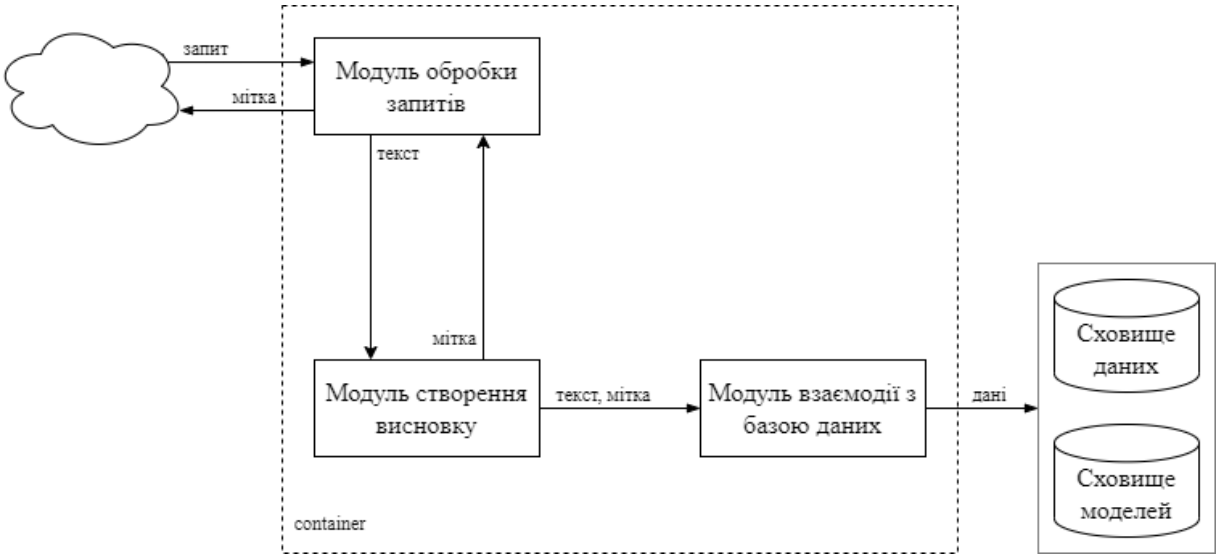


Рисунок 2.5. – Структурна схема засобу висновку

Першим етапом роботи засобу є зчитування тексту повідомлення з вхідного запиту. На цьому етапі працює блок обробки запитів, його основною задачею є встановлення зв’язку з зовнішнім хостом та зчитування його веб запитів. Результатом роботи блоку є зчитане повідомлення.

Далі, отриманий текст повідомлення передається до модулю створення висновку. На цьому етапі, отримане повідомлення спочатку подається до токенізатору і після цього до самої класифікаційної моделі. Далі вихідна мітка класу повідомлення повертається на блок обробки запитів і повертається до запитуваного хосту. Паралельно з цим, текст повідомлення і отримана мітка записуются в БД для подальшого аналізу ефективності роботи класифікатору.

2.3 Алгоритм роботи засобу

Загальну схему алгоритму роботи засобу для визначення емоційного забарвлення повідомлення показано на рисунку 2.6.

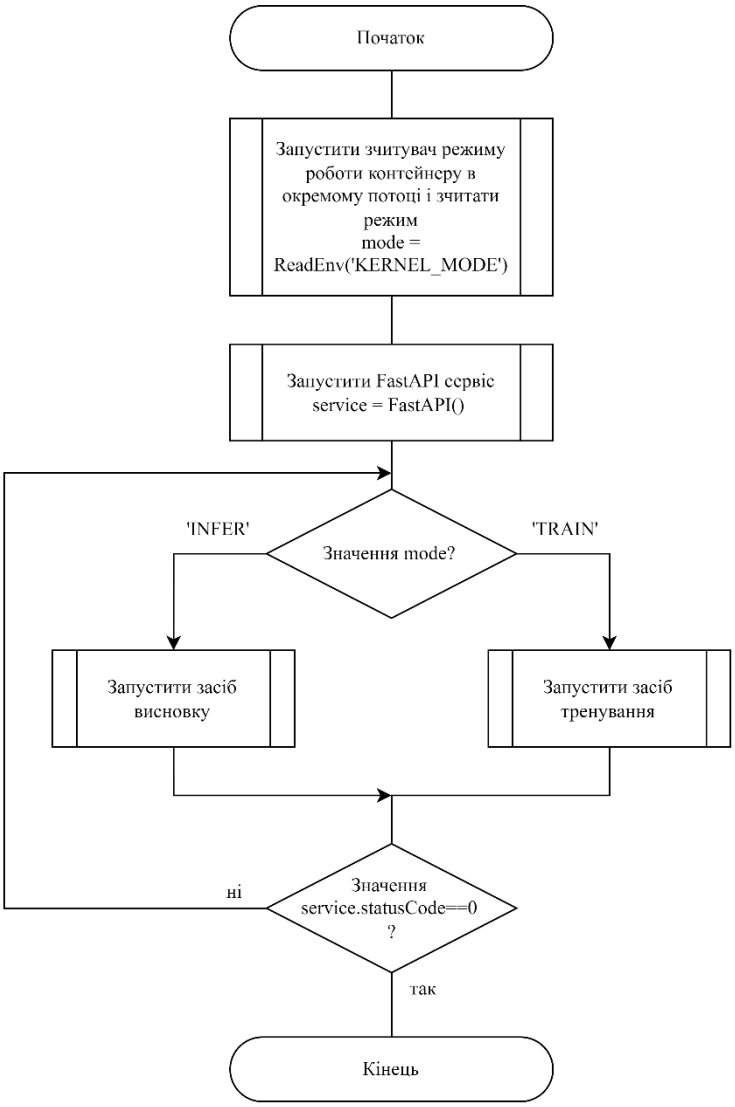


Рисунок 2.6 – Загальний алгоритм роботи засобу

2.3.1 Алгоритм колектору даних

Алгоритм модулю колектору даних складається з виконання наступних дій. Першочергово необхідно завантажити конфігурацію колектору, яка містить ключову інформацію про необхідні кінцеві ресурси. Файл конфігурації містить такі дані як: тип ресурсу(форум, соціальна мережа і т.д.), назва ресурсу, список посилань на необхідні теми або пости, і список посилань на топіки, які не треба обробляти.

Далі необхідно вибрати обробник запитів та сторінок, який залежить від назви ресурсу, оскільки сторінки ресурсів, які навіть належать до одного типу, досить сильно відрізняються. Обробник запитів, повинен формувати вихідні дані, чітко за заданою схемою результуючого JSON файлу. Схему результуючого файлу показано у додатку А.

Наступним кроком в обробник передається посилання, після чого воно оброблюється алгоритмом, визначеним в реалізації обробника, і отримані вихідні дані записуються в БД через модуль взаємодії з базою даних. Алгоритм колектору даних показано на рисунку 2.7.

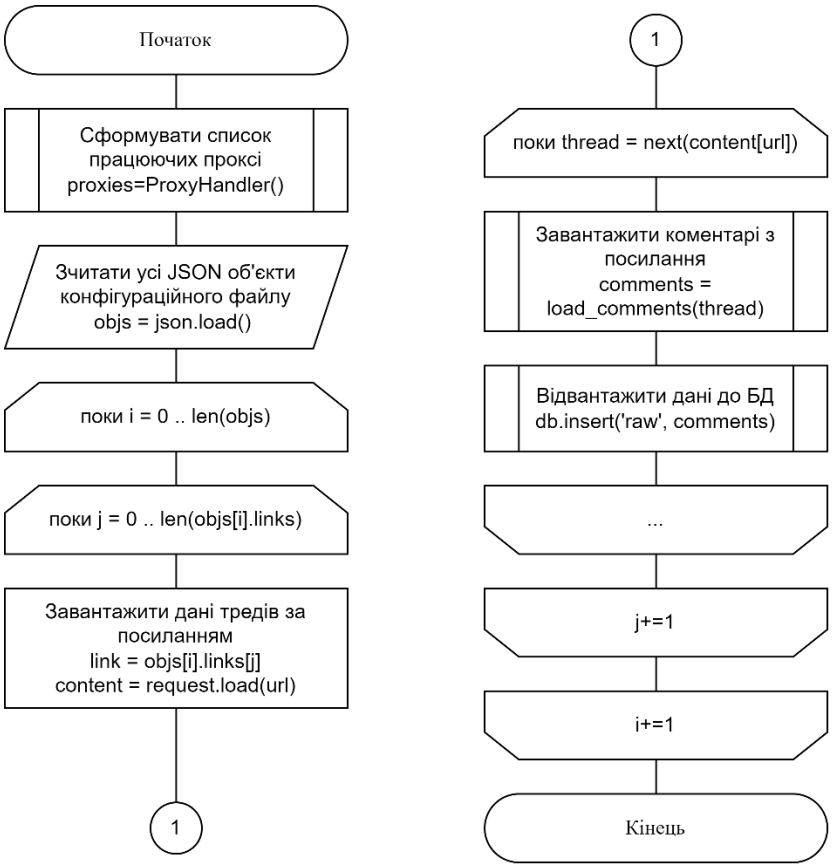


Рисунок 2.7. – Алгоритм роботи колектору даних

Наступним кроком є передобробка повідомлень.

2.3.2 Алгоритм передобробки повідомлень

Суть алгоритму передобробки повідомлень полягає у фільтрації повідомлень на основі його характеристик, та фільтрації самого вмісту повідомлень. Фільтрація повідомлень на основі характеристик полягає у відкиданні повідомлення в разі якщо воно являє собою картинку або пустий рядок. До фільтрації вмісту включається: видалення смайликів та емодзі, видалення посилань вставлених в текст, видалення символів табуляції, переносу каретки і т.д. Однак, передбачається, що фільтрація повідомлень може здійснюватись різними способами, в залежності від того, який експеримент ми проводимо (наприклад не фільтрувати емодзі). Тому алгоритм включає в себе

такі кроки. Спочатку модуль зчитує вхідні параметри, які відповідають за обробник фільтрації. Далі, відбувається зчитування інформації з бази даних. Наступний кроком дані подаються до обробнику фільтрації і результати його роботи записуються у БД. Схема алгоритму модулю передобробки даних показана на рисунку 2.8.

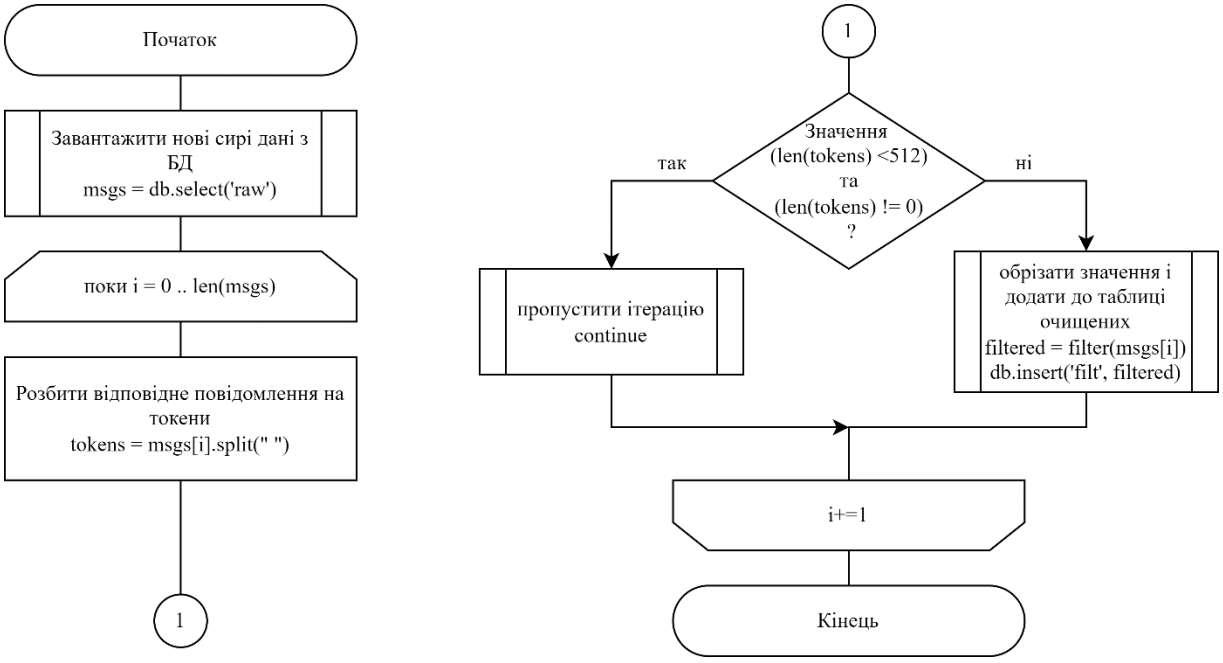


Рисунок 2.8. – Алгоритм передобробки даних

Наступним є модуль маркування даних.

2.3.3 Алгоритм маркування даних

Маркування даних здійснюється вручну за допомогою GUI інтерфейсу. Спочатку користувач вводить свій унікальний ідентифікатор, який використовується для формування вибірки необроблених даних. Далі, повідомлення з вибірки даних поступово ітеруються і виводяться на екран користувача у вигляді радіокнопок з відповідними назвами класів. Користувач вибирає відповідну мітку класу і натискає відповідну кнопку для відправки запиту на модуль роботи з БД. Запит містить в собі текст повідомлення та мітку класу, які необхідно додати до відповідної таблиці. Загальний алгоритм роботи модулю маркування даних показано на рисунку 2.9.

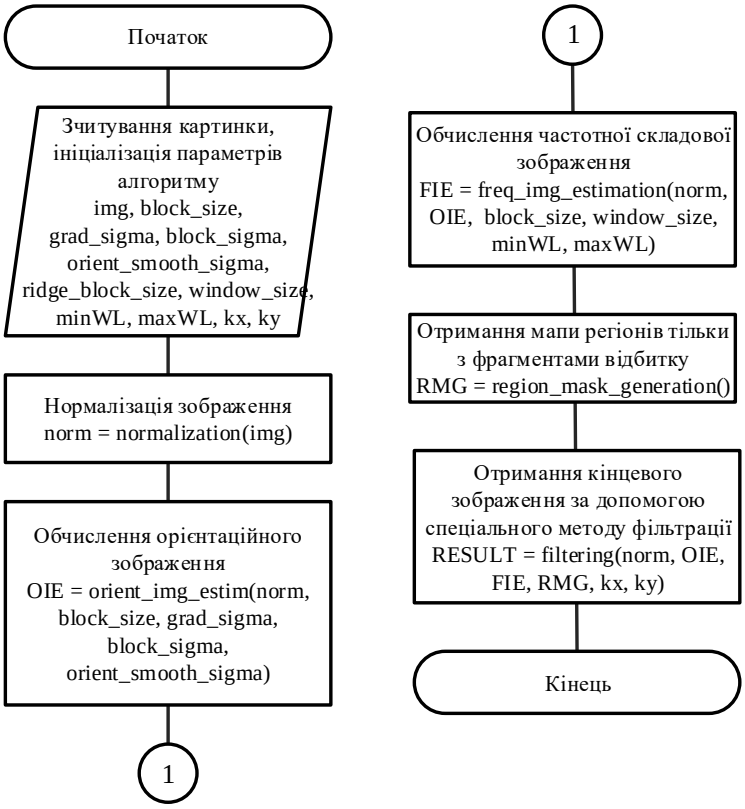


Рисунок 2.9. – Алгоритм маркування даних

2.3.4 Алгоритм тренування мережі

Робота алгоритму полягає у створенні скрипту тренування (пайплайну) відповідно до вхідних параметрів, проведення тренувальних сесій (експериментів) використовуючи вхідні дані та запис натренованих моделей та історії їх тренування в базу даних.

Першим кроком роботи алгоритму є зчитування параметрів тренування, такими параметрами є:

1) Назва попередньо натренованої моделі, яка буде використовуватись в якості кодувальника повідомлень. Такі моделі зберігаються у зовнішньому хмарному сховищі і завантажуються під час виконання програми;
2) Назва класифікатору, який буде використовуватись для класифікації закодованих повідомлень. Класифікатори зберігаються у базі даних;
3) Максимальна кількість токенів – визначає максимальну довжину речення і розмірність вхідних даних претренованої моделі.

Після зчитування параметрів, завантажуються відповідні моделі і зчитуються усі вхідні дані з БД. Головною метою роботи алгоритму є тренування класифікатору, тому процес обробки інформації в алгоритмі тренування можна описати наступною чіткою послідовністю: вхідні дані, мережа кодувальник, мережа класифікатор, вихідні дані.

Першим кроком вхідні дані потрапляють до кодувальника. Для того, щоб подати текстові дані на вхід до BERT мережі, необхідно їх токенізувати. Архітектура BERT використовує так званий WordPiece токенізатор. Суть його роботи полягає у розкладанні слів або в повну форму (одне слово стає одним токеном), або в часткову форму (одне слово розбивається на кілька токенів). Наприклад, слово «surf» отримає токен «surf», а «surfering» розіб'ється на «surf» та «##ing». Використання даного методу дозволяє BERT з легкістю ідентифікувати взаємопов'язані слова, оскільки вони завжди будуть мати одні і ті самі токени. Токенізатор, окрім набору лем повертає також і індекси відповідних токенів, а також маску уваги (attention mask). Маска уваги використовується з метою подання в мережу речень із різною кількістю слів.

Після того, як вхідні повідомлення було токенізовано, вони подаються на вхід попередньо натренованій мережі BERT. Мережа повертає матрицю чисел розмірністю $(1, \text{кількість токенів}, 768)$ [19]. Для задачі класифікації, із отриманої матриці достатньо взяти зріз по нульовому токену в якості вектору речення. Отриманий вектор подається до моделі класифікатора. Далі, необхідно розпочати процес тренування отриманої мережі. Оскільки досліджувана архітектура є класифікатором, то мета її тренування полягає у тому, щоб знайти взаємозв'язок між вхідними даними (вектором повідомлення) та відповідними їм мітками класів.

Обрахування якості навчання під час тренування здійснюється за допомогою функцій витрат (Loss Functions) [20]. В загальному, їх можна розділити на три основні класи:

- Регресійні функції витрат. Використовуються для моделей, що вирішують регресійну задачу, тобто передбачення реального числа на основі багатьох параметрів;
- Функції витрат бінарної класифікації. Використовуються для моделей, що вирішують класифікаційну задачу з двома класами, тобто передбачення однієї з двох міток;
- Мультикласові класифікаційні функції витрат. Використовується для моделей, що вирішують класифікаційну задачу з більш ніж двома класами;

Для поставленої задачі підходять лише мультикласові класифікаційні функції витрат, оскільки вона не є регресійною, а цільових класів більше ніж 2. Мультикласовими класифікаційними функціями витрат є [20]:

1) Multi-Class Cross-Entropy Loss:

$$L(\hat{y}, y) = -\sum_{k}^{K} y^{(k)} \log\left(\hat{y}^{(k)}\right),$$

де $y^{(k)}$ дорівнює 1, якщо мітка класу k вірно класифікована, інакше 0.

2) Sparse Multiclass Cross-Entropy Loss:

$$L(\hat{y}, y) = -\sum_{i=1}^{N} y_i \log(\hat{y}_i)$$

3) Kullback Leibler Divergence Loss:

$$L(\hat{y}, y) = -\sum_{i=1}^{N} \hat{y}_i \log(\hat{y}_i - y_i)$$

Після того, як за допомогою функції витрат було обраховано помилку мережі, необхідно оновити вагові коефіцієнти за допомогою оптимізатора. Ключової задачею оптимізатора є пошук глобального мінімуму на гіперплощині параметрів нейромережі. Тобто знаходження такої точки на площині ваг, при якій результуюча похибка моделі буде мінімальною. В

загальному, алгоритми оптимізації поділяють на дві категорії: градієнтні методи та імпульсні (Momentum) методи [21].

До методів градієнтного спуску належать такі алгоритми, як: градієнтний спуск, стохастичний градієнтний спуск та міні-пакетний градієнтний спуск. Алгоритм градієнтного спуску (Gradient descent) є найвідомішим алгоритмом і активно використовується в задачах лінійної регресії та класифікаційних алгоритмах. Градієнтний спуск є оптимізаційним алгоритмом першого порядку, який залежить від значення похідної функції втрат першого порядку. За допомогою похідної, алгоритм розуміє у який бік слід змінити ваги, щоб функція досягла мінімуму. За допомогою зворотного поширення отримане значення переносяться з одного шару на інший, а ваги моделі, змінюються залежно від втрат, щоб можна було мінімізувати втрати[22].

$$\theta = \theta - \alpha * \nabla J(\theta)$$

Перевагами алгоритму є легкість обчислення, легкість реалізації та легка засвоюваність алгоритму його роботи. Недоліками є низька стійкість до локальних мінімумів – точок, які є хибними рішеннями задачі оптимізації; ваги мережі оптимізуються після обчислення градієнту на всьому наборів даних, таким чином при великій розмірності даних, час за який алгоритм шукатиме мінімум не буде адекватним.

Для вирішення проблем звичайного градієнтного спуску було розроблено алгоритм стохастичного градієнтного спуску (SGD). Він намагається оновлювати параметри моделі частіше. При цьому параметри моделі змінюються після обчислення втрат на кожному навчальному прикладі. Отже, якщо набір даних містить 1000 рядків, SGD оновить параметри моделі 1000 разів за один цикл проходу набору даних [22]:

$$\theta = \theta - \alpha * \nabla J\big(\theta; x(i); y(i)\big),$$

де $\{x(i); y(i)\}$ – тренувальні семпли.

Оскільки параметри моделі часто оновлюються, параметри мають високу дисперсію та флуктуації функцій втрат з різною інтенсивністю.

Перевагами алгоритму є: більша швидкість сходження до точки глобального мінімум часте оновлення параметрів моделі, вимагає менше ресурсів, оскільки не потрібно зберігати значення функцій втрат, може отримати нові мінімуми. В свою чергу основними недоліками є висока дисперсія параметрів моделі, може затрягти в локальному мінімумі.

Іншим типом оптимізаційних алгоритмів стали імпульсні методи. Імпульс був винайдений для зменшення високої дисперсії в SGD і згладжування збігання функції. Таким чином прискорюється сходження функції в необхідному напрямку і зменшується флуктуація в невідповідному напрямку. У цьому методі використовується ще один параметр(імпульс), що позначається як «γ» [22]:

$$V(t) = \gamma * V(t-1) + \alpha * \nabla J(\theta)$$

$$\theta = \theta - V(t)$$

Параметр імпульсу γ зазвичай встановлюється 0,9 або в близькому діапазоні. Перевагами імпульсного методу є: зменшення осциляцій та високої дисперсії параметрів; збігається швидше, ніж градієнтний спуск. Недоліками алгоритму вважається збільшення складності обчислень через додавання нового гіперпараметру, який потрібно вибрати вручну при цьому вгадати його ідеальне значення для конкретної матриці.

Алгоритм Адам (Adaptive Moment Estimation) працює з моментами першого та другого порядку. Логіка роботи адаму полягає в тому, що алгоритму не треба так швидко збігатись тільки тому, що можемо перестрибнути через потенційний мінімум, а отже треба зменшити швидкість для ретельного пошуку. На додаток до зберігання експоненціально спадного середнього минулих квадратів градієнтів, таких як AdaDelta, Адам також зберігає експоненціально спадну середню величину минулих градієнтів M(t).

M(t) і V(t) – значення першого моменту, який є середнім, і другого моменту, який є нецентрованою дисперсією градієнтів відповідно. Тут ми

беремо середнє значення M(t) і V(t), щоб E[m(t)] могло бути рівним E[g(t)], де E[f(x)] – очікуване значення f(x) [22]:

$$m_t = \frac{m_t}{1 - \beta_1^t},$$

$$v_t = \frac{v_t}{1 - \beta_2^t},$$

Щоб оновити параметр [22]:

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{v_t} + \epsilon} m_t$$

при b1 = 0.9, b2 = 0.999 та theta = 10*e-8.

Перевагами Адаму є висока швидкість сходження, а також корекція затухаючого кроку навчання. Із недоліків алгоритму виділяють лише його складність обчислення [21].

В якості оптимізатора моделі було вибрано саме алгоритм Адам, оскільки він найкращим чином серед інших справляється з задачею оптимізації ваг. Перед запуском модулю тренування встановлюється кількість епох, яка визначає кількість повних проходжень датасету нейромережею; та розмір батчу, який визначає скільки семплів даних подається в мережу за одну ітерацію.

Після визначення функції витрат та оптимізатору для тренування мережі, необхідно перейти до роботи з даними. Тренувальні (train) і валідаційні (valid) завантажуються з диску з відповідних файлів за допомогою класу генератору. Клас генератору даних приймає на вхід токенізатор повідомлень, який відповідає за перетворення строкових повідомлень у вигляд, придатний для подання до BERT кодувальника. Перевагою використання класу генератору є те, що необхідні для епохи не завантажуються повністю у пам’ять, а подаються до мережі лише за викликом [23].

Набори даних train і valid являють собою кортежі значень (x, y), де: $x = [t(msg_1), t(msg_2), \dots, t(msg_n)]$ – набір повідомлень, оброблених

токенізатором, довжиною n; $y = [label_1, label_2, \dots, label_n]$ – набір міток класів емоцій відповідних повідомлень.

Схематично, загальний процес тренування показано на рисунку 2.10.

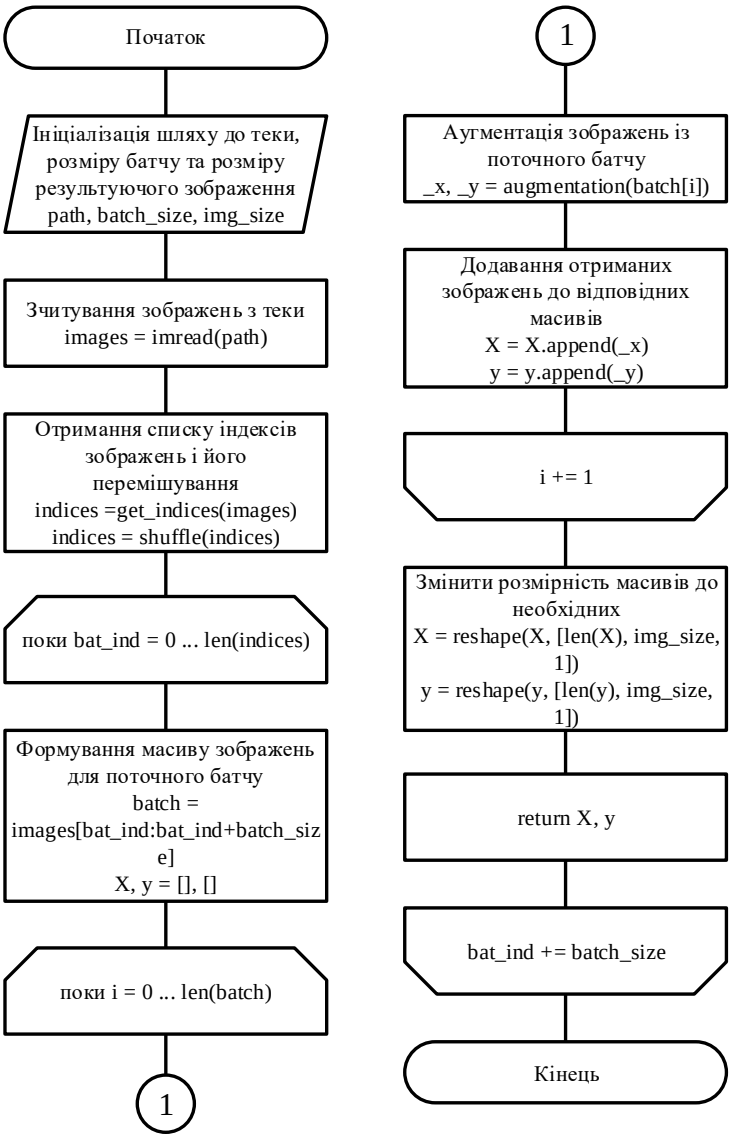


Рисунок 2.10. – Алгоритм тренування мережі

Після розробки алгоритмів можна приступати до реалізації програмного засобу, описаного у наступному розділі. В даному розділі було створено теоретичну основу для подальшого впровадження в програмному коді засобу для визначення емоційного контексту:

- розроблено схеми роботи алгоритмів визначення емоційного контексту повідомлення;
- запропоновано архітектуру класифікаційної мережі потенційно ефективної для вирішення поставлених задач;

- описано метрики точності та вибрано відповідно до специфіки задачі;
- описано архітектуру повноцінного життєвого циклу засобу.

Опираючись на результати, отримані в цьому розділі, можна приступати до наступного етапу роботи – розробка програмної реалізації продукту.

3 РЕАЛІЗАЦІЯ СТРУКТУР І АЛГОРИТМІВ МІКРОСЕРВІСУ

3.1 Аналіз та обгрунтування вибору програмних засобів

На даний момент, бібліотеки для реалізації побудови та тренування моделей машинного навчання існують майже в будь-якій сучасній мові програмування. Для розробки застосунку було обрано Python версії 3.6.5. Основні причини вибору мови Python [24,25]:

- підтримка парадигми ООП;
- кроссплатформеність, саме тому добре підходить для розгортання, керування та відслідковування проектів.
- має найбільший набір пакетів, призначених для роботи в сфері штучного інтелекту (статистика, обробка сигналів, машинне навчання, візуалізація, деплой);

Основними недоліками обраної мови є:

- оскільки Python – це інтерпретована мова, великим мінусом є швидкість виконання програм [26].
- існування Global Interpreter Lock і як наслідок відсутність багатоковості;

Також, для реалізації архітектури нейронної мережі застосунку було обрано фреймворк Pytorch. Причинами для цього стали такі його переваги [26]:

- висока швидкість деплою моделей та легкість використання;
- якісна документація і підтримка від широкої спільноти; претренеровані моделі, які можна використовувати просто завантаживши їх ваги;
- легке використання суміжної бібліотеки huggingface;

Оскільки розроблюваний застосунок використовує досить великі за об’ємом моделі, то з метою пришвидшення навчання мережі було встановлено засоби архітектури CUDA від NVIDIA. CUDA – це програмно-апаратна архітектура, яка підвищує обчислювальну потужність фреймворку за допомогою використання графічних процесорів NVIDIA для пришвидшення

розрахунків [27]. CUDA версії 9 і вище підтримується Pytorch без необхідності встановлення зайвих GPU-орієнтованих бібліотек.

Для реалізації розподіленого навчання, яке також значно підвищує швидкість обчислення за рахунок розподілення даних для тренування між графічними процесорами, було використано фреймворк horovod.

3.2 Реалізація описаних алгоритмів

3.2.1 Реалізація алгоритму колектору даних

Для успішної реалізації засобу знадобиться досить великий набір даних, який складається з повідомлення та відповідної мітки настрою (нейтральне, позитивне, негативне). Перш за все, перед збором необроблених даних було визначено формат файлу конфігурації, який зчитується парсером перед початком роботи.

Таким чином, файл конфігурації являє собою список JSON об’єктів, поля якого відповідають за тип ресурсу (форум чи соціальна мережа), назва форуму, посилання на теми з релевантними для задачі топіками та поле посилань які необхідно пропустити. Далі було сформовано вимоги до вихідних даних, які є результатом роботи парсеру.

При реалізації парсеру було враховано факт того, що більшість ресурсів при великій кількості запитів до їх серверу, будуть заморожувати трафік з метою попердження DDoS-атаки. Тому було вирішено на початку роботи парсеру знаходити усі можливі адреси проксі-серверів, через які можна тунелювати трафік. Для цього спочатку необхідно зробити запит за url адресою ресурсу з проксі серверами і опрацювати отриману сторінку. В якості навантаження було додано поле user-agent, яке видає себе за браузер.

Отримавши контент сторінки з запиту, він передається до конструктора бібліотеки для обробки веб-контенту і далі, за допомогою взаємодії з об’єктом цього класу відбувається зчитування IP адрес та портів необхідних серверів

Також, для запобігання випадку використання тунелів, які не працюють було реалізовано метод перевірки швидкості. Оскільки не було знайдено

бібліотек для визначення швидкості, було реалізовано простішу версію: через необхідний проксі сервер формується запит на випадкову сторінку форуму зі вказуванням параметра очікування відповіді рівним двом секундам, якщо за цей час приходить відповідь з серверу, то хост живий, інакше – ні.

3.2.2 Реалізація алгоритму тренування мережі

Першим чином було створено модель мережі за допомогою обраного фреймворку Pytorch. Її основою стала архітектура BERT, яка виступає в ролі кодувальника повідомлення. Вихід з енкодера являє собою вектор чисел розмірністю (762, 1), саме він подається далі на вхід повнозв'язній мережі з трьома виходами, які відповідають трьом необхідним класам повідомлення. Схематична структура моделі показана на рисунку 3.2.

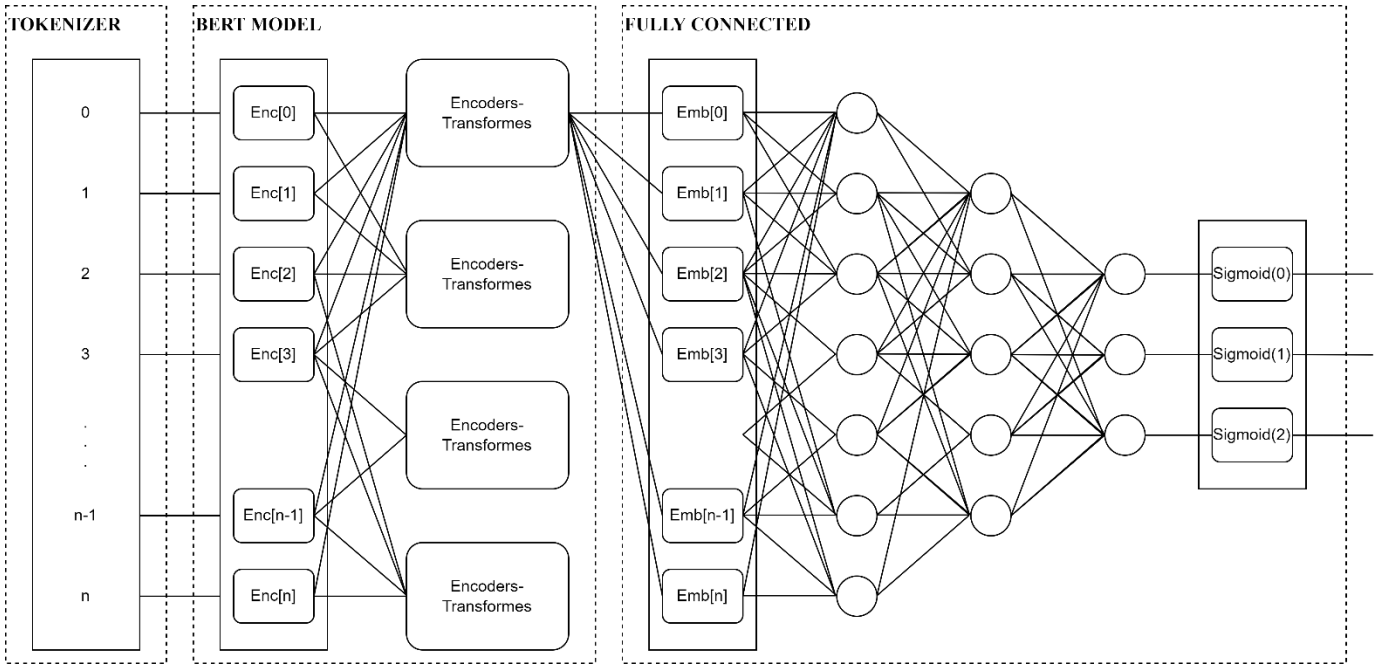


Рисунок 3.2 – Модель розробленої мережі для тренування

Спочатку на вхід до моделі подаються токенізоване повідомлення – шар «TOKENIZER»; далі, токенізовані повідомлення подаються до BERT моделі, яка за допомогою налаштованих енкодерів та трансформерів перетворює вхідні токени у матрицю розмірністю $(12, batchsize, tokensnum, 334)$, де batchsize – к-ть семплів в одному батчі, tokensnum – к-ть вхідних токенів. Однак для задачі класифікації, нам необхідно взяти зріз матриці лише по першому токену і по останньому прихованому шару з 12, таким чином вихідний вектор матимете

розмірність $(batchsize, 334)$, отриманий вектор подається до повнозв'язної мережі-класифікатора «FULLY CONNECTED». Вихідними даними логіт вектор з трьох чисел, який по своїй суті являє впевненність моделі в приналежності вхідних даних до кожного із трьох класів. Таким чином, отримавши вектор $(0.8,0.01,0.1)$ ми беремо індекс максимального числа, в даному випадку 0 і розуміємо, що вектор відносить до нульового класу.

Функцією активації для вихідного шару мережі є сигмоїдальна функція, її застосовуюь для задач багатоміткової класифікації, оскільки вихідними значеннями є числа в проміжку(0, 1). Для проміжних шарів доцільно використовувати ReLU функцію, оскільки вона попереджує виникнення проблеми «затухаючих» градієнтів, яка майже унеможливлює треування. Функція сигмоїди та релу мають наступний вигляд:

$$Sigmoid: \sigma(x) = \frac{1}{1+e^{-x}};$$

$$ReLU: f(x) = \max(0, x);$$

та відповідають за нелінійність збудження нейрону. Графік залежності вихідного сигналу від вхідного показано на рисунку 3.3 [41].

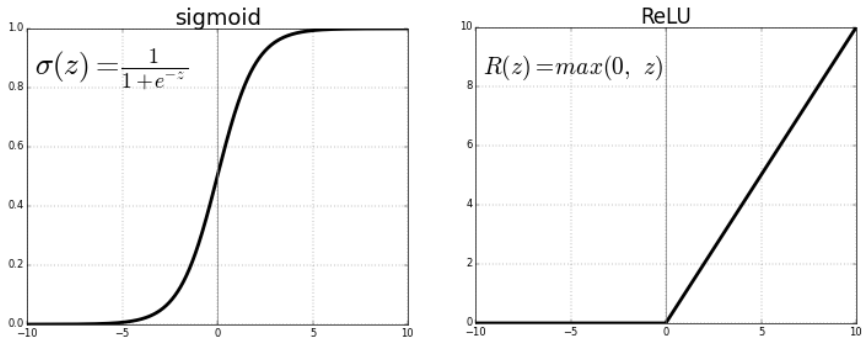


Рисунок 3.3 – Приклади функцій Sigmoid(зліва) та ReLU(справа)

3.2.3 Реалізація алгоритму взаємодії з базою даних.

3.3 Попереднє натренування моделі

Попереднє тренування моделі являє собою гіпотезу використання класифікатора за трьома емоціями, натренованого на попередньо розміченому

датасеті, для маркування сирих даних, зібраних колектором. Сперше треба зібрати дані з відритих джерел, які б задовольняли умовам задачі. Далі залишається лише тренування мережі, її валідація та збереження образу моделі на диск для подальшого використання.

3.3.1 Аналіз датасету Toxic Russian Comments

Даний датасет знаходиться у відкритому доступі на популярному ресурсі для змагань по машинному навчанню Kaggle. Датасет містить розмічені коментарі з популярної Російської соціальної мережі ok. Всього, було іиділено 4 мітки емоційності, які відображають різні рівні «токсичності» повідомлення. Кожному повідомленню присвоїно одну або декілька наступних міток:

1) __label__NORMAL – коментарі з нейтральний забарвленням;
2) __label__INSULT – коментарі, які принижують людину;
3) __label__THREAT – коментарі з явним натяком на загрозу;
4) __label__OBSCENITY – коментарі з описом або загрозою сексуального насилля.

Для початку поглянемо на вміст датасету, а також оцінимо відповідність міток до вмісту повідомлення (рис 3.4).

```
for label, texts in data.groupby('label'):
    print(f'Повідомленя: {texts.iloc[0, 0]}')
    print(f'Мітка класу: {label}\n')
```

```
Повідомленя: чтобы каждый день как петуха драли )в ж[redacted]
Мітка класу: ['__label__INSULT', '__label__OBSCENITY', '__label__THREAT']

Повідомленя: [redacted]в [redacted] этому [redacted], а не респект
Мітка класу: ['__label__INSULT', '__label__OBSCENITY']

Повідомленя: заколоть этого плешивого урода что бы крякнул как [redacted] всю вакцину ему в бошку что бы конкрет
но его завернуло
Мітка класу: ['__label__INSULT', '__label__THREAT']

Повідомленя: скотина! что сказать
Мітка класу: ['__label__INSULT']

Повідомленя: я сегодня проезжала по рабочей и между домами снитенко и гомолысовой магазином ( на пустыре) бежала кошка похожего
окраса. может, я и ошиблась, но необычный окрас бросился в глаза.
Мітка класу: ['__label__NORMAL']

Повідомленя: изнасиловать в[redacted]черенком от снеговой лопаты!!!
Мітка класу: ['__label__OBSCENITY', '__label__THREAT']

Повідомленя: эти генеральши знают где [redacted] и раздвинуть [redacted] и [redacted] ,позор страны
Мітка класу: ['__label__OBSCENITY']

Повідомленя: надо было его собаку на заборе повесить ,жалко скотинку конечно но это было бы предсказание...
Мітка класу: ['__label__THREAT']
```

Рисунок 3.4 – Приклади повідомлень в датасеті

Відповідно, можемо зробити висновок, що мітки є доцільними відносно тексту. Далі необхідно проаналізувати датасет на розмірність, наявність пропусків в даних, а також баланс класів – кількість повідомлень для кожного класу емоції. Інформацію про розмірініть та пропуски показано на рисунку 3.5.

```
print('Розмірність датасету:', data.shape)
print(f'Кількість пропущениих даних:\n{data.isna().sum()}')
```

```
Розмірність датасету: (248290, 2)
Кількість пропущениих даних:
text     0
label    0
dtype: int64
```

Рисунок 3.5 – Дані про розмірність та наявність пустих рядків

Для аналізу балансу класів було побудовано графік з кількістю семплів на відповідний клас, отриманий результат показано на рисунку 3.6.

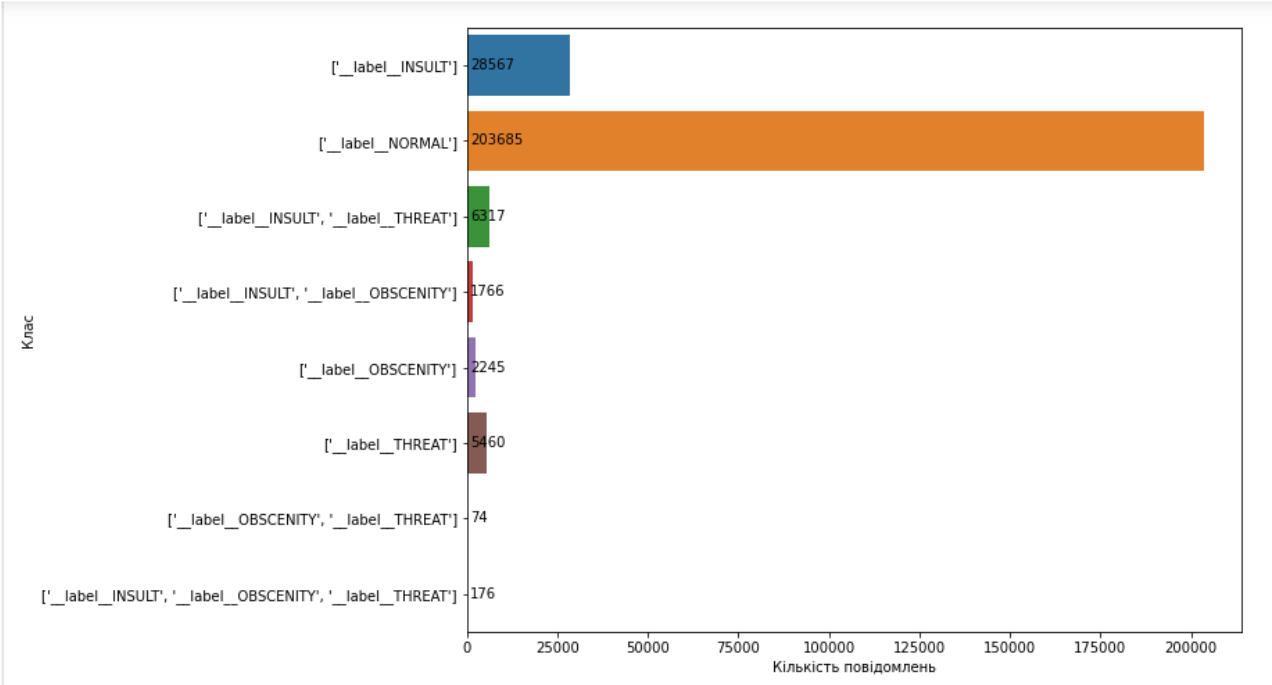


Рисунок 3.6 – графік кількості повідомлень для класів емоцій

Для того, щоб отримати повну картину про баланс даних, було вирішено звести датасет до вигляду, відповідного задачі, а саме обробити його таким чином, щоб всі не нейтральні мітки відносились до класу негативних повідомлень. Результат показано на рисунку 3.7

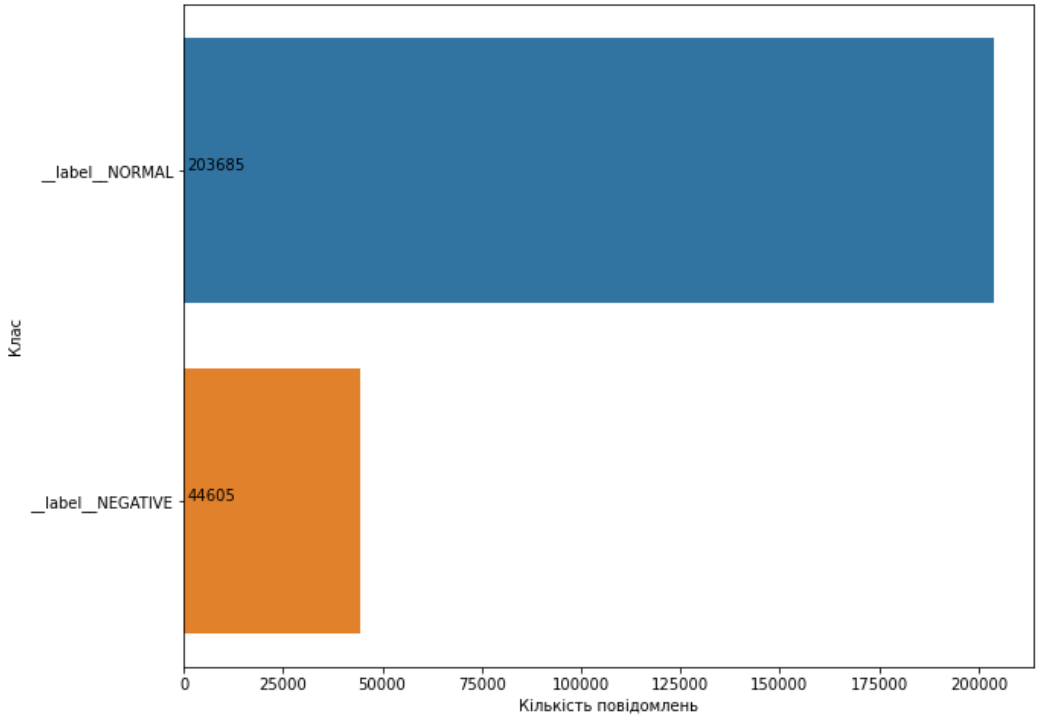


Рисунок 3.7 – Графік кількості повідомлень відносно двох класів емоцій

Таким чином, можна зробити висновок, що хоч і датасет має проблему з балансом класів, однак вона не є досить критичною, оскільки співвідношення кількості семплів в класах дорівнює $\frac{203685}{44605} = 4.56$.

Також необхідно провести аналіз повідомлень за їх токенізованою довжиною, оскільки максимальна кількість вхідних токенів для BERT моделей – 512 штук. Для цього, порядково розіб'ємо повідомлення по пробілу і запишемо отриману токенів в окрему колонку. Графік розподілу довжин показано на рисунку 3.8.

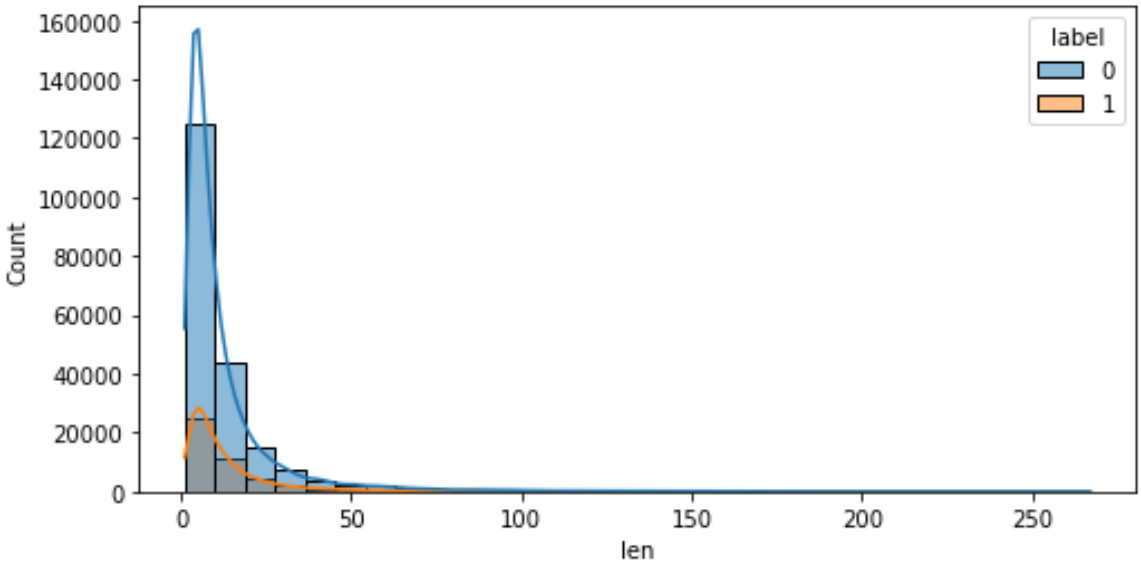


Рисунок 3.8. – Розподіл довжин повідомлень

З рисунку можна зрозуміти, що найбільша кількість повідомлень має до 25 токенів, при цьому для обох класів міток, щопонвістю задовольняє умові кількості вхідних даних BERT моделі.

3.3.2 Аналіз датасету RuSentiment

Наступний датасет також знаходиться у відкритому доступі на платформі Git. Повідомлення для датасету збирались з коментарів соціальної мережі «ВКонтакте». Далі, зібрані речення розмічувались вручну відповідно наступному набору правил:

1) класи негативних «negative» і позитивних «positive» настроїв охоплюють як неявні, так і явні почуття, як для вираження емоцій, так і ставлення;
2) нейтральний клас (не позначений для настроїв);
3) клас мовленнєвої дії «speech»: дописи в соціальних мережах часто містять шаблонні привітання, пости подяки та вітальні пости, які можуть виражати або не виражати справжні почуття відправника;
4) клас «skip» додано для незрозумілих випадків, галасливих дописів, вмісту, який, ймовірно, створений не самими користувачами (вірші, тексти, жарти тощо).

5) випадки змішаних настроїв анотовані для домінуючих настроїв пости, а рекомендації охоплюють 6 частих випадків змішаних настроїв для покращення угоди між анотаторами;
6) хештеги та смайли не розглядаються як автоматичні мітки настроїв.

Для початку поглянемо на вміст датасету, а також оцінимо відповідність міток до вмісту повідомлення (рис 3.9).

```
for label, texts in data.groupby('label'):
    print(f'Повідомленя: {texts.iloc[1, 1]}')
    print(f'Мітка класу: {label}\n')
```

```
Повідомленя: Блин, почему эта жизнь столь не справедлива ((((
Мітка класу: negative

Повідомленя: Просто пост :)
Мітка класу: neutral

Повідомленя: урря!я дождался этой овцы)
Мітка класу: positive

Повідомленя: где еще встречать свой день рождения как не на кладбище))))
Мітка класу: skip

Повідомленя: С Днем Рождения желаю много счастья, любви и успехов во всем:) И еще с днем Защитника Отечества;)))))))))
Мітка класу: speech
```

Рисунок 3.9 – Приклади повідомлень в датасеті

Мітки є доцільними відносно контексту емоції відомлення, а отже далі необхідно проаналізувати розмірність даних, наявність в них пропусків, а також баланс класів міток. Інформацію про розмірініть та пропуски показано на рисунку 3.10.

```
print('Розмірність датасету:', data.shape)
print(f'Кількість пропущениих даних:\n{data.isna().sum()}')
```

```
Розмірність датасету: (31185, 2)
Кількість пропущениих даних:
label    0
text     0
dtype: int64
```

Рисунок 3.10 – Дані про розмірність та наявність пустих рядків

Для аналізу балансу класів було побудовано графік кількості семплів для відповідних класів. Результат показано на рисунку 3.11.

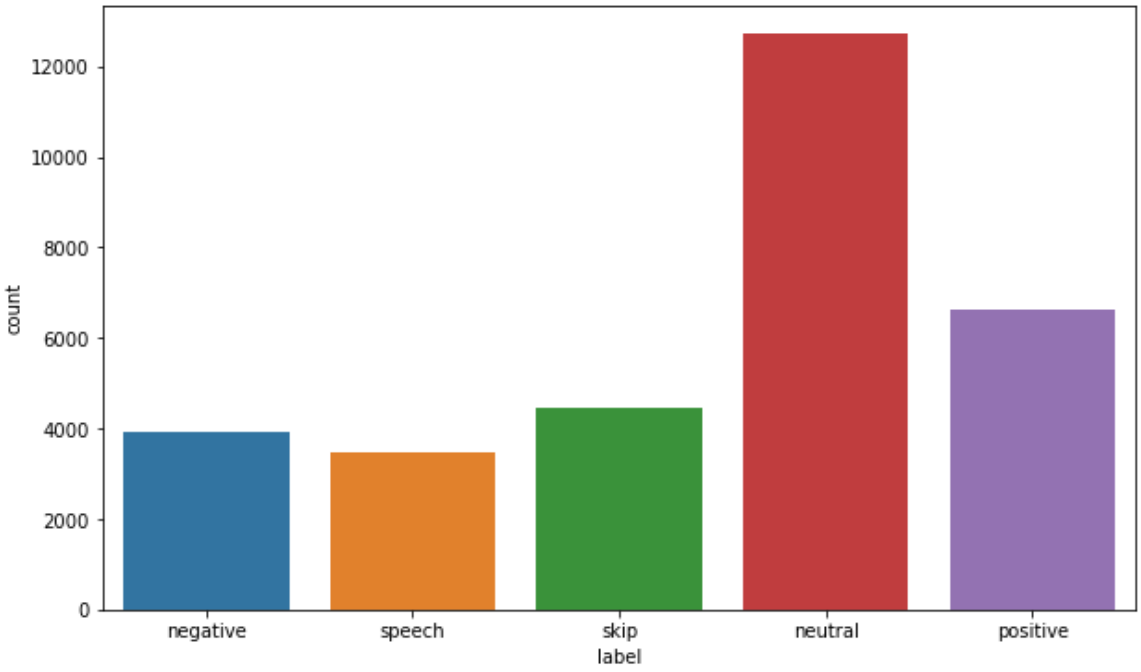


Рисунок 3.11 – графік кількості повідомлень для класів емоцій

Оскільки під час розробки засобу було вирішено будувати модель для виявляення трьох емоцій – відкинемо непотрібні нам мітки та семпли і залишимо лише необхідні класи. Результат показано на рисунку 3.12

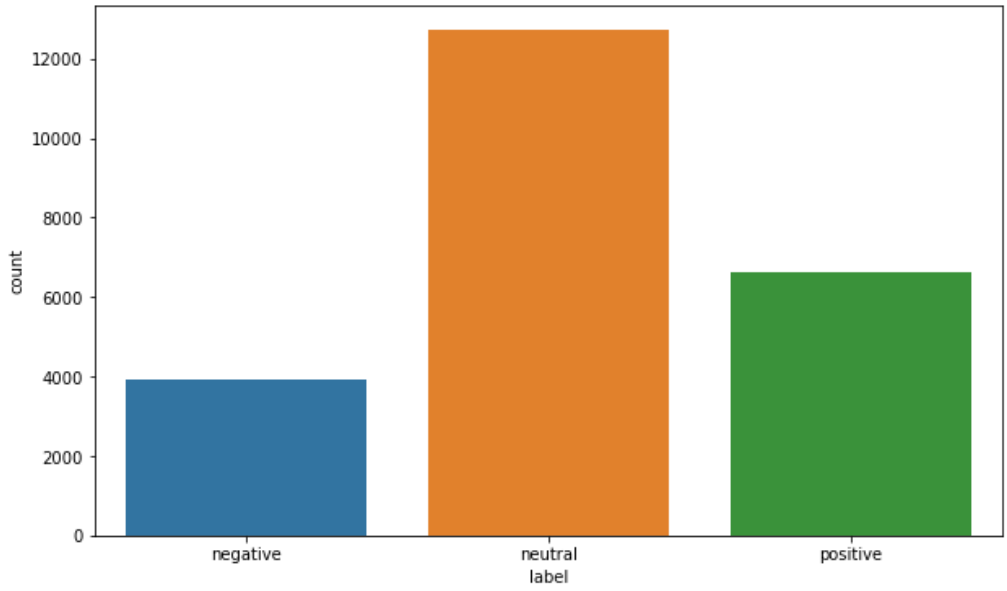


Рисунок 3.12 – Графік кількості повідомлень відносно двох класів емоцій

Отже максимальне значення відношення класів дорівнює $\frac{13021}{3897} = 3.32$, що на перший погляд свідчить про несерйозну проблему з балансом даних,

однак необхідно враховувати те, що найменший клас має близько 4-х тисяч записів, що є досить невеликою кількістю повідомлень

Останнім кроком проведемо аналіз повідомлень за їх токенізованою довжиною. Алгоритм отримання довжин токенів овідомлень аналогічний описаному в попередньому пункті. Графік розподілу довжин показано на рисунку 3.13.

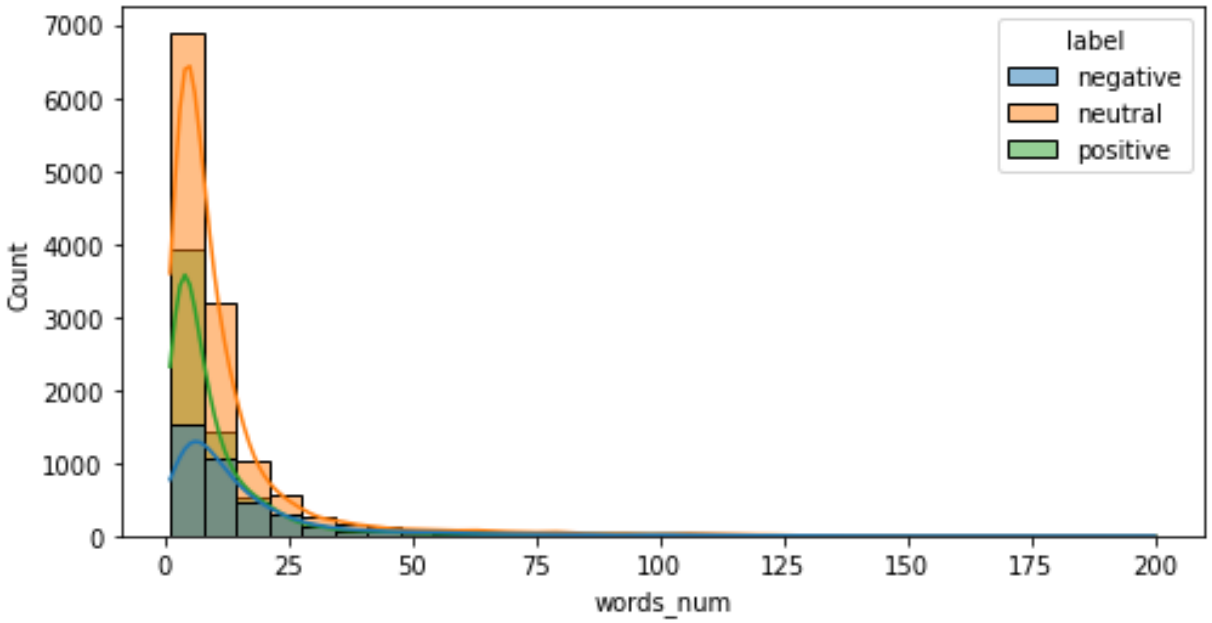


Рисунок 3.13. – Розподіл довжин повідомлень

З рисунку можна зрозуміти, що найбільша кількість повідомлень має до 25 токенів, при цьому для усіх трьох класів міток, що понвістю задовольняє умові розмірості вхідних даних BERT моделі.

3.3.3 Тренування моделі

Після численних експериментів з тренуванням класифікатору і поадльшому вивченні похибок було з’ясовано, що результат роботи класифікатору сильно залежить від seed числа генератору випадкових чисел, факту перемішування даних у вибірках, зміні параметрів вхідного токенізатору і т.д.

В результаті, після спроб підібрати необхідні параметри було отримано тренувальну сесія, графік функції витрат якої показано на рисунку 3.15.

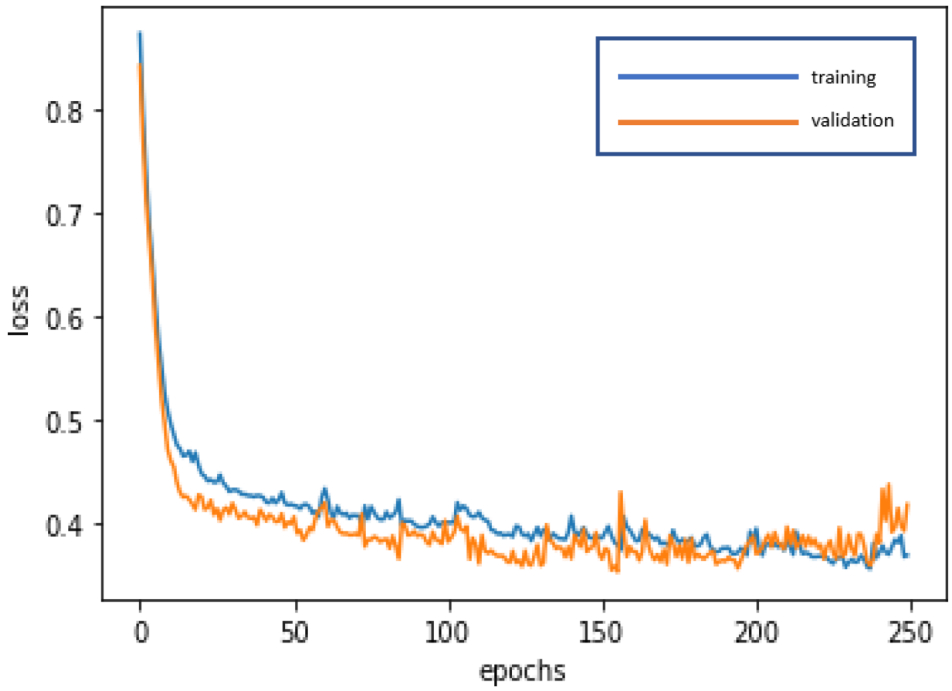


Рисунок 3.15 – Графік функції витрат

Для даного експерименту було отримано наступні результати: Покращення матриці помилок під час тренування класифікатору, показано на рисунку 3.14

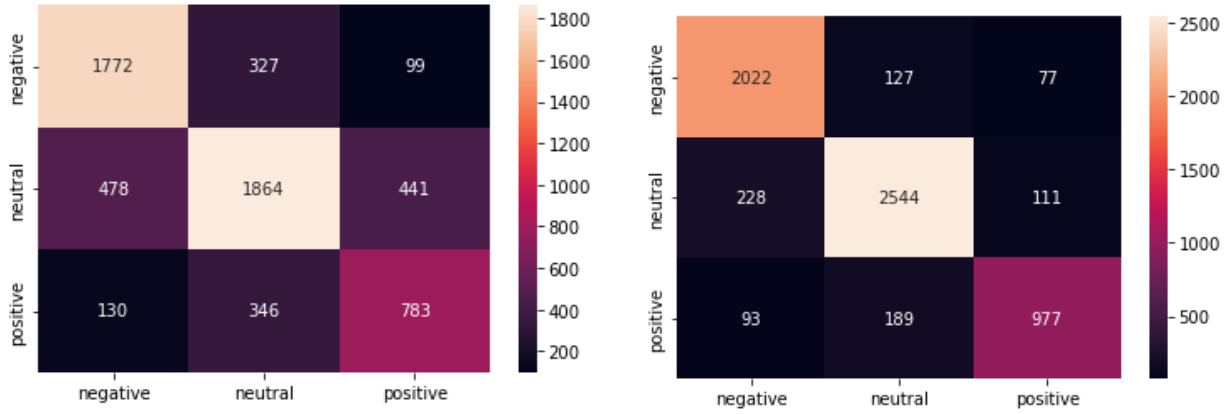


Рисунок 3.14 – Матриця помилок для експерименту

Першу матрицю похибок було отримано на першій епохі тренування, а другу – на 150 епосі. Графіки добре відображають факт того, що модель змогла підлаштуватись навіть під імбалансний клас повідомлень.

З метою аналізу некоректно передбачених семплів можемо вивести їх разом із мітками їх класів (рисунок 3.15).

Повідомлення	Проставлена мітка	Цільова мітка
“Я бы тебя сжёг…”	Агресія	Агресія
“классно придумано красота”	Позитивний	Позитивний
“маразматики озлобленые”	Нейтральний	Агресія
“скотина! Что сказать…”	Агресія	Агресія
“кот само спойствие и пофигизм, а собачка боится???”	Нейтральний	Нейтральний
“утопить бы того,, , кто так поступил.”	Нейтральний	Агресія

Рисунок 3.15 – Аналіз передбачених міток нейромережею

Далі проаналізуємо похибки першого і другого роду для натренованого класифікатору:

$$Recall = \frac{\sum TP}{\sum(TP + FN)} = \frac{(2022 + 2544 + 977)}{(2022 + 204) + (2544 + 339) + (977 + 282)} = 0.87$$

$$Precision = \frac{\sum TP}{\sum(TP + FP)} = \frac{(2022 + 2544 + 977)}{(2022 + 321) + (2544 + 316) + (977 + 188)} = 0.85$$

Як видно із отриманих результатів метрик, класифікатор натренувався досить добре та гарно відрізняє класи повідомлень один від одного. Існує невелика похибка з розпізнаванням позитивного класу, оскільки у вхідних даних був невеликий зсув в балансі класів. Однак результатами тренування став досить ефективний класифікатор повідомлень.

4 ЕКОНОМІЧНА ЧАСТИНА

Метою економічної частини магістерської кваліфікаційної роботи є обґрунтування економічної доцільності розробки методу та засобу розпізнавання емоційного забарвлення повідомлень. Для цього необхідно виконати такі етапи робіт:

- оцінити комерційний потенціал розробки;
- спрогнозувати витрати на виконання наукової роботи та впровадження її результатів;
- спрогнозувати комерційний ефект від реалізації результатів розробки;
- розрахувати ефективність вкладених інвестицій та період їх окупності.

4.1 Оцінювання комерційного потенціалу розробки (технологічний аудит розробки)

Об’єктом дослідження магістерської кваліфікаційної роботи є метод та засіб розпізнавання фішингових інформаційних ресурсів.

З метою проведення технологічного аудиту розробки, було залучено трьох незалежних експертів: Войтович О.П., Шелепало Г.В., Баришев Ю.В. Спочатку, залучені експерти ознайомлювалися з розробкою, після чого, надавали свої оцінки відповідно таблиці, яка визначає рекомендовані критерії оцінювання комерційного потенціалу розробки та їх можливу оцінку в балах. Наступним кроком етапу оцінювання було обрахування середньоарифметичної суми оцінок та отримання висновку про рівень комерційного потенціалу розробки.

Оцінювання експертами потенціалу розробки було здійснено за дванадцятьма критеріями, наведеними в таблиці 4.1.

Таблиця 4.1 – Критерії оцінювання комерційного потенціалу розробки та їх відповідна оцінка

Критерії оцінювання та бали (за 5-ти бальною шкалою)					
Критерій	0	1	2	3	4
Технічна здійсненність концепції:					

1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено роботоздатність продукту в реальних умовах
Ринкові переваги (недоліки):					
2	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів
4	Технічні та споживчі властивості продукту значно гірші, ніж в аналогів	Технічні та споживчі властивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в аналогів	Технічні та споживчі властивості продукту значно кращі, ніж в аналогів
5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів
Ринкові перспективи					
6	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою
7	Активна конкуренція великих компаній на ринку	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає
8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних фахівців	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так і з комерційної реалізації ідеї
9	Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеї відсутні	Потрібні незначні фінансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуютьс я у військово-	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно

		промисловому комплексі			використовуються у виробництві
11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10-ти років	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій більше 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від 3-х до 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій менше 3-х років
12	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на виробництво та реалізацію продукту	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що вимагає значних коштів та часу	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	Необхідно тільки повідомлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь-які регламентні обмеження на виробництво та реалізацію продукту

Результати експертного оцінювання комерційного потенціалу розробки наведено в таблиці 4.2.

Таблиця 4.2 – Результати експертного оцінювання

Критерії	Прізвище, ініціали експерта		
	Експерт 1	Експерт 2	Експерт 3
	Бали, виставлені експертами:		
1	3	3	3
2	1	2	2
3	4	4	4
4	2	3	2
5	3	2	2
6	2	2	2
7	1	1	2
8	4	4	4
9	2	1	1
10	4	4	4
11	4	4	4
12	4	4	4
Сума балів	СБ1=x1	СБ2=x2	СБ3=x3
Середньоарифметична сума балів $\overline{СБ}$	$\overline{СБ} = \frac{\sum_{1}^{i} СБ_i}{i} = \frac{x1+x2+x3}{3} =$		

Розрахувавши середньоарифметичну суму балів з оцінок експертів, наведених в таблиці 4.2, було отримано оцінку 34, значення якої відповідно до

таблиці 4.3, свідчить про те, що нова розробка володіє комерційним потенціалом вище середнього.

Таблиця 4.3 – Рівні комерційного потенціалу розробки

Середньоарифметична сума балів	Рівень комерційного потенціалу
0-10	Низький
11-20	Нижче середнього
21-30	Середній
31-40	Вище середнього
41-50	Високий

На українському ринку майже повністю відсутні засоби визначення емоційного забарвлення тексту, а отже існує необхідність в розробці таких методів та засобів. Існує дійсно велика кількість способів застосування таких засобів та їх монетизація, особливо враховуючи повну відсутність конкуренції на локальному ринку.

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

Для здійснення прогнозування витрат на виконання науково-дослідної роботи, дослідно-конструкторської та конструкторсько-технологічної роботи необхідно виконати наступні кроки:

1) Розрахувати витрати, що стосуються виконавців даного розділу роботи;
2) Розрахувати загальні витрати на виконання даної роботи;
3) Спрогнозувати можливі загальні витрати, пов’язані з виконанням та впровадженням результатів даної роботи.

Для розрахунку витрат, затрачених на розробку методики дослідження необхідно використати такі дані:

– основний та додатковий доходи розробників;

– нарахування на заробітну плату розробників;
– витрати на сервіси, використані під час розробки програмного продукту;
– амортизація використаного обладнання;
– витрати на електроенергію;
– інші витрати.

Розробкою даного продукту займалися один спеціаліст та один науковий керівник. Величину основної заробітної плати спеціаліста можна визначити за формулою:

$$З_О = \sum_{i=1}^{k} \frac{М_{ni} \cdot t_i}{Т_р},$$

де k – кількість посад дослідників залучених до процесу досліджень;

$М_{ni}$ – місячний посадовий оклад конкретного дослідника, грн;

t_i – число днів роботи конкретного дослідника, дн.;

$Т_р$ – середнє число робочих днів у місяці, $Т_р$ = 22 дні.

Оплата за робочий день розробника проекту становить 1200 грн, а наукового керівника – 1200 грн.

Заробітна плата наукового керівника:

$$З_{О_{НК}} = \frac{26400}{22} \times 7 = 1200 * 10 = 12000(\text{грн})$$

Заробітна плата розробника:

$$З_{О_Р} = \frac{26400}{22} \times 7 = 1200 * 20 = 24000(\text{грн})$$

Витрати на оплату праці, основна заробітна плата:

$$З_О = З_{О_{НК}} + З_{О_Р} = 12000 + 24000 = 36000(\text{грн})$$

Результати розрахованої заробітної плати для дослідників наведені в таблиці 4.3.

Таблиця 4.3 – Витрати на заробітну плату дослідників

Найменування посади	Місячний посадовий оклад, грн	Оплата за робочий день, грн	Число днів роботи, t	Витрати на оплату праці, грн
Науковий керівник проекту	26400	1200	10	12000
Розробник	26400	1200	20	24000
Всього:				36000

Витрати на основну заробітну плату робітників ($З_р$) за відповідним найменуваннями робіт розраховано за формулою:

$$З_р = \sum_{i=1}^{n} C_i \cdot t_i ,$$

де C_i – погодинна тарифна ставка робітника відповідного розряду, за виконану відповідну роботу, грн/год;

t_i – час роботи робітника при виконанні визначеної роботи, год.

Погодинну тарифну ставку робітника відповідного розряду C_i визначено за формулою:

$$C_i = \frac{M_M \cdot K_i}{T_p \cdot t_{зм}} ,$$

де M_M – розмір прожиткового мінімуму працездатної особи або мінімальної заробітної плати (в залежності від діючого законодавства), грн;

K_i – коефіцієнт міжкваліфікаційного співвідношення для встановлення тарифної ставки робітнику відповідного розряду;

T_p – середнє число робочих днів у місяці, T_p = 22 дні.

$t_{зм}$ – тривалість зміни, год.

Таблиця 4.4 – Величина витрат на основну заробітну плату робітників

Найменування робіт	Тривалість роботи, год	Розряд роботи	Тарифний коефіцієнт	Погодинна тарифна ставка, грн	Величина оплати на робітника, грн
Розробка	12	4	1,5	65,34	784.08

Тестування	6	4	1,5	65,34	392,04
Впровадження	2	3	1,3	50,63	101.26
Всього					1277.38

Розрахуємо додаткову заробітну плату. Додаткова заробітна плата розраховується як 10...12% від суми основної заробітної плати дослідників та робітників за формулою:

$$З_{\text{дод}} = (0.1, \dots, 0.12) \times З_{\text{О}} \quad (5.4)$$

Додаткова заробітна плата:

$$З_{\text{дод}} = 0.2 \times (З_{\text{О}_{\text{НК}}} + З_{\text{О}_{\text{Р}}}) = 0.12 \times (12000 + 1277{,}3) = 1593{,}27(\text{грн})$$

Нарахування на заробітну плату дослідників та робітників розраховується як 22% від суми основної та додаткової заробітної плати за формулою:

$$H_{ЗП} = (З_о + З_р + З_{дод}) \times \frac{\beta}{100},$$

де $З_о$ – основна заробітна плата розробників, грн;

$З_{дод}$ – додаткова заробітна плата розробників, грн;

$З_р$ – заробітна плата робітників, грн;

β – ставка єдиного внеску на загальнообов'язкове державне страхування.

Єдиний соціальний внесок на загальнообов'язкове державне соціальне страхування (ЄСВ) – об'єднаний страховий внесок в Україні, збір якого здійснюється в системі загальнообов'язкового державного страхування в обов'язковому порядку та на регулярній основі. Відповідно до даних офіційного сайту міністерства фінансів в 2021 році ЄСВ становить 22%.

$$\text{Н}_{\text{ЗП}} = (36000 + 1277.3 + 1593{,}27) \times 0.22 = 11080.57$$

Розрахунок витрат на амортизацію обладнання можна провести з використанням прямолінійного методу амортизації за формулою:

$$A_{обл} = \frac{Ц_б}{T_в} \times \frac{t_{вик}}{12},$$

де $Ц_б$ – балансова вартість обладнання, програмних засобів, приміщень тощо, які використовувались для проведення досліджень, грн;

$t_{вик}$ – термін використання обладнання, програмних засобів, приміщень під час досліджень, місяців;

$T_в$ – строк корисного використання обладнання, програмних засобів, приміщень тощо, років.

Розробка застосунку проводилось на ноутбуці загальною вартістю 26000 грн. та за допомогою віддаленого серверу з відеокартами, загальною вартістю близько 70000. Амортизаційні відрахування для пристроїв наведені у таблиці 4.5.

Таблиця 4.5 – Амортизаційні відрахування

Найменування	Ціна, грн	Корисний час використання, роки	Термін використання, міс.	Сума амортизації, грн
Ноутбук	26000	4	0,73	395,4
Віддалений сервер з GPU	70000	6	0.5	486,08
Всього	881,4			

Тепер, розрахуємо витрати на комплектуючі.

Витрати на силову електроенергію розраховуються за формулою:

$$B_E = B \times П \times Ф \times K_П,$$

де B – вартість 1кВт-години електроенергії (B=4,88 грн/кВт);

$П$ – установлена потужність комп'ютеру ($П$=0,1кВт);

$Ф$ – фактична кількість годин роботи комп'ютеру ($Ф$ = 8 год х 50 днів);

$K_П$ – коефіцієнт використання потужності ($K_П$<1, $K_П$ = 0,9).

Потужність ноутбука складає 30 Вт/год = 0,3кВт/год + потужність на освітлення, тоді П = 0,1кВт/год

$$B_E = 4{,}88 \times 0{,}1 \times (8 \times 50) \times 0{,}9 = 175{,}68$$

Розрахуємо інші витрати $В_{ін}$.

Інші витрати $В_{ін}$ можна прийняти як (100-300)% від суми основної заробітної плати розробників, які виконували дану роботу, тобто:

$В_{ін} = (1...3)х(З_о+З_р)$.

Отже, розрахуємо інші витрати:

$В_{ін} = 1х(47272,95) = 47272,95$ (грн).

Усі витрати складають:

$$B = 47272{,}95 + 4727{,}3 + 11440{,}06 + 850 + 390{,}61 + 46{,}22 + 47272{,}95 = 112000{,}09 \text{ (грн)}$$

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

$$B_{заг} = \frac{B}{\alpha}$$

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

$$B_{заг} = \frac{112000{,}09}{1} = 112000{,}09 \text{ (грн)}$$

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

$$ЗВ = \frac{B_{заг}}{\beta}$$

Отже, розрахуємо прогнозовані загальні витрати:

$$ЗВ = \frac{112000{,}09}{0{,}7} = 160000{,}13 \text{ (грн)}$$

4.3 Прогнозування комерційних ефектів від реалізації результатів розробки

В економічному розділі магістерської кваліфікаційної роботи обґрунтовується економічна доцільність розробки методу та засобу визначення емоційного забарвлення повідомлення. На виконання усіх необхідних робіт необхідно 68 робочі дні.

Зростання чистого продукту для даного засобу можна оцінити у теперішній вартості грошей. Зростання чистого прибутку забезпечить підприємству (організації) надходження додаткових коштів, які дозволять покращити фінансові результати діяльності.

Величина збільшення чистого прибутку підприємства $\Delta П_i$ для кожного із років, протягом яких очікується отримання позитивних результатів від впровадження розробки, розраховується за формулою:

$$\Delta П_i = \sum_{1}^{n} (\Delta П_Я \times N + П_Я \Delta N)_i$$

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

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

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

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

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

В результаті впровадження результатів наукової розробки витрати на розробку алгоритму зменшаться на 200 грн (що автоматично спричинить збільшення чистого прибутку підприємства на 200 грн), а кількість користувачів збільшиться: протягом першого року – на 90 користувачів, протягом другого року – на 100 користувачів, протягом третього року – на 80 користувачів.

Реалізація продукції до впровадження результатів наукової розробки складала 400 шт., а прибуток, що його отримувало підприємство (організація) на одиницю продукції до впровадження результатів наукової розробки – 350 грн.

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

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

$$\Delta П_1 = 200 \times 400 + (350 + 200) \times 90 = 80000 + 550 \times 90 = 129500 \text{ (грн)}$$

Протягом другого року:

$$\Delta П_2 = 200 \times 400 + (350 + 200) \times (90 + 100) = 80000 + 550 \times 190 = 184500 \text{ (грн)}$$

Протягом третього року:

$$\Delta П_3 = 200 \times 400 + (350 + 200) \times (90 + 100 + 80) = 80000 + 550 \times 270 = 228500 \text{ (грн)}$$

4.4 Розрахунок ефективності вкладених інвестицій та період їх окупності

Розрахунок ефективності вкладених інвестицій передбачає проведення таких робіт:

1) Розрахуємо теперішню вартість інвестицій PV, що вкладаються в наукову розробку. Такою вартістю можемо вважати прогнозовану величину загальних витрат ЗВ на виконання та впровадження результатів НДДКР, розраховану раніше, тобто будемо вважати, що ЗВ = PV = 180000,13 грн.
2) Розрахуємо очікуване збільшення прибутку $\Delta П_i$, що його отримає підприємство (організація) від впровадження результатів наукової розробки, для кожного із років, починаючи з першого року впровадження. Таке збільшення прибутку також було розраховане нами раніше та становить:

   $\Delta П_1 = 129500$ грн, $\Delta П_2 = 184500$ грн, $\Delta П_3 = 228500$ грн.
3) Для спрощення подальших розрахунків необхідно побудувати вісь часу, на яку наносять всі платежі (інвестиції та прибутки), що мають місце під час виконання науково-дослідної роботи та впровадження її результатів.

Якщо загальні витрати ЗВ на виконання та впровадження результатів НДДКР (або теперішня вартість інвестицій PV) дорівнюють 180000,13 грн., а результати вкладених у наукову розробку інвестицій почнуть виявлятися вже в кінці другого року впровадження. То ці результати виявляться у тому, що у першому році підприємство отримає збільшення чистого прибутку на 129500 грн. відносно базового року, у другому році – збільшення чистого прибутку на 184500 грн (відносно базового року), у третьому році – збільшення чистого прибутку на 228500 грн (відносно базового року).

Тоді рисунок, що характеризує рух платежів (інвестицій та додаткових прибутків) буде мати вигляд, наведений на рис. 4.1.

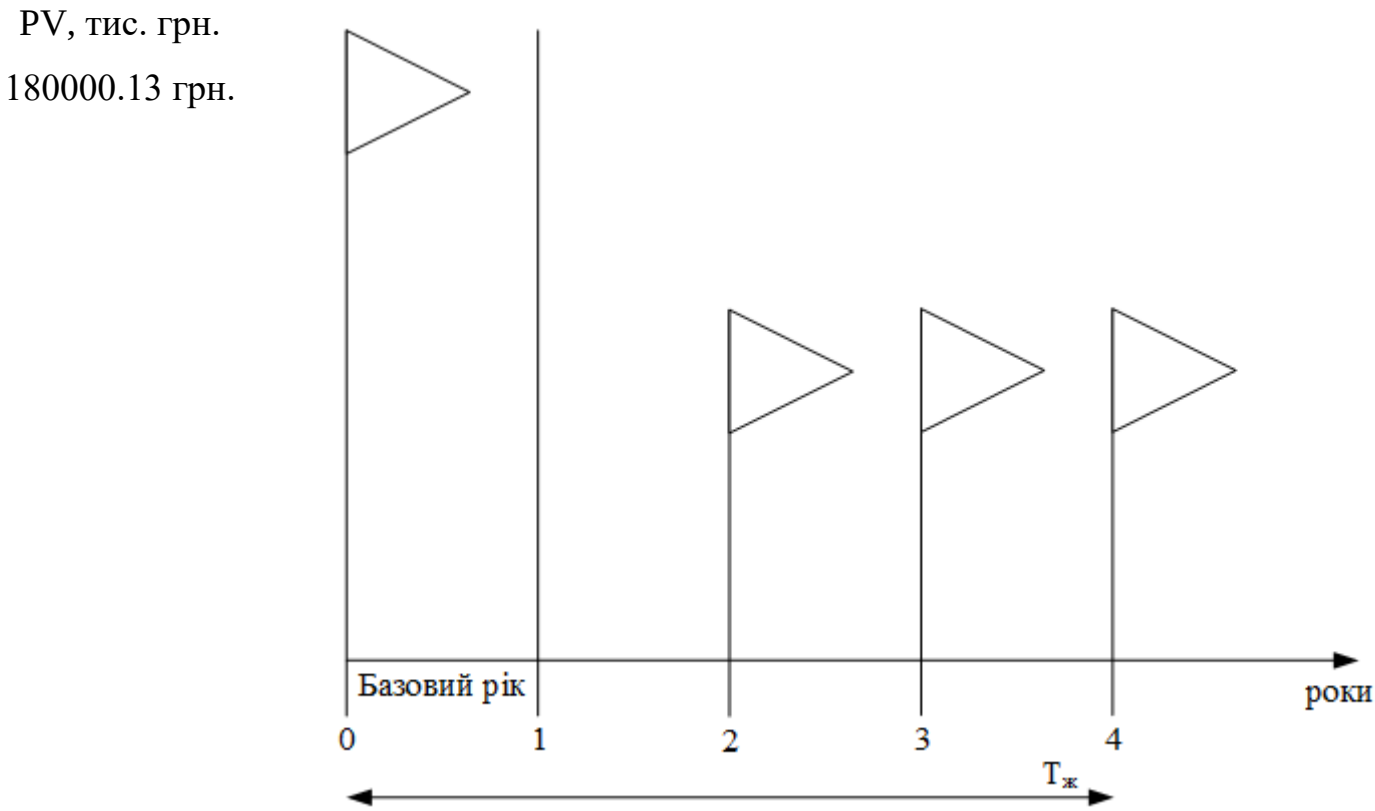


Рисунок 4.1 – Вісь часу з фіксацією платежів, що мають місце під час розробки та впровадження результатів НДДКР

4-й крок. Розрахуємо абсолютну ефективність вкладених інвестицій $E_{абс}$. Для цього скористаємося формулою:

$$E_{абс} = (ПП - PV)$$

де ПП – приведена вартість всіх чистих прибутків, що їх отримає підприємство (організація) від реалізації результатів наукової розробки, грн.;

PV – теперішня вартість інвестицій PV = ЗВ = 180000,13 грн.

У свою чергу, приведена вартість всіх чистих прибутків ПП розраховується за формулою:

$$ПП = \sum_{1}^{T} \frac{\Delta П_i}{(1+\tau)}$$

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

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

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

t – період часу (в роках) від моменту отримання чистого прибутку до точки «0».

$$ПП = \frac{129500}{(1+0,1)^2} + \frac{184500}{(1+0,1)^3} + \frac{228500}{(1+0,1)^3} = \frac{129500}{1,21} + \frac{184500}{1,331} + \frac{228500}{1,4641} =$$

$$107024,79 + 138617,58 + 156068,57 = 401710,94\ (грн)$$

$$E_{абс} = 401710,94 - 180000,13 = 221710,81\ (\text{грн})$$

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

5-й крок. Розрахуємо відносну (щорічну) ефективність вкладених в наукову розробку інвестицій E_B. Для цього використаємо формулу:

$$E_B = \sqrt[T_ж]{1 + \frac{E_{абс}}{PV}} - 1,$$

де $E_{абс}$ – абсолютна ефективність вкладених інвестицій, грн; PV – теперішня вартість інвестицій PV = ЗВ, грн; $T_ж$ – життєвий цикл наукової розробки, роки.

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

$$\tau = d + f$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; в 2021 році в Україні d = (0,14...0,2);

f – показник, що характеризує ризикованість вкладень; зазвичай, величина f = (0,05...0,1), але можу бути і значно більше.

Якщо величина $E_B > \tau_{мін}$, то інвестор може бути зацікавлений у фінансуванні даної наукової розробки. В іншому випадку фінансування наукової розробки здійснюватися не буде.

Спочатку спрогнозуємо величину $\tau_{мін}$. Припустимо, що за даних умов

$\tau_{мін} = 0{,}15 + 0{,}05 = 0{,}2$.

Тоді відносна (щорічна) ефективність вкладених інвестицій в проведення наукових досліджень та впровадження їх результатів складе:

$$E_B = \sqrt[4]{1 + \frac{221710{,}81}{180000{,}13}} - 1 = \sqrt[4]{1 + 1{,}23} - 1 = \sqrt[4]{2{,}23} - 1 = 1{,}22 - 1 = 0{,}22 \text{ або } 22\%.$$

Оскільки $E_B = 22\% > \tau_{мін} = 0{,}2 = 20\%$, то інвестор буде зацікавлений вкладати гроші в дану наукову розробку.

6-й крок. Розраховують термін окупності вкладених у реалізацію наукового проекту інвестицій. Термін окупності вкладених у реалізацію наукового проекту інвестицій $T_{ок}$ можна розрахувати за формулою:

$$T_{ОК} = \frac{1}{E_B}$$

Якщо $T_{ОК}$<3...5-ти років, то фінансування даної наукової розробки в принципі є доцільним. В інших випадках потрібні додаткові розрахунки та обґрунтування.

$$T_{ОК} = \frac{1}{0{,}26} = 3{,}8 \text{року}$$

$T_{ОК}$<5 років, що свідчить про доцільність фінансування даної наукової розробки.

Висновки: рівень комерційного потенціалу засобу для визначення фішингових інформаційних ресурсів є вище середнього. Загальні витрати на створення нового програмного продукту склали 180000,13 грн. Абсолютна ефективність капіталовкладень для даної розробки за 3 роки 221710,81 грн. Показники ефективності показують, що даний метод є доцільним і буде цікавий для інвестора. Термін окупності розробленого проекту менше 5-ти років, що підтверджує доцільність вкладання коштів в дану розробку.

ВИСНОВКИ

Провівши аналіз предметної області оцінки емоційного забарвлення тексту та порівнявши декілька методів, було обрано та покращено один із методів нейромережевого визначення контексту. На основі описаних методів було створено систему, яка здатна працювати в двох режимах: навчання та створення висновку.

Систему було реалізовано у вигляді ПЗ, яке дозволяє розгортати методи в контейнері Docker. Позитивними сторонами системи є висока гнучкість, наявність методів маркування даних, а також можливість використання GPU ресурсів машини під час тренування та висновку. Негативними аспектами можна виділити необхідність маркування даних в ручну та відсутність можливості змінити внутрішню архітектуру класифікатора.

Проведене дослідження несе значну практичну цінність як для державних органів так і для приватних підприємств. Цінність для державних органів заключається у боротьбі з підривною діяльністю і мережі Інтернет, а цінністю для приватних підприєств є можливість поведінкового аналізу клієнта для покращення взаємодії з ним.

За допомогою проведення економічних розрахунків було доведено комерційний потенціал та доцільність розробки системи визначення емоційного забарвлення повідомлення. У ході дослідження визначено термін окупності – 2 роки і 6 місяців.

Дослідження вважається завершеним в рамках даної роботи і може бути застосовано, як окремий продукт систем на підприємствах. Моделі можуть дотреновуватися у процесі використання, за рахунок чого точність буде підвищуватись.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. D. T. and N. Deepa, "A novel intervention method for aspect-based emotion Using Exponential Linear Unit (ELU) activation function in a Deep Neural Network," 2021 5th International Conference on Intelligent Computing and Control Systems (ICICCS), 2021, pp. 1671-1675, doi: 10.1109/ICICCS51141.2021.9432223.
2. N. Irtiza Tripto and M. Eunus Ali, "Detecting Multilabel Sentiment and Emotions from Bangla YouTube Comments," 2018 International Conference on Bangla Speech and Language Processing (ICBSLP), 2018, pp. 1-6, doi: 10.1109/ICBSLP.2018.8554875.
3. Word2Vec | Tensorflow Core [Електронний ресурс], URL: https://www.tensorflow.org/tutorials/text/word2vec
4. M. Abdullah, M. Hadzikadicy and S. Shaikhz, "SEDAT: Sentiment and Emotion Detection in Arabic Text Using CNN-LSTM Deep Learning," 2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA), 2018, pp. 835-840, doi: 10.1109/ICMLA.2018.00134.
5. Chatterjee A, Narahari KN, Joshi M, Agrawal P. SemEval-2019 task 3: EmoContext contextual emotion detection in text. Paper presented at: Proceedings of the 13th International Workshop on Semantic Evaluation; 2019:39-48.
6. How do Transformers Work in NLP? A Guide to the Latest State-of-the-Art Models, URL: https://www.analyticsvidhya.com/blog/2019/06/ understanding-transformers-nlp-state-of-the-art-models/
7. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. In Advances in neural information processing systems (pp. 5998-6008).
8. Andrea Chiorrini, Claudia Diamantini, Alex Mircoli, Domenico Potena: Emotion and sentiment analysis of tweets. URL: http://ceur-ws.org/Vol-2841/DARLI-AP_17.pdf
9. Understanding TF-ID: A Simple Introduction URL: https://monkeylearn.com/blog/what-is-tf-idf/

10. TF-IDF from scratch in python on a real-world dataset. URL: https://towardsdatascience.com/tf-idf-for-document-ranking-from-scratch-in-python-on-real-world-dataset-796d339a4089
11. Clustering - scikit-learn 10.0.1 documentation. URL: https://scikit-learn.org/stable/modules/clustering.html#k-means
12. Pairwise metrics, Affinities and Kernels. URL: https://scikit-learn.org/stable/modules/metrics.html#metrics
13. Edy Umargono, Jadmiko Endro Suseno, Vincensius Gunawan S. K., K-Means Clustering Optimization using the Elbow Method and Early Centroid Determination Based-on Mean and Median URL: https://www.scitepress.org/Papers/2019/99084/99084.pdf
14. David M., Andrew Y. Ng, Michael I. Jordan. Latent Dirichlet Allocation. URL: https://www.jmlr.org/papers/volume3/blei03a/blei03a.pdf
15. Латентное размещение Дирихле (LDA). URL: https://lambda-it.ru/post/tematicheskoe-modelirovanie-v-deistvii-lda
16. What is tSNE and when should I use it? - Sonrai Analytics. URL: https://sonraianalytics.com/what-is-tsne/
17. t-SNE Machine Learning Algorithm — A Great Tool. URL: http://datareview.info/article/algoritm-t-sne-illyustrirovannyiy-vvodnyiy-kurs/
18. Evaluation Metrics For Machine Learning For Data Scientists. URL: https://www.analyticsvidhya.com/blog/2020/10/quick-guide-to-evaluation-metrics-for-supervised-and-unsupervised-machine-learning/
19. BERT - Hugging Face. URL : https://huggingface.co/docs/transformers/model_doc/bert
20. 37 моя
21. Optimization Algorithms - Dive into Deep Learning URL:https://d2l.ai/chapter_optimization/
22. How to Choose an Optimization Algorithm - https://machinelearningmastery.com/tour-of-optimization-algorithms/

23. GENERATOR URL: https://pytorch.org/docs/stable/generated/torch.Generator.html
24. What are the pros and cons of Python versus R for machine learning and data science purposes? [Електронне джерело] – 2018 – Режим доступу до ресурсу: https://www.quora.com/What-are-the-pros-and-cons-of-Python-versus-R-for-machine-learning-and-data-science-purposes
25. Python Vs R: What's Best for Machine Learning [Електронне джерело] – 2019 – Режим доступу до ресурсу: https://towardsdatascience.com/python-vs-r-whats-best-for-machine-learning-93432084b480
26. Pros and Cons of Python in Machine Learning [Електронне джерело] – 2018 – Режим доступу до ресурсу: https://www.zarantech.com/blog/pros-and-cons-of-python-in-machine-learning/
27. ПАРАЛЛЕЛЬНЫЕ ВЫЧИСЛЕНИЯ С CUDA [Електронне джерело] – 2016 – Режим доступу до ресурсу https://www.nvidia.com.ua/object/cuda-parallel-computing-ru.html

ДОДАТКИ

ДОДАТОК А

Міністерство освіти і науки України
Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра захисту інформації

ЗАТВЕРДЖУЮ
Завідувач кафедри ЗІ, д. т. н., проф.
_____В. А. Лужецький
«____»_____2021 р.

ТЕХНІЧНЕ ЗАВДАННЯ
на виконання магістерської кваліфікаційної роботи
на тему: «Метод та засіб розпізнавання емоційного забарвлення повідомлення під час інформаційної війни»
08-20.МКР.019.00.000 ТЗ

Керівник магістерської дипломної
роботи к. т. н., доцент каф. ЗІ Войтович О. П.

Вінниця 2021

1 Підстави для проведення робіт

Робота проводиться на підставі наказу ректора ВНТУ від 24 вересня 2021 року № 277.

Дата початку роботи 09.09.21 р.

Дата закінчення роботи 18.12.21 р.

2 Мета та призначення НДР

Метою магістерської кваліфікаційної роботи є покращення розпізнавання емоційного забарвлення повідомлення під час інформаційної війни

Об'єктом дослідження є процеси, задіяні при розпізнавання емоційного забарвлення повідомлень.

Предметом дослідження є методи та засоби, що використовуються для визначення емоційного забарвлення повідомлень.

Актуальність теми. Сьогодні, задачі обробки натуральної мови являються найбільш актуальними, особливо ті, які стосуються контексту емоційного забарвлення. Стрімкий розвиток соціальних мереж призвів до того, що рівень довіри до них став занадто високим і людьми стало значно простіше керувати через агітативні повідомлення.

В контексті інформаційної війни існує ризик можливого впливу на людину, шляхом маніпуляцій емоціями суперника. Одним з засобів, що допомагає детермінувати даний процес є визначення емоційного контексту інформаційних повідомлень.

3 Джерела розробки

7.1. Дудатьев А.В. Комплексный метод противодействия информационно-психологическим операциям // Проблемы управления и информатики. - Киев: Институт кибернетики им. В. М. Глушкова НАН Украины, 2017. - № 1.

7.2. Дудатьев А.В., Лужецкий В.А., Коротаев Д.О. Метод оценки информационной устойчивости социотехнических систем в условиях информационной войны / А.В. Дудатьєв, В.А Лужецкий, Д.А. Коротаев // Восточно-Европейский журнал передовых технологий. – 2016. – № 1.

7.3.Лужецький В. А., Дудатьєв А. В. Концептуальна модель системи інформаційного впливу // Український науковий журнал з інформаційної безпеки. - 2017. - Т. 23, № 1. 97

7.4. Ткачук, Т. Ю. Механізми протидії інформаційним загрозам зовнішніх джерел // Вісник НТУУ «КПІ». Політологія. Соціологія. Право : збірник наукових праць. – 2017. – № 1/2 (33/34).

7.5. Stone, P. A computer approach to content analysis: Studies using the general inquirer system // Spring Joint Computer Conference, AFIPS '63 (Spring) – New York: ACM, 1963. – p. 241–256.

7.6. Daniel W. Otter, Julian R. Medina, Jugal K. Kalita: A Survey of the Usages of Deep Learning for Natural Language Processing // IEEE

TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS, VOL. XX, NO. X, JULY 2019, URL: https://arxiv.org/pdf/1807.10854.pdf

4 Виконавці НДР

Студент групи 1БС-20м Майстренко В'ячеслав Олегович

5 Вимоги до виконання НДР

Для пришвидшення процесів оцінювання тональності тесту та реакції на виявлений вплив необхідно розв’язати такі задачі:

– науково-досліджене обґрунтування необхідності дослідження та розробки системи оцінювання тональності інформаційного повідомлення;

– аналіз наукової літератури щодо існуючих методів та засобів визначення тональності тексту;

– удосконалення лесемного методу оцінки тональності, шляхом розгляду не тільки окремих слів, а й словосполучень, що складаються з двох слів;

– розробка алгоритмів функціонування модулів системи;

– розробка системи згідно розроблених алгоритмів функціонування;

– експериментальне дослідження розробленої системи.

6 Вимоги до супровідної документації

Графічна і текстова документація повинна відповідати діючим стандартам України.

7 Вимоги до супровідної документації

Робота з теми виконується в чотири етапи

Зміст етапу	Початок	Закінчення	Результат
Аналіз завдання. Вступ	09.09.2021	14.09.2021	Вступ, технічне завдання
Розробка технічного завдання	15.09.2021	21.09.2021	Проект технічного завдання
Аналіз літературних джерел за напрямком магістерської кваліфікаційної роботи	22.09.2021	29.09.2021	Аналіз відомих методів. Постановка завдання
Покращення класифікатору емоційного забарвлення	30.09.2020	11.10.2020	Покращений класифікаційний метод емоційного забарвлення
Створення макету системи	11.10.2021	01.11.2021	Макет системи для класифікації емоційного забарвлення

Зміст етапу	Початок	Закінчення	Результат
Експериментальні дослідження	02.11.2021	07.11.2021	Система, яка реалізує розроблений метод
Розробка економічного розділу	08.11.2021	15.11.2021	Економічні показники дослідження
Оформлення пояснювальної записки	16.11.2021	10.12.2021	Пояснювальна записка

8 Очікувані результати та порядок реаліації НДР

Передбачається розробка удосконалення існуючого методу визначення тональності інформаційного повідомлення. Заплановане створення програмного засобу, який може бути використаний як в державних установах так і на пдприємствах.

9 Матеріали які подаються після закінчення НДР

По завершенню роботи подається пояснювальна записка та ілюстративна частина.

10 Порядок приймання НДР та її етапів

Апробація на науково-технічних конференціях та семінарах. Результати роботи будуть розглядатися на засіданні ДЕК із захисту магістерських кваліфікаційних робіт.

Попередній захист та доопрацювання МКР грудень 2021 р.

Представлення МКР до захисту 18 грудня 2021 р.

Захист МКР 23.12.21.

11 Вимоги до розроблення документації

Документація буде виконуватись за допомогою комп’ютерного набору у відповідності вимог ДСТУ 3008:2015 «Інформація та документація. Звіти у сфері науки і техніки. Структура та правила оформлювання».

12 Вимоги щодо технічного захисту інформації з обмеженим доступом

У зв’язку з тим, що дана робота не містить інформації, що потребує захисту у відповідності до законів України, заходи з її технічного захисту не передбачаються.

Розробив студент 1БС-20м . Майстренко В. О.

ДОДАТОК Б

Текст програми

Trainer.py

```
import os
import pickle

import pandas as pd
import torch
import time

from torch.utils import data
from tqdm.cli import tqdm as tqdmn
#from tqdm.notebook import tqdm as tqdmn
import random
import numpy as np

# Preliminaries

from torch.utils.data import Dataset, DataLoader
import torch.utils.data as data_utils
from torchtext.legacy.data import Field, TabularDataset, BucketIterator, Iterator

# Models

import torch.nn as nn
from transformers import BertTokenizer, BertForSequenceClassification, AutoModel, AutoTokenizer,
AutoModelForSequenceClassification

# Training

import torch.optim as optim
from transformers import AdamW, get_linear_schedule_with_warmup

# Evaluation

from sklearn import metrics, model_selection, preprocessing
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
from sklearn.preprocessing import LabelBinarizer
from transformers.file_utils import cached_path

model_baseline = "models/pretrained/cointegrated-tiny"
model_bert = 'bert-base-uncased'

config = {
    'method': 'random', #grid, random, bayesian
    'metric': {
      'name': 'auc_score',
      'goal': 'maximize'
    },
    'parameters': {
        'lr':0.0004,
        'batch_size':32,
        'epochs':100,
        'dropout':0.1,
        'tokenizer_max_len':128,
        'bert_model_name':model_baseline,
        'output_path':f"models/trained/{model_baseline}/",
        'device':torch.device('cuda:1'),
        'drop_perc':0.9,
    },
    'seed':8,
}
ru_sent = pd.read_csv('data/input/rusentiment_nnp.csv')
labelEncoder = LabelBinarizer().fit(ru_sent.label.values)
del ru_sent
```

```
def seed_everything(seed=42):
    random.seed(seed)
    os.environ['PYTHONHASHSEED'] = str(seed)
    np.random.seed(seed)
    torch.manual_seed(seed)
    torch.cuda.manual_seed(seed)
    torch.cuda.manual_seed_all(seed)
    # Some cudnn methods can be random even after fixing the seed
    # unless you tell it to be deterministic
    torch.backends.cudnn.deterministic = True

class TextDataset(Dataset):
    def __init__(self, path, tokenizer, max_len, drop=None):
        self.tokenizer = tokenizer
        self.MAX_LEN = max_len
        # mapper = {
        #     'negative':0,
        #     'neutral':1,
        #     'positive':2
        # }
        self.data = pd.read_csv(path)
        self.data['text'] = self.data.text.apply(self.__tokenize)
        # self.data['label'] = self.data.label.map(mapper)
        self.data = self.data.values
        # da1 = self.data[self.data.label=='neutral'][:10]
        # da2 = self.data[self.data.label=='positive'][:20]
        # da3 = self.data[self.data.label=='negative'][:20]
        # self.data = pd.concat([da1, da2, da3]).values
#         else:
#             data = pd.read_csv(path)
#             output = []
# #             for grname, group in data.groupby('label'):
# #                 group['label'] = np.full(len(group), grname, dtype=int)
# #                 num_of_samples = int(drop*len(group))
# #                 output.append(group[:num_of_samples])
#             self.data = pd.concat(output).values[:10, :]
        self.len = self.data.shape[0]

    def __getitem__(self, index):
        txt, label = self.data[index, :]
        return txt, labelEncoder.transform([label]).squeeze()

    def __tokenize(self, txt):
        return self.tokenizer.encode(
                txt, padding="max_length", truncation=True,
                max_length=self.MAX_LEN, return_tensors='pt')

    def __len__(self):
        return self.len


def load_model_tokenizer(model_name):
    model = AutoModel.from_pretrained(f"{model_name}-model")
    tokenizer = AutoTokenizer.from_pretrained(f"{model_name}-tokenizer")
    return model, tokenizer

def load_datasets(root_path, train_name, valid_name, tokenizer):
    MAX_SEQ_LEN = config['parameters']['tokenizer_max_len']
    BATCH_SIZE = config['parameters']['batch_size']
    drop = None
    drop = config['parameters']['drop_perc']

    train = TextDataset(path=f"{root_path}/{train_name}",
                        tokenizer=tokenizer,
                        max_len = MAX_SEQ_LEN,
                       drop=drop)
    valid = TextDataset(path=f"{root_path}/{valid_name}",
                        tokenizer=tokenizer,
```

```
                        max_len = MAX_SEQ_LEN,
                        drop=drop)

    train = DataLoader(dataset=train, batch_size=BATCH_SIZE, drop_last=True)
    valid = DataLoader(dataset=valid, batch_size=BATCH_SIZE, drop_last=True)

    return train, valid

class BERT_Classifier(nn.Module):
    def __init__(self, bert_model, classes_num, do_prob=0.1):
        super(BERT_Classifier, self).__init__()
        self.encoder = bert_model
        self.dropout = nn.Dropout(do_prob)
        self.out = nn.Linear(768, classes_num)
        self.sigmoid = nn.Sigmoid()

    def forward(self, ids):
        ids = torch.squeeze(ids, 1)
        mask = torch.clamp(ids, 0, 1)
        out = self.encoder(ids, attention_mask=mask)["pooler_output"]
        out = self.dropout(out)
        out = self.out(out)
        out = self.sigmoid(out)
        return out

class tinyBERT_Classifier(nn.Module):
    def __init__(self, bert_model, classes_num, do_prob=0.1):
        super(tinyBERT_Classifier, self).__init__()
        self.encoder = bert_model
        self.encoder.eval()
        self.dropout = nn.Dropout(do_prob)
        self.hidden = nn.Sequential(
            nn.Linear(312, 312), nn.ReLU(),
            nn.Linear(312, 128), nn.ReLU(),
            # nn.Linear(128, 32), nn.ReLU()
            # nn.Linear(128, classes_num),
        )
        self.out = nn.Linear(128, classes_num)
        self.sigmoid = nn.Softmax(dim=1)
        self.weights_init_normal(self.out)
        self.weights_init_normal(self.hidden)

    def weights_init_normal(self, m):
        '''Takes in a module and initializes all linear layers with weight
           values taken from a normal distribution.'''

        classname = m.__class__.__name__
        # for every Linear layer in a model
        if classname.find('Linear') != -1:
            y = m.in_features
        # m.weight.data shoud be taken from a normal distribution
            m.weight.data.normal_(0.0, 0.02)
        # m.bias.data should be 0
            m.bias.data.fill_(0)

    def embed_bert_cls(self, ids, mask):
        with torch.no_grad():
            model_output = self.encoder(ids, mask)
        embeddings = model_output.last_hidden_state[:, 0, :]
        return torch.nn.functional.normalize(embeddings)

    def forward(self, ids):
        ids = torch.squeeze(ids, 1)
        mask = torch.clamp(ids, 0, 1)
        out = self.embed_bert_cls(ids, mask)
        out = self.dropout(out)
        out = self.hidden(out)
        out = self.out(out)
        out = self.sigmoid(out)
```

```
        return out

# def loss_fn(predicted, true):
#     if true is None: return None
#     return nn.BCELoss()(predicted, true.float())
def loss_fn(outputs, labels):
    if labels is None:
        return None
    return nn.BCEWithLogitsLoss()(outputs, labels.float())

def ret_optimizer(model):
    '''
    Taken from Abhishek Thakur's Tez library example:
    https://github.com/abhishekkrthakur/tez/blob/main/examples/text_classification/binary.py
    '''
    param_optimizer = list(model.named_parameters())
    no_decay = ["bias", "LayerNorm.bias"]
    optimizer_parameters = [
        {
            "params": [
                p for n, p in param_optimizer if not any(nd in n for nd in no_decay)
            ],
            "weight_decay": 0.001,
        },
        {
            "params": [
                p for n, p in param_optimizer if any(nd in n for nd in no_decay)
            ],
            "weight_decay": 0.0,
        },
    ]
    opt = AdamW(optimizer_parameters, lr=config['parameters']['lr'])
    return opt

def ret_scheduler(optimizer, num_train_steps):
    sch = get_linear_schedule_with_warmup(
        optimizer, num_warmup_steps=0, num_training_steps=num_train_steps)
    return sch


def log_metrics(preds, labels):

    '''
    ##Method 1 by taking transpose and picking each column for averaging

    auc_micro_list = []
    for i in range(n_labels):
      current_pred = preds.T[i]
      current_label = labels.T[i]
      fpr_micro, tpr_micro, _ = metrics.roc_curve(current_label.T, current_pred.T)
      auc_micro = metrics.auc(fpr_micro, tpr_micro)
      auc_micro_list.append(auc_micro)

    return {"auc": np.array(auc_micro).mean()}
    '''

    ## Method 2 using ravel()
    labels = labels[..., None]
    preds = preds[..., None]
    acc = metrics.accuracy_score(labels, preds)


    return {"acc": acc}

def train_fn(data_loader, model, optimizer, device, scheduler, criterion, epoch_n):
    '''
        Modified from Abhishek Thakur's BERT example:
        https://github.com/abhishekkrthakur/bert-sentiment/blob/master/src/engine.py
    '''
```

```
    train_loss = 0.0
    bn = len(data_loader)
    model.train()
    with tqdmn(data_loader, unit='batch') as tepoch:
        tepoch.set_description(f"Epoch {epoch_n}")
        for batch in tepoch:
            ids, targets = batch
        #targets = torch.reshape(targets, targets.shape+tuple([1]))

            ids = ids.to(device, dtype=torch.long)
            targets = targets.to(device, dtype=torch.float)

            optimizer.zero_grad()
            outputs = model(ids=ids)

            loss = criterion(outputs, targets)
            train_loss += loss.item()
            loss.backward()
            # torch.nn.utils.clip_grad_norm_(model.parameters(), 1.0)
            optimizer.step()
            #scheduler.step()

            tepoch.set_postfix(loss=train_loss/bn)

    return train_loss


def eval_fn(data_loader, model, device, criterion, epoch_n):
    '''
        Modified from Abhishek Thakur's BERT example:
        https://github.com/abhishekkrthakur/bert-sentiment/blob/master/src/engine.py
    '''
    eval_loss = 0.0
    model.eval()
    bn = len(data_loader)
    fin_targets = []
    fin_outputs = []
    with torch.no_grad(), tqdmn(data_loader, unit='batch') as tepoch:
        tepoch.set_description(f"Epoch {epoch_n}")
        for batch in tepoch:
            ids, targets = batch
            #targets = torch.reshape(targets, targets.shape+tuple([1]))

            ids = ids.to(device, dtype=torch.long)
            targets = targets.to(device, dtype=torch.float)

            outputs = model(ids=ids)
            loss = criterion(outputs, targets)
            eval_loss += loss.item()
            fin_targets.extend(targets)
            fin_outputs.extend(outputs)
            tepoch.set_postfix(loss=eval_loss/bn)

                            return       eval_loss,       torch.stack(fin_outputs).argmax(dim=1),
torch.stack(fin_targets).argmax(dim=1)

def save_dict(path, filename, data):
    if not os.path.exists(path):
        os.mkdir(path)
    with open(f"{path}/{filename}", 'wb+') as f:
        pickle.dump(data, f)

def trainer():
    params =  config["parameters"]
    bert_model, tokenizer = load_model_tokenizer(params['bert_model_name'])
    train_dataset, valid_dataset = load_datasets(root_path='data/input/',
                                                 train_name='rusentiment_nnp_train.csv',
                                                 valid_name='rusentiment_nnp_valid.csv',
```

```
                                                tokenizer=tokenizer)
    for param in bert_model.parameters():
        param.requires_grad = False
    print("Length of Train Dataloader: ", len(train_dataset))
    print("Length of Valid Dataloader: ", len(valid_dataset))

    device = params["device"]
    dropout = params['dropout']
    opath = params['output_path']
    n_train_steps = len(train_dataset) // params["batch_size"]


    model = tinyBERT_Classifier(bert_model=bert_model,
                            classes_num=len(labelEncoder.classes_),
                            do_prob=dropout)
    #optimizer = ret_optimizer(model)
    #scheduler = ret_scheduler(optimizer, n_train_steps)
    #compute the class weights
    # tr, ts = pd.read_csv(f'../data\input\rusentiment_nnp.csv')
    # class_count = [i for i in get_class_distribution(y_train).values()]
    # class_weights = 1./torch.tensor(class_count, dtype=torch.float)
    # weights= torch.tensor(class_weights,dtype=torch.float)
    # weights = weights.to(device)

    # define the loss function
    criterion  = torch.nn.BCEWithLogitsLoss().to(device)
    optimizer = torch.optim.AdamW(model.parameters(), lr=params['lr'])
    scheduler = None
    model.to(device)
    n_epochs = params['epochs']

    best_val_loss = 100
    for epoch in tqdmn(range(n_epochs)):
        train_loss = train_fn(train_dataset, model, optimizer, device, scheduler, criterion,
epoch)
        eval_loss, preds, labels = eval_fn(valid_dataset, model, device, criterion, epoch)

        yp = preds.cpu().detach().numpy().ravel()
        yt = labels.cpu().detach().numpy().ravel()
        # auc_score = log_metrics(yp, yt)
            avg_train_loss, avg_val_loss = train_loss / len(train_dataset), eval_loss /
len(valid_dataset)
        yp, yt = np.round(yp), np.round(yt)
        cm = metrics.confusion_matrix(yp, yt)
        save_dict(
            opath, f'epoch_{epoch}.pkl',
            {
                "epoch": epoch + 1,
                "train_loss": avg_train_loss,
                "val_loss": avg_val_loss,
                # "metrics": auc_score,
                "cm": cm
            }
        )
        # print(auc_score)
        # print("Average Train loss: ", avg_train_loss)
        # print("Average Valid loss: ", avg_val_loss)

        if avg_val_loss < best_val_loss:
            best_val_loss = avg_val_loss
            torch.save(model.state_dict(), f"{opath}/best_model.pt")
            print("Model saved as current val_loss is: ", best_val_loss)

if __name__ =='__main__':
    seed_everything(config['seed'])
    trainer()
```

Worker.py

```
from fastapi import FastAPI, HTTPException, Query, Request
from fastapi.middleware.cors import CORSMiddleware
from easynmt import EasyNMT
from typing import Optional, Union, List
import time
import os
import json
import time
import datetime
import requests
import http3



app = FastAPI()
app.add_middleware(
    CORSMiddleware,
    allow_origins=["*"],
    allow_credentials=True,
    allow_methods=["*"],
    allow_headers=["*"],
)


IS_BACKEND = os.getenv('ROLE', 'FRONT') == 'BACKEND'
BACKEND_URL = os.getenv('BACKEND_URL', 'http://localhost:8080')

print("Booted as backend: {}".format(IS_BACKEND))

model_name = os.getenv('EASYNMT_MODEL', 'opus-mt')
model_args = json.loads(os.getenv('EASYNMT_MODEL_ARGS', '{}'))
print("Load model: "+ model_name)
model = EasyNMT(model_name, load_translator=IS_BACKEND, **model_args)


@app.get("/translate")
async def translate(target_lang: str, text: List[str] = Query([]), source_lang: Optional[str] =
'', beam_size: Optional[int] = 5, perform_sentence_splitting: Optional[bool] = True):
    """
    Translates the text to the given target language.
    :param text: Text that should be translated
    :param target_lang: Target language
    :param source_lang: Language of text. Optional, if empty: Automatic language detection
    :param beam_size: Beam size. Optional
      :param perform_sentence_splitting: Split longer documents into individual sentences for
translation. Optional
    :return:  Returns a json with the translated text
    """

    if not IS_BACKEND:
        async_client = http3.AsyncClient()
       data = {'target_lang': target_lang, 'text': text, 'source_lang': source_lang, 'beam_size':
beam_size, 'perform_sentence_splitting': perform_sentence_splitting}
        x = await async_client.post(BACKEND_URL+'/translate', json=data, timeout=3600)
        if x.status_code != 200:
            error_msg = "Error: " + x.text
            try:
                error_msg = x.json()['detail']
            except:
                pass

            raise HTTPException(403, detail=error_msg)
```

```
        return x.json()

    else:
        #Check input parameters
                        if 'EASYNMT_MAX_TEXT_LEN' in os.environ and len(text) >
int(os.getenv('EASYNMT_MAX_TEXT_LEN')):
                raise ValueError("Text was too long. Only texts up to {} characters are
allowed".format(os.getenv('EASYNMT_MAX_TEXT_LEN')))

            if beam_size < 1 or ('EASYNMT_MAX_BEAM_SIZE' in os.environ and beam_size >
int(os.getenv('EASYNMT_MAX_BEAM_SIZE'))):
            raise ValueError("Illegal beam size")

        if len(source_lang.strip()) == 0:
            source_lang = None

        #Start the translation
        start_time = time.time()
        output = {"target_lang": target_lang, "source_lang": source_lang}

        if source_lang is None:
            detected_langs = model.language_detection(text)
            output['detected_langs'] = detected_langs
            #TODO Grouping and individual translation
            #Exception: add original text

        try:
                        output['translated'] = model.translate(text, target_lang=target_lang,
source_lang=source_lang,                                                    beam_size=beam_size,
perform_sentence_splitting=perform_sentence_splitting,
batch_size=int(os.getenv('EASYNMT_BATCH_SIZE', 16)))
        except Exception as e:
            raise HTTPException(403, detail="Error: "+str(e))

        output['translation_time'] = time.time()-start_time
        return output

@app.post("/translate")
async def translate_post(request: Request):
    """
    Post method for translation
    :return:
    """
    data = await request.json()
    return await translate(**data)


@app.get("/translate")
async def translate(target_lang: str, text: List[str] = Query([]), source_lang: Optional[str] =
'', beam_size: Optional[int] = 5, perform_sentence_splitting: Optional[bool] = True):
    """
    Translates the text to the given target language.
    :param text: Text that should be translated
    :param target_lang: Target language
    :param source_lang: Language of text. Optional, if empty: Automatic language detection
    :param beam_size: Beam size. Optional
      :param perform_sentence_splitting: Split longer documents into individual sentences for
translation. Optional
    :return:  Returns a json with the translated text
    """

    if not IS_BACKEND:
        async_client = http3.AsyncClient()
       data = {'target_lang': target_lang, 'text': text, 'source_lang': source_lang, 'beam_size':
beam_size, 'perform_sentence_splitting': perform_sentence_splitting}
        x = await async_client.post(BACKEND_URL+'/translate', json=data, timeout=3600)
        if x.status_code != 200:
            error_msg = "Error: " + x.text
            try:
```

```
                error_msg = x.json()['detail']
            except:
                pass

            raise HTTPException(403, detail=error_msg)
        return x.json()

    else:
        #Check input parameters
                        if 'EASYNMT_MAX_TEXT_LEN' in os.environ and len(text) >
int(os.getenv('EASYNMT_MAX_TEXT_LEN')):
                raise ValueError("Text was too long. Only texts up to {} characters are
allowed".format(os.getenv('EASYNMT_MAX_TEXT_LEN')))

            if beam_size < 1 or ('EASYNMT_MAX_BEAM_SIZE' in os.environ and beam_size >
int(os.getenv('EASYNMT_MAX_BEAM_SIZE'))):
            raise ValueError("Illegal beam size")

        if len(source_lang.strip()) == 0:
            source_lang = None

        #Start the translation
        start_time = time.time()
        output = {"target_lang": target_lang, "source_lang": source_lang}

        if source_lang is None:
            detected_langs = model.language_detection(text)
            output['detected_langs'] = detected_langs
            #TODO Grouping and individual translation
            #Exception: add original text

        try:
                        output['translated'] = model.translate(text, target_lang=target_lang,
source_lang=source_lang,                                                    beam_size=beam_size,
perform_sentence_splitting=perform_sentence_splitting,
batch_size=int(os.getenv('EASYNMT_BATCH_SIZE', 16)))
        except Exception as e:
            raise HTTPException(403, detail="Error: "+str(e))

        output['translation_time'] = time.time()-start_time
        return output

@app.post("/translate")
async def translate_post(request: Request):
    """
    Post method for translation
    :return:
    """
    data = await request.json()
    return await translate(**data)

@app.get("/lang_pairs")
async def lang_pairs():
    """
    Returns the language pairs from the model
    :return:
    """
    return model.lang_pairs

@app.get("/get_languages")
async def get_languages(source_lang: Optional[str] = None, target_lang: Optional[str] = None):
    """
    Returns the languages the model supports
    :param source_lang: Optional. Only return languages with this language as source
    :param target_lang: Optional. Only return languages with this language as target
    :return:
    """
    return model.get_languages(source_lang=source_lang, target_lang=target_lang)
```

```python
@app.get("/language_detection")
async def language_detection(text: str):
    """
    Detects the language for the provided text
    :param text: A single text for which we want to know the language
    :return: The detected language
    """
    return model.language_detection(text)

@app.post("/language_detection")
async def language_detection_post(request: Request):
    """
     Pass a json that has a 'text' key. The 'text' element can either be a string, a list of strings, or
    a dict.
    :return: Languages detected
    """
    data = await request.json()
    if isinstance(data['text'], list):
        return [model.language_detection(t) for t in data['text']]
    elif isinstance(data['text'], dict):
        return {k: model.language_detection(t) for k, t in data['text'].items()}
    return model.language_detection(data['text'])



@app.get("/model_name")
async def model_name():
    """
    Returns the name of the loaded model
    :return: EasyNMT model name
    """
    return model._model_name
```

cuda11.builder.dockerfile

```
FROM pytorch/pytorch:1.8.0-cuda11.1-cudnn8-runtime
LABEL maintainer="Nils Reimers <info@nils-reimers>"

####################################### Same code for all docker files ###############

## Install dependencies
RUN apt-get update && apt-get -y install build-essential
RUN pip install --no-cache-dir "uvicorn[standard]" gunicorn fastapi
COPY ./requirements.txt /requirements.txt
RUN pip install --no-cache-dir -r /requirements.txt
RUN python -m nltk.downloader 'punkt'

#### Scripts to start front- and backend worker

COPY ./start_backend.sh /start_backend.sh
RUN chmod +x /start_backend.sh

COPY ./start_frontend.sh /start_frontend.sh
RUN chmod +x /start_frontend.sh

COPY ./start.sh /start.sh
RUN chmod +x /start.sh

COPY ./gunicorn_conf_backend.py /gunicorn_conf_backend.py
COPY ./gunicorn_conf_frontend.py /gunicorn_conf_frontend.py

#### Woking dir

COPY ./src /app
```

```
WORKDIR /app/
ENV PYTHONPATH=/app
EXPOSE 80

####

# Create cache folders
RUN mkdir /cache/
RUN mkdir /cache/easynmt
RUN mkdir /cache/transformers
RUN mkdir /cache/torch

ENV EASYNMT_CACHE=/cache/easynmt
ENV TRANSFORMERS_CACHE=/cache/transformers
ENV TORCH_CACHE=/cache/torch

# Run start script
CMD ["/start.sh"]
```

scriptrunner.sh

```
#! /usr/bin/env sh
set -e

if [ -f /app/app/main.py ]; then
    DEFAULT_MODULE_NAME=app.main
elif [ -f /app/main.py ]; then
    DEFAULT_MODULE_NAME=main
fi
MODULE_NAME=${MODULE_NAME:-$DEFAULT_MODULE_NAME}
VARIABLE_NAME=${VARIABLE_NAME:-app}
export APP_MODULE=${APP_MODULE:-"$MODULE_NAME:$VARIABLE_NAME"}

DEFAULT_GUNICORN_CONF=/gunicorn_conf_backend.py
export GUNICORN_CONF=${GUNICORN_CONF:-$DEFAULT_GUNICORN_CONF}
export WORKER_CLASS=${WORKER_CLASS:-"uvicorn.workers.UvicornWorker"}

# If there's a prestart.sh script in the /app directory or other path specified, run it before
starting
PRE_START_PATH=${PRE_START_PATH:-/app/prestart.sh}
echo "Checking for script in $PRE_START_PATH"
if [ -f $PRE_START_PATH ] ; then
    echo "Running script $PRE_START_PATH"
    . "$PRE_START_PATH"
else
    echo "There is no script $PRE_START_PATH"
fi

# Start Gunicorn
ROLE=BACKEND exec gunicorn -k "$WORKER_CLASS" -c "$GUNICORN_CONF" "$APP_MODULE"
```

Parser.py

```
import requests
import base64
from bs4 import BeautifulSoup
import urllib.request
import time
import locale
import os
import json
import numpy as np
from random import choice
from proxy_hand import ProxyHandler
locale.setlocale(locale.LC_TIME, 'uk_ua')
```

```
class RelativeContext(object):
    def __init__(self, filename, access_method):
        self.filename = os.path.join(os.getcwd(), filename)
        self.access_method = access_method

    def __enter__(self):
        self.file = open(self.filename, self.access_method)
        return self.file

    def __exit__(self, *args):
        self.file.close()

class ScrapperTools:

    @staticmethod
    def load_json(path):
        with RelativeContext(path, 'r') as file:
            return json.load(file)

    @staticmethod
    def basic_load(url, proxy=None) -> BeautifulSoup:
        r = requests.get(url, proxies=proxy)
        return BeautifulSoup(r.content)

    @staticmethod
    def async_load(url, proxies=None) -> BeautifulSoup:
        connection_exception = True
        while connection_exception:
            proxy = ScrapperTools.form_proxy(proxies)
            try:
                if proxy is None:
                    req = urllib.request.Request(url)
                    with urllib.request.urlopen(req) as response:
                        page = response.read()
                else:
                    proxy_handler = urllib.request.ProxyHandler(proxy)
                    opener = urllib.request.build_opener(proxy_handler)
                    urllib.request.install_opener(opener)
                    with urllib.request.urlopen(url) as response:
                        page = response.read()
                    connection_exception = False
            except:
                pass

        return BeautifulSoup(page)

    @staticmethod
    def check_alive(proxy, test = "https://forum.pravda.com.ua"):
        try:
            r = requests.get(test, proxies=proxy, timeout=1)
            return proxy
        except Exception as err:
            return None

    @staticmethod
    def form_proxy(proxies:list)->dict:
        if proxies is None: return None
        prx = choice(list(proxies.keys()))
        if prx is None: return None
        proxy = {
            "https":prx,
            "http": prx
        }
        return ScrapperTools.check_alive(proxy)

    @staticmethod
    def write_json(path, filename, content):
        with open(path+'\\'+filename, 'w+', encoding='utf8') as json_file:
            json.dump(content, json_file, ensure_ascii=False)
```

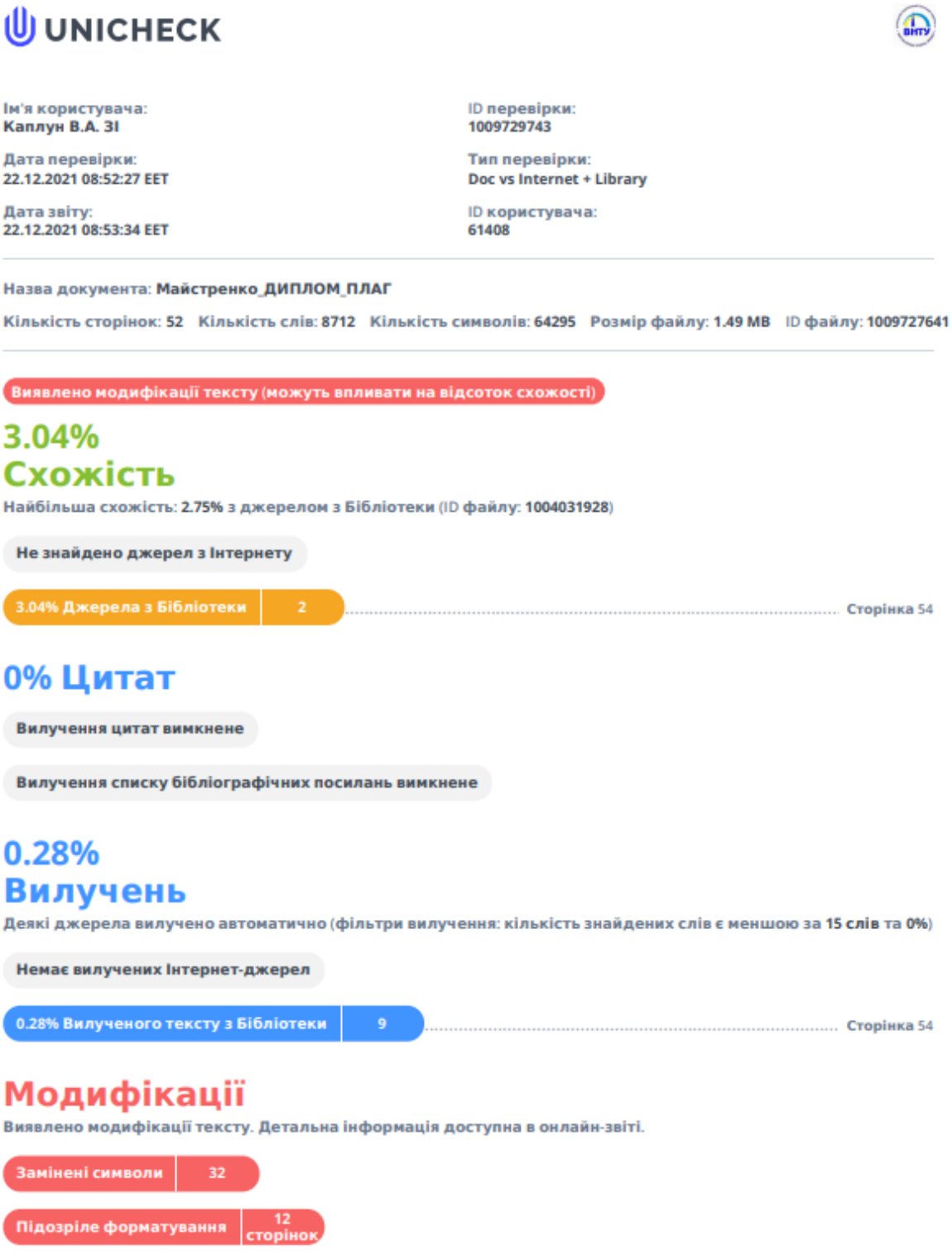
```
def get_pages_indices(content, max_pages):
    paging = content.find("div", "paging")
    if paging is not None:
        last_page = int(paging.findAll('a')[-2].text)
        start_page = max(0, last_page-max_pages)
        return [i*40 for i in range(start_page, last_page)]
    else:
        return [0]

def parse(config, proxies, parsed_themes):
    max_pages, max_topics = 50, 250
    checkout_step = 1

    for source in config:
        rtype = source['type']
        name = source['name']
        links = source['links']
        skiplist = source['skip']
        for url in links:
            bs = ScrapperTools.basic_load(url, None)
            title = bs.title.string
            forum_data = {
                'link': url,
                'title': title,
                'type': rtype
            }
            print(f"{rtype}:{name}:{title}")
            topics = []
            for top_count, tr in enumerate(bs.findAll("tr")[1:]):
                tid = tr.find("h4").a.get("href").split('topic=')[1][:-2]
                 if f"https://forum.pravda.com.ua/index.php?topic={tid}.0" in skiplist or tid in
parsed_themes:
                    continue
                topic_name = tr.find("h4").a.get_text('\n', strip=True)
                tds = tr.findAll("td")
                msg_num = int(tds[1].get_text().replace(' ', ''))
                view_num = int(tds[2].get_text().replace(' ', ''))
                author_name = tds[3].get_text()
                author_uid = tds[3].a.get("href").split("u=")[1]

                content = ScrapperTools.async_load(
                    f"https://forum.pravda.com.ua/index.php?topic={tid}.0",
                    alive_proxy
                )

                thread = {
                    'name': topic_name,
                    'tid': tid,
                    'msg_num': msg_num,
                    'view_num': view_num,
                    'author_name': author_name,
                    'author_uid': author_uid
                }

                messages = []
                pages_indices = get_pages_indices(content, max_pages)
                for page_ind in pages_indices:
                    content = ScrapperTools.async_load(
                        f'https://forum.pravda.com.ua/index.php?topic={tid}.{page_ind}',
                        alive_proxy
                    )
                    print(f"\t{topic_name}:{page_ind}")

                    for i, msg in enumerate(content.findAll("div", 'message')):
                        if msg.find("div", "generic_b") is None:
                            author = msg.find("div", "message-author icon-user")
                            message = msg.find("div", 'msg').get_text('\n', strip=True)
                            if len(message) == 0:
```

```
                                continue
                            date = msg.find('span', 'datetimeinfo').get_text()
                            date = date.replace('сьогодні в', time.strftime("%d %B %Y"))
                            uname = author.get_text(strip=True)
                        uid = None if author.a is None else int(author.a.get("href").split('u=')[1])
                            quoteheader = msg.find("div", "quoteheader")
                            quotetext = msg.find("blockquote", "bbc_standard_quote")
                            message_json = {
                                'date': date,
                                'uid': uid,
                                'uname': uname
                            }
                            if quoteheader is not None:
                                try:
                                    re_uname = quoteheader.a.b.get_text()
                                     re_date = quoteheader.a.get_text().replace(re_uname + ', ',
''')
                                    re_date = re_date.replace('сьогодні в', time.strftime("%d %B
%Y"))
                                    re_msg = quotetext.get_text()
                                    message_json['re'] = [{
                                        're_date': re_date,
                                        're_uname': re_uname,
                                        're_msg': re_msg
                                    }]
                                   message = '\n'.join(msg.find('div','msg').find_all(text=True,
recursive=False))
                                except Exception as e:
                                    pass
                            message_json['message'] = message
                            messages.append(message_json)
                    time.sleep(2)

                thread['messages'] = messages
                topics.append(thread)
                if top_count % save_count == 0:
                    forum_data['topics'] = topics
                    ScrapperTools.write_json('../data/output',
                                                          f'{rtype}-{name}-{title}-{topic_name}-
{time.time()}.json',
                                             forum_data
                    )
                if top_count == max_topics:
                    break
            break



def main(cfg, proxies):
    parsed_files = os.listdir("../data/output")
    parsed_themes = []
    for file in parsed_files:
        with open('../data/output/'+file, encoding='utf-8') as f:
            jsn = json.load(f)
        for topic in a['topics']:
            parsed_themes.append(topic['tid'])

    parse(cfg, proxies, parsed_themes)
    print("END")

def drop_dead(proxies):
    return { k : v for k,v in proxies.items() if v!=-1}

if __name__=='__main__':
    proxy_handler = ProxyHandler("https://free-proxy-list.net/")
    alive_proxy = drop_dead(proxy_handler.elapsed_list)
    config = ScrapperTools.load_json('data\\input\\forums.json')
    print(config)
```

```
main(config, alive_proxy)
```

ДОДАТОК В

Протокол перевірки на наявність плагіату



UNICHECK

Ім'я користувача:
Каплун В.А. 3І

ID перевірки:
1009729743

Дата перевірки:
22.12.2021 08:52:27 EET

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
22.12.2021 08:53:34 EET

ID користувача:
61408

Назва документа: Майстренко_ДИПЛОМ_ПЛАГ

Кількість сторінок: 52 Кількість слів: 8712 Кількість символів: 64295 Розмір файлу: 1.49 MB ID файлу: 1009727641

Виявлено модифікації тексту (можуть впливати на відсоток схожості)

3.04%
Схожість

Найбільша схожість: 2.75% з джерелом з Бібліотеки (ID файлу: 1004031928)

Не знайдено джерел з Інтернету

3.04% Джерела з Бібліотеки 2 ..... Сторінка 54

0% Цитат

Вилучення цитат вимкнене

Вилучення списку бібліографічних посилань вимкнене

0.28%
Вилучень

Деякі джерела вилучено автоматично (фільтри вилучення: кількість знайдених слів є меншою за 15 слів та 0%)

Немає вилучених Інтернет-джерел

0.28% Вилученого тексту з Бібліотеки 9 ..... Сторінка 54

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

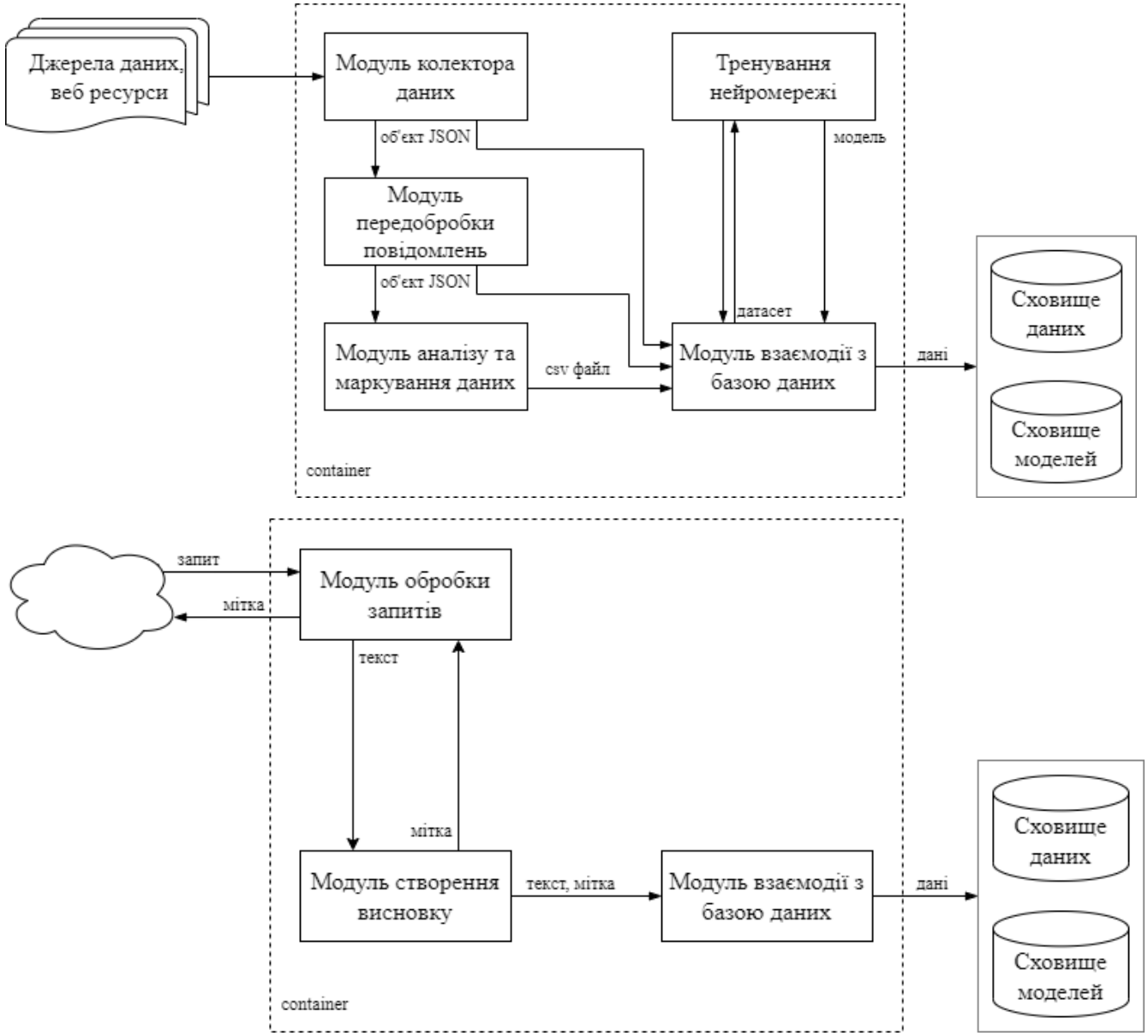
Замінені символи 32

Підозріле форматування 12 сторінок

ІЛЮСТРАТИВНА ЧАСТИНА

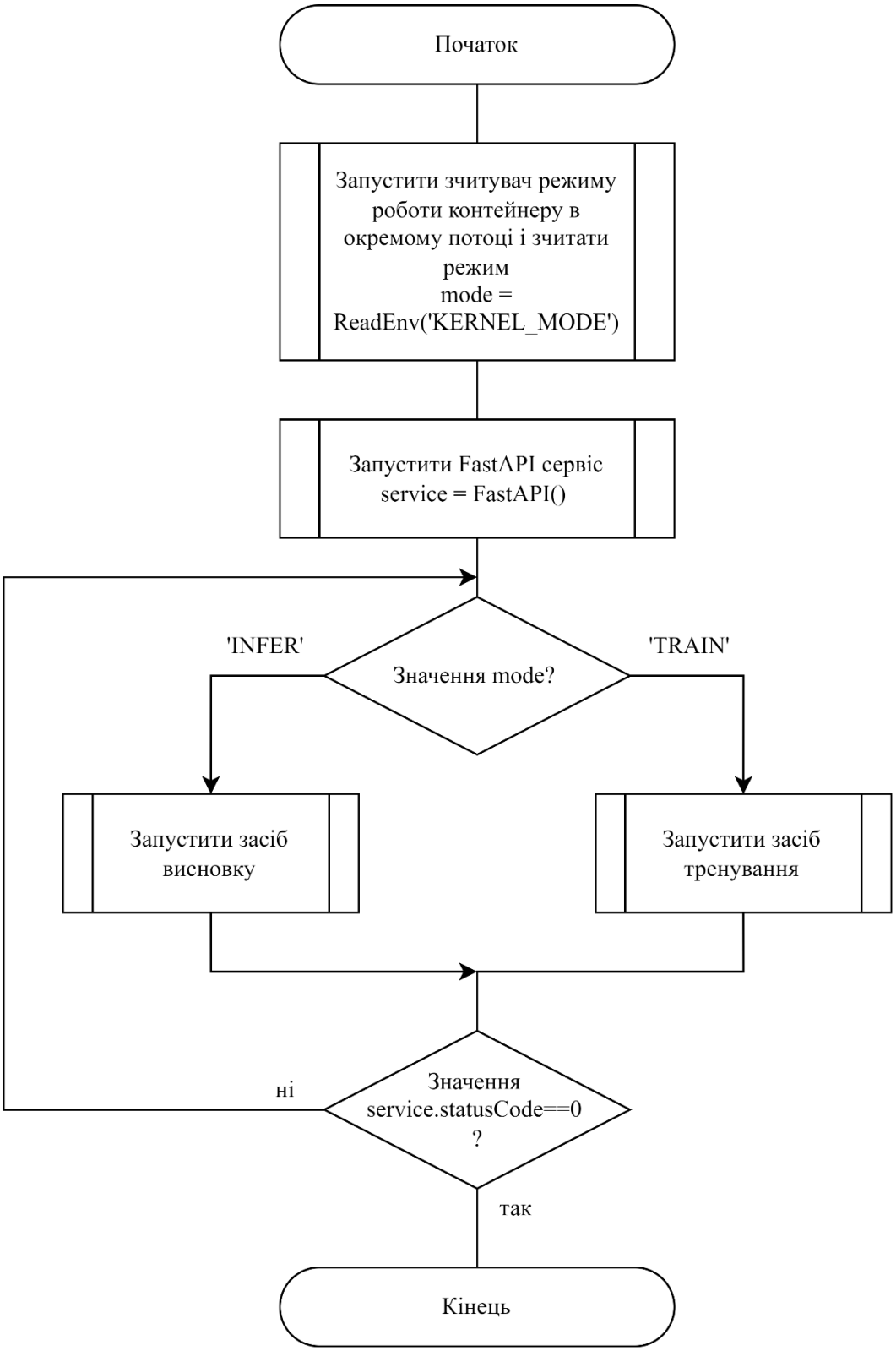
МЕТОД ТА ЗАСІБ РОЗПІЗНАВАННЯ ЕМОЦІЙНОГО ЗАБАРВЛЕННЯ ПОВІДОМЛЕННЯ ПІД ЧАС ІНФОРМАЦІЙНОЇ ВІЙНИ

СТРУКТУРА ЗАСТОСУНКУ ДЛЯ ВИЗНАЧЕННЯ ЕМОЦІЙНОГО КОНТЕКСТУ ПОВІДОМЛЕНЬ



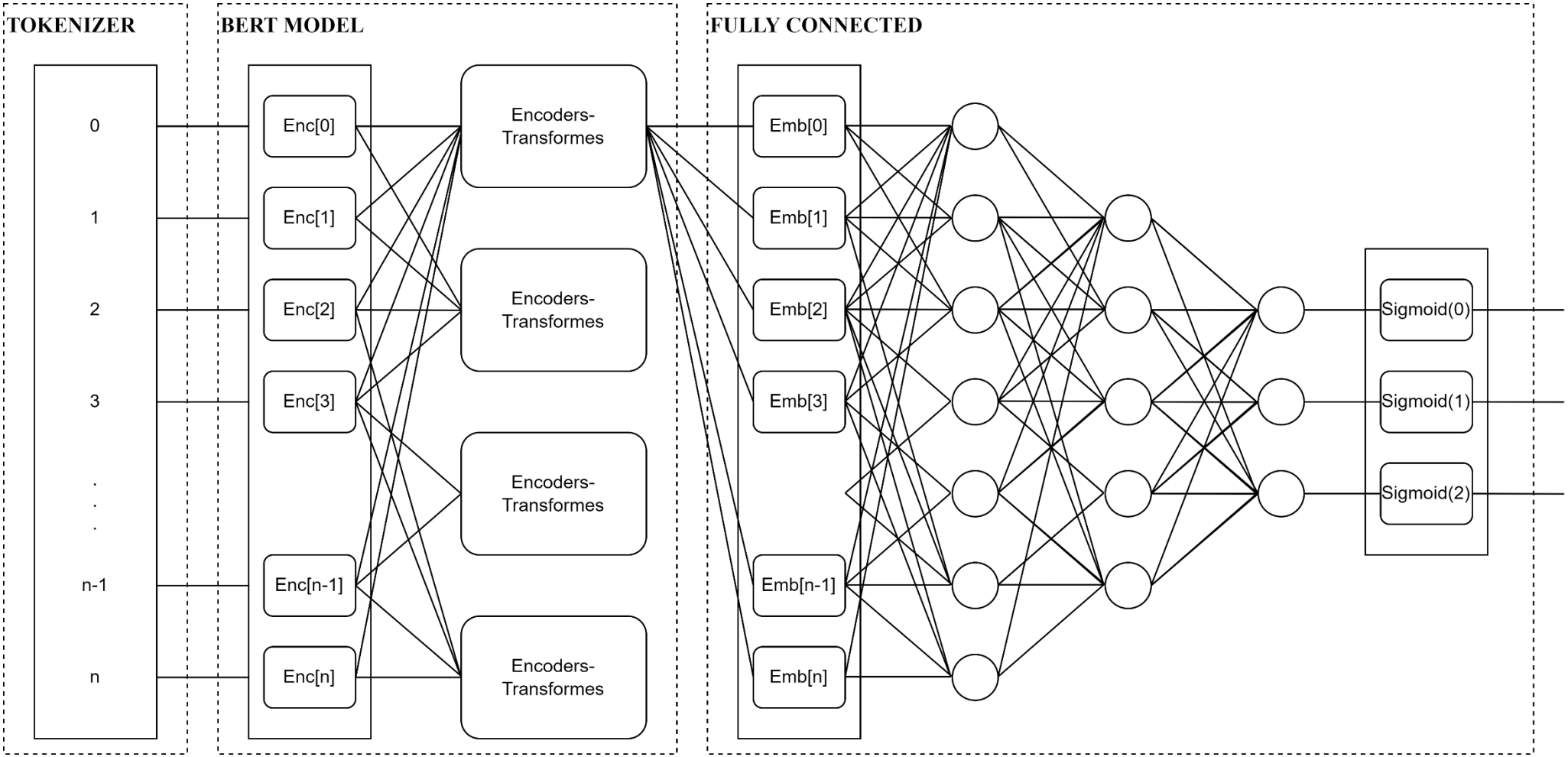
					08-20.МКР.019.00.000 ІЧ1		
Змн.	Арк.	№ докум.	Підпис	Дата			
Розроб.		Майстренко В.О.			Метод та засіб виялення емоційного забарвлення. Структура застосунку для визначення емоційного контексту повідомлень	Літ.	Арк. 1 / Акрушів 1
Перевір.		Войтович О. П.					
Реценз.		Войцеховська О.В.				ВНТУ, 1БС-20м	
Н. Контр.							
Затверд.							

АЛГОРИТМ РОБОТИ ЗАСТОСУНКУ ДЛЯ ВИЗНАЧЕННЯ ЕМОЦІЙНОГО КОНТЕКСТУ



					08-20.МКР.019.00.000 ІЧ2		
Змн.	Арк.	№ докум.	Підпис	Дата			
Розроб.		Майстренко В.О.			Метод та засіб виялення емоційного забарвлення. Алгоритм роботи застосунку для визначення емоційного контексту	Літ.	Арк. 1 / Акрушів 1
Перевір.		Войтович О. П.					
Реценз.		Войцеховська О.В.				ВНТУ, 1БС-20м	
Н. Контр.							
Затверд.							

СТРУКТУРА НЕЙРОННОЇ МЕРЕЖІ ВИЗНАЧЕННЯ ЕМОЦІЙНОГО КОНТЕКСТУ

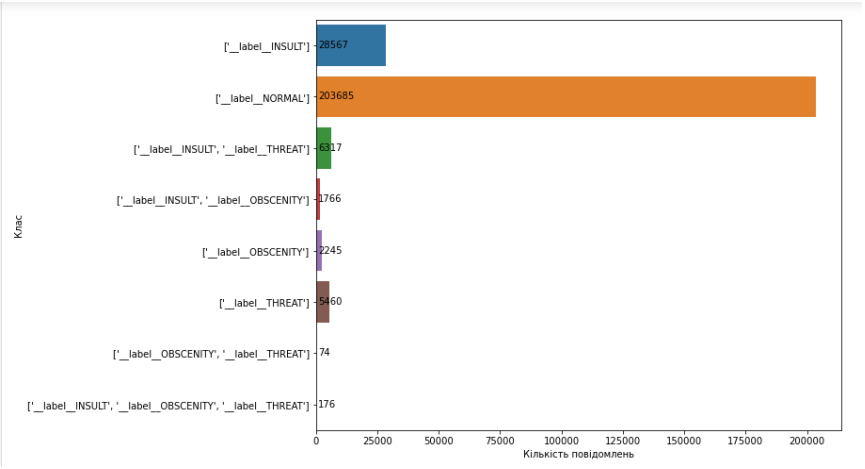


					08-20.МКР.019.00.000 ІЧ3		
Змн.	Арк.	№ докум.	Підпис	Дата			
Розроб.		Майстренко В.О.			Метод та засіб виялення емоційного забарвлення. Структура нейронної мережі визначення емоційного контексту	Літ.	Арк. 1 / Акрушів 1
Перевір.		Войтович О. П.					
Реценз.		Войцеховська О.В.				ВНТУ, 1БС-20м	
Н. Контр.		Войтович О. П.					
Затверд.		Лужецький В.А					

АНАЛІЗ ДАНИХ

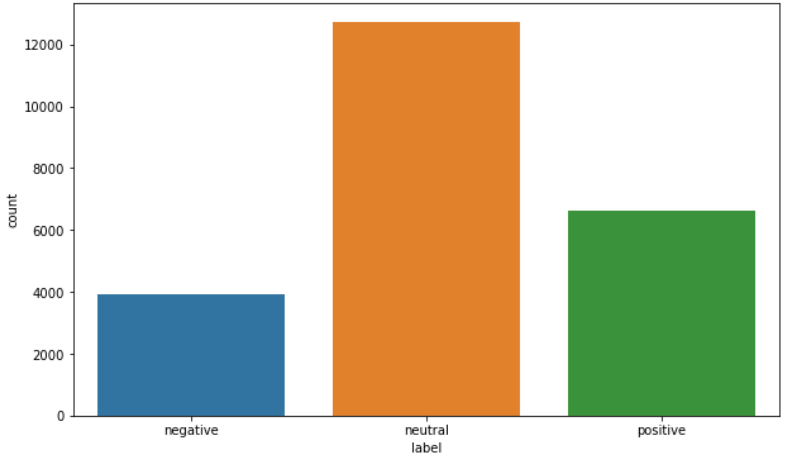
```
for label, texts in data.groupby('label'):
    print(f'Повідомленя: {texts.iloc[0, 0]}')
    print(f'Мітка класу: {label}\n')
```

```
Повідомленя: чтобы каждый день как петуха драли )в ж[redacted]
Мітка класу: ['__label__INSULT', '__label__OBSCENITY', '__label__THREAT']

Повідомленя: [redacted]в [redacted] этому [redacted], а не респект
Мітка класу: ['__label__INSULT', '__label__OBSCENITY']

Повідомленя: заколоть этого плешивого урода что бы крякнул как [redacted] всю вакцину ему в бошку что бы конкрет
но его завернуло
Мітка класу: ['__label__INSULT', '__label__THREAT']

Повідомленя: скотина! что сказать
Мітка класу: ['__label__INSULT']

Повідомленя: я сегодня проезжала по рабочей и между домами снитенко и гомолысовой магазином ( на пустыре) бежала кошка похожего
окраса. может, я и ошиблась, но необычный окрас бросился в глаза.
Мітка класу: ['__label__NORMAL']

Повідомленя: изнасиловать в[redacted]черенком от снеговой лопаты!!!
Мітка класу: ['__label__OBSCENITY', '__label__THREAT']

Повідомленя: эти генеральши знают где [redacted] и раздвинуть [redacted] и [redacted] ,позор страны
Мітка класу: ['__label__OBSCENITY']

Повідомленя: надо было его собаку на заборе повесить ,жалко скотинку конечно но это было бы предсказание...
Мітка класу: ['__label__THREAT']
```



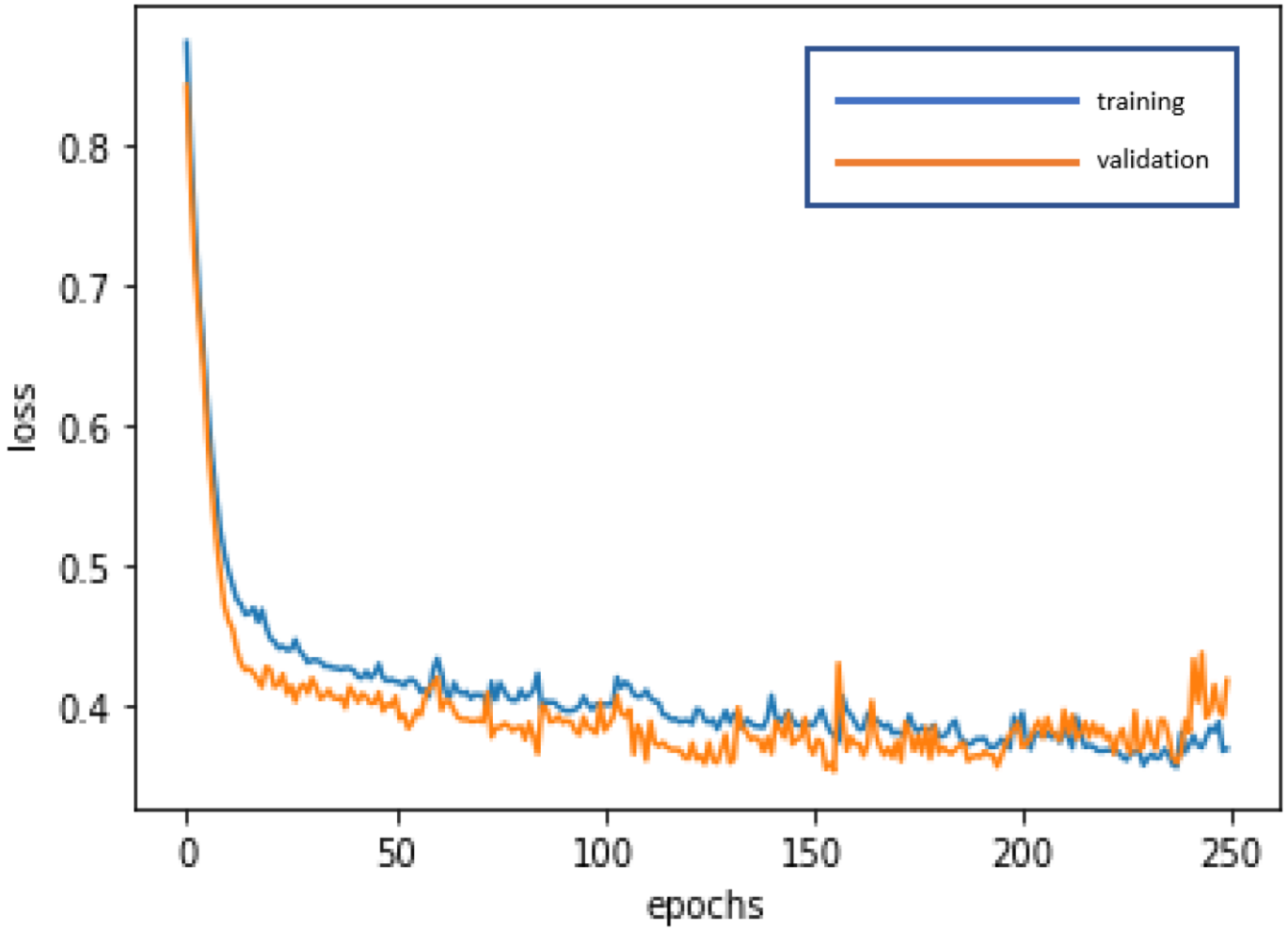

```
for label, texts in data.groupby('label'):
    print(f'Повідомленя: {texts.iloc[1, 1]}')
    print(f'Мітка класу: {label}\n')
```

```
Повідомленя: Блин, почему эта жизнь столь не справедлива ((((
Мітка класу: negative

Повідомленя: Просто пост :)
Мітка класу: neutral

Повідомленя: урря!я дождался этой овцы)
Мітка класу: positive

Повідомленя: где еще встречать свой день рождения как не на кладбище))))
Мітка класу: skip

Повідомленя: С Днем Рождения желаю много счастья, любви и успехов во всем:) И еще с днем Защитника Отечества;)))))))))
Мітка класу: speech
```



					08-20.МКР.019.00.000 І4		
Змн.	Арк.	№ докум.	Підпис	Дата			
Розроб.		Майстренко В.О.			Метод та засіб виялення емоційного забарвлення. Аналіз даних.	Літ.	Арк. / Акрушів: 1 / 1
Перевір.		Войтович О. П.					
Реценз.		Войцеховська О.В.				ВНТУ, 1БС-20м	
Н. Контр.							
Затверд.							

АНАЛІЗ ФУНКЦІЇ ВИТРАТ



					08-20.МКР.019.00.000 ІЧ5		
Змн.	Арк.	№ докум.	Підпис	Дата			
Розроб.		Майстренко В.О.			Метод та засіб виялення емоційного забарвлення. Структура застосунку для визначення емоційного контексту повідомлень	Літ.	Арк. 1 / Акрушів 1
Перевір.		Войтович О. П.					
Реценз.		Войцеховська О.В.				ВНТУ, 1БС-20м	
Н. Контр.							
Затверд.							

РЕЗУЛЬТАТИ ТЕСТУВАННЯ

Повідомлення	Проставлена мітка	Цільова мітка
“Я бы тебя сжёг…”	Агресія	Агресія
“классно придумано красота”	Позитивний	Позитивний
“маразматики озлобленые”	Нейтральний	Агресія
“До речі, ти теж мене почав реально задовбувати. Будеш відкривати рот і брехати про мене, то і тебе мій юрист із під землі дістане.”	Агресія	Агресія
кот само спойствие и пофигизм, а собачка боится???	Нейтральний	Нейтральний
“утопить бы того,, , кто так поступил.”	Нейтральний	Агресія

					08-20.МКР.019.00.000 ІЧ6		
Змн.	Арк.	№ докум.	Підпис	Дата			
Розроб.		Майстренко В.О.			Метод та засіб виялення емоційного забарвлення. Результати тестування	Літ.	Арк. 1 / Акрушів 1
Перевір.		Войтович О. П.					
Реценз.		Войцеховська О.В.				ВНТУ, 1БС-20м	
Н. Контр.							
Затверд.							