НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»
Факультет інформатики та обчислювальної техніки
Кафедра інформаційних систем та технологій

"На правах рукопису"
УДК *004.89*

ДО ЗАХИСТУ ДОПУЩЕНО
Завідувач кафедри

_____ Олександр РОЛІК

“ ____” _____2023 р.

МАГІСТЕРСЬКА ДИСЕРТАЦІЯ

на здобуття ступеня магістра
за освітньо-науковою програмою
«*Інтегровані інформаційні системи*»
зі спеціальності 126 «*Інформаційні системи та технології*»
на тему:
«Система розпізнавання та аналізу пропаганди на базі моделі текстової класифікації та методів статистичної обробки даних»

Виконав:
студент VI курсу, групи **ІА-11мн**
Безлюдний Юрій Сергійович _____
Керівник:
Доцент кафедри ІСТ, кандидат технічних наук, доцент
Шимкович Володимир Миколайович _____
Рецензент:
Доцент кафедри ОТ, кандидат технічних наук, доцент
Волокита Артем Миколайович _____

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2023 року

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки
Кафедра інформаційних систем та технологій
Рівень вищої освіти – другий (магістерський)
Спеціальність – 126 «Інформаційні системи та технології»
Освітньо-наукова програма «Інтегровані інформаційні системи»

ЗАТВЕРДЖУЮ
Завідувач кафедри

_____ Олександр РОЛІК

«__» _____2023 р.

ЗАВДАННЯ
на магістерську дисертацію студенту
Безлюдний Юрій Сергійович

1. Тема дисертації «Система розпізнавання та аналізу пропаганди на базі моделі текстової класифікації та методів статистичної обробки даних»
науковий керівник дисертації Шимкович Володимир Миколайович, кандидат технічних наук, доцент, затверджені наказом по університету від «20 березня» 2023 р. № 1275-с

2. Строк подання студентом дисертації “ 12 ” травня 2023 р.

Об’єкт дослідження: процес розпізнавання та аналітики пропаганди.
4. Предмет дослідження: методи та способи розпізнавання та аналітики пропаганди в текстах.
5. Перелік завдань, які потрібно розробити: дослідити методи пропаганди, визначити які з них використовуються проросійськими ЗМІ; розглянути існуючі методи текстової класифікації та статистичної обробки даних; розглянути та проаналізувати існуючі рішення в області визначення ступеня політичної полярності текстів, розпізнавання та аналітики пропаганди; розробити способи визначення політичної

полярності джерела повідомлень, побудови мереж найпопулярніших хештегів та популярних телеграм-каналів (та підмереж на її основі), а також збору статистичних даних по джерелах повідомлень; розробити архітектуру системи та компоненти, що її складають; програмно реалізувати систему розпізнавання та аналізу пропаганди.

6. Перелік графічного (ілюстративного) матеріалу: діаграма варіантів використання розробленого застосунку; схема структурна розробленого застосунку; діаграма послідовності процесу класифікації джерела повідомлень; діаграма послідовності процесу побудови підмережі заданого типу; діаграма послідовності процесу отримання статистичних даних; діаграма послідовності процесу оновлення класифікаційної моделі; діаграма послідовності процесу побудови мережі заданого типу; діаграма послідовності процесу збору статистичних даних; діаграма пакетів сервісу надання графічного інтерфейсу; діаграма пакетів сервісу обробки запитів.

7. Орієнтовний перелік публікацій

Pro-Russian propaganda recognition and analytics system based on text classification model and statistical data processing methods / Yurii Bezliudnyi [та ін.] // Адаптивні системи автоматичного управління. – 2023. – Т. 1, № 42. – С. 15–31.

8. Консультанти розділів дисертації

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

9. Дата видачі завдання “ 31 ” січня 2023 р.

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Строк виконання етапів магістерської дисертації	Примітка
1	Огляд літератури	10.02	
2	Огляд методів та засобів розв’язання задачі	17.02	
3	Розробка методів розв’язання задачі	24.02	
4	Формування набору даних	01.03	
5	Розробка класифікаційної моделі	11.03	
6	Проєктування та реалізація застосунку	14.04	
7	Подання роботи на попередній захист	20.04	
8	Подання роботи на основний захист	17.05	

Студент _____ *Юрій БЕЗЛЮДНИЙ*

Науковий керівник _____ *Володимир ШИМКОВИЧ*

РЕФЕРАТ

Магістерська дисертація: 147 с., 35 рис., 11 табл., 3 додатків, 35 джерел.

Актуальність. Оскільки з початку повномасштабного вторгнення російських військ у 2022 році, країною-агресором ведеться активна психологічна війна, є актуальним розроблення програмної системи, що могла б допомогти перевіряти інформацію, яку публікують користувачі соціальними мереж та месенджерів на наявність в них проросійської риторики. Корисною є також можливість отримувати інформацію про те які телеграм-канали чи хештеги є проукраїнськими та проросійськими, як вони взаємопов'язані між собою, та як змінюється їх популярність і політична полярність з часом.

Мета дослідження – розробка нової системи на базі нейронних мереж, що вміє як визначати політичну полярність переданих джерел повідомлень, так і будувати різноманітну аналітику на основі зібраних даних, що стосуються українсько-російської війни.

Об'єкт дослідження – процес розпізнавання та аналітики пропаганди.

Предмет дослідження – методи та способи розпізнавання та аналітики пропаганди в текстах.

Методи дослідження – методи штучного інтелекту, а саме нейронні мережі, для класифікації тексту; статистичні методи групування, ранжування та усереднення даних.

Відповідно до мети роботи, був окреслені наступні **завдання**, які необхідно вирішити при розробці системи:

- дослідити методи пропаганди, визначити які з них використовуються проросійськими ЗМІ;
- розглянути існуючі методи текстової класифікації та статистичної обробки даних;

— розглянути та проаналізувати існуючі рішення в області визначення ступеня політичної полярності текстів, розпізнавання та аналітики пропаганди;
— розробити спосіб визначення політичної полярності джерела повідомлень;
— розробити спосіб побудови мережі найпопулярніших хештегів та підмереж на її основі;
— розробити спосіб побудови мережі популярних телеграм-каналів та підмереж на її основі;
— розробити спосіб збору статистичних даних по джерелах повідомлень;
— розробити архітектуру системи та компоненти, що її складають;
— розробити програмне забезпечення, що реалізує систему розпізнавання та аналізу пропаганди.

Новизна роботи:

— сформовано набір даних для класифікації політичної полярності повідомлень в контексті українсько-російської війни на основі популярних облікових записів в мережі Twitter та телеграм-каналів;
— розроблено спосіб класифікації джерел повідомлень на основі розробленої класифікаційної моделі;
— розроблено спосіб побудови мережі найпопулярніших хештегів в соціальній мережі Twitter, що відрізняється наявністю інформації про політичну полярність ребер мережі та можливістю побудови підмереж;
— розроблено спосіб побудови мережі популярних телеграм-каналів.

Публікації. Результати досліджень представлені у друкованому виданні “Адаптивні системи автоматичного управління” [1].

ОБРОБКА ПРИРОДНЬОЇ МОВИ, НЕЙРОННІ МЕРЕЖІ, КЛАСИФІКАЦІЯ ТЕКСТУ, ТЕКСТОВІ ТРАНСФОРМЕРИ, НОВИНИ, РОСІЙСЬКА ПРОПАГАНДА, ХЕШТЕГИ, ГРАФИ, TELEGRAM, TWITTER.

ABSTRACT

Master's thesis: 147 p., 35 fig., 11 tbls., 3 appendices, 35 sources.

Relevance. Since the beginning of the full-scale invasion of Russian forces in 2022, the aggressor country has been conducting an active psychological warfare. It is essential to develop a software system that could help verify the information shared by users on social media platforms and messengers for the presence of pro-Russian rhetoric. It would also be beneficial to have the ability to gather information about which Telegram channels or hashtags are pro-Ukrainian or pro-Russian, how they are interconnected, and how their popularity and political polarity change over time.

Research objective – the development of a new system based on neural networks that can both determine the political polarity of the transmitted message sources and build various analytics based on the collected data related to the Ukrainian-Russian war.

Research object – .

Research subject – methods and ways of recognizing and analyzing propaganda in texts.

Research methods – methods of artificial intelligence, namely neural networks, for text classification; statistical methods of data grouping, ranking, and averaging.

The research **methods** are artificial intelligence methods, namely neural networks, for text classification; statistical methods of grouping, ranking, and averaging data.

In accordance with the goal of the work, the following **tasks** were outlined to be addressed during the development of the system:

— investigate propaganda methods, identify which ones are used by pro-Russian media;

— consider existing methods of text classification and statistical data processing;

— consider and analyze existing solutions in the field of determining the degree of political polarity of texts, recognizing and analyzing propaganda;

— develop a method for determining the political polarity of message sources;

— develop a method for building a network of the most popular hashtags and subnetworks based on it;

— develop a method for building a network of popular Telegram channels and subnetworks based on it;
— develop a method of collecting statistical data on message sources;
— develop the architecture of the system and its components;
— develop software that implements the system for recognizing and analyzing propaganda.

Novelty of the work:

— a dataset has been formed for classifying the political polarity of messages in the context of the Ukrainian-Russian war based on popular Twitter accounts and Telegram channels;
— a method for classifying message sources has been developed based on the developed classification model;
— a method for building a network of the most popular hashtags on the social network Twitter has been developed, which differs by the presence of information about the political polarity of network edges and the ability to build subnetworks;
— a method for building a network of popular Telegram channels has been developed.

Publications: the research results are presented in the printed edition "Adaptive systems of automatic control" [1].

NATURAL LANGUAGE PROCESSING, NEURAL NETWORKS, TEXT CLASSIFICATION, TEXT TRANSFORMERS, NEWS, RUSSIAN PROPAGANDA, HASHTAGS, GRAPHS, TELEGRAM, TWITTER.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, СКОРОЧЕНЬ І ТЕРМІНІВ

API – Application Programming Interface; інтерфейс програмування застосунків, що дозволяє різним програмним застосункам взаємодіяти між собою та обмінюватися даними.

BERT – Bidirectional Encoder Representations from Transformers; алгоритм машинного навчання, який використовується для обробки природної мови.

CORS – Cross-Origin Resource Sharing; механізм, який дозволяє веб-сторінкам запитувати ресурси з іншого джерела (домену), а також передавати дані назад, забезпечуючи безпеку мережевих запитів.

GRU – Gated Recurrent Unit; модифікація рекурентної нейромережі.

HTML – HyperText Markup Language; стандартна мова розмітки для створення веб-сторінок та веб-додатків.

HTTP – Hyper Text Transfer Protocol; протокол передачі документів у мережі Інтернет.

LSTM – Long Short-Term Memory; модифікація рекурентної нейромережі.

REST – Representational State Transfer; архітектурний підхід до мережевих протоколів, які надають доступ до інформаційних ресурсів.

RNN – Recurrent Neural Network; тип нейромережі, призначений для обробки послідовних даних, що мають залежності в часі.

SVM – Support Vector Machine; алгоритм машинного навчання для розв'язування задач класифікації та регресії, що використовується для знаходження границі розділення між різними категоріями даних.

URL – Uniform Resource Locator; уніфікована адреса ресурсу в мережі Інтернет.

ER – Entity Relationship; взаємозв’язок сутностей бази даних.

ВСТУП

В сучасному світі кожен свідомий громадянин намагається бути обізнаним про події що відбуваються в його країні та світі. За даними опитувань [2], основними джерелами інформації для жителів України є національне телебачення (57,7% респондентів), інтернет-сторінки (55,4%) та соціальні мережі чи месенджери (39,8%).

Якщо за достовірність інформації на національному телебаченні відповідає Національна рада України з питань телебачення і радіомовлення, то контроль за достовірністю та якістю вибірки інформації, що публікується засобами масової інформації чи відомими особистостями на інтернет-сторінках є більш вибірковим та хаотичним. Ще гіршою є ситуація з соціальними мережами та месенджерами, оскільки країна не може заблокувати окремих користувачів чи канали новин на таких популярних месенджерах як Telegram чи Viber або соціальних мережах як Facebook чи Twitter, оскільки рішення за блокування аккаунтів можуть приймати лише їх власники.

Водночас не всі громадяни є обізнаними з правилами інформаційної гігієни [3, 4], яка так важлива в умовах масового поширення фейків та пропаганди та в час коли світогляд та дії кожної людини як ніколи впливають на майбутнє української нації.

Пропаганда являє собою поширення певних поглядів шляхом донесення до мас різних аргументів, правдивих чи напівправдивих фактів, чуток або відвертої брехні з метою маніпуляції громадською свідомістю [5]. Для здійснення пропаганди використовуються багато методів, які базуються на дослідженнях соціальної психології, тому звичайному громадянину може бути важко усвідомити під який тип пропаганди він потрапив читаючи те чи інше джерело масової інформації [6]. Також деякі верстви населення є більш піддатливими до пропаганди – люди без вищої освіти, діти та пенсіонери, надмірно емоційні люди, деякі жителі південних та східних областей України [7].

З моменту вторгнення військ країни-агресора на територію України в 2014 році, здійснюється інтенсивна інформаційна кампанія, що має багато цілей, наприклад, заохочення до ведення бойових дій, виправдання воєнних злочинів, отримання підтримки населення серед інших країн, дискредитація дій ЗСУ тощо. Після початку повномасштабного вторгнення російських військ у 2022 році, дана психологічна війна лише посилилась і продовжується.

Відповідно, є актуальним розроблення програмної системи, що могла б допомогти перевіряти інформацію, яку публікують користувачі соціальними мереж та месенджерів на наявність в них проросійської риторики. Корисною є також можливість отримувати інформацію про те які джерела повідомлень чи хештеги є проукраїнськими та проросійськими, як вони взаємопов’язані між собою, та як змінюється їх популярність і політична полярність з часом.

Метою дослідження є розробка нової системи на базі нейронних мереж, що вміє як визначати політичну полярність переданих джерел повідомлень, так і будувати різноманітну аналітику на основі зібраних даних, що стосуються українсько-російської війни.

1 ОГЛЯД МЕТОДІВ ТА ЗАСОБІВ РОЗВ'ЯЗАННЯ ЗАДАЧІ РОЗПІЗНАВАННЯ ТА АНАЛІТИКИ ПРОПАГАНДИ В ТЕКСТАХ

1.1 Постановка задачі виявлення пропаганди та загальна схема розв'язання

Виявлення пропаганди є складним завданням, яке передбачає аналіз різних джерел інформації, виявлення закономірностей, а також розуміння контексту та намірів, що стоять за повідомленнями, які передаються.

Окреслимо загальну схему рішення даної задачі виявлення пропаганди.

Крок 1. Визначення джерел для збору даних.

Крок 2. Збір даних із обраних джерел.

Крок 3. Попередня обробка даних шляхом очищення та перетворення тексту у формат, придатний для аналізу. Це може включати скорочення надлишкових символів та слів, видалення стоп-слів, стемінг, лематизацію слів тощо.

Крок 4. Вибір релевантних способів вилучення ознак з тексту, таких як побудова моделей "торба слів", N-грам та вкладання слів. Обрані способи мають охоплювати семантику та синтаксис тексту, а також мовленнєві та стилістичні особливості, пов'язані з пропагандистськими прийомами.

Крок 5. Вибір моделі машинного навчання для класифікації пропагандистського тексту, такі як логістична регресія, метод опорних векторів, метод випадкових лісів або нейронні мережі. Ці моделі повинні мати можливість навчатися на основі створених ознак і точно передбачати пропагандистську техніку, використану в тексті.

Крок 6. Навчання моделі на обраному наборі даних і оцінка її продуктивності за допомогою обраних показників.

Крок 7. Розгортання моделі у робочому середовищі та її оновлення. Це передбачає моніторинг продуктивності моделі та її періодичне перенавчання, щоб гарантувати, що вона залишається точною та актуальною.

1.2 Постановка задачі вибору джерел для збору даних

Тренування класифікаційної моделі на правильних даних є ключовим для її точної роботи. Вибір джерел може проводитись за допомогою засобів машинного навчання, з використанням наявних в інтернеті даних про джерела, за результатами експертного аналізу джерел чи комбінованим методом. Загалом, виявлення пропаганди вимагає мультидисциплінарного підходу, який поєднує різноманітні техніки та методи з таких галузей, як журналістика, дослідження медіа, обробка природної мови та машинне навчання.

Окреслимо загальну схему вибору джерел для збору даних в контексті задачі класифікації пропаганди.

Крок 1. Визначення джерел засобів масової інформації та збір інформації про їх репутацію, популярність тощо. Це можна зробити за допомогою різних методів, таких як аналіз джерел та перевірка фактів.

Крок 2. Аналіз вмісту обраних джерел інформації. Це передбачає ідентифікацію шаблонів і ключових слів, пов’язаних із пропагандистськими методами, такими як емоційні звернення, навантажена мова та логічні помилки.

Крок 3. Визначення упереджень у ЗМІ, враховуючи використану мову та тон, а також обрамлення та контекст повідомлення. Це можна зробити шляхом порівняння ЗМІ з іншими джерелами та аналізу використаної мови та образів.

Крок 4. Контекстуалізація інформації шляхом вивчення історичного, соціального та політичного контексту, в якому створювалися ЗМІ. Це передбачає розуміння основоположних припущень і цінностей, які просуваються через ЗМІ.

Крок 5. Перевірка інформації, що публікують ЗМІ, шляхом перевірки фактів і перехресних посилань на інші джерела. Це можна зробити за допомогою різних інструментів перевірки фактів та баз даних.

Крок 6. Оцінка впливу медіа на аудиторію, за допомогою аналізу реакцій на повідомлення. Це передбачає моніторинг соціальних медіа, онлайн-форумів та інших каналів для зворотного зв’язку та взаємодії.

1.3 Огляд способів збору даних з джерел

Збір даних із джерел повідомлень може бути проведений за допомогою різних методів та інструментів, залежно від конкретного джерела, його типу та кількості необхідних даних.

Опишемо найпоширеніші способи збору даних з джерел.

Використання API. Багато соціальних мереж та месенджерів надають API, які дозволяють розробникам отримувати дані з мережі. Ці API зазвичай вимагають аутентифікації та можуть мати обмеження на використання або плату.

Веб-скрапінг. Веб-скрапінг передбачає використання автоматизованих інструментів для отримання даних шляхом аналізу HTML-коду веб-сторінок. Веб-скрапінг можна використовувати для отримання публічних даних, але це може порушувати умови обслуговування соціальних мереж чи месенджерів.

Використання сторонніх інструментів. Існує багато сторонніх бібліотек і служб, які можна використовувати для отримання необхідних даних. Ці інструменти часто пропонують розширені функції аналітики та звітності.

Ручний пошук. Ручний пошук передбачає використання функцій пошуку, що надає соціальна мережа чи месенджер для отримання певних повідомлень або тем. Цей метод займає багато часу, але може бути корисним для отримання невеликих обсягів даних або моніторингу певних розмов.

Загалом, збір даних вимагає поєднання технічних і аналітичних навичок, а також розуміння політики конфіденційності даних конкретної соціальної мережі чи месенджера.

1.4 Огляд способів попередньої обробки тексту

Попередня обробка тексту являє собою перетворення тексту в чистий і послідовний формат, який потім можна ввести в класифікаційну модель для її подальшого аналізу та навчання. Методи попередньої обробки тексту можуть бути загальними, щоб їх можна було застосувати до багатьох типів програм, або вони можуть бути спеціалізованими для конкретного завдання.

Найпоширеніші способи попередньої обробки тексту [8]:

— скорочення надлишкових символів та слів;

— канонізація тексту;

— виправлення орфографії;

— сегментація;

— видалення стоп-слів;

— стемінг;

— лематизація;

— зміна регістру слів.

Метод скорочення надлишкових символів та слів виконує видалення або заміну символів та слів у тексті на їх популярніші аналоги. Наприклад: знайдені в тексті спеціальні HTML-символи “&amp”, “&#60” та “&#x3E” будуть замінені на символи “&”, “<” та “>” відповідно; символи “×”, “÷”, “é” будуть замінені на “*”, “/” та “e”; множинні пробіли та знаки табуляції будуть замінені на одинарний пробіл.

Канонізація перетворює текст на його канонічне представлення. Поширеним застосуванням є обробка публікацій у соціальних мережах, де введений текст скорочується або слова пишуться по-різному. Наприклад, слово “hello” може бути написано як “hellooo” або слово “something” може виглядати як “smth.”, і різні люди можуть обирати, як писати “real time”: як “real-time” або “realtime”. Канонізація тексту замінює всі слова їх відповідним канонічним представленням. В останньому прикладі всі три форми слова “real time” будуть перетворені в “realtime”, оскільки воно є

канонічним. Канонізація тексту також може включати заміну емодзі в тексті відповідним словом чи набором слів. Наприклад, емодзі “:-)” після етапу канонізації заміниться на слова “happy face”.

Метод виправлення орфографії замінює всі слова на їх орфографічно правильні версії. Наприклад, неправильно написане слово “finaly” буде замінене на слово “finally”, а слово “lisrener” заміниться на “listener” і т. д.

Сегментація передбачає розбиття тексту на відповідні речення. Хоча це може здатися тривіальним завданням, воно має кілька труднощів. Наприклад, в англійській мові крапка зазвичай позначає кінець речення, але багато скорочень, зокрема “mr.” “ms.”, “inc.”, а також усі дробові числа містять крапки і вносять невизначеність, якщо правила закінчення речення не враховують ці винятки.

Метод видалення стоп-слів видаляє слова, котрі часто використовуються для побудови речень, проте не грають великої ролі в задачах аналізу чи обробки тексту. В англійській мові стоп-словами є “the”, “is”, “are”, “of”, “in” , “and” тощо. Для таких задач як класифікація тексту, аналіз настроїв і фільтрація спаму, ці слова зазвичай є зайвими, оскільки їх видалення зазвичай не впливає на метрики продуктивності роботи моделі.

Стемінг — це процес обрізання слів у тексті до їхньої кореневої частини. Наприклад, слово “learn” є коренем для таких слів як “learns”, “learning” та “learned”. Зазвичай для пошуку слова та його відповідної основи використовується таблиця пошуку. Багато пошукових систем використовують стемінг для отримання документів та сайтів, які запитують користувачі. Стемінг також використовується на етапі попередньої обробки для таких програм, як ідентифікація емоцій і класифікація тексту.

Лематизація є більш розвиненою формою стемінгу і передбачає перетворення всіх слів на відповідну кореневу форму, яка називається “лема”. В той час як процес стемінгу зводить усі слова до їх основи за допомогою таблиці пошуку, він не використовує жодних знань про контекст слова чи правила словотворень. Наприклад,

в процесі лематизації слово “less” перетвориться на “little”, “slept” на “sleep”, “wrote” на “write” тощо.

Порівняння результатів стемінгу та лематизації для деяких слів зображено на рисунку 1.1.

Word	Stemming	Lemmatization
information	inform	information
informative	inform	informative
computers	comput	computer
feet	feet	foot

Рисунок 1.1 – Порівняння результатів стемінгу та лематизації

Метод лематизації використовує більшу кількість правил мови та контекстної інформації, ніж метод стемінгу, відповідно для його реалізації потрібно більше ресурсів та часу.

Метод зміни регістру слів передбачає перетворення всього тексту на малі або великі літери, щоб усі рядки слів відповідали узгодженому формату.

1.5 Огляд способів вилучення ознак з тексту

Основна проблема в роботі з моделями обробки природної мови полягає в тому, що алгоритми машинного навчання не можуть працювати безпосередньо з необробленим текстом. Отже, необхідні певні методи вилучення ознак, щоб перетворити текст на матрицю (або вектор) ознак.

Деякі з найпопулярніших методів вилучення ознак з тексту [9]:

— побудова моделі “торба слів”;

— побудова моделі N-грам;

— вкладання слів.

1.5.1 Модель “торба слів”

Модель “торба слів” являє собою представлення тексту як набору слів, ігноруючи граматику та навіть порядок слів.

Приклад побудови моделі “торба слів” з речень наведено на рисунку 1.2.

	she	loves	pizza	is	delicious	a	good	person	people	are	the	best
She loves pizza, pizza is delicious	1	1	2	1	1	0	0	0	0	0	0	0
She is a good person	1	0	0	1	0	1	1	1	0	0	0	0
good people are the best	0	0	0	0	0	0	1	0	1	1	1	1

Рисунок 1.2 – Приклад побудови моделі “торба слів”

Модель “торба слів” зазвичай використовується в методах класифікації тексту, де частота зустрічі кожного слова використовується як ознака для навчання класифікатора.

1.5.2 Модель N-грам

N-грами являють собою послідовності з N елементів, які формуються з тексту. Елементами можуть бути фонеми, склади, слова або букви, залежно від того, який підхід використовується.

Приклад розкладу речення на словесні N-грами різної довжини зображений на рисунку 1.3.

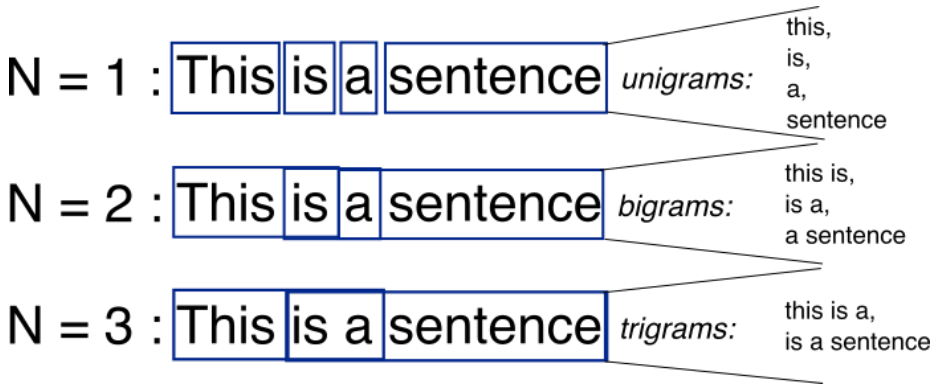

Рисунок 1.3 – Приклад розкладу речення на словесні N-грами різної довжини

Подібно до моделі “торба слів”, модель N-грам може використовуватись для класифікації тексту, де частота зустрічі N-грам використовується як ознака для навчання класифікатора.

1.5.3 Вкладання слів

Вкладання слів являє собою техніку, яка представляє слова або фрази як числові вектори у просторі великої розмірності. Мета вкладання слів полягає в тому, щоб охопити значення слів таким чином, щоб модель машинного навчання могла зрозуміти їхні зв’язки та схожість. Ключовою характеристикою вкладання слів є те, що слова зі схожими значеннями представлені векторами, розташованими близько один до одного у багатовимірному просторі, тоді як слова з різними значеннями представлені векторами, розташованими далеко один від одного.

Вкладання слів можуть будуватись як на основі нейронних мереж, так і методом зниження розмірності матриці суміжності слів, або з використанням ймовірнісних моделей.

Приклад побудови вкладання слів зображено на рисунку 1.4.

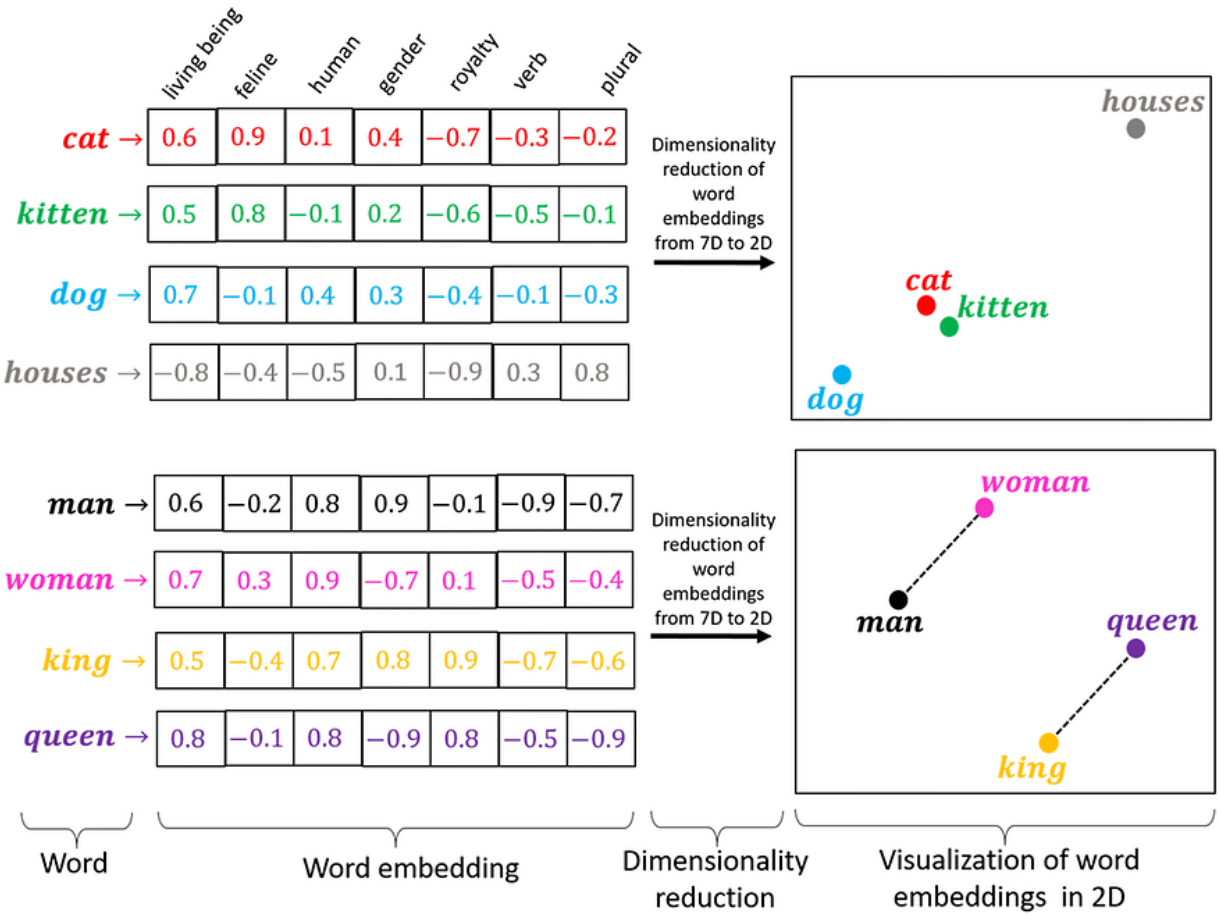

Рисунок 1.4 – Приклад побудови вкладання слів

1.6 Огляд методів текстової класифікації

Текстова класифікація являє собою процес визначення класу, до якого належить текст, за допомогою класифікаційної моделі, серед множини всіх класів, котрі дана модель здатна класифікувати. Визначення класу, до якого належить переданий до моделі текст, відбувається на основі ймовірнісного розподілу приналежності тексту до певного класу на множині всіх класів, якими оперує модель. Для точної класифікації бажано обирати таку множину класів, де кожен елемент, що належить до одного класу, не належить до іншого.

Основні методи текстової класифікації наступні:

— логістична регресія;

— метод опорних векторів;

— метод випадкових лісів;

— нейронні мережі.

1.6.1 Логістична регресія

Логістична регресія являє собою різновид множинної регресії, загальне призначення якої полягає в аналізі зв'язку між декількома незалежними змінними (котрі називають регресорами або предикторами) та залежною змінною [9]. Для відображення зв’язку між предикторами та значенням залежної змінної використовується логістична функція, котра визначається наступним чином:

$$\sigma(z) = \frac{1}{1 + e^{-z}} \tag{1.1}$$

де z — лінійна комбінація змінних предиктора та їхніх коефіцієнтів.

Формула для z має наступний вигляд:

$$z = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p \tag{1.2}$$

де β_0 — коефіцієнт зміщення;

$\beta_1, \beta_2, \ldots, \beta_p$— коефіцієнти для змінних предикторів $x_1, x_2, \ldots, x_p$ відповідно.

Логістична функція перетворює лінійну комбінацію предикторів у значення від 0 до 1, що представляє ймовірність двійкового результату. Зокрема, логістична функція відображає будь-які реальні вхідні дані z на ймовірнісні вихідні дані $\sigma(z)$ від 0 до 1.

Модель логістичної регресії оцінює значення коефіцієнтів $\beta_0, \beta_1, \beta_2, \ldots, \beta_p$ шляхом максимізації ймовірності спостережуваних даних, заданих моделлю, тобто функції правдоподібності.

Функція правдоподібності L визначається як:

$$L(\beta_0, \beta_1, \ldots, \beta_p) = \prod_{i=1}^{n} \sigma(z_i)^{y_i} \left(1 - \sigma(z_i)\right)^{1-y_i} \quad (1.3)$$

де n — розмір вибірки;

y_i — двійковий результат для i-го спостереження;

z_i — лінійна комбінація змінних предиктора для i-го спостереження.

Максимізація функції правдоподібності передбачає знаходження значень β_0, β_1, β_2, …, β_p, які максимізують ймовірність спостереження заданих результатів за допомогою предикторів. Зазвичай це робиться за допомогою алгоритму оптимізації, такого як градієнтний спуск.

Після того, як значення β_0, β_1, β_2, …, β_p знайдені, модель логістичної регресії може бути використана для прогнозування ймовірності бінарного результату для нових спостережень на основі їхніх предикторних значень. Прогнозовану ймовірність можна перетворити на двійковий результат за допомогою порогового значення, наприклад 0,5.

Модель логістичної регресії можна подати у вигляді одношарової нейронної мережі з сигмоїдальною функцією активації, вагами якої є коефіцієнти β_0, β_1, β_2, …, β_p, а на входи подаються значення предикторів x_1, x_2, …, x_p.

В контексті текстової класифікації, предикаторами регресійної моделі слугують вилучені з тексту набори ознак, наприклад, модель “торба слів” та модель N-грам, а двійковим результатом спостереження є приналежність тексту до певного класу.

1.6.2 Метод опорних векторів

Метод опорних векторів (SVM) являє собою алгоритм машинного навчання, який використовується для класифікації та регресійного аналізу [10].

Для заданого набору точок, що представляють навчальні дані, $(\boldsymbol{x_1}, y_1), (\boldsymbol{x_2}, y_2), \ldots, (\boldsymbol{x_n}, y_n)$, де $\boldsymbol{x_i}$ — p-вимірний вектор ознак, а y_i — відповідна мітка, SVM прагне знайти гіперплощину, яка розділяє точки різних класів на максимальну відстань.

Рівняння гіперплощини можна виразити наступною формулою:

$$\boldsymbol{w} \cdot \boldsymbol{x} + b = 0 \tag{1.4}$$

де $\boldsymbol{w}$ — вектор вагових коефіцієнтів;

b — коефіцієнт зсуву.

Відстань між гіперплощиною та найближчими точками даних кожного класу називається запасом, і метою SVM є максимізувати цей запас.

Щоб знайти оптимальну роздільну гіперплощину, SVM вирішує наступну задачу оптимізації:

$$\left\{ \underset{\boldsymbol{w}}{\arg\min} \frac{1}{2} \|\boldsymbol{w}\|^2 \;\middle|\; y_i(\boldsymbol{w} \cdot \boldsymbol{x}_i + b) \geq 1 \right\}, \forall i = 1, \ldots, n \tag{1.5}$$

де y_i — це мітка точки даних $\boldsymbol{x_i}$.

Перший член у цільовій функції представляє запас, а другий член представляє регулювання для запобігання перенавчанню. Обмеження гарантують, що всі точки даних правильно класифіковані та лежать за межами поля.

Функція Лагранжа для цієї задачі оптимізації задається наступною формулою:

$$L(\boldsymbol{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}\|\boldsymbol{w}\|^2 - \sum_{i=1}^{n} \alpha_i [y_i(\boldsymbol{w} \cdot \boldsymbol{x}_i + b) - 1] \tag{1.6}$$

де $\boldsymbol{\alpha} = (\alpha_1,\ \alpha_2, \ldots, \alpha_n)$ — множники Лагранжа.

Розв'язання задачі оптимізації можна знайти шляхом мінімізації $L(\boldsymbol{w}, b, \boldsymbol{\alpha})$ щодо $\boldsymbol{w}$, b і максимізації його відносно $\boldsymbol{\alpha}$ з урахуванням обмежень $\alpha_i \geq 0$. Тоді параметри оптимальної гіперплощини визначаються наступним чином:

$$\boldsymbol{w}^* = \sum_{i=1}^{n} \alpha_i^* y_i \boldsymbol{x}_i \tag{1.7}$$

$$b^* = y_j - \sum_{i=1}^{n} \alpha_i^* y_i (\boldsymbol{x}_i \cdot \boldsymbol{x}_j) \tag{1.8}$$

де α_i^* — оптимальне значення множника Лагранжа;

$\boldsymbol{x}_j$ — будь-яка точка даних, що лежить на границі запасу.

Точки даних з ненульовими множниками Лагранжа називаються опорними векторами, і вони лежать на границі запасу.

Метод опорних векторів можна розширити для обробки нелінійно розділених даних за допомогою ядрових методів. При такому підході точки даних відображаються у високовимірному просторі ознак, де їх можна лінійно розділити, а функція ядра дозволяє проводити ефективні обчислення без явного обчислення векторів ознак.

Приклад побудови оптимальної роздільної гіперплощини методом опорних векторів для двох класів зображено на рисунку 1.5.

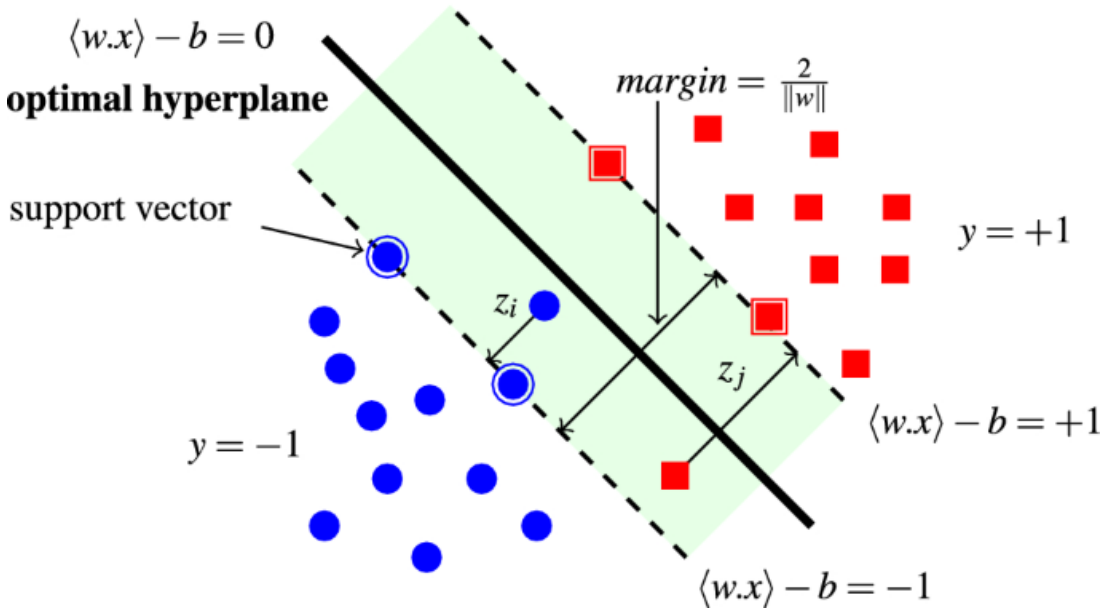

Рисунок 1.5 – Приклад побудови оптимальної роздільної гіперплощини методом опорних векторів

Розглядаючи метод опорних векторів у контексті текстової класифікації, набором навчальних даних є комбінація вилучених з текстів наборів ознак, та міток, що відображають до якого класу належить кожен текст.

1.6.3 Метод випадкових лісів

Метод випадкових лісів — це популярний алгоритм машинного навчання, який використовується для завдань класифікації [11]. У випадкових лісах набір дерев рішень навчається на випадково вибраних підмножинах навчальних даних, а остаточний прогноз робиться більшістю голосів, отриманих як результат прогнозів окремих дерев. Класифікація екземпляру даних за допомогою методу випадкових лісів зображено на рисунку 1.6.

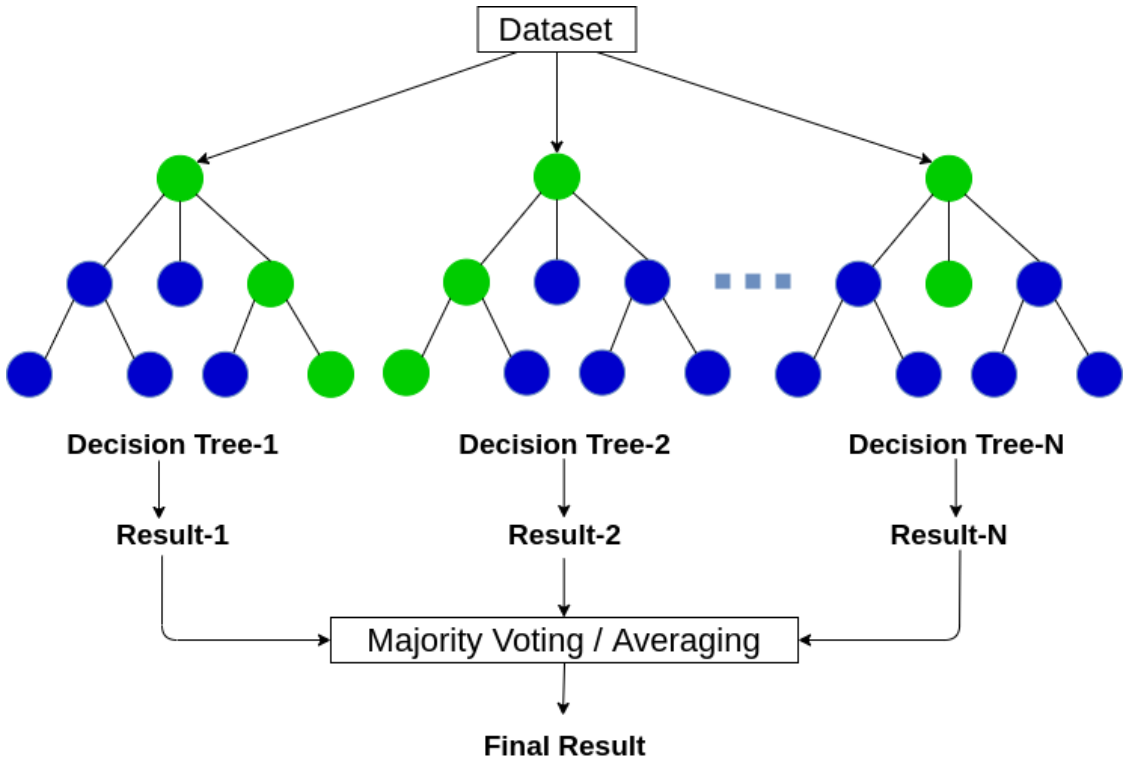

Рисунок 1.6 – Класифікація екземпляру даних за допомогою методу випадкових лісів

Нехай X — це множина вхідних даних розміром N елементів, Y — множина відповідних міток для завдання класифікації, M – розмірність простору ознак. Опишемо алгоритм випадкових лісів.

Крок 1. Обирається випадкова підвибірка з повторенням D_j з навчальних даних розміром N' елементів.

Крок 2. Будується дерево рішень, котре класифікує елементи з D_j, причому, при створенні чергового вузла дерева, ознака, на основі якої проводиться розбиття, обирається не з усіх M ознак, а лише з m випадково вибраних (зазвичай $m \approx \sqrt{M}$).

Крок 3. Повторюються кроки 1 і 2, допоки не буде створена бажана кількість дерев рішень T.

Щоб класифікувати новий екземпляр x, мітка класу прогнозується шляхом отримання більшості голосів прогнозів з окремих дерев:

$$y(x) = \underset{y \in Y}{\arg\max} \sum_{j=1}^{T} I(y_j = y) \tag{1.9}$$

де y_j – прогноз j-го дерева;

$I(x)$ – характеристична функція.

1.6.4 Нейронні мережі

За останні десятиріччя штучні нейронні мережі показали значний прогрес в задачах семантичного аналізу, класифікації, перекладу, генерації тексту тощо. Задача обробки природної мови потребує особливої архітектури нейронної мережі, яка має виконувати наступні вимоги, що випливають з особливостей текстових даних:

— підтримка послідовностей даних змінної довжини;

— відслідковування залежностей різної дальності між даними;

— підтримка інформації про послідовність надходження даних.

Найпростішою моделлю, що виконує всі поставлені вище вимоги є класична модель рекурентних нейронних мереж (RNN). З плином часу, дана модель зазнавала покращень, породивши такі моделі як довга короткочасна пам’ять (LSTM) та вентиль рекурентні вузли (GRU).

Останніми роками моделям на основі RNN стали створювати конкуренцію моделі на основі глибинної архітектури текстового трансформера. На відміну від RNN, трансформер не потребує обробки послідовностей по порядку. Радше, механізм уваги забезпечує контекст для будь-якого положення у послідовності входу. Наприклад, якщо дані входу є реченням природної мови, то трансформера не потрібно обробляти

його початок, перш ніж взятися за обробку його кінця. Натомість трансформер визначає контекст, який надає значення кожному слову в цій послідовності. Ця властивість уможливлює розпаралелювання задач, і відтак знижує тривалість тренування.

З часу виходу моделі трансформера, з'явився ряд моделей, що використовують її, або її частини як основу в своїй архітектурі. Такими є моделі BERT, GPT, CTRL, XLM, XLNET, T5, REFORMER, LONGFORMER, ELECTRA тощо.

1.6.4.1 Класична модель RNN

Рекурентні нейронні мережі являють собою клас штучних нейронних мереж, які спеціально розроблені для обробки послідовних даних. RNN широко використовуються в багатьох областях, таких як обробка природної мови, розпізнавання мови, створення підписів до зображень, генерація тексту та аудіозаписів, завдяки їхній здатності моделювати тимчасові залежності в послідовних даних.

В основі RNN лежить набір рекурентних з'єднань, які дозволяють мережі зберігати пам'ять про попередні вхідні дані [12]. Мережа приймає послідовність векторів як вхідні дані. RNN обробляє послідовність крок за кроком, і на кожному кроці оновлює свій прихований стан, використовуючи поточний вхідний вектор і попередній прихований стан.

Прихований стан RNN можна розглядати як стиснуте представлення всієї послідовності вхідних даних, які спостерігалися до цього моменту часу. Це стиснуте представлення можна використовувати для прогнозування наступного елемента в послідовності або для класифікації всієї послідовності.

Класична модель RNN в її замкнутому та розгорнутому вигляді зображена на рисунку 1.7.

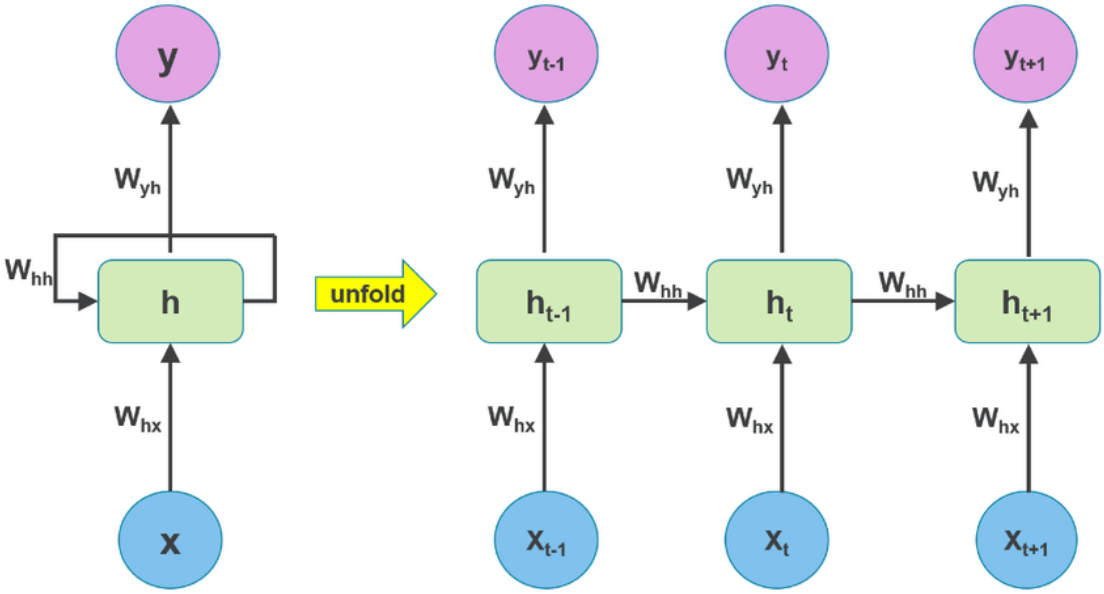

Рисунок 1.7 – Класична модель RNN

Відповідно до поставленої задачі, модель RNN може мати один чи декілька входів та виходів. Загалом, за кількістю входів та виходів, виділяють 4 типи моделі RNN: 1:1, 1:N, M:1 та M:N.

Модель 1:1 (один вхід до одного виходу), являє собою класичний перцептрон.

Модель 1:N (один вхід до багатьох виходів), зазвичай застосовується для таких задач як створення підписів для зображень або генерація аудіозапису за переданим жанром.

Модель M:1 (багато входів до одного виходу), застосовується, наприклад, для визначення емоційної тональності переданого тексту.

Модель M:N (багато входів до багатьох виходів), знаходить застосування в задачах перекладу тексту з однієї мови на іншу, або класифікації кожного слова тексту в залежності від контексту.

Модель RNN зазвичай навчають за допомогою алгоритму зворотного поширення в часі (Backpropagation Through Time, BPTT), який є варіантом алгоритму зворотного поширення, адаптованого для послідовних даних. BPTT передбачає розгортання мережі в часі та розглядання її як прямої нейронної мережі, де кожен часовий крок

розглядається як окремий рівень. Потім ваги мережі оновлюються за допомогою стандартного алгоритму зворотного поширення, але з додаванням тимчасового виміру, який враховує залежності між часовими кроками. Схема процесу оновлення ваг RNN за допомогою алгоритму BPTT зображено на рисунку 1.8.

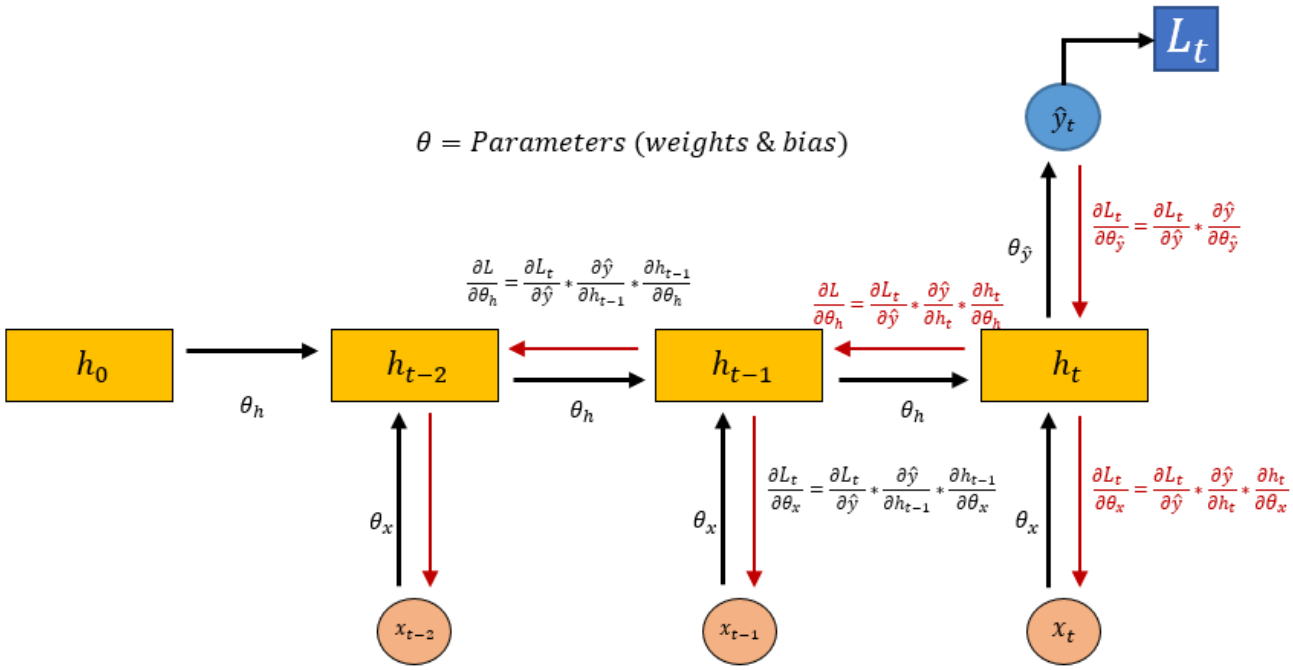

Рисунок 1.8 – Процес оновлення ваг RNN за допомогою алгоритму BPTT

1.6.4.2 Модель LSTM

Модель LSTM являє собою особливий різновид архітектури RNN, здатний до навчання довгостроковим залежностям, оскільки пам'ять, що реалізується за допомогою класичної модель RNN, виходить короткою — на кожному кроці навчання інформація в пам'яті поєднується з новою і через кілька ітерацій повністю перезаписується [13].

Модель LSTM не використовує функцію активації усередині своїх рекурентних компонентів. Таким чином, значення, що зберігається, не розмивається в часі і

градієнт не зникає при використанні BPTT. Структура LSTM-модуля зображена на рисунку 1.9.

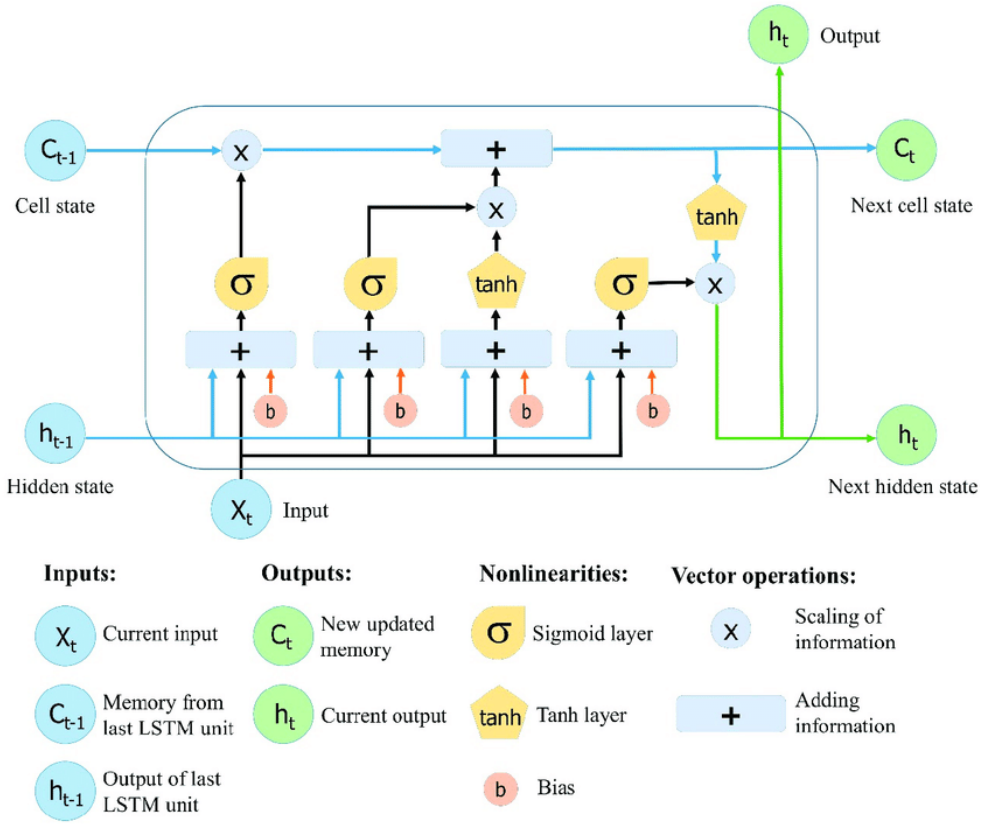

Рисунок 1.9 – Структура LSTM-модуля

Ключовими компонентами LSTM-модуля є стан комірки та різноманітні фільтри.

Про стан комірки можна говорити, як про пам'ять мережі, яка передає відповідну інформацію по всьому ланцюжку модулів. Таким чином, інформація з ранніх кроків буде присутньою на значно пізніших кроках, нівелюючи ефект короткочасної пам'яті. У міру того, як відбувається навчання, стан комірки змінюється, інформація додається або видаляється зі стану комірки структурами, які називаються фільтрами.

Фільтри контролюють потік інформації на входах та на виходах модуля на підставі деяких умов. Вони складаються з шару сигмоїдальної нейронної мережі та операції поточкового множення.

Сигмоїдальний шар повертає числа в діапазоні [0; 1], які позначають, яку частку кожного блоку інформації слід пропустити далі через мережу. Множення на це значення використовується для пропуску чи заборони потоку інформації всередину і назовні пам'яті. Наприклад, вхідний фільтр контролює міру входження нового значення в пам'ять, а фільтр забування контролює міру збереження значення в пам'яті. Вихідний фільтр контролює міру того, якою мірою значення, що знаходиться в пам'яті, використовується при розрахунку вихідної функції активації.

1.6.4.3 Модель текстового трансформера

Модель текстового трансформера використовує механізми звернення уваги для обробки послідовностей текстових даних [14].

У своїй класичній формі трансформер складається з двох окремих частин — кодера, який зчитує введений текст і переводить його в проміжну репрезентацію, і декодера, який на основі даної репрезентації створює прогноз для завдання.

Архітектура трансформера зображена на рисунку 1.10.

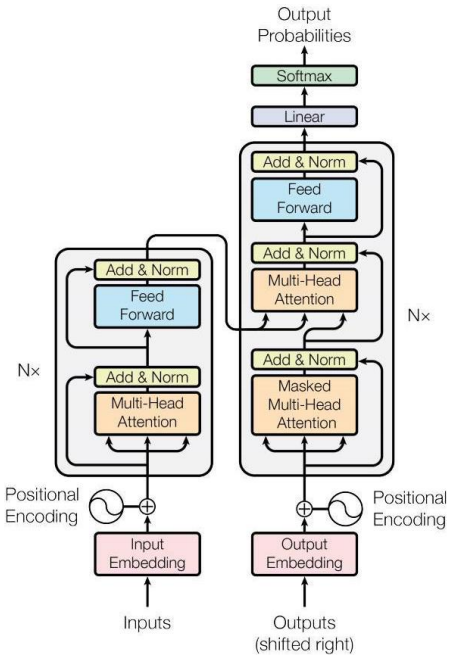

Рисунок 1.10 – Модель трансформера

Кодер складається з шарів кодування, які обробляють вхідні дані послідовно один шар за іншим, тоді як декодер складається з шарів декодування, які роблять те саме з виходом кодера. Функція кожного шару кодера полягає у створенні репрезентації, яка містить інформацію про те, які частини вхідних даних є релевантними одна одній. Кожний шар кодера передає свою репрезентацію на наступний рівень кодера як вхідні дані. Кожен шар декодера робить протилежне, беручи всі репрезентації та використовуючи їхню контекстну інформацію для генерації вихідної послідовності. Для цього кожен рівень кодера та декодера використовує механізм звернення уваги. Для кожної частини вхідних даних механізм звернення уваги обчислює ступінь важливості кожної іншої частини та використовує їх для отримання результату. Кожен рівень декодера має додатковий механізм уваги, який отримує інформацію з виходів попередніх декодерів, перш ніж шар декодера отримає інформацію з репрезентацій кодера. Як кодер, так і декодер мають нейронну мережу прямого зв'язку для додаткової обробки виходів і містять залишкові з'єднання та шар нормалізації.

Функцію уваги можна описати як відображення запиту та набору пар ключ-значення на вихідні дані, де запит, ключі, значення та вихідні дані є векторами. Вихід обчислюється як зважена сума значень, де вага, призначена кожному значенню, обчислюється функцією сумісності запиту із відповідним ключем. На практиці функція уваги обчислюється на наборі запитів одночасно, упакованих разом у матрицю Q. Ключі та значення також упаковані разом у матриці K та V. Матриця виходів A обчислюється за наступною формулою:

$$A(Q,K,V) = softmax\left(\frac{QK^T}{\sqrt{\dim(K)}}\right)V \tag{1.10}$$

Замість обчислення єдиної функції уваги за допомогою d_{model}-вимірних ключів, значень і запитів, автори моделі трансформера виявили корисним робити лінійну

проекцію запитів, ключів і значень h разів з різними лінійними проекціями на виміри d_k, d_k і d_v відповідно. По кожній з цих проекцій запитів, ключів і значень паралельно обчислюється функція уваги, даючи d_v-вимірне вихідне значення. Вони з'єднуються та знову проектуються, в результаті чого отримуються кінцеві значення функції багатосторонньої уваги за наступною формулою:

$$\mathrm{M}HA(Q,K,V) = Concat(head_1,\dots,head_h)W^O \tag{1.11}$$

де W^O – матриця параметрів, що задає лінійну проекцію з'єднаних вихідних значень функції уваги кожного шару кодера на вихідний вимір d_{model};

$head_i$ – значення функції уваги з i-го центру уваги, що визначається за наступною формулою:

$$head_i = A\left(QW_i^Q, KW_i^K, VW_i^V\right) \tag{1.12}$$

де W_i^Q, W_i^K, W_i^V – матриці параметрів, що задають лінійні проекції для матриць Q, K, V на виміри d_k, d_k і d_v відповідно.

Порівняння структур блоків для обчислення функцій уваги та багатосторонньої уваги зображено на рисунку 1.11.

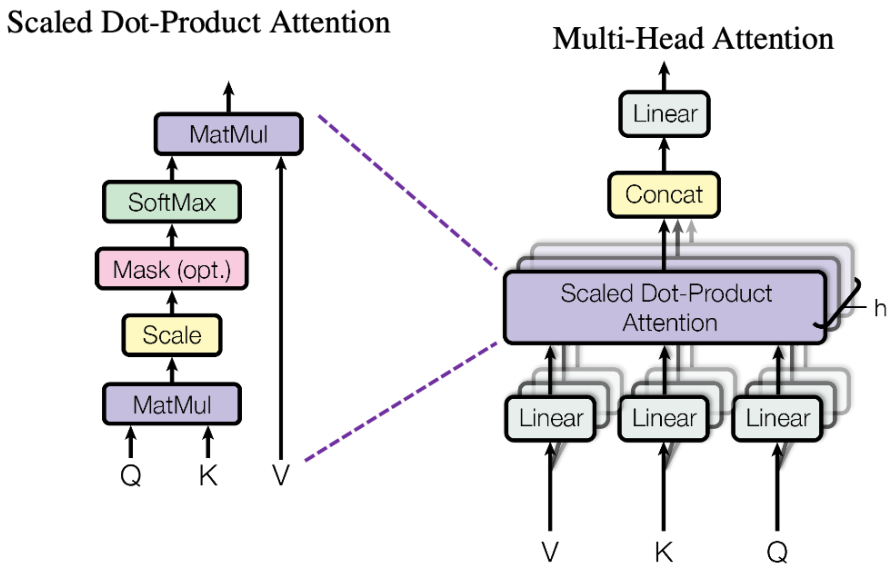

Рисунок 1.11 – Порівняння структур блоків для обчислення функцій уваги (зліва) та багатосторонньої уваги (справа)

1.6.4.4 Модель BERT

BERT являє собою попередньо навчену модель, що складається зі стеку кодерів трансформера. BERT є однією з найпопулярніших і широко використовуваних моделей для завдань обробки природної мови, таких як аналіз настроїв, класифікація тексту та переклад мови [15].

Ключовою технічною інновацією BERT є застосування двонаправленого навчання. Даний підхід відрізняється від попередніх, які розглядали послідовність тексту зліва направо або поєднували навчання зліва направо та справа наліво. Результати показують, що мовленнєва модель, яка навчається двонаправлено, має глибше відчуття мовного контексту, ніж однонаправлені мовленнєві моделі.

Для реалізації двонаправленого навчання модель BERT використовує дві стратегії: побудова маскувальної моделі мови (Masked Language Modeling, MLM) та передбачення наступного речення (Next Sentence Prediction, NSP).

Стратегія MLM полягає в тому що перед подачею послідовностей слів у BERT 15% слів у кожній послідовності замінюються маркером [MASK]. Потім модель

намагається передбачити початкове значення замаскованих слів на основі контексту, наданого іншими, незамаскованими, словами.

Опишемо алгоритм передбачення замаскованих слів.

Крок 1. Додається шар класифікації поверх виходу кодера.

Крок 2. Вихідні вектори множаться на проекційну матрицю, тим самим переводячи їх у вимір словника.

Крок 3. Обчислюються ймовірності кожного слова в словнику за допомогою функції softmax.

Функція втрат BERT бере до уваги лише передбачення замаскованих значень і ігнорує передбачення незамаскованих слів.

Стратегія NSP полягає в тому що у процесі навчання модель BERT отримує пари речень як вхідні дані та вчиться передбачати, чи є друге речення в парі наступним реченням у вхідному тексті. Під час навчання 50% вхідних даних є парою, у якій друге речення є наступним реченням у вхідному тексті, а в інших 50% друге речення обирається випадково зі збірки речень.

Щоб допомогти моделі розрізнити два речення під час навчання, перед входом у модель вхідні дані обробляються за алгоритмом наведеним нижче.

Крок 1. Маркер [CLS] вставляється на початку першого речення, а маркер [SEP] — наприкінці кожного речення.

Крок 2. Вектор проекції речень, який вказує на речення A або речення B, додається до кожного токена.

Крок 3. Вектор проекції позицій додається до кожного токена, щоб вказати його позицію в послідовності.

Щоб передбачити, чи справді друге речення пов’язане з першим, виконуються нижчеописані кроки.

Крок 1. Уся вхідна послідовність проходить через модель.

Крок 2. Вихід з маркера [CLS] перетворюються на вектор у формі 2×1 за допомогою простого шару класифікації.

Крок 3. Обчислюється ймовірність слідування другого речення за першим за допомогою функції softmax.

Під час навчання моделі BERT, стратегії MLM та NSP виконуються разом з метою мінімізації комбінованої функції втрат двох стратегій.

Для використання моделі BERT у процесі вирішення поставлених задач спочатку виконується попереднє навчання моделі на великій текстовій збірці. Стандартними текстовими збірками для попереднього навчання моделі BERT є Wikipedia Corpus (~4,4 мільйони статей) та Books Corpus (~11 тисяч книг).

Після попереднього навчання виконується фактичне навчання моделі на переданій базі даних. Попереднє навчання та подальше налаштування моделі BERT під конкретну задачу зображено на рисунку 1.12.

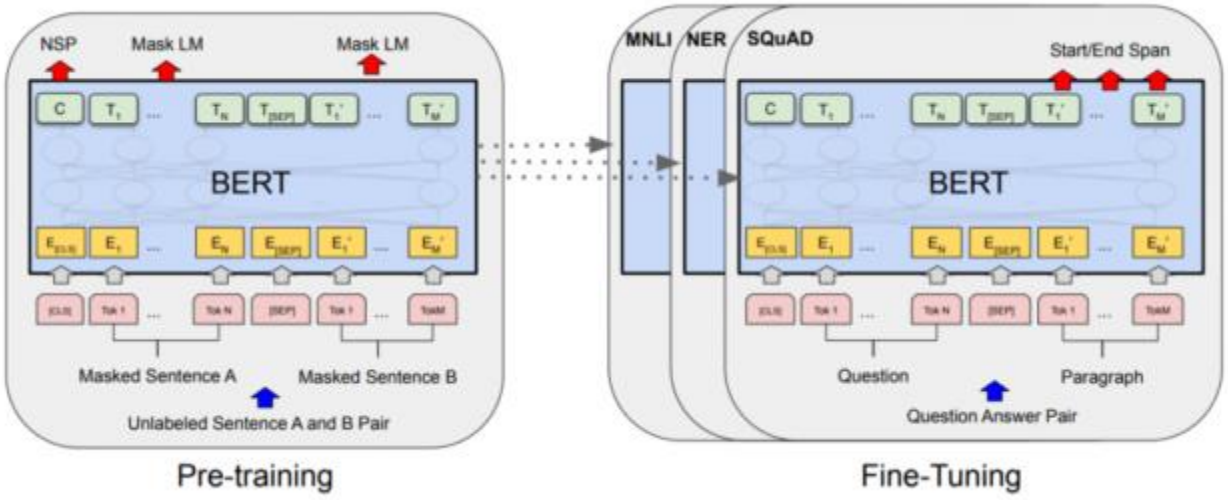

Рисунок 1.12 – Попереднє навчання та подальше налаштування моделі BERT під конкретну задачу

1.7 Порівняння методів текстової класифікації

Враховуючи велику кількість існуючих технік машинного навчання для класифікації тексту є доцільним порівняти кожен метод, описавши їх недоліки та

переваги в контексті задачі текстової класифікації. Дане порівняння наведено в таблиці 1.1.

Таблиця 1.1 – Порівняння методів текстової класифікації

Назва методу	**Переваги методу**	**Недоліки методу**
Логістична регресія	— простий в реалізації; — може обробляти двійкові та багатокласові проблеми класифікації.	— припускає лінійний зв'язок між незалежними змінними та залежною змінною; — неефективний з даними великої розмірності; — не підтримує інформацію про послідовність даних, що надходять та їх порядок.
SVM	— ефективно обробляє багатовимірні дані; — може обробляти нелінійні зв'язки між змінними; — добре обробляє великі масиви даних.	— вимагає ретельного налаштування гіперпараметрів; — може бути дорогим з точки зору обчислень; — не підтримує інформацію про послідовність даних, що надходять та їх порядок.
Випадкові ліси	— ефективний при обробці даних великої розмірності;	— може бути схильний до перенавчання;

	— може обробляти нелінійні зв'язки між змінними; — може обробляти дані з деякими відсутніми значеннями та з присутнім шумом.	— може бути дорогим з точки зору обчислень; — не підтримує інформацію про послідовність даних, що надходять та їх порядок.
Нейронні мережі	— підтримує інформацію про послідовність даних, що надходять та їх порядок; — може працювати з нелінійними зв'язками між змінними; — може обробляти дані великого розміру.	— для ефективного навчання потрібна велика кількість даних; — може бути схильний до перенавчання; — може бути обчислювально дорогим.

1.8 Огляд методів статистичної обробки даних

Для виділення з даних корисної і зрозумілої для людини інформації існує багато популярних статистичних методів. Деякі з найбільш широко використовуваних [16]:

— регресійний аналіз;

— аналіз головних компонент;

— кластерний аналіз;

— аналіз часових рядів;

— аналіз мереж;

— аналіз просторових даних.

1.8.1 Регресійний аналіз

Регресійний аналіз використовується для моделювання зв’язку між залежною змінною та однією чи кількома незалежними змінними. Метою регресійного аналізу є оцінка параметрів лінійної або нелінійної функції, які можуть найкраще пояснити спостережувану варіацію залежної змінної на основі значень незалежних змінних.

Існує багато різних типів регресійного аналізу, включаючи просту лінійну регресію, множинну лінійну регресію, логістичну регресію та нелінійну регресію. Кожен з перелічених типів обирається в залежності від кількості змінних та типу зв’язку між залежними та незалежними змінними.

1.8.2 Аналіз головних компонент

Аналіз головних компонент (Principal component analysis, PCA) використовується для зменшення розмірності великого набору даних, зберігаючи якомога більше вихідної інформації. Це досягається шляхом ідентифікації головних компонент, або напрямків найбільшої варіації, в даних і проектування даних на ці компоненти.

Перша головна компонента — це напрямок, у якому дані варіюються найбільше, а кожен наступна головна компонента є ортогональною або перпендикулярною до попередніх і фіксує решту варіацій даних.

Етапи виконання PCA є наступними.

Етап 1. Стандартизацію даних, щоб гарантувати, що кожна змінна має однаковий масштаб і що середнє значення центрується на нулі.

Етап 2. Розрахунок коваріаційної матриці, яка вимірює лінійні зв’язки між змінними.

Етап 3. Обчислення власних векторів і власних значень коваріаційної матриці. Власні вектори представляють головні компоненти, а власні значення представляють величину дисперсії.

Етап 4. Вибір перших n основних компонентів, які відображають найбільшу дисперсію в даних.

Етап 5. Проектування даних на вибрані головні компоненти для отримання нового представлення даних з меншою розмірністю.

1.8.3 Кластерний аналіз

Метою кластерного аналізу є визначення угруповань або шаблонів у даних, які можуть надати краще розуміння структури даних.

Опишемо деякі поширені методи, які використовуються в кластерному аналізі.

K-mean кластеризація. K-mean кластеризація — це метод поділу набору точок даних на k кластерів, де k — заздалегідь визначена кількість кластерів. Алгоритм працює шляхом ітеративного призначення точок даних найближчому центроїду або центру кластера та повторного обчислення центроїдів, доки кластери не зійдуться.

Ієрархічна кластеризація. Ієрархічна кластеризація — це метод групування точок даних у вкладені кластери, де кожен кластер є підмножиною більшого кластера. Алгоритм працює шляхом ітеративного злиття кластерів, доки всі точки даних не належатимуть одному кластеру або поки не буде виконано критерій зупинки.

Кластеризація на основі щільності. Кластеризація на основі щільності — це метод кластеризації точок даних на основі їх щільності або кількості точок даних у певній області простору даних. Алгоритм працює, ідентифікуючи області з високою щільністю та групуючи точки даних, які належать до цих областей, у кластери.

1.8.4 Аналіз часових рядів

Аналіз часових рядів використовується для аналізу та моделювання даних, зібраних протягом певного часу. Часовий ряд – це послідовність даних, які вимірюються через рівні проміжки часу протягом певного періоду часу.

Аналіз часових рядів знаходить застосування при визначенні закономірностей і тенденцій у даних, а також для прогнозування майбутніх значень на основі історичних даних.

Опишемо найпоширеніші методи аналізу часових рядів.

Аналіз тенденцій. Використовується для визначення довгострокового напрямку часового ряду.

Сезонний аналіз. Використовується для виявлення періодичних шаблонів у часовому ряді, наприклад тижневих або місячних циклів.

Автокореляційний аналіз. Використовується для визначення ступеня залежності між значеннями часового ряду в різні моменти часу.

1.8.5 Аналіз мереж

Аналіз мереж використовуються для дослідження даних, що можна представити у вигляді мережі. Зазвичай мережі представляють у вигляді графів, де вузли є певними сутностями або акторами, а ребра представляють зв’язки або відносини між ними.

Опишемо найпоширеніші методи аналізу мереж.

Визначення показників центральності. Показники центральності дають можливість визначити найважливіші вузли в мережі на основі різних критеріїв, таких як їх ступінь (кількість з’єднань, які вони мають), їх проміжність (кількість найкоротших шляхів, які проходять через них) або центральність власного вектора (міра їхнього впливу на всю мережу);

Виявлення спільнот. Методи виявлення спільнот групують вузли разом на основі подібності їхніх з’єднань, щоб ідентифікувати кластери або спільноти в мережі;

Прогнозування зв’язку. Методи передбачення зв’язку визначають ймовірність формування з’єднання між двома вузлами в мережі на основі характеристик вузлів та їхніх зв’язків з іншими вузлами;

Візуалізація мережі. Методи візуалізації мережі використовують графічні представлення, щоб допомогти аналітикам зрозуміти структуру та поведінку мереж.

1.8.6 Аналіз просторових даних

Аналіз просторових даних використовуються для дослідження та моделювання даних, які мають географічний або просторовий компонент. Просторові дані можуть містити інформацію про розташування, розмір, форму та інші атрибути географічних об’єктів, таких як країни, області, міста тощо.

Опишемо найпоширеніші методи, які використовуються в аналізі просторових даних.

Просторова статистика. Просторова статистика — це галузь статистики, яка зосереджена на аналізі даних, які мають географічний або просторовий компонент.

Просторова інтерполяція. Просторова інтерполяція – це техніка, яка використовується для оцінки значення змінної в певному місці на основі значень, спостережених у сусідніх місцях.

Просторова регресія. Просторова регресія – це статистичний метод, який використовується для моделювання зв’язку між залежною змінною та однією або декількома незалежними змінними з урахуванням просторових залежностей між спостереженнями.

1.9 Огляд існуючих рішень щодо розпізнавання та аналітики пропаганди в текстах

На сьогоднішній день існує ряд науково-дослідних робіт, що стосуються задачі класифікації тексту на його політичну полярність чи на наявність пропаганди в ньому.

Так, у досліджені, котре проводилось дослідниками Рао А. та Спасоєвич Н. [17], застосовується підхід, що використовує техніку вкладання слів (word embeddings) і модель LSTM для вирішення проблеми класифікації повідомлень, зібраних з

соціальної мережі Twitter на наявність в них політичної полярності, де повідомлення класифікуються як демократичні чи республіканські.

Для збору повідомлень автори обрали користувачів у мережі Twitter, чиї політичні погляди є загальновідомими. Далі автори зчитали повідомлення від даних користувачів із цих списків, які вони публікували протягом останніх 3 місяців. Повідомленням, опублікованим відомими демократами, згідно зі списками мережі Twitter, присвоювалась мітка “0”, а повідомленням від відомих республіканців присвоювалась мітка “1”. Навчальний набір в даній роботі містить 336 000 повідомлень, а тестовий – 84 000 повідомлень.

Розроблена модель здатна класифікувати повідомлення з високою точністю 87,57% на тестовому наборі. Також дослідники розгорнули систему, що використовує навчену модель, і надають доступ до неї для можливості інтеграції.

В іншій роботі, дослідниками з Монтерейського інституту технологій [18] було проведено класифікацію настроїв (позитивні / негативні) повідомлень користувачів мережі Twitter, пов’язаних з українсько-російською війною, у різних країнах світу.

Для формування бази даних, автори взяли існуючу збірку повідомлень з мережі Twiiter, що стосуються українсько-російської війни, під назвою “Ukraine Conflict Twitter Dataset” та розміром в 11,2 мільйонів повідомлень на 64 різних мовах. Для задачі класифікації настроїв повідомлень автори обрали тільки повідомлення на англійській мові та виконали попередню обробку даних повідомлень (видалили всі посилання, емодзі, стоп-слова, перевели слова в нижній регістр). Далі автори використали попередньо навчений аналізатор настроїв VADER, щоб оцінити настрої групи повідомлень і призначити кожному повідомленню мітку “0” або “1”, що відповідають позитивним та негативним повідомленням відповідно.

Аналізатор VADER використовує лексикон слів і фраз із заздалегідь визначеними оцінками настроїв, а також правила, які враховують контекст, у якому ці слова та фрази з’являються, щоб визначити настрої певного тексту. VADER також враховує

інші лінгвістичні особливості, такі як пунктуація та великі літери, щоб підвищити точність аналізу настроїв.

Після присвоєння повідомленням відповідних міток, автори сформували зі сформованої бази даних тренувальні, перевірочні та тестові набори для тренування моделі RNN.

Даний підхід показав гарну точність на перевірочному наборі в 93% і точність в наборі для тестування близьку до 90%.

Детальне дослідження повідомлень, що стосуються українсько-російської війни, провели дослідники з університету Любліна [19]. В своїй роботі автори поставили за задачу відповісти на наступні питання:

— які користувачі мережі Twitter є найпопулярнішими в контексті українсько-російської війни?
— чи існують конкретні користувачі чи хештеги, які є суто проросійськими чи проукраїнськими, і чи утворюють вони чіткі спільноти?
— чи є помітні відмінності в публікаціях повідомлень до і після заборони мережі Twitter у росії?

Результатами роботи дослідників стали:

— класифікатор на базі нейронних мереж, котрий визначає ступінь проросійського спрямування повідомлень;
— мережа найбільш вживаних хештегів;
— дві мережі користувачів з різними типами з’єднань (згадок і пересланих повідомлень);
— тристороння мережа хештегів, повідомлень і користувачів.

Розроблений авторами класифікатор має за основу новітню модель трансформера RoBERTa. Для навчання моделі автори використали два набори даних, пов’язані з українсько-російською війною. Перший набір даних [20] було зібрано з 1 січня 2022 року до 5 березня 2022 року з використанням певних ключових слів для пошуку, наприклад “Ukrainian war”, “Russian troops”, “Ukrainian troops” тощо. Другий набір

[21] містить понад 25 мільйонів повідомлень і оновлюється щодня з 27 лютого 2022 р. Він містить повідомлення, геолокація яких була виявлена в Україні або містить такі хештеги, як “#SlavaUkraini”, “#Russia”, “#RussiaUkraineWar”, “#Putin”, “#ukraineunderattack”, “#StopPutinNow” тощо. Далі автори призначили повідомленням мітки “0” або “1”, що відповідають повідомленням котрі підтримують українську та російську сторони відповідно. З обох наборів для навчання використовувались обрані випадковим чином перші 300 тисяч повідомлень.

Розподіл кількості повідомлень за ступенем їх проросійського спрямування для обох наборів даних (першому з описаних наборів автори дали назву 65D, а другому – 25M) зображено на рисунку 1.13.

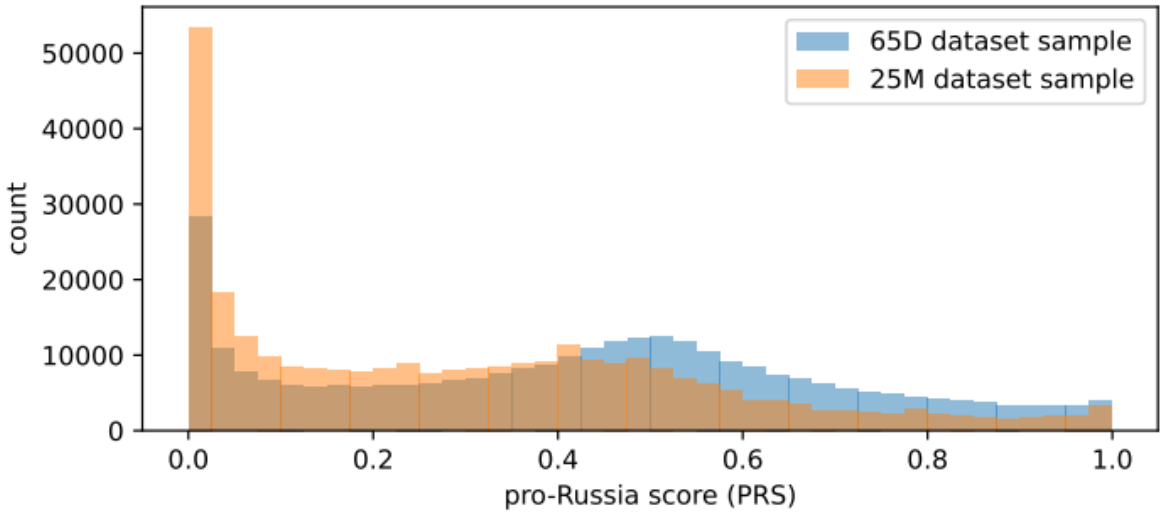

Рисунок 1.13 – Розподіл кількості повідомлень за ступенем їх проросійського спрямування для наборів даних 65D та 25M

Для решти аналізу автори використовували наступні алгоритми:

— алгоритм Infomap для виявлення спільнот [22];

— алгоритм K-Core декомпозиції для визначення ядра та периферії мереж [23];

— алгоритм PageRank для вимірювання важливості вузлів в мережах [24].

Ядро побудованої мережі найбільш вживаних хештегів в контексті українсько-російської війни зображено на рисунку 1.14.

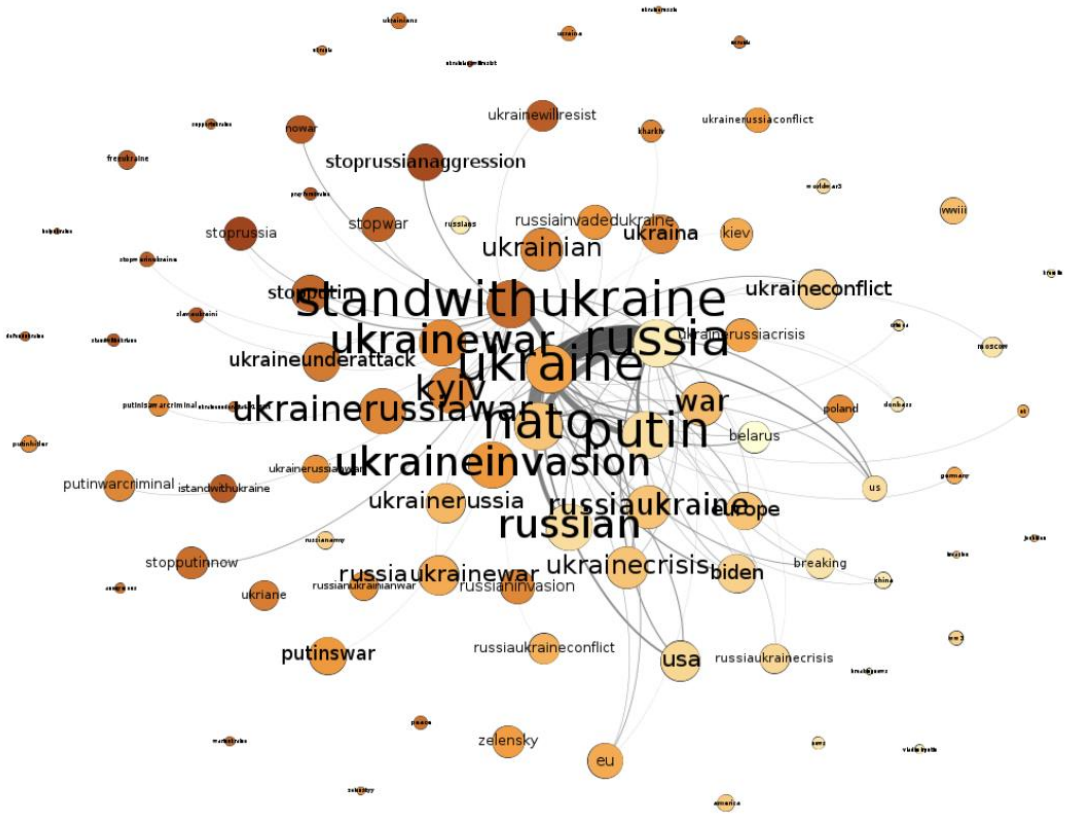

Рисунок 1.14 – Ядро побудованої мережі найбільш вживаних хештегів

На даному зображені вершини представляють хештеги, а ребра представляють повідомлення, де хештеги, котрі належать вершинам, що пов’язані даним ребром, зустрічаються в парі. Мінімальний ступінь входження кожної вершини графу мережі рівний 51. Товщина ребер на зображені пропорційна кількості повідомлень, де хештеги зустрічаються в парі. Розмір самих вершин пропорційний до кількості появи хештегів, котрі належать їм. Деякі тонкі ребра приховані на зображені. Більш світлий колір вершин означає вищий ступінь проросійської спрямованості хештегу.

Підмережа мережі пересланих повідомлень користувачів у контексті українсько-російської війни, що має мінімальний ступінь входження кожної вершини графу мережі рівний 5, зображено на рисунку 1.15.

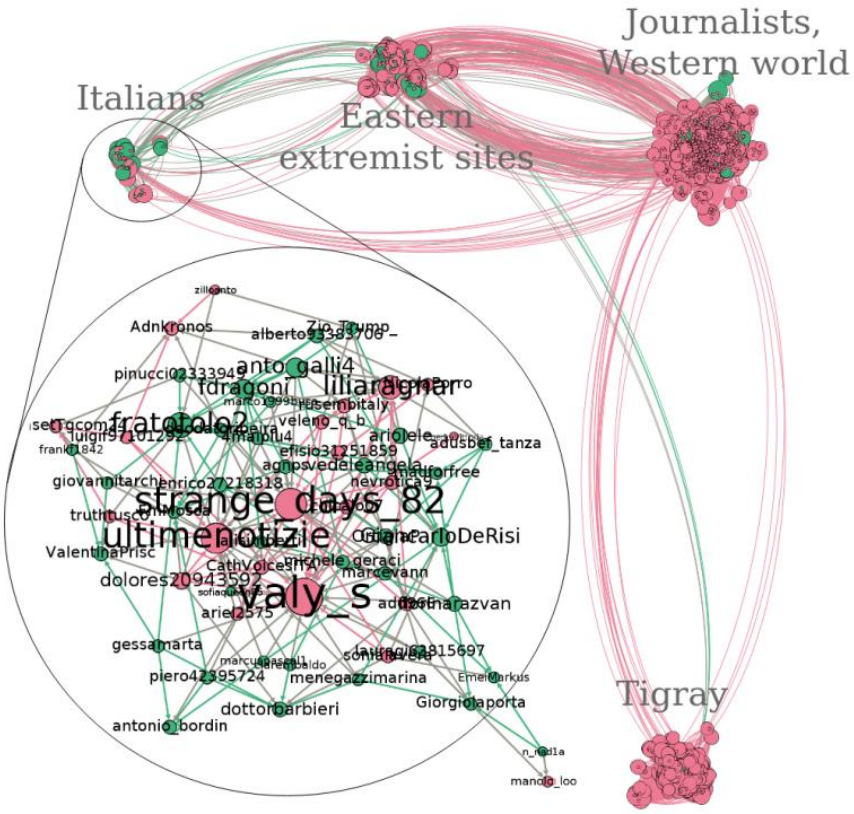

Рисунок 1.15 – Підмережа мережі пересланих повідомлень користувачів

На даному зображені вершини представляють користувачі, а ребра представляють повідомлення, котрі пересилались користувачем, що представляє першу вершину, від користувача, що представляє другу вершину з пари вершин, котрі формують ребро. Розмір вершини рівний кількості повідомлень, котрі були переслані від даного користувача. Товщину ребер визначає кількість пересланих повідомлень від першого користувача до другого. Червоний колір позначає ступінь проукраїнської спрямованості користувачів чи повідомлень, а зелений – ступінь проросійської спрямованості. Найбільша зв’язана компонента в даній підмережі містить 8575 вершин та 47851 ребро.

Висновки до розділу

В даному розділі було проведено детальний огляд методів та засобів розв’язання задачі розпізнавання та аналітики пропаганди в текстах. Поставлено задачу виявлення пропаганди та запропоновано загальну схему її розв’язання. Проаналізовано питання вибору джерел для збору даних, оглянуто різні способи збору даних з джерел та методи попередньої обробки тексту. Розглянуто різні методи вилучення ознак з тексту, зокрема модель “торба слів”, модель N-грам та вкладання слів. Проаналізовано методи текстової класифікації, зокрема логістичну регресію, метод опорних векторів, метод випадкових лісів та різні моделі нейронних мереж, включаючи класичну модель RNN, модель LSTM, модель трансформера та модель BERT. Проведено порівняння методів текстової класифікації. Також оглянуто методи статистичної обробки даних, зокрема регресійний аналіз, аналіз головних компонент, кластерний аналіз, аналіз часових рядів, аналіз мереж та аналіз просторових даних. Наостанок, було проведено огляд існуючих рішень щодо розпізнавання та аналітики пропаганди в текстах.

2 РОЗРОБКА МЕТОДУ ТА СПОСОБІВ РОЗВ’ЯЗАННЯ ЗАДАЧІ

2.1 Змістовна постановка задач

Розроблювана система має вирішувати наступний набір задач.

Задача 1. На основі переданого посилання на джерело повідомлень визначати його політичну полярність як усереднене значення політичної полярності зчитаних з джерела повідомлень.

Задача 2. На основі зібраних даних з мережі Twitter, будувати мережу найпопулярніших хештегів, що стосуються українсько-російської війни. Вершини графу, що представляє мережу, мають відображати хештеги. Ребра графу мають відображати повідомлення, в котрих попарно зустрічались хештеги, що відповідають вершинам на кінцях ребра.

Задача 3. На основі зібраних даних з месенджеру Telegram, будувати мережу популярних телеграм-каналів, що стосуються українсько-російської війни. Вершини графу, що представляє мережу, мають відображати телеграм-канали. Ребра графу мають відображати повідомлення, в котрих канал, що відповідає першій вершині на кінці ребра, посилається на канал, що відповідає другій вершині на кінці ребра.

Задача 4. На основі побудованих мереж будувати підмережі з використанням заданих параметрів.

Задача 5. Збирати статистичні дані про кількість підписників на популярні джерела повідомлень, разом з інформацією про реакції на повідомлення підписниками, та про політичну полярність даних джерел.

2.2 Розробка способу визначення політичної полярності джерела повідомлень

Опишемо спосіб визначення політичної полярності джерела повідомлень за переданим посиланням можна описати наступним чином (для скорочення викладок,

тут і далі опускаються перевірки на коректність даних та обробки виняткових ситуацій).

Дано:

— ідентифікатор I джерела повідомлень;

— значення кількості останніх повідомлень k ($k \geq 20$), котре потрібно зчитати з джерела для визначення його політичної полярності;

— множина методів попередньої обробки повідомлень M.

Знайти:

Політичну полярність джерела повідомлень x^{src}.

Розв’язання:

Крок 1. Зчитати k останніх повідомлень з джерела повідомлень з ідентифікатором I.

Крок 2. Виконати попередню обробку зчитаних повідомлень методами з множини M.

Крок 3. Обрати серед оброблених повідомлень ті, що мають довжину від 50 до 5000 символів, тим самим прибравши занадто короткі та занадто довгі повідомлення.

Крок 4. Виконати переклад повідомлень на англійську мову.

Крок 5. Для кожного перекладеного повідомлення сформувати вилучити його ознаки за обраним підходом.

Крок 6. Для кожного перекладеного повідомлення визначити його політичну полярність за допомогою попередньо навченого класифікатора та вилучених з повідомлення ознак. Класифікатор розглядається як функція, що ставить у співвідношення елементи з множини повідомлень елементи з множини результатів класифікації. В даному алгоритмі, результатом класифікації є трійка $x = (u, r, n)$, де u, r та n ($u, r, n \in \mathbb{R}; u, r, n \in [0; 1]$) відображають ступінь впевненості моделі щодо приналежності повідомлення до класу проукраїнських, проросійських та нейтральних відповідно.

Крок 7. Усереднити політичну полярність повідомлень, повернути усереднене значення x^{src} як результат визначення політичної полярності джерела повідомлень та завершити алгоритм.

2.3 Розробка способу побудови мережі найпопулярніших хештегів

Дамо опис способу побудови графу G, що представляє мережу найпопулярніших хештегів.

Дано:

— множина A акаунтів мережі Twitter, $|A| \geq 50$;

— кількість повідомлень n ($n \in \mathbb{N}; n \geq 100$), котрі потрібно зчитати з акаунтів множини A;

— множина методів попередньої обробки повідомлень M;

— значення параметра l_{min} ($l_{min} \in \mathbb{N};\ l_{min} \geq 20$);

— значення параметрів m_{min} ($m_{min} \in \mathbb{N};\ m_{min} \geq 20$) та m_{max} ($m_{max} \in \mathbb{N};\ m_{max} > m_{min}$);

— значення параметрів p та p' ($p,\ p' \in \mathbb{R};\ p,\ p' \in (0; 1]$);

— значення параметра r_{min} ($r_{min} \in \mathbb{R};\ r_{min} \in [0; 1]$);

— значення параметра f_{min} ($f_{min} \in \mathbb{N}$).

Знайти:

Граф G, що представляє мережу найпопулярніших хештегів.

Розв’язання:

Крок 1. Зчитати з аккаунтів множини A останні n повідомлень.

Крок 2. Виконати попередню обробку зчитаних повідомлень методами з множини M.

Крок 3. Обрати серед оброблених повідомлень ті, що мають довжину від 50 до 5000 символів, та містять в собі непорожню множину хештегів.

Крок 4. Згрупувати оброблені повідомлення за хештегами, тим самим сформувавши множину $B = \{b_1, b_2, \dots, b_m\}$, де $b_i = (h_i, L_i)$, h_i – хештег, L_i – множина повідомлень, що містять хештег h_i та за умови $|L_i| \geq l_{min}$.

Крок 5. На основі множини B побудувати множину вершин C графу G.

Крок 5.1. Для кожного хештегу h_i, що знаходиться в множині B, взяти p відсотків повідомлень з множини повідомлень L_i, де цей хештег зустрічається, проте не менше за значення параметра m_{min} та не більше за значення параметра m_{max}. Позначити нову підмножину повідомлень як L'_i.

Крок 5.2. Сформувати множину $B' = \{b'_1, b'_2, \dots, b_k'\}$, де $b'_j = (h_j, L'_j)$.

Крок 5.3. Перекласти на англійську всі повідомлення L'_i, що знаходяться в множинні B'.

Крок 5.4. Для кожного перекладеного повідомлення визначити його політичну полярність $x_{i,j}$ за допомогою попередньо навченого класифікатора та вилучених з повідомлення ознак.

Крок 5.5. Визначити усереднену політичну полярність повідомлень x_i^{avg}.

Крок 5.6. Сформувати множину вершин $C = \{c_1, c_2, \dots, c_k\}$, де $c_i = \left(h_i, L_i, x_i^{avg}\right)$.

Крок 6. На основі множини C, побудувати множину ребер D графу G.

Крок 6.1. Сформувати з множини C множину всіх можливих пар вершин графу $E = C \times C$.

Крок 6.2. Для кожної пари вершин $(c_i, c_j) \in E$ визначити множину повідомлень $L_{i,j}$ в яких попарно зустрічались хештеги h_i та h_j, що представляють дані вершини.

Крок 6.3. Сформувати з множини E підмножину E', де $\left|L_{i,j}\right| \geq max\left(r_{min} * min\left(|L_i|, \left|L_j\right|\right), f_{min}\right)$.

Крок 6.4. Для кожної пари $(c_i, c_j) \in E'$ взяти p' відсотків повідомлень з множини повідомлень $L_{i,j}$, проте не менше за значення параметра m_{min} та не більше за значення параметра m_{max}. Позначити дану підмножину повідомлень як $L'_{i,j}$.

Крок 6.5. Перекласти на англійську всі повідомлення $L'_{i,j}$.

Крок 6.6. Для кожного перекладеного повідомлення визначити його політичну полярність $y_{i,j,k}$ за допомогою попередньо навченого класифікатора та вилучених з повідомлення ознак.

Крок 6.7. Визначити усереднену політичну полярність повідомлень $y_{i,j}^{avg}$.

Крок 6.8. Сформувати множину ребер $D = \cup\{d_{i,j}\}$, де $d_{i,j} = \left(h_{i,j}, L_{i,j}, y_{i,j}^{avg}\right)$, $h_{i,j} = \left(h_i, h_j\right)$.

Крок 7. Повернути граф $G = \langle C, D \rangle$ та завершити алгоритм.

2.4 Розробка способу побудови мережі популярних телеграм-каналів

Опишемо спосіб побудови графу G, що представляє мережу популярних телеграм-каналів.

Дано:

— множина A_0 телеграм-каналів, $|A_0| \geq 1$;

— кількість повідомлень n ($n \in \mathbb{N}; n \geq 100$), котрі потрібно зчитати з акаунтів множини A_0;

— множина методів попередньої обробки повідомлень M;

— значення параметра s_{min} ($s_{min} \in \mathbb{N};\ s_{min} \geq 10000$);

— значення параметра a_{dest} ($a_{dest} \in \mathbb{N};\ a_{dest} \geq |A_0|$);

— значення параметра k ($k \in \mathbb{N};\ k \geq 20$);

— значення параметра r_{min} ($r_{min} \in \mathbb{N};\ r_{min} \geq 1$).

Знайти:

Граф G, що представляє мережу популярних телеграм-каналів.

Розв’язання:

Крок 1. Зчитати з телеграм-каналів множини $A_0 = \{a_{0,1}, a_{0,2}, \dots, a_{0,m}\}$ останні n повідомлень. Визначити множину опрацьованих каналів $A_{proc} = \{\}$.

Крок 2. З множини зчитаних повідомлень, для кожного каналу $a_{0,i}$ обрати підмножину повідомлень, що містять посилання на інші канали.

Крок 3. Виконати попередню обробку повідомлень з підмножини за обраними методами.

Крок 4. Обрати серед оброблених повідомлень ті, що мають довжину від 50 до 5000 символів та сформувати з них множину $L_{0,i}$.

Крок 5. Сформувати множину посилань на канали $B_{0,i}$, які знайдені в повідомленнях з множини $L_{0,i}$ які мають кількість підписників, більшу за параметр s_{min} та які не належать множині A_0 і множині попередньо опрацьованих каналів A_{proc}.

Крок 6. Почергово виконавши пошук посилань на інші канали з каналів з множини A_0, додати канали з множини A_0 в множину A_{proc} та перевірити чи кількість опрацьованих каналів $|A_{proc}|$ рівна бажаній a_{dest}. Якщо так – перейти до кроку 7, інакше – сформувати множину каналів $A_1 = \cup B_{0,j}$, і почергово виконати кроки 2 – 5 для каналів з множини A_1, розширюючи множину A_{proc} після обробки чергового каналу, поки:

— кількість опрацьованих каналів $|A_{proc}|$ буде рівною бажаній a_{dest} – тоді перейти до кроку 7;

— всі канали з множини A_1 опрацьовані, проте $|A_{proc}| < a_{dest}$ і $B_1 \neq \emptyset$ – тоді виконати крок 6, замінюючи множину A_0 на A_1 (або множину A_{i-1} на множину в A_i у загальному випадку);

— всі канали з множини A_1 опрацьовані, проте $|A_{proc}| < a_{dest}$ і $B_1 = \emptyset$ – тоді перейти до кроку 7.

Крок 7. Для кожного каналу $a_i \in A_{proc}$ зчитати останні k повідомлень.

Крок 8. Виконати попередню обробку повідомлень з множини зчитаних повідомлень методами з множини M.

Крок 9. Обрати серед оброблених повідомлень ті, що мають довжину від 50 до 5000 символів та сформувати з них множину R_i.

Крок 10. Сформувати множину $C = \{(a_i, R_i) | a_i \in A_{proc}\}$.

Крок 11. На основі множини C побудувати множину вершин D графу G.

Крок 11.1. Перекласти на англійську всі повідомлення множини R_i, що знаходяться в множинні C.

Крок 11.2. Для кожного перекладеного повідомлення визначити його політичну полярність $x_{i,j}$ за допомогою попередньо навченого класифікатора та вилучених з повідомлення ознак.

Крок 11.3. Визначити усереднену політичну полярність повідомлень x_i^{avg}.

Крок 11.4. Сформувати множину вершин $D = \{d_1, d_2, \dots, d_k\}$, де $d_i = (a_i, L_i, s_i, x_i^{avg})$, s_i – кількість підписників каналу a_i, L_i – множина повідомлень з каналу a_i з наявними посиланнями на інші канали, що була знайдена на кроці 5.

Крок 12. На основі множини D побудувати множину ребер E графу G.

Крок 12.1. Сформувати з множини D множину всіх можливих пар вершин графу $H = D \times D$.

Крок 12.2. Для кожної пари вершин $(d_i, d_j) \in H$ визначити множину повідомлень $L_{i,j}$ в яких зустрічались посилання на канали a_i та a_j, що представляють дані вершини.

Крок 12.3. Сформувати з множини H підмножину H', де $|L_{i,j}| \geq r_{min}$.

Крок 12.4. Перекласти на англійську всі повідомлення $L_{i,j}$.

Крок 12.5. Для кожного перекладеного повідомлення визначити його політичну полярність $y_{i,j,k}$ за допомогою попередньо навченого класифікатора та вилучених з повідомлення ознак.

Крок 12.6. Визначити усереднену політичну полярність повідомлень $y_{i,j}^{avg}$.

Крок 12.7. Сформувати множину ребер $E = \cup\{e_{i,j}\}$, де $e_{i,j} = \left(a_{i,j}, L_{i,j}, y_{i,j}^{avg}\right)$, $a_{i,j} = \left(a_i, a_j\right)$.

Крок 13. Повернути граф $G = \langle D, E \rangle$ та завершити алгоритм.

2.5 Розробка способу побудови підмережі найпопулярніших хештегів за заданою мережею

Опишемо спосіб побудови підмережі найпопулярніших хештегів за заданою мережею.

Дано:

— мережа найпопулярніших хештегів, що представлена графом $G = \langle C, D \rangle$, де $C = \{c_1, c_2, \dots, c_k\}$, $c_i = \left(h_i, L_i, x_i^{avg}\right)$, $D = \cup\{d_{i,j}\}$, $d_{i,j} = \left(h_{i,j}, L_{i,j}, y_{i,j}^{avg}\right)$ (h_i, L_i, x_{avg} та $h_{i,j}$, $L_{i,j}$, y_{avg} описані в алгоритмі побудови мережі найпопулярніших хештегів);

— значення параметрів t_{min}, t_{max} ($t_{min} \geq \min(|L_i|)$, $t_{max} \leq \max(|L_i|)$);

— значення параметрів s_{min}, s_{max} ($s_{min} \geq \min\left(|L_{i,j}|\right)$, $s_{max} \leq \max\left(|L_{i,j}|\right)$);

— значення параметрів x_{min}, x_{max} ($x_{min} \geq \min(X)$, $x_{max} \leq \max(X)$, $X = \cup\{x_i^{avg}\}$);

— значення параметрів y_{min}, y_{max} ($y_{min} \geq \min(Y)$, $y_{max} \leq \max(Y)$, $Y = \cup\{y_{i,j}^{avg}\}$).

Знайти:

Підграф G', що представляє підмережу найпопулярніших хештегів.

Розв’язання:

Крок 1. Сформувати множину $C' = \left\{ c_j \middle| \begin{matrix} c_j \in C,\ c_j = \left(h_j, L_j, x_j^{avg}\right), \\ t_{min} \leq |L_j| \leq t_{max}, \\ x_{min} \leq x_j^{avg} \leq x_{max} \end{matrix} \right\}$.

Крок 2. Сформувати множину $D' = \left\{ d_{i,j} \middle| \begin{matrix} d_{i,j} \in D,\ d_{i,j} = \left(h_{i,j}, L_{i,j}, y_{i,j}^{avg}\right), \\ s_{min} \le |L_{i,j}| \le s_{max}, \\ y_{min} \le y_{i,j}^{avg} \le y_{max} \end{matrix} \right\}$.

Крок 3. Повернути граф $G' = \langle \mathrm{C}', D' \rangle$ та завершити алгоритм.

2.6 Розробка способу підмережі популярних телеграм-каналів за заданою мережею

Опишемо спосіб побудови підмережі популярних телеграм-каналів за заданою мережею.

Дано:

— мережа популярних телеграм-каналів, що представлена графом $G = \langle D, E \rangle$, де $D = \{d_1, d_2, \dots, d_k\}$, $d_i = \left(a_i, L_i, s_i, x_i^{avg}\right)$, $E = \bigcup\{e_{i,j}\}$, $e_{i,j} = \left(a_{i,j}, L_{i,j}, y_{i,j}^{avg}\right)$ (зміні a_i, L_i, t_i, x_{avg} та $a_{i,j}$, $L_{i,j}$, y_{avg} описані в алгоритмі побудови мережі популярних телеграм-каналів);

— значення параметрів l_{min}, l_{max} ($l_{min} \ge \min(|L_i|)$, $l_{max} \le \max(|L_i|)$);

— значення параметрів t_{min}, t_{max} ($t_{min} \ge \min\left(|L_{i,j}|\right)$, $t_{max} \le \max\left(|L_{i,j}|\right)$);

— значення параметрів s_{min}, s_{max} ($s_{min} \ge \min(S)$, $s_{max} \le \max(S)$, $S = \bigcup\{s_i\}$);

— значення параметрів x_{min}, x_{max} ($x_{min} \ge \min(X)$, $x_{max} \le \max(X)$, $X = \bigcup\{x_i^{avg}\}$);

— значення параметрів y_{min}, y_{max} ($y_{min} \ge \min(Y)$, $y_{max} \le \max(Y)$, $Y = \bigcup\{y_{i,j}^{avg}\}$).

Знайти:

Підграф G', що представляє підмережу популярних телеграм-каналів.

Розв’язання:

Крок 1. Сформувати множину $D' = \left\{ d_j \middle| \begin{matrix} d_j \in D,\ d_i = \left(a_i, L_i, s_i, x_i^{avg}\right), \\ l_{min} \le |L_i| \le l_{max}, \\ s_{min} \le s_i \le s_{max}, \\ x_{min} \le x_i^{avg} \le x_{max} \end{matrix} \right\}$.

Крок 2. Сформувати множину $E' = \left\{ e_{i,j} \middle| \begin{matrix} e_{i,j} \in E,\ e_{i,j} = \left(a_{i,j}, L_{i,j}, y_{i,j}^{avg}\right), \\ t_{min} \le |L_{i,j}| \le t_{max}, \\ y_{min} \le y_{i,j}^{avg} \le y_{max} \end{matrix} \right\}$.

Крок 3. Повернути граф $G' = \langle D', E' \rangle$ та завершити алгоритм.

2.7 Розробка способу збору статистичних даних по джерелах повідомлень

Опишемо спосіб збору статистичних даних по джерелах повідомлень

Дано:

— Множина A джерел повідомлень;

— Кількість повідомлень n ($n \in \mathbb{N}; n \geq 100$), котрі потрібно зчитати з джерел множини A.

Знайти:

Множину зібраних статистичних даних S.

Розв’язання:

Крок 1. Зчитати зі заданої множини A джерел повідомлень останні n повідомлень.

Крок 2. Визначити політичну полярність x_i^{avg} джерел повідомлень $a_i \in A$.

Крок 3. Для кожного джерела повідомлень $a_i \in A$ визначити кількість підписників s_i.

Крок 4. З множини зчитаних повідомлень L_i джерел $a_i \in A$, дістати множину реакцій на дані повідомлення $R_i = \{r_{i,1}, r_{i,2}, \dots, r_{i,m}\}$, де $r_{i,j} = (t_{i,j}, c_{i,j})$, $t_{i,j}$ – тип реакції, $c_{i,j}$ – сумарна кількість реакцій на повідомлення з множини L_i типу $t_{i,j}$.

Крок 5. Повернути множину $S = \{(a_i, x_i^{avg}, s_i, R_i) \mid a_i \in A\}$ як результат збору статистичних даних та звершити алгоритм.

Висновки до розділу

У даному розділі виконано змістовну постановку задач, котрі має вирішувати розроблювана система задля виконання розпізнавання та аналізу пропаганди в текстах. Було описано способи:

— визначення політичної полярності джерела повідомлень;

— побудови мережі найпопулярніших хештегів;

— побудови мережі популярних телеграм-каналів;

— побудови підмережі найпопулярніших хештегів за заданою мережею;

— побудови підмережі популярних телеграм-каналів за заданою мережею;

— збору статистичних даних по джерелах повідомлень.

3 ФОРМУВАННЯ НАБОРУ ДАНИХ

3.1 Особливості проросійської та проукраїнської риторики

Російська пропаганда має ряд особливостей, таких як:

— використання підроблених або змінених фактів. Можуть використовуватись неправдиві статистичні дані, фотографії з інших подій, або змінювати контекст дійсної події;

— створення штучних ворогів щоб відволікти увагу від внутрішніх проблем;

— використання загальноприйнятих міфів. Наприклад, міф про “велику росію” для підтримки анексії Криму, міф про "західну загрозу" для виправдання своєї військової агресії;

— використання риторики про втрату заходом традиційних цінностей;

— використання різноманітних риторичних тактик, таких як “Tu quoque” чи “Argumentum ad hominem”.

Переглянемо деякі з відомих проросійських джерел повідомлень для підтвердження наявності в них пропаганди. Для прикладу візьмемо телеграм-канал “РИА Новости”, котрий представляє російське агентство міжнародної інформації та має понад 2.6 мільйонів підписників. Дане агентство займається публікацією багатьох неправдивих та вигідних російській владі новин, що підтверджено статистичними методами та методами перевірки фактів [25]. Приклад фейкової новини в даному телеграм-каналі зображено на рисунку 3.1.

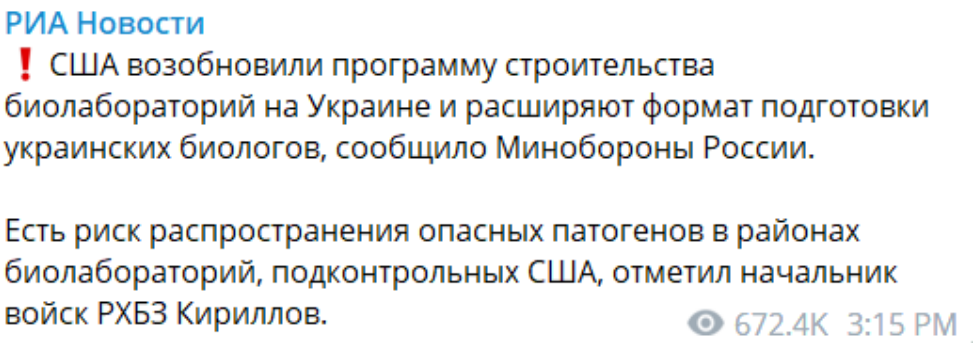

Рисунок 3.1 – Приклад проросійської фейкової новини

Використання популярної для російської пропаганди риторичної тактики “Tu quoque” можна зустріти, наприклад, в іншому проросійському телеграм-каналі “Kadyrov_95” з понад 3.1 мільйонами підписників, що є офіційним каналом глави Чечні Рамзана Кадирова. Приклад повідомлення з даною риторичною тактикою зображено на рисунку 3.2.

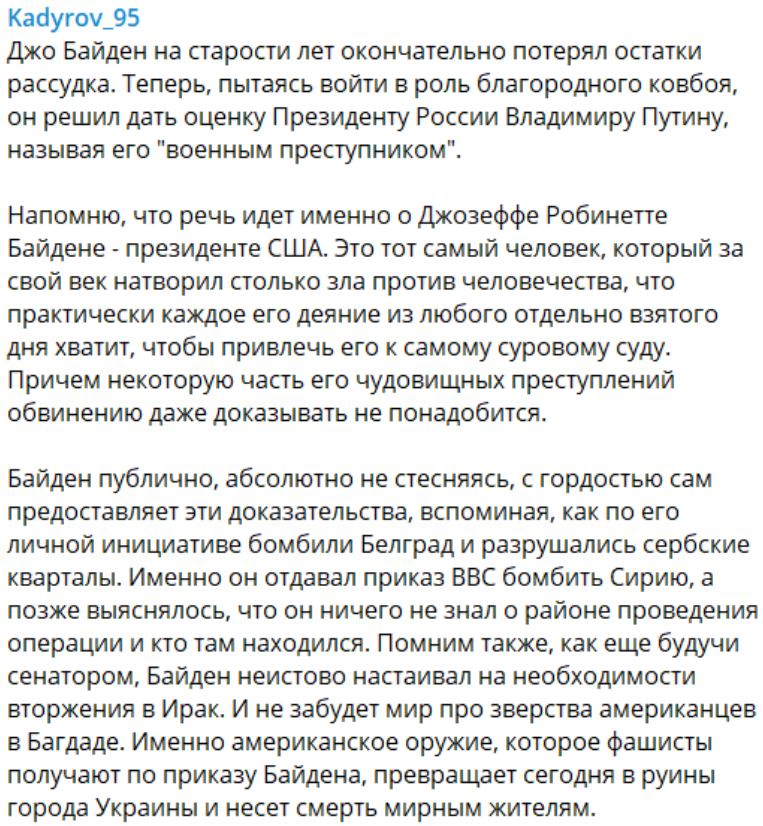

Рисунок 3.2 – Приклад повідомлення з наявною риторичною тактикою “Tu quoque”

Риторику про втрату заходом традиційних цінностей можна зустріти, наприклад, у телеграм-каналі “Мария Захарова” з понад 530 тисячами підписників, що є офіційним каналом російської політичної діячки Марії Захарової. Приклад повідомлення з наявною риторикою зображено на рисунку 3.3.

Рисунок 3.3 – Приклад повідомлення з наявною риторикою про втрату заходом традиційних цінностей

Міф про “західну загрозу”, що часто використовується в якості виправдання росіянами військового вторгнення на територію України, можна зустріти в телеграм-каналі “СОЛОВЬЁВ” з понад 1.3 мільйонами підписників, котрий належить одному з найпопулярніших в росії телеведучому та пропагандисту Володимиру Соловйову. Приклад повідомлення з наявним в ньому міфом зображено на рисунку 3.4.


СОЛОВЬЁВ
Forwarded from Кремлёвская прачка
✔️В любой непонятной ситуации расширяй НАТО по всем возможным направлениям - это рецепт на все случаи жизни у большинства американских политиков и экспертов. Вот и бывший советник Трампа Джон Болтон не отстаёт от собратьев и продвигает идею о том, что «надо больше НАТО». Эта военно-политическая группировка и так-то разрастается как раковая опухоль, а американцам все мало. В болтоновском видении очередная волна расширения альянса должна быть осуществлена за счёт включения в него Японии, Австралии, Израиля и других стран.

Для чего это нужно? Для того, чтобы попытаться разрушить «ось Россия - Китай». Видите ли, после окончания холодной войны мир ненадолго погрузился в «иллюзорный отдых», а теперь вновь «растут угрозы». Прежде всего - от Москвы и Пекина. Поэтому хорош «отдыхать», пора бы и за дело приниматься: расширять известные ущербные военные объединения и повсеместно творить хаос. В практическом плане это означает в том числе: 1) увеличение оборонных бюджетов Вашингтона и его союзников до «рейгановских значений» (6% ВВП); 2) совершенствование всех шарашек под эгидой Вашингтона - НАТО и AUKUS.

Рисунок 3.4 – Приклад повідомлення з наявним міфом про “західну загрозу”

Проросійські наративи можуть також зустрічатись і серед західних політиків правих політичних поглядів. В основному дані політики не вдаються до поширення дезінформації чи возвеличення росії, проте просувають наступні ідеї серед населення своїх країн:

— ідея про те, що народ повинен сконцентруватись на інтересах власної держави, замість витрачання частини бюджету на військову підтримку України;

— ідея про те, що військова підтримка України є грошовою схемою;

— ідея про те, що Україна має поступитись частиною власних територій задля закінчення військового конфлікту з росією;

— ідея про те, що військовий конфлікт України з росією може перерости в ядерну війну при передачі Україні певних видів озброєння.

Відомими західними політиками, котрі поширюють проросійські наративи є, наприклад, угорський державний діяч Віктор Орбан та республіканка Марджорі

Тейлор Ґрін. Приклад повідомлення від Віктора Орбана з грою на “ядерному страхові” в мережі Twitter зображено на рисунку 3.5.

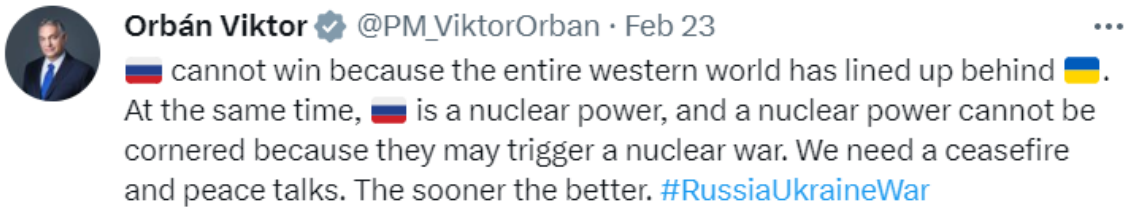

Рисунок 3.5 – Приклад поширення проросійських наративів серед західних політиків

Проукраїнські джерела повідомлень, на відміну від проросійських, відрізняються значно меншою кількістю дезінформації, чіткою позицією щодо питання територіальної цілісності країни, орієнтацією на західні цінності, баченням країни в складі Євросоюзу та НАТО в майбутньому, схваленням підтримки міжнародних партнерів тощо.

3.2 Вибір джерел для збору даних

В рамках даної роботи, джерелами даних є популярні телеграм-канали та користувачі мереж Twitter. Майбутній набір даних повинен містити екземпляри повідомлень, що підпадають під кожен клас, котрий здатна розрізняти класифікаційна модель. Такими класами є джерела проросійських, проукраїнських та політично нейтральних повідомлень.

Кожне джерело повідомлень має свій особливий формат викладу інформації, котрий може відрізнятись кількістю суб’єктивних припущень, емоційною забарвленістю, середньою довжиною повідомлення, наявністю складних лінгвістичних структур тощо. Тому, для точної класифікації реальних даних, класифікаційна модель має бути навчена на гетерогенних екземплярах джерел повідомлень. Відповідно, було вирішено сформувати набір даних з повідомлень,

зібраних з 20 телеграм-каналів та 20 користувачів мережі Twitter, по кожному з класів класифікації.

Визначимо які джерела повідомлень, що стосуються теми українсько-російської війни, є найпопулярнішими. У виборі надаватимемо перевагу джерелам повідомлень, котрі являють собою державні комунікації, представляють політиків або воєнного спрямування. Далі за пріоритетністю – медіа та пабліки. Зазначимо, що джерела повідомлень, котрі мають більше підписників, більшу частоту нових повідомлень та сильнішу емоційну забарвленість, будуть мати більший пріоритет в виборі в рамках однієї категорії. Для визначення популярних проукраїнських телеграм-каналів, використаємо за основу список 100 найпопулярніших телеграм-каналів, котрий сформований редакцією “Детектор медіа” у липні 2022 року [26]. Для визначення популярних проросійських телеграм-каналів, скористаємось сервісом аналітики телеграм-каналів TgStat [27].

Сформований список проукраїнських та проросійських телеграм-каналів, повідомлення з котрих потраплять до навчального набору даних, надано в таблицях 3.1 та 3.2 відповідно.

Таблиця 3.1 – Проукраїнські телеграм-канали для формування набору даних

Ідентифікатор	**Категорія**	**Кількість підписників (тис.)**
@truexanewsua	Паблік	2662
@u_now	Паблік	1714
@lachentyt	Паблік	1136
@TCH_channel	Медіа	955
@V_Zelenskiy_official	Представляє політика	941
@uniannet	Медіа	853
@UkraineNow	Паблік	774

@mykolaivskaODA	Представляє політика	651
@ssternenko	Паблік	508
@KyivCityOfficial	Державні комунікації	501
@Pravda_Gerashchenko	Представляє політика	497
@perepicka_news	Паблік	460
@SBUkr	Державні комунікації	369
@suspilnenews	Медіа	315
@ukraina24tv	Медіа	290
@CinCAFU	Воєнного спрямування	256
@verkhovnaradaukrainy	Державні комунікації	193
@ermaka2022	Представляє політика	183
@kurtievofficial	Представляє політика	180
@OP_UA	Державні комунікації	151

Таблиця 3.2 – Проросійські телеграм-канали для формування набору даних

Ідентифікатор	**Категорія**	**Кількість підписників (тис.)**
@RKadyrov_95	Представляє політика	3174
@novosti_voinaa	Паблік	2874
@rian_ru	Медіа	2679
@readovkanews	Паблік	1764
@ostorozhno_novosti	Паблік	1362
@SolovievLive	Паблік	1326
@wargonzo	Воєнного спрямування	1261
@medvedev_telegram	Представляє політика	1043
@Sladkov_plus	Воєнного спрямування	989
@vv_volodin	Представляє політика	945

@epoddubny	Воєнного спрямування	853
@warfakes	Паблік	712
@rt_russian	Медіа	703
@voenacher	Воєнного спрямування	652
@sashakots	Воєнного спрямування	652
@MariaVladimirovnaZakharova	Представляє політика	530
@milinfolive	Воєнного спрямування	529
@mod_russia	Державні комунікації	480
@voenkorKotenok	Воєнного спрямування	419
@news_kremlin	Державні комунікації	160

Для визначення популярних проукраїнських та проросійських облікових записів мережі Twitter, скористаємось пошуковим рядком, що надає сама мережа. Оберемо розділ “People”, та вкажемо слова “Ukraine” чи “Russia” для отримання рекомендацій щодо популярних проукраїнських чи проросійських облікових записів відповідно. Приклад пошуку популярних проукраїнських облікових записів зображено на рисунку 3.6.

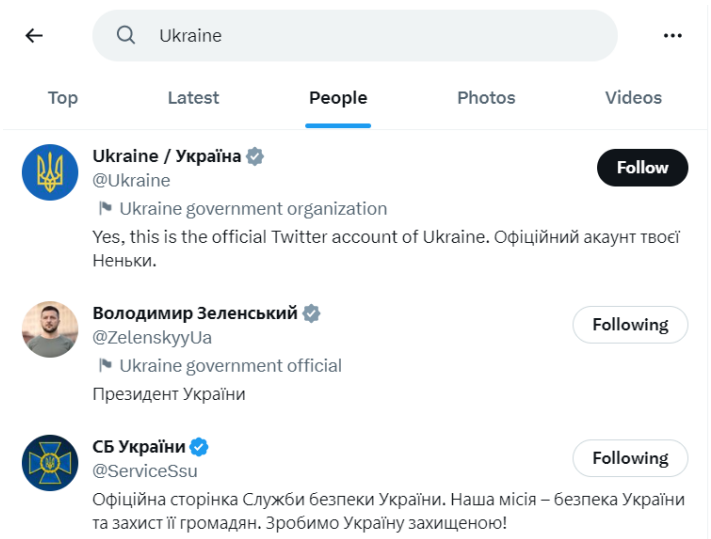

Рис. 3.6 – Пошук популярних проукраїнських облікових записів у мережі Twitter

Сформований список проукраїнських та проросійських облікових записів Twitter, повідомлення з котрих потраплять до навчального набору даних, надано в таблицях 3.3 та 3.4 відповідно.

Таблиця 3.3 – Проукраїнські облікові записи Twitter для формування набору даних

Ідентифікатор	**Категорія**	**Кількість підписників (тис.)**
@ZelenskyyUa	Представляє політика	7200
@DefenceU	Державні комунікації	1800
@poroshenko	Представляє політика	1300
@DmytroKuleba	Представляє політика	1100
@Podolyak_M	Представляє політика	1000
@ukrpravda_news	Медіа	907
@serhiyprytula	Представляє політика	893
@radiosvoboda	Медіа	665
@Vitaliy_Klychko	Представляє політика	635
@Yatsenyuk_AP	Представляє політика	546
@operativno_ZSU	Воєнного спрямування	485
@MFA_Ukraine	Державні комунікації	462
@Turchynov	Представляє політика	289
@BackAndAlive	Воєнного спрямування	269
@MVS_UA	Державні комунікації	257
@DI_Ukraine	Державні комунікації	238
@ZelenskaUA	Представляє політика	196
@KpsZSU	Воєнного спрямування	173
@Denys_Shmyhal	Представляє політика	153
@TDF_UA	Воєнного спрямування	86

Таблиця 3.4 – Проросійські облікові записи Twitter для формування набору даних

Ідентифікатор	**Категорія**	**Кількість підписників (тис.)**
@vesti_news	Медіа	2900
@channelone_rus	Медіа	2900
@RepMTG	Представляє політика	2100
@GazetaRu	Медіа	1800
@KremlinRussia_E	Державні комунікації	1700
@VRSoloviev	Паблік	1400
@MID_RF	Державні комунікації	1200
@Rogozin	Представляє політика	794
@mfa_russia	Державні комунікації	595
@M_Simonyan	Паблік	544
@GovernmentRF	Державні комунікації	341
@kpru	Медіа	317
@mironov_ru	Представляє політика	271
@er_novosti	Державні комунікації	165
@PM_ViktorOrban	Представляє політика	145
@mkomsomolets	Медіа	144
@ldprparty	Державні комунікації	106
@kprf	Державні комунікації	100
@vzglyad	Медіа	85
@wargonzoo	Воєнного спрямування	44

В якості політично нейтральних джерел повідомлень було обрано джерела, що стосуються сфер розваг, мистецтва, науки, програмування, спорту, бізнесу, моди, харчування тощо. Кількість підписників та емоційна забарвленість повідомлень в

даних джерелах не впливає на якість класифікації моделлю, що буде навчатись, тому деталі про джерела опускаються. В якості нейтральних телеграм-каналів, що будуть використані для формування навчального набору даних, було обрано наступні: @startupsi, @nn_for_science, @physics_lib, @sgstudentpromos, @npzbvnkngchtn, @computer_science_and_programming, @cooking_support, @quote, @askmenow, @netflix, @natgeohub, @CryptoWorldNews, @cooking_master, @medcase, @linkedin_learning, @dailycoding, @NutritionistValerie, @discudemy_com, @newgames, @linuxgram. Серед політично нейтральних облікових записів у мережі Twitter було обрано такі: @Discovery, @NASA, @BillGates, @Cristiano, @Tesla, @MarvelStudios, @satyanadella, @ScienceMagazine, @VictoriasSecret, @NYTFashion, @metmuseum, @Literature, @jamieoliver, @GuyFieri, @OpenAI, @Google, @APA, @ScienceNews, @natgeowild, @atlblog.

3.3 Збір даних з джерел

Опишемо алгоритм збору даних з джерел повідомлень для формування навчального набору даних.

Крок 1. Зчитати з джерела N останніх повідомлень.

Крок 2. Виконати попередню обробку тексту повідомлень.

Крок 3. Обрати серед оброблених повідомлень ті, що мають довжину від 50 до 5000 символів, тим самим прибравши занадто короткі та занадто довгі повідомлення.

Крок 4. Обрати перші M повідомлень ($M \leq N$).

Крок 5. Перекласти повідомлення на англійську мову.

Крок 6. Зберегти успішно перекладені повідомлення в базу даних, де кожному повідомленню відповідає запис з вмістом повідомлення.

Оскільки планується оновлювати навчальний набір даних кожні 4 тижні, то в якості бажаної кількості повідомлень, котрі мають бути зібрані з джерел було обрано усереднену кількість повідомлень по обраним джерелам, що публікуються впродовж

даного періоду, тобто 400 ($M = 400$) повідомлень. Враховуючи факт, що не всі повідомлення пройдуть фільтрацію по кількості символів, то зчитуватимемо 500 повідомлень з кожного джерела ($N = 500$).

Отримання повідомлень з месенджеру Telegram вирішено виконати за допомогою бібліотеки telethon, що полегшує взаємодію з Telegram API.

Перш ніж працювати з API Telegram, потрібно мати власний обліковий запис в Telegram та отримати ідентифікатор API та хеш. Опишемо порядок кроків для отримання ідентифікатору API та хешу.

Крок 1. Перейти на сайт конфігурації додатків з використанням Telegram API [28].

Крок 2. Увійти у свій обліковий запис Telegram за допомогою номера телефону облікового запису розробника, який потрібно використовувати.

Крок 3. Обрати розділ “Інструменти розробки API”.

Крок 4. Заповнити дані у новоствореній формі.

Крок 5. Натиснути розділ “Створити додаток”.

Крок 6. Після цього в новому вікні з’являться деталі про новостворений додаток, що містять в собі інформацію ідентифікатор API та хеш.

Ідентифікатор API та хеш потрібні для конфігурації клієнту, що взаємодіє з Telegram API. Під час першого створення клієнт запрошує код підтвердження, що надсилається за номером телефону, до якого прив’язаний обліковий запис розробника. Після введення коду підтвердження клієнт завершує ініціалізацію та створює файл сесії, використовуючи котрий не має необхідності запитувати код підтвердження при наступних запусках додатку.

Отримання повідомлень з соціальної мережі Twitter вирішено виконати за допомогою бібліотеки snscrape. Дана бібліотека була обрана через те, що вона базується на технології веб-скрапінгу, а отже в ній відсутні такі недоліки, котрі є в бібліотеках, заснованих на взаємодії з Twitter API, як:

— необхідність довгого очікування верифікації запиту на отримання персонального токену доступу;

— обмеження в максимальній кількості результатів пошуку повідомлень по ключових словах чи за заданим користувачем;
— обмеження в кількості корисних даних, які можна вилучити зі зчитаних повідомлень.

Після зчитування повідомлень з джерел, виконується процес попередньої обробки кожного з повідомлень, який складається з видалення надлишкових символів, канонізації слів, заміни емодзі відповідними словесними репрезентаціями, зміни регістру слів та анонімізація зовнішніх посилань в повідомленнях. Даний процес виконується задля покращення розуміння класифікаційною моделлю змісту повідомлень, що подаються на вхід.

Оскільки різні повідомлення з обраних джерел можуть бути написаними різними мовами, було вирішено виконати переклад всіх повідомлень на англійську мову. Це зроблено як задля зменшення розмірів словника класифікаційної моделі, так для і покращення якості класифікації повідомлень, оскільки в основі класифікаційної моделі лежатиме попередньо навчений на текстах англійською мовою трансформер. Варто зазначити, що оскільки проукраїнські джерела повідомлень набагато частіше використовують українську мову в якості мови комунікації, то переклад повідомлень на англійську мову зменшує концентрацію уваги класифікаційної моделі на словах певної мови, і водночас збільшує концентрацію уваги на самому змісті повідомлень.

Порівняння необроблених та оброблених і перекладених повідомлень зображено в таблиці 3.5.

Таблиця 3.5 – Порівняння необроблених та оброблених повідомлень

Необроблене повідомлення	**Оброблене повідомлення**
😔Так виглядає церква у Комишувасі Запорізького району після нічного обстрілу.	(pensive face) This is what the church in Komyshuvas, Zaporizhzhia district

[__Фото](https://t.me/gu_dsns_zp/5788)____: ДСНС__	looks like after the night shelling. Photo: DSNS
We celebrate Easter with faith in irreversibility of this victory. Heaven sees our faith & firmness. The world sees valor & invincibility. The enemy sees our strength & determination. UA will see the light of victory. Christ is Risen! https://t.co/FFOOfz4hqn	We celebrate Easter with faith in irreversibility of this victory. Heaven sees our faith & firmness. The world sees valor & invincibility. The enemy sees our strength & determination. (Ukraine) will see the light of victory. Christ is Risen! [link]

3.4 Збільшення кількості даних

Після отримання потрібної кількості повідомлень з джерел було застосовано такі методи збільшення кількості даних як зворотний переклад, перемішування речень та синонімічна заміна.

Метод зворотного перекладу передбачає взяття вихідного речення або фрази мовою оригіналу, переклад його цільовою мовою, а потім переклад його назад на вихідну мову. Отримане речення може бути не ідентичним оригінальному вихідному реченню, але все одно може бути дійсним прикладом навчання.

Приклад створення нового речення методом зворотного перекладу зображено на рисунку 3.7.

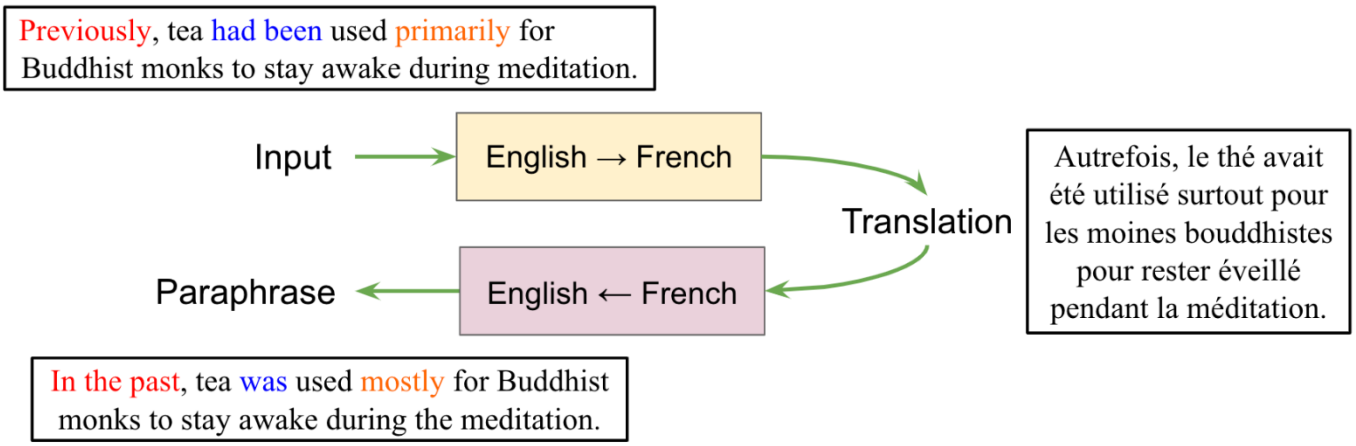

Рисунок 3.7 – Створення нового речення методом зворотного перекладу

Метод синонімічної заміни має схожу мету з подвійним перекладом, але в цій техніці певний відсоток слів у тексті замінюється їхніми синонімами відповідно до словника синонімів. Приклад створення нового речення методом синонімічної заміни зображено на рисунку 3.7.

	Sentence
Original	The quick brown fox jumps over the lazy dog
Synonym (PPDB)	The quick brown fox climbs over the lazy dog

Рисунок 3.8 – Створення нового речення методом синонімічної заміни

Метод перемішування речень передбачає випадкове перемішування речень (або певного їх відсотка) у тексті. Окрім функції збільшення даних цей метод також допомагає запобігти надмірному зверненню уваги класифікаційної моделі до певного порядку речень в тексті.

Приклад створення нового тексту методом перемішування речень зображено на рисунку 3.9.

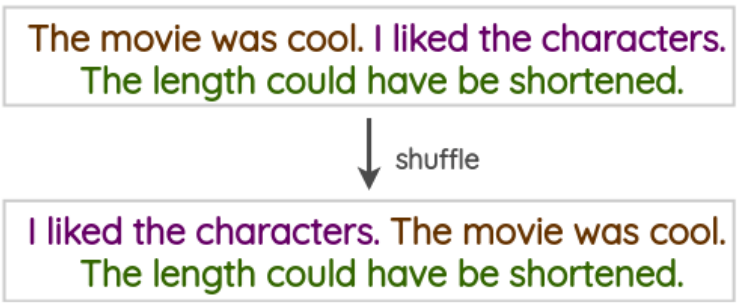

Рисунок 3.9 – Створення нового тексту методом перемішування речень

Кожен з методів збільшення даних було застосовано до трьох підмножин набору даних рівної потужності. Результатом процесу збільшення даних став новий набір з кількістю повідомлень в діапазоні 800 по кожному з джерел.

3.5 Розподіл даних по категоріям

Структура сформованого навчального набору даних у формі ієрархічно згрупованих файлів та директорій зображено на рисунку 3.10.

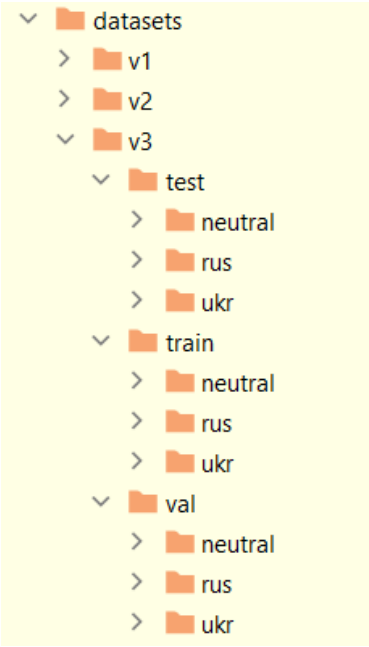

Рисунок 3.10 – Структура сформованого навчального набору даних

Для навчання класифікаційної моделі навчальний набір даних має бути розділений на три категорії – тренувальні дані, перевірочні дані та тестові дані. Стандартним розподілом даних по кожній з категорій є розподіл у співвідношенні 60:20:20. Відповідно, сформований навчальний набір складається з 3 категорій, де кожна з категорій містить дані, що належать до трьох класів, котрі повинна бути здатна класифікувати нейронна мережа. Загальна величина навчального набору є рівною 32000.

Лістинг коду формування навчального набору даних надано в додатку А.

Висновки до розділу

В даному розділі розглянуто процес формування навчального набору даних. Оскільки класифікаційна модель, що навчатиметься на сформованому наборі виконуватиме задачу класифікації політичної забарвленості повідомлень, то першим кроком стало дослідження особливостей проросійської та проукраїнської риторики. Далі було обрано джерела повідомлень, з котрих виконуватиметься збір даних для формування навчального набору. Після цього було виконано збір з обраних джерел за допомогою використання написаних скриптів та бібліотек на мові Python. Останніми кроками стали збільшення кількості зібраних даних, шляхом використання існуючих методів з області NLP, розподіл даних по категоріях за обраним співвідношенням і запис даних на файлову систему.

4 РОЗРОБКА КЛАСИФІКАЦІЙНОЇ МОДЕЛІ

4.1 Структура класифікаційної моделі

Оскільки зібраний набір навчальних даних є досить великим, відсутні обмеження на кількість обчислювальних ресурсів, а також враховуючи той факт, що пропаганда зазвичай містить складні мовленнєві конструкції, то, керуючись порівнянням різних підходів до класифікації тексту, описаних в розділі 1.7, приходимо до висновку що найкращим рішенням для даної задачі є використання нейронних мереж в якості методу класифікації.

На сьогоднішній день, модель BERT є однією з найкращих відкритих моделей для застосування її в задачах класифікації. Використання моделі BERT для вирішення задачі класифікації політичної полярності джерел повідомлень є оптимальним, оскільки пропагандистські матеріали зазвичай містять складні конструкції речень, іншомовні вирази та інші ознаки, які може бути важко виявити та обробити більш простим моделям.

Перед подачею до моделі BERT, вхідні текстові дані потрібно перетворити на числові ідентифікатори токенів і впорядкувати в декілька тензорів. Для цього потрібно мати шар попередньої обробки даних, котрий виконуватиме дану задачу та буде з’єднаний із входами моделі BERT. Оскільки модель BERT повертає послідовність контекстуалізованих вкладень для вхідних токенів, то для виконання задачі класифікації, виходи даної моделі мають бути з’єднані з шаром класифікації, який в свою чергу повертає ймовірнісний розподіл тексту, поданого на вхід класифікаційної моделі, над множиною можливих класів. Між моделлю BERT та вихідним шаром додано дропаут шар, котрий зменшує ймовірність перенавчання класифікаційної моделі на тренувальних даних, шляхом виділення підмережі, ваги котрої повинні оновлюватись, на кожному з етапів навчання.

Структура класифікаційної моделі зображена на рисунку 4.1.

Рисунок 4.1 – Структура класифікаційної моделі

4.2 Деталі реалізації класифікаційної моделі

В якості ядра для класифікаційної моделі було обрано модель BERT з 4 шарами кодування, 8 центрами уваги та розміром вхідного вектору рівному 512 ($L = 4, H = 512, A = 8$).

Класифікаційну модель розроблено з використанням бібліотек TensorFlow та Keras. Шари кодування та попередньої обробки даних не реалізовувались безпосередньо. Замість цього було використано існуючі у відкритому доступі реалізації даних шарів. Для завантаження бажаних шарів використовувалось сховище навчених моделей машинного навчання TensorFlow Hub. В якості шарів кодування, зі

сховища Tensorflow Hub була обрана модель “Smaller BERT” [29], котра попередньо-навчена на текстових збірках Wikipedia Corpus та Books Corpus. В якості шару попередньої обробки даних, зі сховища Tensorflow Hub був обраний шар “BERT English Uncased Preprocess” [30].

В якості відсотку дропауту в дропаут шарі класифікаційної моделі було обрано значення 50%.

Деталі реалізації розробленої моделі зображено на рисунку 4.2.

```
Layer (type)                 Output Shape              Param #   Connected to
==================================================================================================
input_layer (InputLayer)     [(None,)]                 0         []

preprocessing_layer (KerasLaye  {'input_word_ids':   0          ['input_layer[0][0]']
r)                           (None, 128),
                              'input_type_ids':
                             (None, 128),
                              'input_mask': (Non
                             e, 128)}

encoding_layer (KerasLayer)  {'sequence_output':  28763649     ['preprocessing_layer[0][0]',
                              (None, 128, 512),                  'preprocessing_layer[0][1]',
                              'pooled_output': (                 'preprocessing_layer[0][2]']
                             None, 512),
                              'encoder_outputs':
                              [(None, 128, 512),
                              (None, 128, 512),
                              (None, 128, 512),
                              (None, 128, 512)],
                              'default': (None,
                             512)}

dropout_layer (Dropout)      (None, 512)               0         ['encoding_layer[0][5]']

classification_layer (Dense) (None, 3)                 1539      ['dropout_layer[0][0]']

==================================================================================================
Total params: 28,765,188
Trainable params: 28,765,187
Non-trainable params: 1
__________________________________________________________________________________________________
```

Рисунок 4.2 – Деталі реалізації розробленої класифікаційної моделі

Лістинг коду побудови та тренування класифікаційної моделі надано в додатку Б.

4.3 Тренування класифікаційної моделі

Після побудови класифікаційної моделі, виконаємо її тренування на навчальному наборі даних.

Тренування класифікаційної моделі проводитимемо з використанням технології пакетного навчання, при якому модель навчається на кількох зразках даних одночасно, а не оновлює свої параметри на одному зразку даних. Під час пакетного навчання навчальний набір ділиться на невеликі частини, котрі називаються пакетами, і параметри моделі оновлюються після обробки кожного пакету. Такий підхід дозволяє навчити модель більш ефективно, з використанням меншої кількості пам'яті та швидшою конвергенцією.

Для оптимізації рівня навчання було обрано алгоритм AdamW, який спеціально розроблено для навчання глибоких нейронних мереж [31]. Цей алгоритм використовує методи адаптивного навчання, які знаходять індивідуальні показники навчання для кожного параметра нейронної мережі. Основною відмінністю між алгоритмами AdamW і Adam є різний підхід до ослаблення ваг, що допомагає запобігти перенавчанню моделі. В алгоритмі Adam ослаблення ваг зазвичай застосовується після кожного оновлення ваг [32]. Однак у AdamW ослаблення ваг застосовується безпосередньо до градієнтів під час кроку оптимізації, перед оновленням. Ця модифікація допомагає ефективніше регулювати ваги.

В якості функції втрат була використана категоріальна перехресна ентропія. Перехресні ентропійні витрати описують ефективність моделі класифікації, яка повертає ймовірності у діапазоні від 0 до 1. Якщо прогнозована ймовірність відрізняється від фактичної мітки, то значення цих втрат збільшується.

Перехресні ентропійні втрати L визначається за наступною формулою:

$$\mathrm{L} = -\sum_{c=1}^{N} y_{o,c} \ln(p_{o,c}) \tag{4.1}$$

де N – кількість класів;

y – бінарний індикатор, що вказує чи мітка c являє собою правильну класифікацію для спостереження o;

$p_{o,c}$ – ймовірність того, що спостереження o належить класу c, передбачена класифікаційною моделлю.

В якості метрик, котрі відслідковуються були обрані точність, влучність та повнота.

Точність описує кількість правильно класифікованих прикладів.

Влучність P обчислюється як відношення кількості правильно класифікованих позитивних прикладів до загальної кількості прикладів, що були класифіковані як позитивні, тобто визначається формулою:

$$P = \frac{TP}{TP + FP} \tag{4.2}$$

де TP – кількість правильно класифікованих позитивних прикладів;

FP – кількість неправильно класифікованих позитивних прикладів.

Чим більше значення влучності, тим краще модель розпізнає позитивні приклади.

Повнота R визначається як відношення кількості правильно виявлених позитивних прикладів до загальної кількості позитивних прикладів в тестовому наборі даних, тобто за наступною формулою:

$$P = \frac{TP}{TP + FN} \quad (4.3)$$

де TP – кількість правильно класифікованих позитивних прикладів;

FN – кількість неправильно класифікованих негативних прикладів.

Метрика повноти особливо важлива в тих випадках, коли помилка виявлення позитивного прикладу може мати серйозні наслідки.

Виконаємо тренування класифікаційної моделі на навчальному наборі даних впродовж 5 епох. Значення метрик точності, влучності та повноти та значення перехресних ентропійних втрат на кожній з епох описано в таблиці 4.1.

Таблиця 4.1 – Значення метрик та втрат на кожній з епох тренування моделі

Епоха	**Точність**	**Влучність**	**Повнота**	**Втрати**
Тренувальний набір				
1	0.6972	0.7233	0.6660	0.7114
2	0.9124	0.9162	0.9084	0.2383
3	0.9457	0.9475	0.9438	0.1580
4	0.9597	0.9611	0.9586	0.1216
5	0.9667	0.9677	0.9658	0.1043
Перевірочний набір				
1	0.7544	0.7597	0.7481	0.7510
2	0.7961	0.8006	0.7931	0.6118
3	0.8267	0.8297	0.8253	0.5961
4	0.8633	0.8652	0.8614	0.5309
5	0.8667	0.8694	0.8653	0.5300

Бачимо, що різниця між метриками та перехресними ентропійними втратами для перевірочного набору зменшується зі зростанням кількості епох. Так, точність

класифікації для 5-ї епохи є лише на 0.0034 більшою ніж для 4-ї епохи, а значення перехресних ентропійних втрат зменшилось на 0.0009. Приходимо до висновку, що немає необхідності в збільшені кількості тренувальних епох, оскільки вклад додаткових епох в покращення характеристик класифікаційної моделі є досить малим після п’яти епох.

Запуск моделі на тестовому наборі даних показав значення 0.8132 для метрики точності, 0.8153 для метрики влучності та 0.8142 для метрики повноти.

Висновки до розділу

В даному розділі було виконано розробку та тренування класифікаційної моделі. За основу класифікаційної моделі було взято модель BERT, що являє собою попередньо навчену модель текстового трансформера. Далі було описано архітектуру класифікаційної моделі, виконано програмну реалізацію даної моделі з використанням скриптів та бібліотек на мові Python, описано такі деталі реалізованої моделі, як кількість параметрів, зв’язок шарів між собою та розміри кожного з шарів. Наступним етапом стало тренування розробленої класифікаційної моделі. Було обрано метод оптимізації та функцію втрат, що використовуватимуться під час тренування, а також описано раціональність їх використання. Також було обрано метрики, що вказують на якість тренування класифікаційної моделі. Такими метриками стали точність, влучність, повнота та втрати.

Запуск моделі на тренувальному наборі даних показав значення 0.9667 для метрики точності, 0.9677 для метрики влучності та 0.9658 для метрики повноти. Водночас на перевірочному та тестовому наборах даних ці значення є 0.8667 та 0.8132 для метрики точності, 0.8694 та 0.8153 для метрики влучності і 0.8653 та 0.8142 для метрики повноти, що є задовільними показниками для розроблюваної системи.

5 ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАСТОСУНКУ

5.1 Визначення акторів та сценаріїв

Відповідно до змістовної постановки задач до розроблюваної системи, описаної в розділі 2.1, можна виділити єдиного актора, що взаємодіє з системою – звичайного користувача (далі – користувач), оскільки поставлені до системи задачі не потребують різного роду ролей чи динамічної конфігурації параметрів роботи системи адміністраторами.

Можна виділити 4 основні сценарії взаємодії користувача з системою:

— класифікація джерела повідомлення;

— побудова підмережі найпопулярніших хештегів;

— побудова підмережі популярних телеграм-каналів;

— отримання корисних статистичних даних про джерело повідомлень.

Опишемо сценарій класифікації джерела повідомлення.

Крок 1. Користувач вводить посилання на джерело повідомлення в текстову форму графічного інтерфейсу та вибирає в меню тип джерела повідомлення (телеграм-канал чи обліковий запис Twitter).

Крок 2. Користувач натискає кнопку “Submit” для підтвердження запиту на обробку.

Крок 3. Клієнтська частина надсилає запит до серверної частини на отримання розподілу джерела повідомлень над кожним класом.

Крок 4. Серверна частина виконує класифікацію джерела повідомленням відповідно до алгоритму описаного в розділі 2.2 з використанням останньої збереженої на файловій системі версії класифікаційної моделі.

Крок 5. Графічний інтерфейс відображає користувачу отриманий результат з серверної частини у формі таблиці розподілу джерела повідомлень над кожним класом.

Опишемо сценарій побудови підмережі найпопулярніших хештегів.

Крок 1. Користувач вводить параметри, на основі котрих має бути побудована підмережа з існуючої мережі, в форму графічного інтерфейсу. Дана форма має містити наступні поля:

— числові поля “Minimum Vertex Tweets Count” та “Maximum Vertex Tweets Count”, що визначають мінімальну та максимальну кількість повідомлень, в котрих має зустрічатись хештег, щоб з нього була сформована вершина підмережі;
— числові поля “Minimum Edge Tweets Count” та “Maximum Edge Tweets Count”, що визначають мінімальну та максимальну кількість повідомлень, в котрих мають попарно зустрічатись хештеги, щоб з них було сформоване ребро підмережі;
— числове поле “Figure Size”, що визначає величину канви, на котрій буде відображатись підмережа;
— числове поле “Scale”, що визначає коефіцієнт стиснення чи розтягнення побудованого зображення підмережі;
— прапорець “Show Short Node Labels?”, що визначає чи потрібно вказувати над вершинами підмережі лише інформацію про назву хештегу, чи й інформацію про кількість повідомлень, в котрих даний хештег зустрічався;
— прапорець “Show Edge Labels?”, що визначає чи потрібно вказувати над ребрами підмережі інформацію про кількість повідомлень, в котрих попарно зустрічаються хештеги, що відповідають вершинам на кінцях ребра.

Крок 2. Користувач натискає кнопку “Submit” для підтвердження запиту на обробку.

Крок 3. Клієнтська частина надсилає запит до серверної частини на отримання зображення побудованої підмережі хештегів.

Крок 4. Серверна частина виконує побудову підмережі хештегів використовуючи останню збережену на файловій системі версію мережі хештегів відповідно до алгоритму описаного в розділі 2.5.

Крок 5. Графічний інтерфейс відображає користувачу отримане з серверної частини зображення підмережі хештегів.

Опишемо сценарій побудови підмережі популярних телеграм-каналів.

Крок 1. Користувач вводить параметри в форму графічного інтерфейсу, на основі котрих має бути побудована підмережа з існуючої мережі. Дана форма має містити наступні поля:

- числові поля “Minimum Vertex Value” та “Maximum Vertex Value”, що визначають мінімальну та максимальну кількість підписників на каналі, щоб з нього була сформована вершина підмережі;
- числові поля “Minimum Edge Value” та “Maximum Edge Value”, що визначають мінімальну та максимальну кількість повідомлень, в котрих зустрічається посилання на канал, щоб з них було сформоване ребро підмережі;
- числове поле “Scale”, що визначає коефіцієнт стиснення чи розтягнення побудованого зображення підмережі;
- числове поле “Figure Size”, що визначає величину канви, на котрій буде відображатись підмережа;
- прапорець “Show Short Node Labels?”, що визначає чи потрібно вказувати над вершинами підмережі лише інформацію про ідентифікатор каналу, чи й інформацію про кількість підписників на даний канал;
- прапорець “Show Edge Labels?”, що визначає чи потрібно вказувати над ребрами підмережі інформацію про кількість повідомлень, в котрих зустрічаються канали, що відповідають вершинам на кінцях ребра.

Крок 2. Користувач натискає кнопку “Submit” для підтвердження запиту на обробку.

Крок 3. Клієнтська частина надсилає запит до серверної частини на отримання зображення побудованої підмережі каналів.

Крок 4. Серверна частина виконує побудову підмережі каналів використовуючи останню збережену на файловій системі версію мережі каналів відповідно до алгоритму описаного в розділі 2.6.

Крок 5. Графічний інтерфейс відображає користувачу отримане з серверної частини зображення підмережі каналів.

Опишемо сценарій отримання корисних статистичних даних про джерело повідомлень.

Крок 1. Клієнтська частина надсилає запит до серверної частини на отримання статистичних даних про кожне джерело повідомлень, для котрих такі дані збираються. Такими даними є кількість підписників, кількість нових повідомлень з моменту останнього збору статистики, усереднена кількість реакцій на повідомлення за типом реакції.

Крок 2. Серверна частина отримує статистичні дані про джерела повідомлень з БД та повертає їх клієнту.

Крок 3. Графічний інтерфейс відображає статистичні дані про джерело повідомлень у формі гістограми. Також графічний інтерфейс містить меню вибору типу джерела повідомлень, меню вибору джерела повідомлень за обраним типом, а також меню вибору статистичних даних, котрі потрібно відобразити на гістограмі. Для кожного типу статистичних даних гістограма представляє собою зміну певної числової характеристики (або пари характеристик) в часі.

Опис сценаріїв використання розроблюваної системи представлено на плакаті 1 в додатку В.

5.2 Визначення технічних вимог до системи

Розроблювана система має виконувати наступні технічні вимоги:

— система повинна мати можливість гнучкого масштабування;
— система повинна мати високу доступність;
— система має надавати відкрите та інтуїтивно зрозуміле API, до якого можуть звертатись зовнішні системи;
— рівні представлення, бізнес-логіки та зберігання даних повинні бути розділені між собою в рамках окремих сервісів;
— система повинна відповідати сучасним стандартам розробки;
— система має легко розширюватись, без необхідності зміни великої частини коду при додаванні нового функціоналу;
— сервер бази даних повинен відповідати властивостям узгодженості та доступності в контексті теореми CAP, тобто являти реляційну СКБД.

5.3 Загальна архітектура системи

Сформувавши технічні вимоги до системи, та визначивши які сценарії вона повинна підтримувати, перейдемо до проектування загальної архітектури системи.

Оскільки для реалізації всіх сценаріїв не потрібно великої кількості сервісів то найкраще зійде триярусна архітектура, де система складається з тонкого клієнта, сервера застосунків та серверу бази даних. Загальна структура триярусної архітектури зображена на рисунку 5.1.

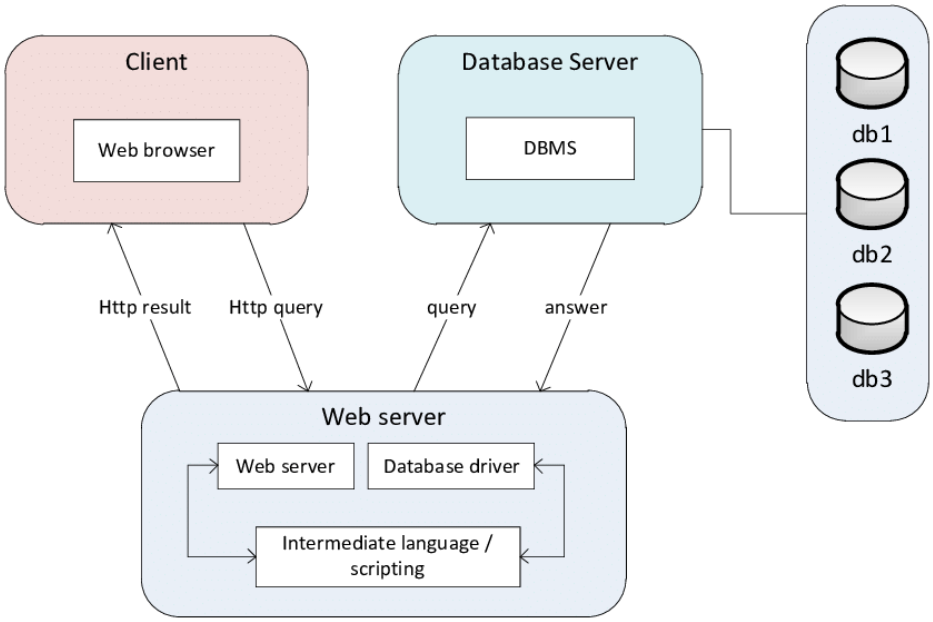

Рисунок 5.1 – Загальна структура триярусної архітектури

Загальна структура розроблюваної системи зображена на плакаті 2 в додатку В.

5.3.1 Загальна структура клієнту

Структурно клієнт розроблюваної системи має являти собою веб-сторінку або набір веб-сторінок, що містить ряд панелей, кожна з яких слугує або суто для відображення певних даних, або для взаємодії з користувачем шляхом отримання від користувача даних через форму, опрацювання переданих даних (на стороні самого клієнта або за допомогою спрямування запиту до серверу застосунків) та відображення результату обробки запиту. Оскільки деякі компоненти веб-сторінок можуть бути використані знову під час додавання нового функціоналу до застосунку то підходящим є використання парадигми компонентно-орієнтованого програмування під час реалізації клієнту. В рамках даної парадигми програмне забезпечення має розбиватись на компоненти, котрі мають ряд загальних характеристик:

— компоненти призначені для підключення до різноманітних додатків без необхідності модифікації чи спеціальних пристосувань;

— компоненти мають можливість комбінування з іншими компонентами задля створення нової поведінки;
— компоненти зі схожими функціями можна міняти місцями;
— компоненти є самодостатніми та розкривають свій функціонал через інтерфейси, приховуючи деталі внутрішніх процесів;
— компоненти повинні мати мінімальну залежність від інших компонентів і могти працювати в різних середовищах і контекстах.

5.3.2 Загальна структура серверу застосунків

Сервер застосунків слугує для отримання запитів з клієнта, їх обробки, за необхідності – звернення до бази даних, файлової системи чи зовнішніх систем, та повернення результату опрацювання запиту клієнту.

Структурно сервер застосунків розроблюваної системи має складатись з рівня доступу до даних, рівня обробки запитів, рівня роботи з класифікаційною моделлю, рівня сервісів, рівня роботи з соціальними мережами та месенджерами, рівня обробки повідомлень та рівня роботи з базою даних.

Рівень доступу до даних виконує перевірку чи може запит, отриманий від певного зовнішнього сервісу, бути опрацьованим. Для цього виконується перевірка таких заголовків запиту, як адрес сервісу (при підтримці механізму CORS) та деталі аутентифікації, за необхідності.

Рівень обробки запитів являє собою набір контролерів, котрі приймають запити, перевіряють джерело запиту, правильність даних що приходять і передають запит до рівня сервісів.

Рівень сервісів виконує опрацювання отриманих запитів та за потреби звертається до інших рівнів. Він містить логіку, пов'язану з обробкою запитів від клієнта (включаючи перевірку даних та опрацювання результатів) та з плануванням запуску періодичних процесів.

Рівень роботи з класифікаційною моделлю використовується для класифікації джерел повідомлень, чи окремих повідомлень за допомогою звернення до попередньо навченої класифікаційної моделі. Також даний рівень займається зберіганням та завантаженням нової версії класифікаційної моделі за необхідністю.

Рівень роботи з соціальними мережами та месенджерами відповідає за отримання повідомлень з різних джерел, таких як облікові записи в соціальних мережах або канали в месенджерах.

Рівень обробки повідомлень виконує попередню обробку та переклад зчитаних з зовнішніх ресурсів повідомлень, задля полегшення опрацювання даних повідомлень класифікаційною моделлю та покращення показників точності класифікації.

Рівень роботи з базою даних відповідає за збереження та отримання даних, що використовуються системою, таких як глобальні налаштування, список джерел, повідомлення від яких використовуються для оновлення навчального набору даних або ж для збору корисної статистики тощо. Цей рівень забезпечує довгострокове зберігання даних та забезпечує їх доступність для інших рівнів системи.

Всі рівні сервера застосунків розроблюваної системи співпрацюють між собою, щоб забезпечити коректну та швидку обробку запитів та підтримку всіх необхідних функціональних вимог системи.

5.4 Формат комунікацій між сервісами

Оберемо формат комунікації клієнту з сервером. Оскільки функціонал системи з часом може розширюватись, а також враховуючи той факт, що розроблювана система не потребує підвищеного рівня безпеки під час комунікації компонентів, що її реалізують, то найбільш оптимальним вибором є комунікація на основі підходу до архітектури мережевих протоколів REST. Комунікація за допомогою підходу REST передбачає виконання HTTP запитів до веб-ресурсів, які ідентифікуються унікальною URL-адресою. В такому підході клієнт надсилає HTTP запит на сервер, який повинен

бути виконаний. Запит містить метод HTTP (GET, POST, PUT, DELETE тощо), URL-адресу веб-ресурсу та необов'язкові параметри запиту. Сервер в свою чергу відповідає на запит, надсилаючи HTTP відповідь, яка містить код статусу обробки запиту, заголовки відповіді та необов'язкове тіло відповіді, яке містить ресурс або інші дані.

Визначимо API, котре має надавати сервер застосунків для реалізації всіх вищеописаних сценаріїв взаємодії користувача з системою. Орієнтуючись на деталі самих сценаріїв приходимо до висновку, що достатньо виділити чотири кінцеві точки.

Загальні відомості по кожній з кінцевих точок API описано в таблиці 5.1.

Таблиця 5.1 – Загальні відомості про кінцеві точки API

Метод HTTP	**URL**	**Тип запиту**	**Тип відповіді**
Класифікація джерела повідомлення			
POST	*/message-sources/classify	application/json	application/json
Побудова підмережі найпопулярніших хештегів			
POST	*/graphs/hashtag-subgraph	application/json	plain/text
Побудова підмережі популярних телеграм-каналів			
POST	*/graphs/channel-subgraph	application/json	plain/text
Отримання корисних статистичних даних про джерела повідомлень			
POST	*/message-sources/stats	application/json	application/json

Деталі запитів до кінцевих точок API описано в таблиці 5.2.

Таблиця 5.2 – Деталі запитів до кінцевих точок API

Структура тіла запиту	**Деталі про тіло запиту**
Класифікація джерела повідомлення	
{ "sourceType": "string",	Опис полів:

"sourceId": "string" }	а) sourceType (обов'язкове) – визначає тип джерела повідомлення; б) sourceId (обов'язкове) – визначає ідентифікатор джерела повідомлення.
Побудова підмережі найпопулярніших хештегів	
{ "minVertexTweetsCount": "int", "maxVertexTweetsCount": "int", "minEdgeTweetsCount": "int", "maxEdgeTweetsCount": "int", "minVertexScore": ["float"], "maxVertexScore": ["float"], "minEdgeScore": ["float"], "maxEdgeScore": ["float"], "figureSize": "int", "scale": "float", "showShortNodeLabels": "bool",	Опис полів: а) minVertexTweetsCount, maxVertexTweetsCount (не обов'язкові) – визначають мінімальну та максимальну кількість повідомлень, в котрих має зустрічатись хештег, щоб з нього була сформована вершина підмережі; б) minEdgeTweetsCount, maxEdgeTweetsCount (не обов'язкові) – визначають мінімальну та максимальну кількість повідомлень, в котрих мають попарно зустрічатись хештеги, щоб з них було сформоване ребро підмережі; в) minVertexScore, maxVertexScore (не обов'язкові) – визначають мінімальні та максимальні значення розподілу хештегу, що представляє вершину мережі, над кожним з класів; г) minEdgeScore, maxEdgeScore (не обов'язкові) – визначають мінімальні та максимальні усереднені значення розподілу повідомлень, що представляють ребра мережі, над кожним з класів;

"showEdgeLabels": "bool" }	д) figureSize (не обов'язкове) – визначає величину канви, на котрій буде відображатись підмережа; е) scale (не обов'язкове) – визначає коефіцієнт стиснення чи розтягнення побудованого зображення підмережі; ж) showShortNodeLabels (не обов'язкове) – визначає чи потрібно вказувати над вершинами підмережі лише інформацію про назву хештегу, чи й інформацію про кількість повідомлень, в котрих даний хештег зустрічався; з) showEdgeLabels (не обов'язкове) – визначає чи потрібно вказувати над ребрами підмережі інформацію про кількість повідомлень, в котрих попарно зустрічаються хештеги, що відповідають вершинам на кінцях ребра.
Побудова підмережі популярних телеграм-каналів	
{ "minVertexValue": "int", "maxVertexValue": "int", "minEdgeValue": "int", "maxEdgeValue": "int", "minVertexScore": ["float"], "maxVertexScore": [	Опис полів: а) minVertexValue, maxVertexValue (не обов'язкові) – визначають мінімальну та максимальну кількість підписників на каналі, щоб з нього була сформована вершина підмережі; б) minEdgeValue, maxEdgeValue (не обов'язкові) – визначають мінімальну та максимальну кількість повідомлень, в

"float"], "minEdgeScore": ["float"], "maxEdgeScore": ["float"], "figureSize": "int", "scale": "float", "showShortNodeLabels": "bool", "showEdgeLabels": "bool" }	котрих зустрічається посилання на канал, щоб з них було сформоване ребро підмережі; в) minVertexScore, maxVertexScore (не обов'язкові) – визначають мінімальні та максимальні значення розподілу каналу, що представляє вершину мережі, над кожним з класів; г) minEdgeScore, maxEdgeScore (не обов'язкові) – визначають мінімальні та максимальні усереднені значення розподілу повідомлень, що представляють ребра мережі, над кожним з класів; д) figureSize (не обов'язкове) – визначає величину канви, на котрій буде відображатись підмережа; е) scale (не обов'язкове) – визначає коефіцієнт стиснення чи розтягнення побудованого зображення підмережі; ж) showShortNodeLabels (не обов'язкове) – визначає чи потрібно вказувати над вершинами підмережі лише інформацію про ідентифікатор каналу, чи й інформацію про кількість підписників на даний канал; з) showEdgeLabels (не обов'язкове) – визначає чи потрібно вказувати над ребрами підмережі інформацію про кількість повідомлень, в котрих

	зустрічаються канали, що відповідають вершинам на кінцях ребра.
Отримання корисних статистичних даних про джерела повідомлень	

{
"sourceType": "string"
}
	Опис полів:
	а) sourceType (не обов'язкове) – визначає тип джерел повідомлень для яких потрібно повернути статистичні дані.

Деталі відповідей від кінцевих точок API описано в таблиці 5.3.

Таблиця 5.3 – Деталі відповідей від кінцевих точок API

Структура тіла відповіді	**Деталі про тіло відповіді**
Класифікація джерела повідомлення	

{
"string": "float"
}
	Відповіддю є словник, котрий містить інформацію про розподіл джерела повідомлень над кожним класом.

Побудова підмережі найпопулярніших хештегів	
"string"	Відповіддю є текст, котрий являє собою зображення побудованої підмережі найпопулярніших хештегів в кодуванні двійкових даних Base64.
Побудова підмережі популярних телеграм-каналів	
"string"	Відповіддю є текст, котрий являє собою зображення побудованої підмережі популярних телеграм-каналів в кодуванні двійкових даних Base64.
Отримання корисних статистичних даних про джерела повідомлень	

{
"result": [
{
"sourceId": "string",
"sourceType": "string",
"stats": [
{
"messagesCount": "int",
"reactions": {
"string": "float"
},
"recordDate": "string",
"score": {
"string": "float"
}
}
]
}
]
}
	Відповіддю є масив статистичних даних для кожного джерела повідомлень.
	Опис полів:
	а) result – визначає масив статистичних даних;
	б) result.sourceId – визначає ідентифікатор джерела повідомлення;
	в) result.sourceType – визначає тип джерела повідомлення;
	г) result.stats – визначає масив записів з статистичними даними по певному джерелу повідомлень;
	д) result.stats.messagesCount – визначає кількість повідомлень в записі;
	е) result.stats.reactions – визначає словник, що містить інформацію про розподіл кількості реакцій по кожному типу в записі;
	ж) result.stats.recordDate – визначає дату запису;
	з) result.stats.score – визначає словник, котрий містить інформацію про розподіл джерела повідомлень над кожним класом.

5.5 Визначення процесів

Окрім процесів, що відбуваються за ініціації користувача в рамках виконання сценаріїв, описаних в розділі 5.1, сервіс обробки запитів здатен власноруч ініціювати заплановані на конкретну дату, періодичні процеси.

Періодичними процесами є:

— щомісячне оновлення навчального набору даних, відповідно до алгоритму, описаного в розділі 3.3;
— щомісячна перебудова мереж найпопулярніших хештегів та популярних телеграм-каналів, відповідно до алгоритмів, описаних в розділах 2.3 та 2.4;
— збір статистики по заданому списку джерел повідомлень кожні 3 дні, відповідно до алгоритму, описаного в розділі 2.7.

Опис процесів класифікації джерела повідомлення, побудови підмереж заданого типу, а також отримання корисних статистичних даних про джерела повідомлень, надано на плакатах 3, 4, 5 в додатку В, відповідно.

Опис процесів оновлення навчального набору даних, перебудови мереж заданого типу, а також збору статистики по заданому списку джерел повідомлень надано на плакатах 6, 7, 8 в додатку В, відповідно.

Висновки до розділу

В даному розділі було визначено акторів, котрі взаємодіятимуть з застосунком, що реалізується, а також сценаріїв використання, які даний застосунок підтримуватиме. Далі було визначено технічні вимоги до застосунку, серед них – гнучкість до масштабування, висока доступність, модульність реалізації тощо. Відповідно до розроблених сценаріїв використання та технічних вимог було окреслено загальну архітектуру системи, її складових, а також формат комунікації між ними. Останнім етапом став опис процесів, котрі проходять в програмному застосунку, та які необхідні для реалізації сценаріїв використання.

6 ПРОГРАМНА РЕАЛІЗАЦІЯ ЗАСТОСУНКУ

6.1 Вибір мов програмування, бібліотек, фреймворків

Відповідно до обраної архітектури, розроблюваний програмний застосунок структурно складається з трьох компонентів:

— сервісу надання графічного інтерфейсу, який грає роль клієнту;

— сервісу обробки запитів, який грає роль серверу застосунків;

— серверу бази даних.

Сервіс надання графічного інтерфейсу вирішено реалізувати з використанням фреймворку Angular. Angular забезпечує більшу узгодженість між компонентами порівняно з іншими веб-фреймворками, такими як Vue чи React. Крім того, Angular структурно розділяє розміщення розмітки HTML, коду TypeScript та CSS стилів, що спрощує роботу з компонентами [33]. Використання мови TypeScript у фреймворку знижує ризик виникнення помилок, пов'язаних з відсутністю перевірки типів.

За допомогою використання мови програмування Python було реалізовано сервіс обробки запитів, оскільки, на сьогоднішній день, існує безліч готових рішень, написаних на даній мові, що пов’язані з роботою з нейронними мережами, графами, API соціальних мереж, обробкою зображень та іншими завданнями. Список використаних зовнішніх бібліотек та їх призначення описано в таблиці 6.1.

Таблиця 6.1 – Використані зовнішні бібліотеки в сервісі обробки запитів

Назва	**Призначення**
Tensorflow, Keras	Робота з нейронними мережами
Numpy	Робота з багатовимірними масивами
Pillow	Робота з зображеннями
Matplotlib	Побудова графіків

Flask	Створення веб-додатків
APScheduler	Планування запусків методів
Telethon	Робота з Telegram API
Snscrape	Веб-скрапінг соціальних мереж
NetworkX	Побудова графів
Psycopg 2	Робота з базами даних
Googletrans	Переклад тексту
NLPAug	Збільшення кількості текстових даних

6.2 Розробка сервісу надання графічного інтерфейсу

Графічний інтерфейс, з котрим взаємодіє користувач, було вирішено вмістити в межах однієї веб-сторінки, основний зміст якої розділений на чотири компоненти. Дані компоненти слугують для взаємодії користувача з системою в рамках кожного з чотирьох описаних сценаріїв.

Головна веб-сторінка структурно може бути поділена на наступні частини:

— шапка сторінки з назвою застосунку та навігацією;

— основний зміст сторінки;

— нижня частина сторінки з певною додатковою інформацією про застосунок чи розробника.

Діаграма пакетів реалізованого сервісу надання графічного інтерфейсу представлена на плакаті 9 в додатку В.

Опишемо основні файли та директорії реалізованого сервісу надання графічного інтерфейсу.

Точкою входу веб-застосунку є файл src/main.ts. Даний файл завантажує головний модуль Angular та керує запуском програми.

Файл src/index.html є сторінкою, яка після відображення на ній необхідних компонентів надаватиметься користувачеві.

Корінним модулем Angular-застосунку є файл src/app/app.module.ts, який імпортує додаткові модулі, а також створені компоненти, що використовуються застосунком.

Директорія src/app/components вміщує в собі всі компоненти застосунку, котрі відображатимуться на сторінці src/index.html в залежності від налаштувань модуля app/app-routing.module.ts.

Директорія src/app/compontents/home описує компонент головної сторінки веб-застосунку.

Директорії src/app/compontents/header, src/app/compontents/footer та src/app/compontents/home описують компоненти шапки, нижньої частини та основного змісту головної сторінки відповідно.

Директорії src/app/compontents/build-channel-subgraph та src/app/compontents/build-hashtag-subgraph описують компоненти для побудови підмереж популярних телеграм-каналів та найпопулярніших хештегів відповідно.

Директорія src/app/components/classify-message-source описує компоненту для класифікації джерел повідомлень.

Директорія src/app/components/message-sources-statistics описує компоненту для відображення корисних статистичних даних по джерелах повідомлень.

Директорія src/app/services містить набір сервісів, які використовуються компонентами для розділення бізнес-логіки від логіки відображення та взаємодії компонентів. Сервіси можуть реалізовувати різноманітну бізнес-логіку. В реалізованому веб-застосунку вони виконують лише завдання формування викликів до інших сервіси та опрацювання отриманих результатів.

Директорія src/assets зберігає всі статичні елементи, такі як картинки, фони та дрібні графічні елементи.

Директорія src/css зберігає глобальні для веб-застосунку стилі CSS.

Директорія src/environments містить інформацію про ті змінні веб-застосунку, котрі залежать від середовища розгортання застосунку.

У директорії src/app/models знаходяться моделі, які необхідні для застосунку.

Відкритий код сервісу надання графічного інтерфейсу наданий в Github-репозиторії propaganda-fighter-ui [34].

6.3 Розробка сервісу обробки запитів

Діаграма пакетів реалізованого сервісу обробки запитів представлена на плакаті 10 в додатку В.

Опишемо основні файли та директорії реалізованого сервісу обробки запитів.

Точкою входу є файл app.py робить виклик методу ініціалізації веб-серверу, котрий описаний в файлі app/__init__.py та запускає веб-сервер на заданому порті.

Файл config.py описує глобальну конфігурацію веб-серверу, а саме інформацію про назву застосунку, середовище розгортання веб-серверу та про необхідність ініціалізації бази даних.

Файл app/__init__.py ініціалізує веб-сервер, використовуючи зчитані дані конфігурації, створює контекст застосунку та робить імпорт скрипту, описаного в файлі app/routes.py.

Директорія sql містить набір SQL-скриптів для ініціалізації структури бази даних та даних в ній, якщо в цьому є необхідність.

Файл app/routes.py декларує ряд контролерів, котрі опрацьовують виклики кінцевих точок API, описаних в розділі 5.4. Контролери виконують завдання перевірки вхідних даних, виклик необхідних скриптів рівня сервісів для опрацювання запиту, та формування і повернення результату. Також даний файл описує метод ініціалізації змінних, котрі застосовуються контролерами та бази даних. Окрім цього, app/routes.py описує ряд методів, котрі запускаються з заданою періодичністю для виконання оновлення навчального набору даних, графів найпопулярніших хештегів і популярних телеграм-каналів та збору статистики по заданих джерелах повідомлень.

Файл configs/config.ini містить деталі конфігурації клієнту взаємодії з Telegram API та клієнту роботи з базою даних.

Директорія datasets містить множину навчальних наборів даних різних версій, де кожна версія представлена піддиректорією vx (x – номер версії).

Директорія models містить множину класифікаційних моделей різних версій, де кожна версія представлена піддиректорією vx.

Директорія graphs складається з двох піддиректорій – channel та hashtag. Дані директорії містять множину серіалізованих мереж популярних телеграм-каналів та найпопулярніших хештегів різних версій, де кожна версія представлена піддиректорією vx під відповідною директорією.

Директорія scripts містить ряд скриптів, котрі описують більшу частину логіки роботи веб-серверу. В даній директорії описані рівень роботи з класифікаційною моделлю, рівень роботи з базою даних, рівень зчитування та обробки повідомлень та більша частина рівня сервісів.

Відкритий код сервісу обробки запитів наданий в Github-репозиторії propaganda-fighter-be [35].

6.4 Розробка таблиць бази даних

В якості реляційної СКБД було вирішено обрати PostgreSQL. PostgreSQL має високу надійність, потужність, гнучкість і розширюваність, що робить її однією з найкращих для проектів будь-якого розміру.

Структуру бази даних складають 4 таблиці, що містять всю необхідну для роботи застосунку інформацію. ER-діаграма бази даних зображена на рисунку 6.1.

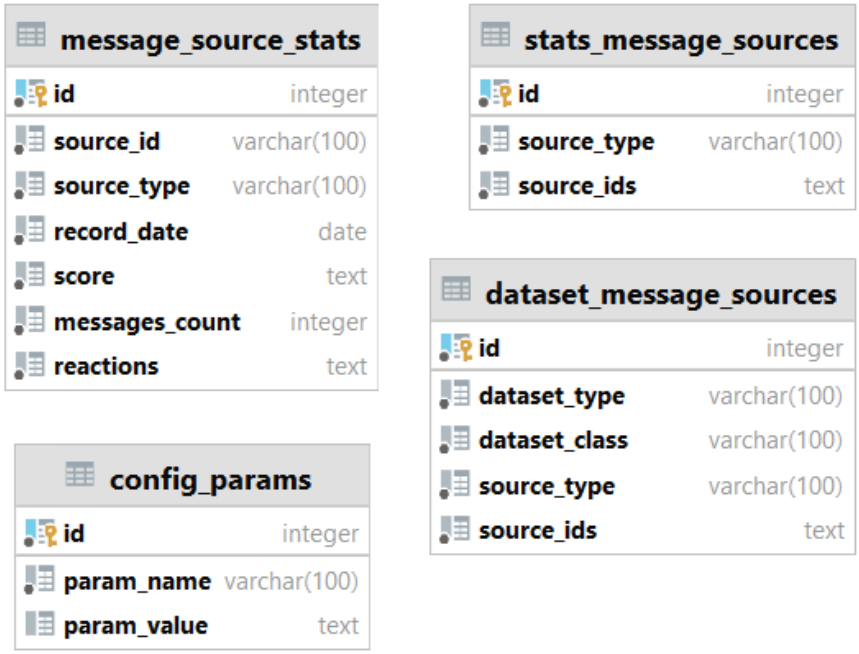

Рисунок 6.1 – ER-діаграма бази даних

Таблиця config_params зберігає список динамічних параметрів, котрі використовуються в роботі застосунку, таких як: поточні версії навчального набору даних, класифікаційної моделі та побудованих мереж; кількість повідомлень котрі потрібно зчитати для формування навчального набору даних та статистичних даних тощо.

Таблиця dataset_message_sources зберігає інформацію про джерела повідомлень, котрі використовуються при формуванні навчального набору даних;

Таблиця stats_message_sources зберігає інформацію про джерела повідомлень, для яких відбувається зчитування корисних статистичних даних;

Таблиця dataset_message_sources зберігає зчитані статистичні дані для кожного з джерел повідомлень.

6.5 Результати роботи застосунку

Головна сторінка реалізованого веб-застосунку зображена на рисунку 6.2. Дана сторінка складається з 4 умовних розділів, кожен з яких відповідає за певний сценарій використання.

Рисунок 6.2 – Головна сторінка веб-застосунку

Розділ “Classify Message Source By Username” відповідає за сценарій класифікації джерел повідомлень. Скріншот класифікації джерела повідомлення зображено на рисунку 6.3.

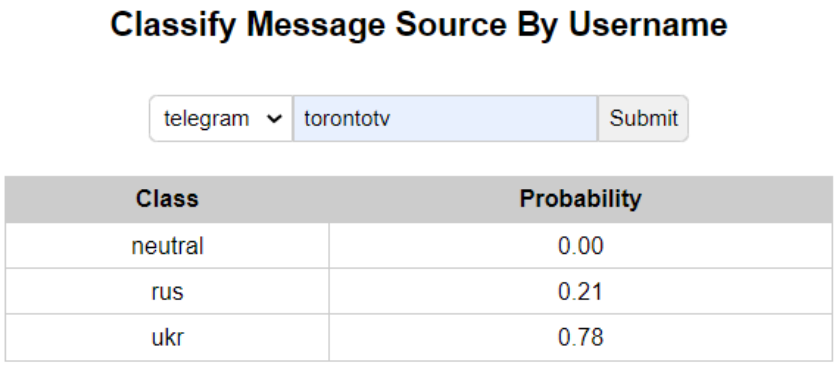

Рисунок 6.3 – Класифікація джерела повідомлення

Розділ “Build Hashtag Subgraph” відповідає за сценарій побудови підмережі найпопулярніших хештегів. Скріншот заповнення форми на побудову такої підмережі та результат її побудови зображено на рисунках 6.4 та 6.5 відповідно.

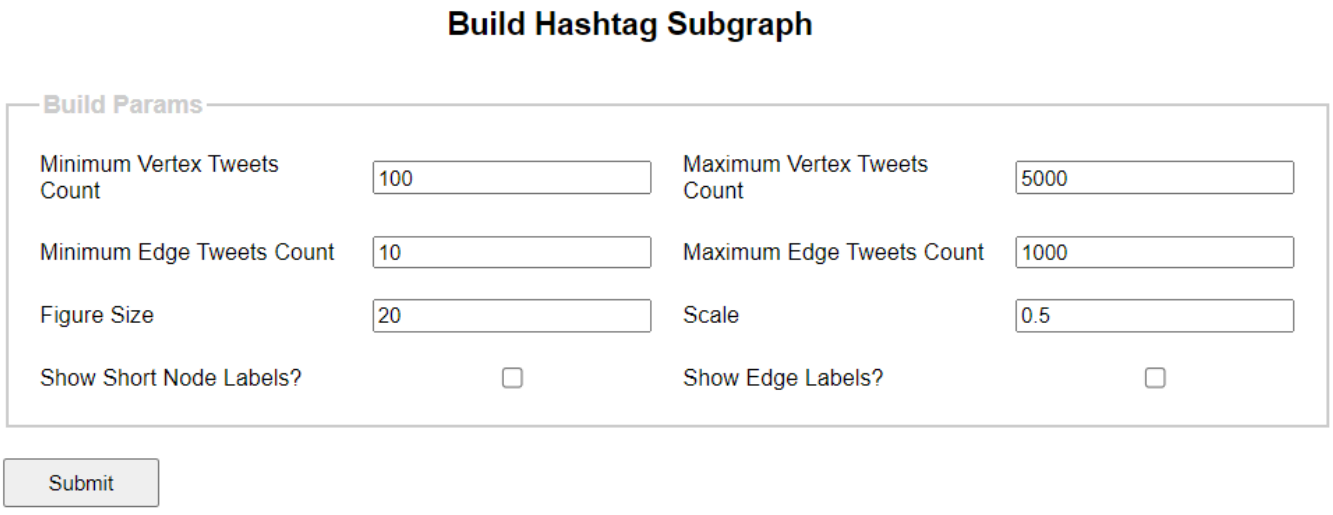

Рисунок 6.4 – Заповнення форми на побудову підмережі найпопулярніших хештегів

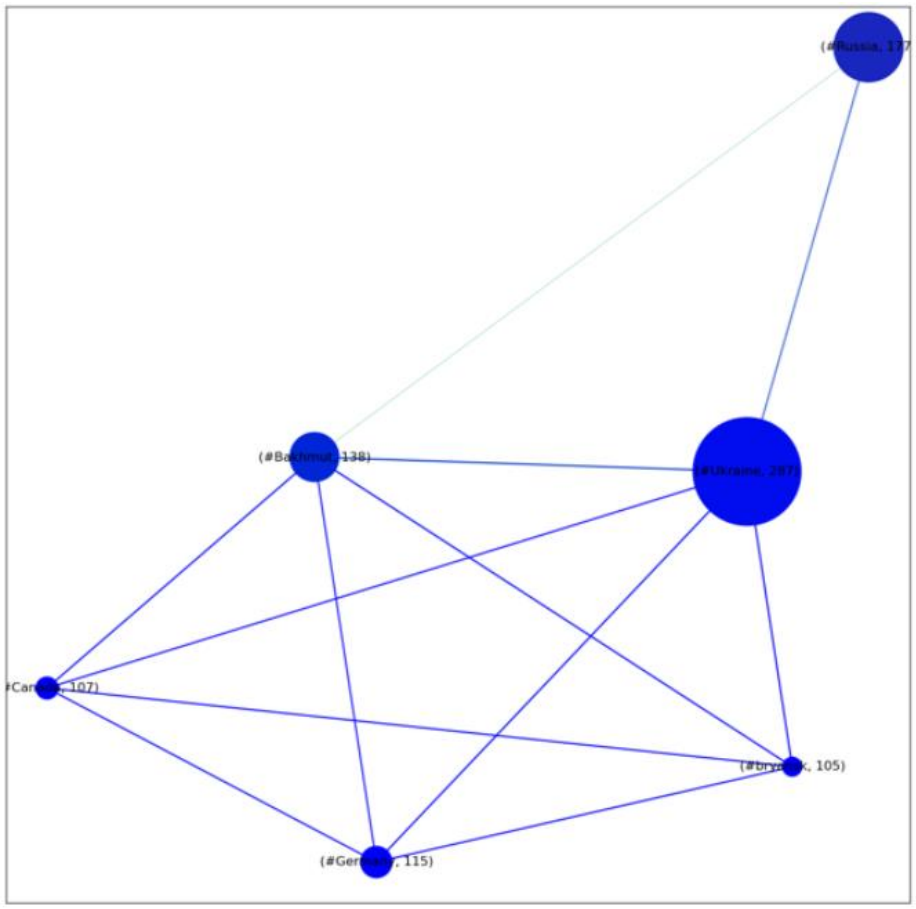

Рисунок 6.5 – Результат побудови підмережі найпопулярніших хештегів

Розділ “Build Channel Subgraph” відповідає за сценарій побудови підмережі популярних телеграм-каналів. Скріншот заповнення форми на побудову такої підмережі та результат її побудови зображено на рисунках 6.6 та 6.7 відповідно.

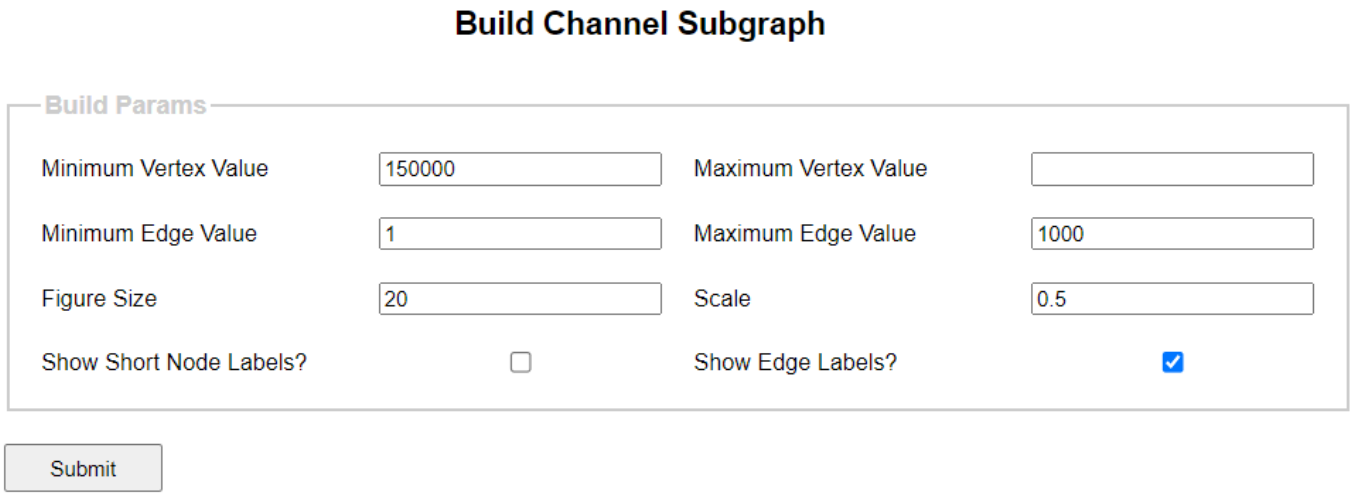

Рисунок 6.6 – Заповнення форми на побудову підмережі популярних телеграм-каналів

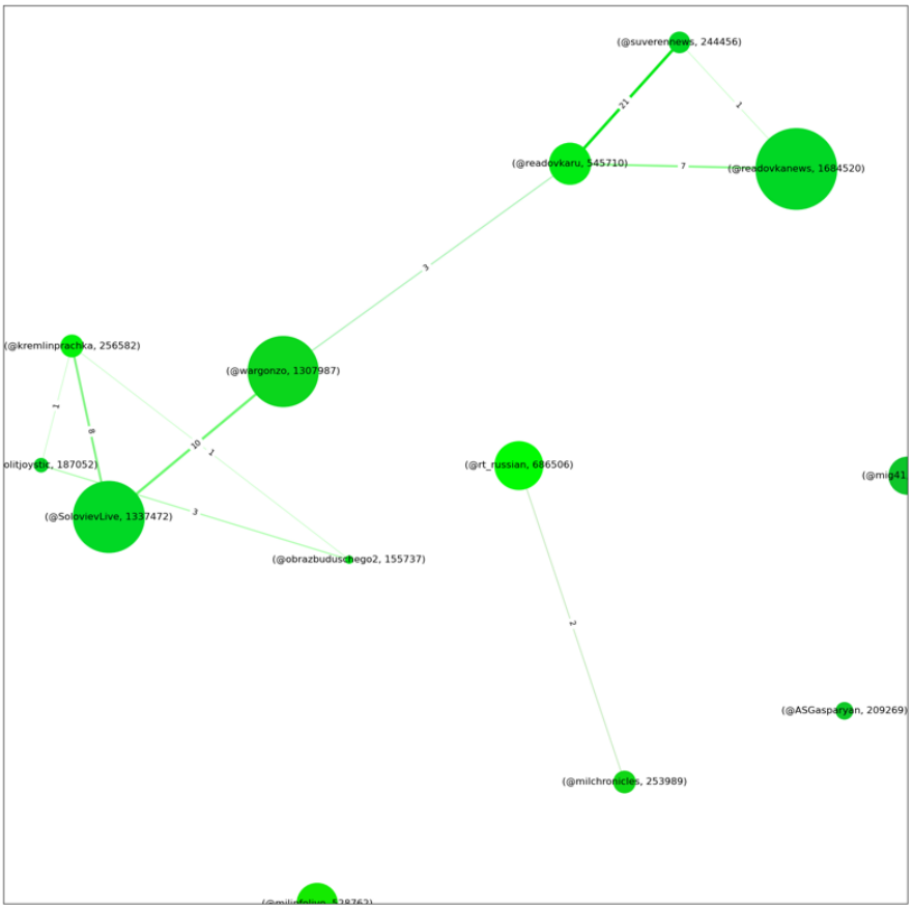

Рисунок 6.7 – Результат побудови підмережі популярних телеграм-каналів

Розділ “Message Sources Statistics” відповідає за сценарій відображення корисної статистики по джерелах повідомлень. Скріншот зі статистикою по обраному джерелу повідомлень зображено на рисунку 6.8.

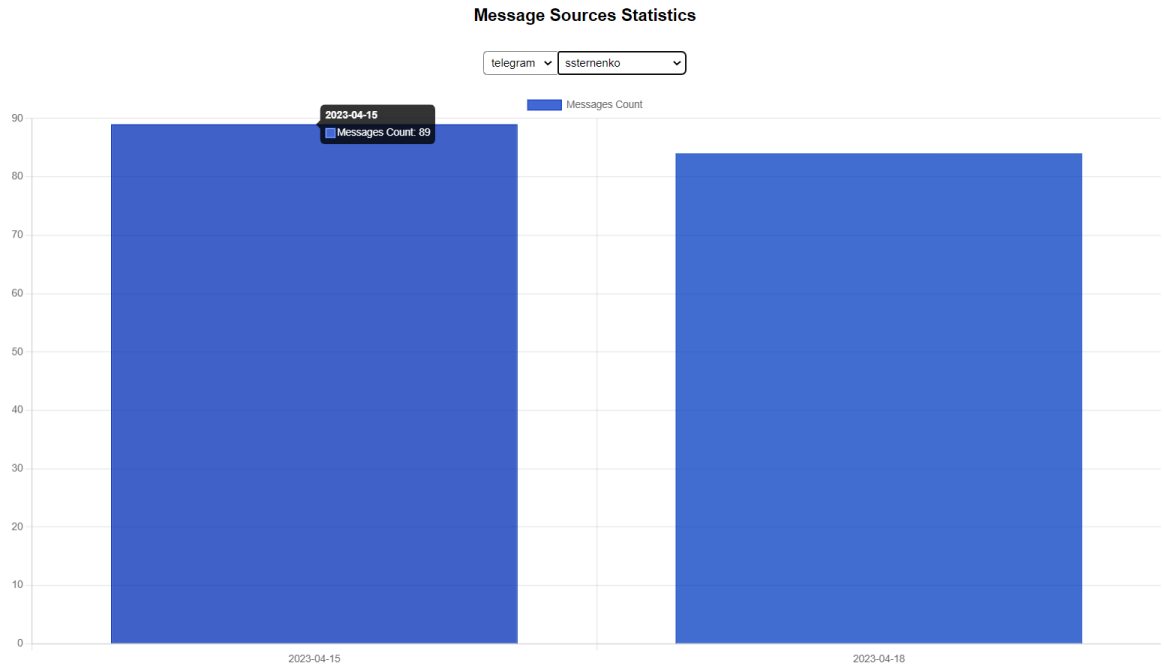

Рисунок 6.8 – Статистика по обраному джерелу повідомлень

Висновки до розділу

В даному розділі виконано програмну реалізацію системи класифікації та аналізу пропаганди. Першим етапом став вибір мов програмування, бібліотек та фреймворків, котрі будуть використанні для розробки застосунку. Для розробки сервісу надання графічного інтерфейсу користувача була обрана мова Angular, а для сервісу обробки запитів – мова Python. В якості СКБД був обраний PostgreSQL. По реалізації всіх сервісів та розробки необхідних таблиць, було надано результати роботи застосунку.

ВИСНОВКИ

В результаті дослідження було розроблено засоби та алгоритми для виявлення та аналізу пропаганди в текстах з використанням методів машинного навчання та статистичної обробки даних.

У розділі 1 проведено огляд методів та засобів розв’язання задачі розпізнавання та аналітики пропаганди в текстах. Поставлено задачу виявлення пропаганди та запропоновано загальну схему її розв’язання. Проаналізовано питання вибору джерел для збору даних, оглянуто різні способи збору даних з джерел та методи попередньої обробки тексту. Розглянуто різні методи вилучення ознак з тексту, текстової класифікації та статистичної обробки даних. Проведено огляд існуючих рішень щодо розпізнавання та аналітики пропаганди в текстах.

У розділі 2 виконано змістовну постановку задач розпізнавання та аналізу пропаганди в текстах. Задля виконання поставлених задач було розроблено алгоритми визначення політичної полярності джерела повідомлень, побудови мережі найпопулярніших хештегів, популярних телеграм-каналів, підмережі найпопулярніших хештегів та популярних телеграм-каналів за заданими параметрами, а також алгоритм збору статистичних даних.

У розділі 3 проаналізовано особливості проросійської та проукраїнської риторики, визначено джерела для збору даних та проведено збір даних з цих джерел. Також було проведено збільшення кількості даних та розподіл даних по категоріям.

У розділі 4 розроблено та натреновано модель класифікації політичної полярності повідомлень. Описано деталі архітектури моделі, обрані методи тренування та показники метрик якості класифікації моделі на тренувальному, перевірочному та тестовому наборах даних.

У розділі 5 виконано проєктування програмного застосунку, що реалізуватиме систему класифікації та аналізу пропаганди. Визначено набір акторів та сценаріїв, котрі має підтримувати система, окреслено загальні технічні вимоги до реалізації

застосунку. Відповідно до розроблених сценаріїв використання та технічних вимог було окреслено загальну архітектуру системи, її складових, а також формат комунікації між ними. Виконано опис процесів, котрі проходять в програмному застосунку, та які необхідні для реалізації сценаріїв використання.

В розділі 6 виконано програмну реалізацію системи класифікації та аналізу пропаганди. Виконано вибір мов програмування, бібліотек та фреймворків, котрі будуть використанні для розробки застосунку. По реалізації всіх сервісів та розробки необхідних таблиць, було надано результати роботи застосунку.

Новизна роботи:

— сформовано набір даних для класифікації політичної полярності повідомлень в контексті українсько-російської війни на основі популярних облікових записів в мережі Twitter та телеграм-каналів;
— розроблено спосіб класифікації джерел повідомлень на основі розробленої класифікаційної моделі;
— розроблено спосіб побудови мережі найпопулярніших хештегів в соціальній мережі Twitter, що відрізняється наявністю інформації про політичну полярність ребер мережі та можливістю побудови підмереж;
— розроблено спосіб побудови мережі популярних телеграм-каналів.

Практичною цінністю роботи є можливість бути використаною для розробки рекомендацій щодо регулювання інформаційного простору, а також сприянню уникненню небажаної дії проросійської пропаганди.

ПЕРЕЛІК ПОСИЛАНЬ

1. Pro-Russian propaganda recognition and analytics system based on text classification model and statistical data processing methods / Yurii Bezliudnyi [та ін.] // Адаптивні системи автоматичного управління. – 2023. – Т. 1, № 42. – С. 15–31.
2. Суспільно-політичні настрої населення України: результати опитування, проведеного 9-17 грудня 2021 року методом особистих (“face-to-face”) інтерв’ю [Електронний ресурс]. – Режим доступу: https://www.kiis.com.ua/?lang=ukr&cat=reports&id=1080&page=1 (дата звернення: 04.05.2023). – Назва з екрана.
3. Дослідження рівня інформаційної гігієни українських користувачів Facebook (листопад 2022) [Електронний ресурс]. – Режим доступу: https://docs.google.com/spreadsheets/d/1m9NYg_TQtB3wbdOX7yqVj0a5CBMvs1CzRweIGu2GKOE/ (дата звернення: 04.05.2023). – Назва з екрана.
4. Інформаційна гігієна під час війни: 7 базових правил [Електронний ресурс]. – Режим доступу: https://osvitoria.media/experience/informatsijna-gigiyena-pid-chas-vijny-7-bazovyh-pravyl/ (дата звернення: 04.05.2023). – Назва з екрана.
5. Пропаганда – що це таке та хто такі пропагандисти [Електронний ресурс]. – Режим доступу: https://termin.in.ua/propahanda/ (дата звернення: 04.05.2023). – Назва з екрана.
6. Драбюк С. Пропаганда та її види. Шляхи протидії пропаганді / Софія Драбюк // Аналітично-порівняльне правознавство. – 2022. – № 1. – С. 153–157. – Режим доступу: https://doi.org/10.24144/2788-6018.2022.01.28 (дата звернення: 04.05.2023). – Назва з екрана.
7. Оцінка вразливості та стійкості мешканців південних та східних областей України до наративів російської пропаганди [Електронний ресурс]. – Режим доступу: https://dif.org.ua/article/otsinka-vrazlivosti-ta-stiykosti-meshkantsiv-

pivdennikh-ta-skhidnikh-oblastey-ukraini-do-narativiv-rosiyskoi-propagandi (дата звернення: 04.05.2023). – Назва з екрана.

8. Jurafsky D. Speech and Language Processing / Dan Jurafsky, James Martin. – 2-ге вид. – Hoboken : Prentice Hall, 2008. – 1032 с.
9. Bengfort B. Applied Text Analysis with Python: Enabling Language-Aware Data Products with Machine Learning / Benjamin Bengfort, Rebecca Bilbro, Tony Ojeda. – Sebastopol : O'Reilly Media, 2018. – 332 с.
10. Schlkopf B. Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond (Adaptive Computation and Machine Learning) / Bernhard Schlkopf, Alexander J. Smola. – Cambridge : The MIT Press, 2001. – 644 с.
11. Hastie T. The Elements of Statistical Learning: Data Mining, Inference, and Prediction / Trevor Hastie, Robert Tibshirani, Jerome Friedman. – 2-ге вид. – New York : Springer, 2016. – 767 с.
12. Goodfellow I. Deep Learning (Adaptive Computation and Machine Learning series) / Ian Goodfellow, Yoshua Bengio, Aaron Courville. – Cambridge : The MIT Press, 2016. – 800 с.
13. Hochreiter S. Long Short-Term Memory [Електронний ресурс] / Sepp Hochreiter, Jürgen Schmidhuber // Neural Computation. – 1997. – Т. 9, № 8. – С. 1735–1780. – Режим доступу: https://doi.org/10.1162/neco.1997.9.8.1735 (дата звернення: 04.05.2023). – Назва з екрана.
14. Attention Is All You Need [Електронний ресурс] / Ashish Vaswani [та ін.] // Computing Research Repository. – 2017. – С. 1–15. – Режим доступу: https://doi.org/10.48550/arXiv.1706.03762 (дата звернення: 04.05.2023). – Назва з екрана.
15. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding [Електронний ресурс] / Jacob Devlin [та ін.] // Computing Research Repository. – 2018. – С. 1–16. – Режим доступу: https://doi.org/10.48550/arXiv.1810.04805 (дата звернення: 04.05.2023). – Назва з екрана.

16. An Introduction to Statistical Learning: with Applications in R (Springer Texts in Statistics) [Електронний ресурс] / Gareth James [та ін.]. – 2-ге вид. – New York : Springer, 2021. – 622 с.
17. Rao A. Actionable and Political Text Classification using Word Embeddings and LSTM [Електронний ресурс] / Adithya Rao, Nemanja Spasojevic // Computing Research Repository. – 2016. – С. 1–9. – Режим доступу: https://doi.org/10.48550/arXiv.1607.02501 (дата звернення: 04.05.2023). – Назва з екрана.
18. Geovany I. A sentiment analysis of the Ukraine-Russia conflict tweets using Recurrent Neural Networks [Електронний ресурс] / Isaac Geovany, Jorge Vargas // Monterrey, 1 черв. 2022 р. – Monterrey, 2022. – С. 1–5. – Режим доступу: https://www.researchgate.net/publication/361275253_A_sentiment_analysis_of_the_Ukraine-Russia_conflict_tweets_using_Recurrent_Neural_Networks (дата звернення: 04.05.2023). – Назва з екрана.
19. Džubur B. Semantic Analysis of Russo-Ukrainian War Tweet Networks [Електронний ресурс] / Benjamin Džubur, Žiga Trojer, Urša Zrimšek // Proceedings of Student Computing Research Symposium : матеріали наук. конф., Ljubljana, 6 жовт. 2023 р. – Ljubljana, 2022. – С. 1–4. – Режим доступу: https://www.scores.si/assets/papers/6258.pdf (дата звернення: 04.05.2023). – Назва з екрана.
20. Russia-Ukraine war – Tweets Dataset (65 days) [Електронний ресурс]. – Режим доступу: https://www.kaggle.com/datasets/foklacu/ukraine-war-tweets-dataset-65-days (дата звернення: 04.05.2023). – Назва з екрана.
21. Ukraine Conflict Twitter Dataset [Електронний ресурс]. – Режим доступу: https://www.kaggle.com/datasets/bwandowando/ukraine-russian-crisis-twitter-dataset-1-2-m-rows (дата звернення: 04.05.2023). – Назва з екрана.

22. Rosvall M. Maps of random walks on complex networks reveal community structure [Електронний ресурс] / M. Rosvall, C. T. Bergstrom // Proceedings of the National Academy of Sciences. – 2008. – Т. 105, № 4. – С. 1118–1123. – Режим доступу: https://doi.org/10.1073/pnas.0706851105 (дата звернення: 04.05.2023). – Назва з екрана.
23. Batagelj V. An O(m) Algorithm for Cores Decomposition of Networks [Електронний ресурс] / Vladimir Batagelj, Matjaž Zaveršnik // Computing Research Repository. – 2003. – С. 1–10. – Режим доступу: https://doi.org/10.48550/arXiv.cs/0310049 (дата звернення: 04.05.2023). – Назва з екрана.
24. Brin S. The anatomy of a large-scale hypertextual Web search engine [Електронний ресурс] / Sergey Brin, Lawrence Page // Computer Networks and ISDN Systems. – 1998. – Т. 30, № 1-7. – С. 107–117. – Режим доступу: https://doi.org/10.1016/s0169-7552(98)00110-x (дата звернення: 04.05.2023). – Назва з екрана.
25. The ‘Goebbels Method’: RIA Novosti as Window into Russian Propaganda [Електронний ресурс]. – Режим доступу: https://www.geopoliticalmonitor.com/the-goebbels-method-ria-novosti-as-window-into-russian-propaganda (дата звернення: 04.05.2023). – Назва з екрана.
26. Від «Трухи» до Гордона: найпопулярніші канали українського сегмента Telegram [Електронний ресурс]. – Режим доступу: https://detector.media/monitorynh-internetu/article/202665/2022-09-09-vid-trukhy-do-gordona-naypopulyarnishi-kanaly-ukrainskogo-segmenta-telegram (дата звернення: 04.05.2023). – Назва з екрана.
27. Telegram channels and groups catalog [Електронний ресурс]. – Режим доступу: https://tgstat.ru/en (дата звернення: 04.05.2023). – Назва з екрана.
28. Telegram Core [Електронний ресурс]. – Режим доступу: https://my.telegram.org (дата звернення: 04.05.2023). – Назва з екрана.

29. Smaller Bert Model [Електронний ресурс]. – Режим доступу: https://tfhub.dev/tensorflow/small_bert/bert_en_uncased_L-4_H-512_A-8 (дата звернення: 04.05.2023). – Назва з екрана.
30. Text preprocessing for BERT [Електронний ресурс]. – Режим доступу: https://tfhub.dev/tensorflow/bert_en_uncased_preprocess (дата звернення: 04.05.2023). – Назва з екрана.
31. Loshchilov I. Fixing Weight Decay Regularization in Adam [Електронний ресурс] / Ilya Loshchilov, Frank Hutter // Computing Research Repository. – 2017. – С. 1–19. – Режим доступу: https://doi.org/10.48550/arXiv.1711.05101 (дата звернення: 04.05.2023). – Назва з екрана.
32. Kingma D. P. Adam: A Method for Stochastic Optimization [Електронний ресурс] / Diederik P. Kingma, Jimmy Ba // Computing Research Repository. – 2017. – С. 1–15. – Режим доступу: https://doi.org/10.48550/arXiv.1412.6980 (дата звернення: 04.05.2023). – Назва з екрана.
33. Angular vs React vs Vue.js: A Comparative Study – 2022 [Електронний ресурс]. – Режим доступу: https://www.zartis.com/angular-vs-react-vs-vuejs (дата звернення: 04.05.2023). – Назва з екрана.
34. Yurets2000/propaganda-fighter-ui, UI part of “Propaganda Fighter” project [Електронний ресурс]. – Режим доступу: https://github.com/Yurets2000/propaganda-fighter-ui (дата звернення: 04.05.2023). – Назва з екрана.
35. Yurets2000/propaganda-fighter-be, BE part of “Propaganda Fighter” project [Електронний ресурс]. – Режим доступу: https://github.com/Yurets2000/propaganda-fighter-be (дата звернення: 04.05.2023). – Назва з екрана.