КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА
Факультет комп'ютерних наук та кібернетики
Кафедра теоретичної кібернетики

Кваліфікаційна робота

на здобуття ступеня бакалавра
за спеціальністю 122 Комп'ютерні науки
на тему:

Розробка системи для прогнозування реакцій на новини на основі даних Телеграм каналів

Виконав: студент 4-го курсу
Олександр КОСУХА _____
(підпис)

Науковий керівник:
професор, доктор фізико-математичних наук
Анатолій ПАШКО _____
(підпис)

Консультанти з застосування алгоритмів машинного навчання:
Technical Trainer in Grid Dynamics, Ph. D, Assoc. Prof.
Анастасія ДЕЙНЕКО _____
(підпис)

Technical Trainer in Grid Dynamics, Data Scientist in WiseTech Global,
Антоніна БОНДАРЧУК _____
(підпис)

Засвідчую, що в цій роботі
немає запозичень з праць інших авторів
без відповідних посилань.
Студент _____
(підпис)

Роботу розглянуто й допущено до захисту
на засіданні кафедри теоретичної кібернетики
«____» _____ 2023 р.,
протокол № ___
Завідувач кафедри
професор, доктор фізико-математичних наук
Юрій КРАК

_____
(підпис)

Київ - 2023

ЗМІСТ

ВСТУП

Обгрунтування вибору теми.

Складно заперечувати роль соціальних мереж в сучасному світі, і окремий інтерес представляють собі месенджери типу Viber та Телеграм, які дозволяють комунікувати не тільки на рівні користувач-користувач, але й користувач-користувачі, що зазвичай характерно для групових чатів та для публічних новинних каналів.

Варто також відмітити популярність Телеграму, якій він завдячує зокрема своїй політиці конфіденційності та певній толерантності у порівнянні з іншими соціальними медіа.

Для Телеграму закономірно виникають задачі інтелектуального аналізу повідомлень, яким сприяє можливість використання чат-ботів в цій мережі ([1] - [3]). Також для неї є можливим зручне тестування алгоритмів аналізу тональності текстів повідомлень та аналізу поведінки користувачів (наприклад, [4] - [5]), оскільки з 30 січня 2021 року користувачі Телеграму можуть публічно залишати реакції (смайли) на повідомлення або новини.

Також потрібно зазначити, що для інших медіа, напрклад, Facebook і Twitter, дослідження контенту користувачів є задачами, які активно розв'язуються ([6] - [10]).

Мета і завдання.

Мета роботи. Мета роботи полягає у формалізації принципів, технологій та алгоритмів, які можна рекомендувати для розробки ефективної системи інтелектуального аналізу та прогнозування реакцій на новини на основі даних Телеграм-каналів, а також створення прототипу такої системи.

Досягнення мети зумовлює розв'язання наступних задач:

1. проаналізувати та автоматизувати процес збору та попередньої обробки новин з мережі Телеграм;
2. розглянути основні підходи до створення моделей обробки природньої мови;
3. розглянути основні моделі для класифікації тексту;
4. розробити програмне забезпечення для прогнозування реакцій на новини.

Об'єкт дослідження. Сентиментальний аналіз новин.

Предмет дослідження. Особливості збору реакцій на новини в мережі Телеграм та створення на їхній основі систем класифікації тексту.

Методи дослідження. Для розв'язання поставлених задач у роботі використовуються:

- алгоритм збору і попередньої обробки даних; використовуються методи векторизації тексту: "торба слів", TF-IDF та вкладення GloVe;
- алгоритми машинного навчання: наївний баєсівський класифікатор, метод опорних векторів, нейромережеві моделі LSTM та GRU.

Фінальні моделі прогнозування оцінені на випробувальному наборі даних з використанням F-міри як показника точності моделі.

Інструменти розроблення. Для програмної реалізації алгоритмів використано мову Python.

Наукова новизна отриманих результатів. У процесі розв'язання поставлених задач отримано нові наукові результати, які полягають у такому:

- запропоновано методику збору і попередньої обробки даних з Телеграм каналів, що дозволяє автоматизовано збирати, а також робити доступними для використання та аналізу повідомлення з заданого каналу та в межах заданого часового проміжку ;
- проведено порівняння поширених підходів для створення моделей класифікації текстових даних у контексті тематики сучасних новин українською мовою;
- розроблено модель для прогнозування реакцій на новини.

Практичне значення одержаних результатів полягає в можливості використання отриманих моделей в тональній оцінці текстів. Також, методика збору повідомлень з Телеграму доступна для збору даних в більших кількостях для подальшого застосування в багатьох задачах обробки природньої мови.

Результати роботи були представлені на XXXVII International Conference "PROBLEMS OF DECISION MAKING UNDER UNCERTAINTIES" November 23 - 25, 2022, Sheki-Lankaran, Republic of Azerbaijan ([11]) та на Міжнародній науково-практичній конференції "Шевченківська весна - 2023", 14 квітня 2023 р., м. Київ, Україна ([12]). Також результати даного дослідження було опубліковано у Віснику Київського національного університету, Серія фізико-математичні науки №3 2022 року ([13]).

Структура та обсяг роботи. Робота складається з 42 стор. та містить в собі такі структурні елементи: титульний аркуш, зміст, основна частина на 36 стор. (складається з вступу, 2 розділів і висновків), список використаної літератури джерел із 40 найменувань на 4 стор.

Робота містить 27 рисунків та 2 таблиці.

Розділ 1. Теоретичні відомості

Для сематичного аналізу текстів було використано алгоритми векторизації тексту та різні варіанти архітектур нейронних мереж класифікації тексту розглянуті далі. Детальніше ці алгоритми розглянуто далі.

1.1 Огляд моделей та визначень, які є фундаментальними складовими застосованих методів

1.1.1 Машинне навчання

Машинне навчання (Machine Learning) - це галузь штучного інтелекту, яка досліджує розвиток алгоритмів, які дозволяють моделям покращувати свою продуктивність на основі даних. В машинному навчанні застосовуються статистичні методи для навчання моделів, робота яких полягає в мінімізації функції втрат моделі. Таким чином, розробнику необхідно лише написати алгоритм навчання, який на базі даних знаходитиме зв'язки самостійно, та налаштувати його гіперпараметри.

У машинному навчанні основні поняття включають:

- Модель. Модель в машинному навчанні — це функція, яка відображає вхідні дані у вихідні, з мметою описати якісь дані або зробити прогноз. Модель характеризується своєю точністю на тренувальному та тестувальному наборах даних, а визначається алгоритмом тренування та вагами, які є параметрами алгоритму тренування і змінюються алгоритмом оптимізації під час навчання.
- Дані. Дані в машинному навчанні використовуються для тренування моделей та оцінки продуктивності їхньої роботи. Вони можуть бути структурованими — коли кожен окремий запис має визначену кількість полів з визначеними можливими значеннями або їх діапазоном. Наприклад, це може бути таблиця з інформацією про квіти: вид, ширина пелюсток, висота квітки, об'єм бутону тощо. Такі дані є суку-

пністю записів, які складаються з ознак або полів (features). Також дані можуть бути неструктурованими (текст, зображення, аудіо)– - вони потребують попередньої обробки, а також часто спеціальних алгоритмів.

– Види навчання. В контексті конкретної задачі та отриманих даних, розрізняють загалом три види навчання:
  – Навчання зі вчителем. Це вид навчання, коли тренувальні дані складаються з розмічених записів. Розмічений запис — це елемент даних, в якому, разом з інформаційними полями елемента, є також поле, яке необхідно передбачувати в рамках конкретної задачі машинного навчання — еталонний вихід або мітка. В контексті задачі класифікації квітів, це може бути вид квітів, в контексті задачі регресії - ціна за квітку. За рахунок даних виду інформація-еталонний вихід, алгоритм навчання навчається знаходити зв'язки між двома, і в кінці навчання може коректно розставляти потрібні мітки як для навчальних, так і для нових даних, за рахунок побудови потрібної для задачі функції.
  – Навчання без вчителя. На відміну від навчання зі вчителем, даний тип навчання не має розмічених даних, а отже еталонних виходів не існує. Ціль — вивчити структуру даних, знайти приховані патерни та залежності, зменшити розмірність даних або провести їх аналіз. Також за допомогою цього виду навчання можна побудувати нові моделі, як-от GloVe(детальніше це буде розглянуто в настому підрозділі).
  – Навчання з підкріпленням. Цей вид машинного навчання вивчає оптимальні стратегії прийняття рішень, шляхом взаємодії агента з навколишнім середовищем. Агент(модель, що проходить тренування) навчається приймати послідовні рішення в середовищі з метою максимізації нагороди, яка накопичується з часом. Навчена модель може приймати правильні дії з можливих, виходячи з наданих станів середовища. Даний вид навчання широко використовується в робототехніці та розробці програм для ігор, таких як AlphaGo [33]

- Функція втрат. Для навчання моделі необхідний показник, який показуватиме її точність на задачі. Цим показником є функція витрат, яка вказує на різницю між тим, як відпрацювала модель та еталонною роботою. Мінімізація цього показника є основною задачею навчання.
- Навчання. Навчання моделі представляє собою процес знаходження вагів моделі, на базі яких і проходить обчислення її виводу. Процес навчання проходить таким чином: певна порція тренувальних даних надається моделі, на базі цих даних модель обчислює вихідні значення. Вихідні значення моделі і, у випадку навчання зі вчителем, еталонні виходи передаються у функцію втрат, яка показує ефективність моделі на даному кроці. На базі значення цієї функції, алгоритм оптимізації змінює ваги моделі таким чином, щоб наступного разу функція втрат була меншою. Під час навчання одні й ті ж дані можуть оброблюватись алгоритмом навчання та оптимізації кілька разів. Коли навчання закінчується, модель має відкоригувати свої ваги таким чином, щоб вхідні дані відображались у вихідні з необхідною точністю. Цей показник точності в деяких випадках має бути близьким до точності роботи фахівця-людини при виконанні даного завдання. Також, після навчання на тренувальному наборі даних, модель має адекватно працювати на даних, які не були використані під час навчання. Тобто, показник точності або функції втрат має бути прийнятним.
- Перенавчання. Під час перенавчання модель має такі ваги, що похибка чи шум у даних сприймається як важливі значення ознак, що впливають на результат. В наслідок даного ефекту модель має велику точність на тренувальному наборі даних, але не здатна працювати адекватно на даних, на яких тренування не відбувалося.
- Недонавчання. Недонавчання вказує на неможливість моделі знайти ваги, які б відображали зв'язок між вхідними ознаками та їх еталонними виходами. Одними з причин недонавчання є мала кількість можливих вагів моделі, занадто великий коефіцієнт регуляризації, неможливість алгоритму навчання відобразити складні зв'язки між даними внаслідок його особливостей роботи.
- Епоха. Повне проходження тренувального набору даних під час навчання називається епохою. Часто для отримання достатньої точності

моделі потрібно провести декілька епох тренування — кількість визначається розробником і є гіперпараметром.

– Гіперпараметри. Гіперпараметри — це величини, за допомогою яких відбувається керування процесом навчання. На противагу параметрам моделі, таким як ваги, які знаходяться алгоритмом оптимізації, гіперпараметри визначаються розробником вручну. Гіперпараметри включають в себе кількість епох, коефіцієнт регуляризації, глибину нейронної мережі та інші величини.
– Градієнтний спуск. Даний метод є одним з основних алгоритмів оптимізації, який використовується для пошуку локального мінімуму функції. Він є основою багатьох методів машинного навчання і нейронних мереж, дозволяючи ефективно оновлювати параметри моделі залежно від градієнту функції втрат. Процес градієнтного спуску починається з визначення ініціалізації параметрів моделі — визначення початкових значень, які часто обираються випадкових чином. Потім, під час навчання, відбуваються ітерації, в кожній з яких обчислюється градієнт функції втрат відносно параметрів моделі. Градієнт представляє собою вектор часткових похідних даної функції по кожному параметру.

  На кожній ітерації значення параметрів оновлюються, зміщуючись в напрямку, протилежному градієнту. Він вказує на найшвидший спад функції втрати. За допогою кроку навчання(learning rate), відбувається вибір величини зміни параметрів у вказаному напрямку. Після зміни параметрів, при правильному виборі learning rate, функція втрат має зменшитись.

  Процес градієнтного спуску продовжується до досягнення заданої точності або до вичерпання максимального числа вказаних розробником ітерацій. В результаті, параметри моделі оновлюються таким чином, щоб мінімізувати значення функції втрат, наближаючи її до локального мінімуму.

  При виборі занадто великих значень learning rate, зміна параметрів на кожній ітерації алгоритму може бути завелика, і може спостерігатись збільшення функції втрат. При виборі занадто малих значень, процес навчання відбуватиметься довго через необхідність у великій кількості оновлень вагів для досягнення мінімуму функції втрат.

1.1.2 Регуляризація

Регуляризація є методикою уникнення ситуацій, коли модель машинного навчання занадто сильно пристосовується до навчальної вибірки і втрачає здатність узагальнюватись на ще не бачені дані. За рахунок впровадження регуляризації може зменшитись точність моделі на тренувальному наборі даних, але збільшитись точність на тестувальній вибірці. На практиці широко застосовні зокрема такі методики регуляризації:

- L1-регуляризація (регуляризація Lasso) та L2-регуляризація (регуляризація Ridge). Перший крок методик полягає в додаванні до функції втрат суми значень всіх вагів моделі(значення ваг попередньо беруться по модулю у випадку використання L1-регуляризації та підносяться в квадрат при використанні L2-регуляризації). І оскільки значення вагів моделі додалися до функції втрат, то тепер алгоритм оптимізації намагатиметься недопустити надто великих значень вагів моделі.

  Для контролювання впливу регуляризації на модель, сума ваг множиться на коефцієнт λ. Значення коефіцієнту, близькі до нуля, мають невеликий вплив на роботу моделі і її здатність до узагальнення. Занадто великі коефіцієнти призводять до надто великого зменшення значення вагів, що може спричинити зменшення точності як на тренувальному, так і на тестувальному наборах.

  При застосуванні L1-регуляризації, є деякі ваги моделі стають близькими до нуля. В результаті модель може краще відсіювати непотрібні ознаки в даних. Це є корисним, коли дані, що приходять на вхід моделі, мають багато ознак, деякі з яких можуть бути незначними або надлишковими.

  При застосуванні L2-регуляризації, всі ваги зменшуються пропорційно, що не призводить до розрідженості моделі, як відбувається у випадку L1-регуляризаці. Це дозволяє не виділяти окремі ознаки, а також перерозподілити вплив кожної ознаки на роботу моделі більш рівномірно.
- Dropout. Ідея регуляризації Dropout полягає в тому, щоб під час чергового тренувального кроку "вимкнути"випадковий набір нейронів в

нейронній мережі під час тренування. Значення вихідних сигналів вимкнутих нейронів стають нульовими, і вони не беруть участі у передачі сигналів протягом даного кроку.

Процес випадкового вимкнення нейронів змушує модель задіювати інші нейрони для коректної роботи, оскільки під час тренування різні нейрони можуть бути вимкнуті на різних кроках. Це зменшує залежність мережі від конкретних зв'язків і допомагає виявляти більш універсальні ознаки. Це допомагає запобігти перенавчанню. Кожен нейрон вимикається з деякою ймовірністю, яка задається користувачем та впливає на силу регуляризації. Чим більша ймовірність - тим більший ефект має регуляризація на модель. Занадто велике значення ймовірності може привести до недостатнього навчання моделі, що означає зменшення точності як на тренувальному, так і на тестувальному наборах.

1.1.3 Нейронні мережі

Нейронна мережа — це модель машинного навчання, що складається з вузлів, які ще називають нейронами. Кожен нейрон отримує дані на вхід, проводить обчислення, і дає результат обчислень на вихід. Вхід нейронів може бути як і вхідними даними моделей, так і результатом обчислень інших нейронів. Зв'язки між нейронами характеризуються вагами, які тренуються алгоритмом оптимізації.

Основні поняття та складові, які є в нейронних мережах:

– Функція активації. Функція активації — це функція, яка отримує вхідні значення нейрона, робить над ними обчислення та переводить на вихід нейрону. Функції активації мають бути нелінійними і необхідні для того, щоб нейронна мережа могла будувати нелінійні моделі. Оскільки структура нейронної мережі є суперпозицією функцій активації від вхідних даних та вагів як аргументів, то при лінійний функції активації або взагалі при її відсутності роботу мережі можна буде звести до лінійної регресії, що не дає достатньої гнучкості та можливісті знаходити зв'язки між вхідними даними та виходами мо-

делі. При застосуванні нелінійної функції активації, можна добитись досить гнучкої моделі нейронної мережі, якої в багатьох випадках неможливо добитись лінійною регресією. Серед функцій активації, що використовуються, можна виділити такі:

- Relu. Обчислюється за формулою: $Relu(z) = max(0, z)$. Функція лінійна на проміжку$(0; + \inf)$. Переваги функції ReLu:
    - Рідше виникає проблема зникання градієнту, оскільки він приймає всього два значення: 0 і 1 (наприклад, в порівнянні з функцією sigmoid)
    - Швидке обчислення функції та похідної

  Недоліком функції ReLu є поява мертвих зон. Існують ситуації, коли нейрон при всіх вхідних значеннях прийматиме значення 0. Тоді ваги, що йдуть до нього, не змінюватимуться, оскільки їх похідна при алгоритмі оптимізації також буде 0. Це призводить до того, що цей нейрон це буде використовуватись, що погіршить навчання мережі. Така ситуація буває, коли ваги ініціалізовані так, що нейрон завжди виходом 0, або при великих навчальних кроках оптимізатора моделі.
- Sigmoid. Обчислюється за формулою: $\sigma(z) = \frac{1}{1+e^{-z}}$.
  На великих за модулем аргументах похідна прямує до 0.
  Переваги:
    - Нелінійна та неперервна, що є достатнім для використання в нейронних мережах
    - Похідну можна представити через значення функції, що дозволяє пришвидшити обчислення

  Недоліки:
    - Зникнення градієнту — при великих за модулем значеннях нейронів, похідна sigmoid в таких точках буде близькою до нуля, що значно сповільнить процес навчання і зміни значень вагів всієї мережі. Таке може статись при ініціалізації вагів надто великими числами.
    - При роботі алгоритму backpropagation, градієнт буде передаватись від останніх шарів до початкових. І оскільки максимальне значення похідної — 0.25 (при

аргументі 0) то щоразу градієнт буде зменшуватись мінімум в 4 рази. Це призводить до повільнішого навчання шарів мережі, які ближче до вхідного шару в порівнянні з тими, що ближче до вихідного.

- Softmax. Обчислюється за формулою: $\sigma(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$, $i = 1,2,\ldots,K$, де K - кількість класів

  Функція переводить K-вимірний вектор довільних значень в K-вимірний вектор значень від 0 до 1, сума яких дорівнює 1. Дана функція є корисною в представленні розподілу ймовірності при K можливих варіантах. В нейронних мережах вона часто використовується в задачах класифікації як функція активації останнього шару, і її вихід є виходом всієї нейронної мережі. Він розцінюється як розподіл ймовірності віднесення вхідні даних до конкретного класу.

– Зворотнє поширення помилки. Алгоритм зворотного поширення помилки (backpropagation) є алгоритмом для знаходження градієнту функції втрат нейронних мереж відносно її вагів. Процес backpropagation включає два основних етапи: пряме поширення сигналу і зворотне поширення помилки. Під час прямого поширення вхідні дані проходять через мережу, з активацією нейронів і обчисленням вихідного значення. Потім отриманий вихід порівнюється з еталонним вихідним значенням, різниця між ними є втратою.

  Після обчислення втрати починається зворотне поширення помилки. У цьому етапі значення втрати поширюється від кінцевого шару назад до початкових шарів мережі, пропорційно вагам, що пов'язані з кожним нейроном. Основна ідея полягає в тому, що кожен нейрон у мережі отримує помилку від наступного шару, та обчислює свою власну помилку шляхом поширення отриманої через свої ваги. Для кожного нейрона обчислюється градієнт, який вказує, як зміниться втрата при зміні його ваги. Градієнт обчислюється шляхом застосування правил диференціювання складної функції до похідних функції активації та ваг.

  Потім градієнти передаються алоритму оптимізації — градієнтному спуску, який оновлює ваги.

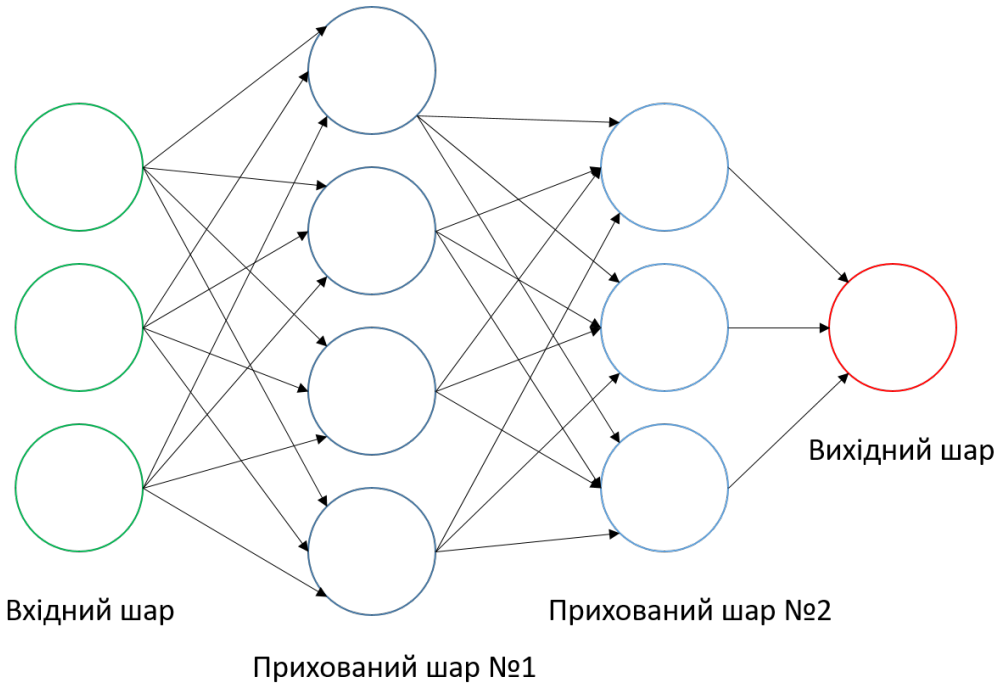

Рисунок 1 — Приклад чотирьохшарової нейронної мережі

– Шари мережі. Нейронна мережа складається з послідовностей шарів нейронів. Один нейрон має зв'язки у вигляді вагів від всіх нейронів з попереднього шару, а від нього йдуть зв'язки до всіх нейронів наступного шару. Сам шар нейронів може мати один чи багато нейронів, між якими немає зв'язків. Також виділяють вхідний та вихідний шари. Вхідний шар є початковим шаром мережі і не має вхідних зв'язків, результат вихідного шару є результатом роботи моделі, а отже не має вихідних зв'язків. Між вхідним і вихідним шарами може бути довільна кількість шарів, які називаються прихованими. Вони мають як вхідні, так і вихідні зв'язки. Модель з чотирма шарами нейронів зображена на Рис. 1. Стрілочками позначено зв'язки між нейронами, колами - самі нейрони. Кількість шарів нейронної мережі, а також кількість нейронів в кожному прихованому шарі є гіперпараметром мережі та визначається розробником.

1.1.4 Механізм attention, його застосування та різновиди

- Attention(Увага) [32], [34]. Застосування уваги в машинному навчанні є методикою, що надає можливість моделям виділяти надавати вагу частинам даних таким чином, щоб більш важливі частини входу були підсилені, а менш важливі — пригнічені. Це дозволяє моделі краще розподіляти свої ресурси, опрацьовуючи лише важливі частини входу.
- Self-attention. При використанні self-attention формується матриця уваги, в якій кожному слову надається своє значення уваги відносно кожного іншого слова у вхідній послідовності. Даний механізм дозволяє знайти зв'язки між частинами речення та зрозуміти, як одні слова залежать від інших. Наприклад, коли модель знаходить займенник в реченні, вона ставить підсилену увагу тим словам, на які вказує цей займенник. Далі значення уваги використовуються для покращення репрезентації всього речення загалом.
- Multi-headed attention [35]. При використанні Multi-headed attention формується деяка кількість матриць значень self-attention. В результаті кожна матриця має різнопланову увагу, що збільшує можливість моделі бачити зв'язки різного роду. Наприклад, в реченні, в якому є займенник, одна матриця self-attention може відображати зв'язок займенника з підметом, а інші — з присудком або означенням.
- Transformer(Трансформер) [35]. Модель машинного навчання, яка складається з кодувальника(encoder), декодера(decoder) та вихідної нейронної мережі. Кодувальник — це послідовність повторюваних блоків, кожен з яких отримує вхідну матрицю від попереднього, застосовує механізм multi-headed attention; отримані значення уваги передаються до нейронної мережі, яка на виході передає свою матрицю в наступний блок. Після проходження блоків кодувальника вивід передається декодеру, який складається з блоків такої ж архітектури, як і блоки кодувальника. Кожен блок декодера отримує вивід кодувальника, а також вивід попереднього блоку декодера. Завдання декодера - перевести вивід кодувальника у потрібний вивід для задачі, який потім за допомогою вихідної нейронної мережі переводиться в потрібну форму — наприклад, номери слів в словнику.

1.1.5 BERT

BERT(Bidirectional Encoder Representations from Transformers) [36]. Це мовна модель, яка використовує, між інших, частину архітектури Трансформерів — а саме encoder. Вона зроблена для створення якісних відображень слів та їх послідовностей у векторні відповідники. Тренування відбувається шляхом видалення частини слів з тексту і навчання моделі правильно передбачувати видалені слова. В результаті отримана модель добре розрізняє значення слів відносно контексту їх вживання, а також є хорошим початковим блоком для моделей обробки природної мови. На відміну від Трансформерів, які необхідно вивчати на кожній задачі з новим набором розмічених даних, BERT тренується на великих об'ємах текстів один раз, і потім отриману модель, яка має навчилась неявно вирізняти структуру мови, можна використовувати при вирішенні нових задач обробки природної мови з мінімальними змінами.

1.2 Огляд алгоритмів векторизації текстової інформації

В роботі будуть використані кілька різних методик векторизації тексту — тобто переведення тексту в число або вектор чисел. Вони наведені нижче:

- Модель "торба слів". Під час використання даного підходу створюється словник, що містить всі унікальні слова з корпусу повідомлень. Кожен документ або речення перетворюються в вектор, де кожному слову кожному слову співвідноситься частота його появи у всьому корпусі повідомлень. Дана модель припускає, що значення кожного слова в тексті не залежить від контексту і є сталими.
- TF-IDF. Цей метод враховує два основних фактори:
  - Term Frequency (TF) — відображає частоту вживання слова в повідомленні. Зазвичай, використовується частота входження слова у порівнянні з загальною кількістю слів у документі.
  - Inverse Document Frequency (IDF) — відображає важливість слова в корпусі текстів. Вона показує, наскільни унікальне є

дане слово для повідомлення шляхом обчислення логарифму відношення загальної кількості повідомлень до кількості повідомлень, в яких зустрічається це слово.

При використанні цієї методики вага слова прямо пропорційна частоті вживань цього слова у повідомленні, і обернено пропорційна частоті вживання у інших повідомленнях датасету.

- Вкладеннння слів. Вкладення відображають кожне слово у векторний простір, в якому відстань між словами пов'язана з семантичною подібністю. Дана модель зазвичай тренується на великих корпусах даних з подальшим застосуванням в конкретних задачах
- Вкладення GloVe ([27]) . Дані вкладення будуються на матриці суміжності, в якій кожному слову відображається кількість разів його появи в контексті іншого слова. Контекст слова — це кількість слів, які стоять зліва і справа від нього; конкретне значення контексту є гіперпараметром моделі та залежить від корпусу для навчання та інших характеристик. Після отримання матриці суміжності проводиться розклад даної матриці таким чином, щоб кожному слову відповідав вектор ознак визначеної розмірності(на практиці вкладення найкращої якості мають розмірність 300, в загальному розмірність варіюється між 200 та 400). Даний вектор ознак і є вкладенням.
- Вкладення BERT. Дані вкладення є шарами нейронної мережі, тренованої на задачі вгадування пропущених слів в тексті. На практиці для отримання вкладення слова використовуються як і комбінації значень шарів різної глибини, та і значення якогось окремого шару — часто останнього. Дані вкладання є контекстними, що означає залежність вкладання слова від інших слів в реченні та їх порядку. Також є методики, завдяки яким можна отримувати вкладання не кожного окремого слова, а речення вцілому, що покращує спроможність вкладень відобразити сенс написаного тексту.

1.3 Огляд алгоритмів для прогнозування

На даний момент є багато алгоритмів для роботи з текстом. Вони включають в себе як і класичний наївний баєсівський класифікатор([17] - [21]) та метод опорних векторів ([22] - [26]), алгоритми навчання LSTM([30]) та GRU ([28]), які є рекурентими нейронними мережами, та BERT — мовна модель, яка використовує архітектуру трансформерів. В цьому розділі буде наведено короткий опис даних алгоритмів.

Наївний баєсівський класифікатор. Грунтується на використанні теореми Баєса для обчислення ймовірності належності елемента вибірки до одного з запропонованих класів, при припущенні про незалежність змінних. У нашому дослідженні ми множимо між собою умовні ймовірності слів однієї новини для двох класів (позитивний та негативний). Після чого отримані дві ймовірності порівнюються і надається перевага тому класу, ймовірність якого більша.

Метод опорних векторів. Базується на використанні алгоритмів навчання, які називають опорно-векторними машинами (ОВМ, англ. support vector machines, SVM). В цьому методі для тренувального набору даних формується модель, яка відображає кожне повідомлення у точку в просторі. Тоді в цьому просторі шукається "найширший коридор", який дозволяє розділити тренувальний набір даних на два окремих підпростори. В подальшому, кожному новому зразку при роботі ОВМ автоматично за відповідними йому координатами ставиться у відповідність його клас.

Рекурентна нейронна мережа. Це тип нейронних мереж, що може обробляти послідовності даних, як-от текстові дані чи звук. Її робота розбивається на кроки, де на новий крок дається новий елемент послідовності. Внутрішні вузли мережі використовуються для запам'ятовування попередньої інформації та обробки нових входів.

LSTM (ДКЧП — Довга короткочасна пам'ять). Це архітектура штучних рекурентних нейронних мереж, яка створена для того, щоб добре вловлювати залежності між словами, що знаходяться далеко одне від одного. Вона використовує спеціальні вузли, які мають три вентилі: для нової інформації, для запам'ятовування та забування інформації, і вихідний вентиль. Таком чином, правильно натренована мережа вловлює важливу інформацію, яка не

втрачається після багатьох кроків, та правильно поєднує її з новими словами на кожному кроці.

GRU (ВРВ — Вентильні рекурентні вузли). Підвид LSTM, який поєднує вхідний та запам'ятовувальний вентилі в один, вихід якого стає вихіднидним вентилем. Дані мережі є обчислювально легшими за LSTM, і дають схожі результати в обробці природної мови та моделюванні звукових сигналів.

Нейронна мережа на основі BERT. Текст для обробки передається в натреновану модель BERT, яка видає вкладення всього тексту. Ці вкладення передається нейронній мережі, яка опрацьовує його у відповідності з поставленою метою.

Описані алгоритми та методи векторизації тексту дають достатній інструментар для створення моделей обробки природньої мови, зокрема для задач класифікації. Ми оцінимо їхню точність на наборах новин українською мовою в наступному розділі.

1.4 Висновки до розділу 1

Описані алгоритми та методи векторизації тексту дають достатній інструментар для створення моделей обробки природньої мови, зокрема для задач класифікації. Ми оцінимо їхню точність на наборах новин українською мовою в наступному розділі.

Розділ 2. Програмна реалізація

Програмна реалізація є послідовністю Jupyter-ноутбуків, що виконуються один після іншого. На вхід першому ноутбуку приходять дані, зібрані за допомогою Telegram API. Всі ноутбуки розміщено у відкритому доступі [16]. Схема розробленої системи вказана на Рис. 2.

Структура проекту:

- reactions_prediction_preprocessing — переведення даних з одного каналу в формат .csv та попередня обробка тексту;
- rp_concat_to_pos_neg — збирання файлів з різних каналів та дат в один та розбиття повідомлень на негативний та позитивний класи;
- rp_wordcloud — створення word clouds — хмар слів;
- reactions_prediction_model — застосування наївного баєсівського класифікатору та метода опорних векторів на базі методик векторизації "торба слів", та TF-IDF;
- reactions_prediction_model_without_stopwords — використання тих алгоритмів, що в попередньому ноутбуці, але з попереднім видаленням шумових слів в датасеті;
- reaction_prediction_lemmatized — використання алгоритмів reactions_prediction_model, але з попередньою лематизацією шумових слів в датасеті;
- reactions_predictions_glove — застосування GloVe із використанням нейронних мереж(LSTM і GRU);
- reactions_bert — застосування BERT з використанням нейронних мереж.

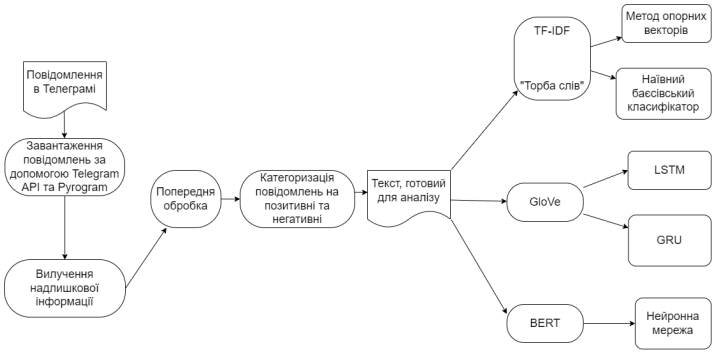

Рисунок 2 — Блок-схема роботи системи

На вхід першому ноутбуку приходять дані, зібрані за допомогою Telegram API (детальніше в підрозділі 2.1).

Далі відбувається:

- попередня обробка та тематичний аналіз повідомлень Телеграм (reactions_prediction_preprocessing, rp_concat_to_pos_neg, rp_wordcloud ; детальніше в підрозділі 2.2);
- прогнозування реакцій на основі обробленого датасету. Детальніше про це в підрозділі 2.3:
  - reactions_prediction_model ;
  - reactions_prediction_model_without_stopwords;
  - reaction_prediction_lemmatized;
  - reactions_bert;
  - reactions_predictions_glove.

  reaction_prediction_lemmatized, а також reactions_predictions_glove).

2.1 Збір даних

Однією із задач дослідження було формування навчальної вибірки для інтелектуального аналізу повідомлень в Телеграм-каналах.

Було створено датасет, сформований з 63494 новинних повідомлень з реакціями з Телеграм-каналів ТСН [15], "Громадське" та "Ukraine NOW" з 09 березня 2022 р. Основними критеріями при виборі каналів були: україномовний контент; відсутність орієнтації на специфічну цільову аудиторію, як, наприклад, у Телеграм-каналів районів, міст і т.п.; активність підписників. Дані канали налічують від однієї до кількох сотень тисяч підписників; кількість переглядів однієї новини в каналах ТСН та "Ukraine NOW" варіюється в межах 150-200 тисяч переглядів, для каналу "Громадське" — приблизно 35 тисяч, тобто трохи менше, ніж третина всіх підписників.

Тобто на основі цього можна обгрунтовано припускати репрезентативність сформованого датасету повідомлень з цих джерел.

Для отримання доступу до даних Телеграм-каналів було використано фреймворк Pyrogram [37], який працює з Telegram API ([14]). Було написано код додатку, що дозволяє завантажувати повідомлення з будь-якого каналу,

```
async def main():
    with open("bigchat_ukraine_now_from_1_march.json", "w+", encoding="utf-8") as output:
        output.write('[')
        async with Client('my_account', api_id, api_hash) as app:
            async for message in app.iter_history(UKRAINE_NOW_ID):
                mesdict = message.__dict__
                mesdict['date'] = datetime.datetime.fromtimestamp(mesdict.get('date'))
                if mesdict['date'] >= start_date:
                    json.dump(dict_to_prim(mesdict), output, indent=2, ensure_ascii=False)
            output.write(']')

asyncio.run(main())
```

Рисунок 3 — Функція для завантаження повідомлень з Телеграм-каналів

до якого має доступ користувач, починаюючи з визначеної дати start_date. На Рис. 3 зображено реалізовану функцію завантаження. Спочатку створюється підключення до акаунту Telegram API, та відбувається ітерація по всіх повідомленнях каналу, який характеризується унікальним числовим ключем — ID (на рисунку каналом є "Ukraine Now", ID якого записано у відповідну константу). Якщо дата повідомлення пізніше за порогову дату — воно додається до результатів, інакше пропускається. Результат роботи записується у файл bigchat.json.

Після завантаження, в повідомленнях є багато полів, які є технічними або не представляють для нас інтересу. Також існують повідомлення, які не мають тексту, а мають лише фотографії, стікери чи відео. Для очищення даних від таких повідомлень та полів, файл bigchat.json проходить додаткову обробку, яка наведена на Рис. 4. На вхід функції приходить об'єкт, що позначає окреме повідомлення, кожне поле якого перевіряється на те, чи є воно в списку потрібних атрибутів. На вихід дається очищене повідомлення та значення, що позначає наявність чи відсутність тексту.

Фінальний набір даних має такі поля: дата і час повідомлення, кількість переглядів, його текст, типи реакцій, кількість реакцій кожного типу відповідно.

В результаті роботи функції отримуємо набір даних, елемент якого вказаний на Рис. 5.

Щоб з даними було зручно працювати, було виконано їх переведення в табличний формат, приклад показано на Рис. 6.

Як видно з прикладу, для полегшення подальшої роботи потрібно кожній реакції зробити свою колонку з вказанням числа реакцій. Також для кожного повідомлення було проведено переведення абсолютної кількості ре-

```python
attr_to_keep = ['message_id', 'date', 'text', 'caption', 'views', 'reactions']
def clear_message_dict(message_dict):
    new_comment = dict()
    to_write = True
    is_none_text = False
    is_none_caption = False
    for key, val in message_dict.items():
        if key in attr_to_keep:
            if key == 'text':
                if val is None:
                    is_none_text = True
                else:
                    new_comment['msg_text'] = val
            elif key == 'caption':
                if val is None:
                    is_none_caption = True
                else:
                    new_comment['msg_text'] = val
            else:
                new_comment[key] = val
    if is_none_text and is_none_caption:
        to_write = False
    return new_comment, to_write
```

Рисунок 4 — Функція для очищення повідомлень


```
{
  "message_id": 53897,
  "date": "2022-09-21 19:51:27",
  "msg_text": "🐱‍🏍 Цей день настав! Кіт Степан повернувся до України
  "views": 116700,
  "reactions": [
    {
      "_client": null,
      "emoji": "😻",
      "count": 2649,
      "chosen": false
    },
    {
      "_client": null,
      "emoji": "🔥",
      "count": 431,
      "chosen": false
    },
```


Рисунок 5 — Приклад очищеного повідомлення

	message_id	date	msg_text	views	reactions
0	68765	2023-01-10 15:58:31	😢 Донька міністра оборони РФ Сергія Шойгу Ксен...	2011	[{'_client': None, 'emoji': '😠', 'count': 4, '...
1	68764	2023-01-10 15:48:51	⚡НАТО та ЄС підписали декларацію про співпрац...	22569	[{'_client': None, 'emoji': '👍', 'count': 363,...
2	68762	2023-01-10 15:46:28	🤬RuУ Мегіоні на Росії все стабільно: на вулиці...	27773	[{'_client': None, 'emoji': '👍', 'count': 473,...
3	68761	2023-01-10 15:32:42	🤬😡 У Києві під суд піде викладачка гімназії, ...	49711	[{'_client': None, 'emoji': '🤬', 'count': 1015...
4	68760	2023-01-10 15:26:41	😱🎥 З'явилось відео наслідків російської атаки ...	55128	[{'_client': None, 'emoji': '🤬', 'count': 508,...

Рисунок 6 — Дані в табличному форматі

```
def standardize_reaction(data):
  for index, val in data.reactions.items():
    s = str(val).replace('None', 'null')
    s = s.replace('False', 'false')
    s = s.replace('True', 'true')
    s = s.replace('\'', '\"')
    s = s.replace('\\', '')
    reaction = pd.read_json(s, orient = 'records', encoding_errors = 'backslashreplace')

    if reaction.empty:
      empty_reactions_indexes.append(i - 1)
      standardized_reaction = pd.DataFrame(data=0,
                                           index = ['count'], columns=recorded_emoji)
    else:
      reaction.drop(columns = ['_client', 'chosen'], inplace = True)
      reaction = reaction.T
      reaction.columns = reaction.iloc[0]
      reaction.drop('emoji', inplace = True)

      standardized_reaction = pd.DataFrame(data=0,
                                           index = ['count'], columns=recorded_emoji)
      for emj, cnt in reaction.iteritems():
        if emj in recorded_emoji_set:
          standardized_reaction[emj] = cnt
      standardized_reaction.loc['count'] /= standardized_reaction.loc['count'].sum()

    yield standardized_reaction
```

Рисунок 7 — Переведення реакцій в табличний формат та трансформація абсолютної кількості у відносну

Рисунок 8 — Типи реакцій, які використовувались під час обробки даних

акцій окремої категорії в відносну частку відносно всієї кількості реакцій. Така обробка робить різні оцінки настроїв більш наочними, а також спрощує подальші операції з даними та створення моделей. Реалізація вказана на Рис. 7.

Було видалено всі повідомлення, які з'явились менш ніж за добу до збору даних, оскільки даного часу мало, щоб повідомлення набрало приблизно середню кількість переглядів.

Множину типів реакцій звужено до типів, які вказані на Рис. 8.

	msg_text	views	clown_face	thumbs_up	thumbs_down	red_heart	fire	smiling_face_with_hearts	clapping_hands
0	Міноборони висміяло фейки РФ про "знищені" М14...	230149	0	0.855035	0.007056	0.074407	0.025657	0.003849	0.004811
1	❗Вибухи в Бєлгородській області\n\nГучно цьо...	222552	0	0.683468	0.003136	0.016801	0.173835	0.076165	0.008289
2	Challenger 2: що відомо про танк, який Велика ...	221416	0	0.800264	0	0.14064	0.026081	0	0.015847

Рисунок 9 — Дані, готові до аналізу

👍 , ❤ , 🔥 , 🥰 , 👏 , 😁 , 😱 , 🎉 , 🤩

👎 , 🤡 , 🤔 , 😱 , 🤬 , 😢 , 🤮 , 💩

Наведені типи реакцій, окрім смайла лиця клоуна — стандартний набір реакції для користувача Телеграм. Смайл лиця клоуна до 18 вересня 2022 р. був доступний тільки для преміум-користувачів, але все одно зустрічається дуже часто, тому його включено в множину типів реакцій.

Отримали табличний вигляд даних, готовий для інтелектуальної обробки, показано на Рис. 9.

2.2 Попередня обробка та тематичний аналіз повідомлень в Телеграм

Наступним етапом дослідження було перетворення отриманого набору даних в стандартизований формат, з якими могли б працювати алгоритми машинного навчанння.

Оскільки для даної роботи не була потрібна деталізована інформація про реакції користувачів, їх було поділено на дві категорії:

– позитивні;

– негативні.

Якщо відносна сумарна частка позитивних реакцій більша ніж 0.5, то повідомлення маркувалось як позитивне, і як негативне в іншому випадку, приклад наведено на Рис. 11. У сформованому наборі даних налічується 36647 позитивних та 26847 негативних повідомлень (Рис. 10).

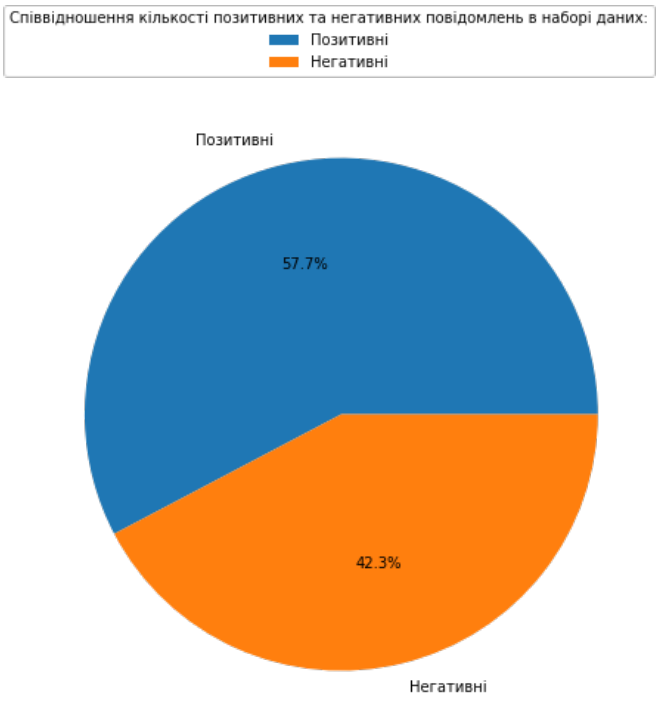

Рисунок 10 — Результати тематичного аналізу навчального набору даних

	msg_text	reaction_type
5	У Білорусі силовики розігнали організоване вол...	0
6	У Росії ще є багато ракет, які вона може випус...	0
7	Росіяни не розуміють, що війну, в принципі, пр...	1
8	Ситуація в Харкові станом на червня дуже важка...	0
9	Фронтмен Imagine Dragons Ден Рейнольдс присвят...	1
10	Оформити українські паспорти тепер можна в Пол...	1

Рисунок 11 — Приклад негативних і позитивних повідомлень. Число 1 позначає позитивне, число 0 - негативне

Також було проведено попередню обробку тексту, програмну реалізацію якої наведено на Рис. 12:

– вилучено всі смайли;
– видалено знаки пунктуації;
– видалено гіперпосилання на зовнішні ресурси;
– весь текст переведено в нижній регістр;
– видалено авторські підписи каналів.

Після цього було створено додаткові версії датасетів:

– з видаленням шумових слів. Було складено список слів, що не несуть смислового навантаження, як-от применники, сполучники, займенники та частки. Слова з даного списку були видалені з повідомлень;

```python
def preprocess_text(text):
  import re
  import string

  new_text = emoji.replace_emoji(text, replace = '')
  new_text = new_text.replace('\n', ' ')
  new_text = new_text.lower()
  new_text = re.sub(r'http\S+', '', new_text)

  # deleting all punctuation
  punct = string.punctuation.replace('\'', '')
  mappingtbl = new_text.maketrans({sign: ' ' for sign in punct + '—–«»№£€'})
  new_text = new_text.translate(mappingtbl)

  # removing all numbers
  new_text = re.sub(r' *\d+ *', ' ', new_text)

  # removing multiple spaces
  new_text = ' '.join(new_text.split())

  # removing channel signature
  new_text = new_text.replace('підписатися youtube тсн підтримати тсн ua', ' ')
  new_text = new_text.replace('підписатися youtube тсн підтримати', ' ')

  # removing multiple spaces the second time
  new_text = ' '.join(new_text.split())

  return new_text
```

Рисунок 12 — Попередня обробка тексту

– з лематизацією всіх слів (перетворення слів у їхню "первісну"форму, наприклад, в називний відмінок замість інших, однина замість множини, переведення з минулої та майбутньої форм в теперішню), з використанням бібліотеки Stanza([29]), яка надає інструменти для обробки природної мови, як-от синтаксичний розбір речення чи лематизація слів.

При використанні бібліотеки Stanza для обробки документу створюється Pipeline, який складається з процесорів, кожен з яких виконує свою задачу мовленнєвого аналізу. В даній роботі використані такі процесори:

– tokenize — розбиває вхідний текст на речення, які складаються з токенів, що позначають слова. Також вирізняються токени, які складаються з багатьох слів, але мають єдине значення — multi-word tokens. Наприклад, “20 000”, “т. д.” будуть вважатись multi-word tokens, але фразеологізми — ні;
– mwt(multi-word tokens) [38], [39] — проводить обробку multi-word tokens. Дані токени ставляться у відповідність кожному слову, з яких складаються;
– pos(part of speech) — кожному слову надає три властивості:
  – pos — частина мови;
  – xpos — частина мови відповідно до нотації tree-bank [40];

```
nlp = stanza.Pipeline('uk', processors = ['tokenize', 'mwt', 'pos', 'lemma'])
```

Рисунок 13 — Приклад створення Pipeline з додаванням потрібних процесорів

```
doc = nlp("У Києві запроваджені екстрені відключення електроенергії. Мешканців закликають зберігати спокій.")
```

```
print(doc.sentences[0].words)
```

```
[{
  "id": 1,
  "text": "У",
  "lemma": "у",
  "upos": "ADP",
  "xpos": "Spsl",
  "feats": "Case=Loc",
  "start_char": 0,
  "end_char": 1
}, {
  "id": 2,
  "text": "Києві",
  "lemma": "Київ",
  "upos": "PROPN",
  "xpos": "Npmsln",
  "feats": "Animacy=Inan|Case=Loc|Gender=Masc|Number=Sing",
  "start_char": 2,
  "end_char": 7
}, {
```

Рисунок 14 — Приклад обробки речення за допомогою Stanza

```
list_all_words = [remove_sm(word) for msg in data.msg_text for word in msg.split()]
set_all_words = set(list_all_words)
len(set_all_words)
```

Рисунок 15 — Створення масиву всіх слів та їх обробка за допомогою Pipeline nlp

  - feats(universal morphological features) — наявні морфологічні ознаки. Наприклад, рід, однина чи множина, пасивний чи активний стан дієслова та інші;
- lemma — проводить лематизацію слова.

У даній роботі для проведення лематизації спочатку був створений Pipeline, який показано на Рис. 13.

Приклад використання вказано на Рис. 14.

Було створено словник переведення слів в лематизовану форму, оскільки обробка кожного повідомлення бібліотекою Stanza займає багато часу й обчислювальних ресурсів. Лематизацію даних було виконано так: створення словника лематизації та збереження його у файл, потім, за рахунок його використання, швидка лематизація при побудові моделей.

Процес створення словнику показаний на Рис. 15 - 16.

В подальшому даний словник можна використовувати так, як вказано на Рис. 17.

```
word_to_lemma = {}
for doc in out_docs:
    for sentence in doc.sentences:
        for word in sentence.words:
            word_to_lemma[word.text] = word.lemma

df_word_to_lemma = pd.DataFrame.from_dict(word_to_lemma, orient='index')
df_word_to_lemma.to_csv('word2lem.csv')
```

Рисунок 16 — Створення словнику лематизації з попередньо обробленого тексту, та збереження словника у файл

```
word_to_lemma = pd.read_csv(f, index_col = 0)

word_to_lemma = word_to_lemma.to_dict(orient='dict')
word_to_lemma = word_to_lemma['0']

def lemmatize(text):
  return ' '.join([word_to_lemma.get(w, '') for w in text.split()])

data.msg_text = data.msg_text.apply(lemmatize)
```

Рисунок 17 — Лематизація повідомлень на основі словника

В наступних розділах буде розглянуто вплив видалення шумових слів та лематизації на точність прогнозування моделей.

В рамках початкового аналізу отриманого набору даних, було отримано word cloud — графіки, в яких розмір шрифту слова прямопропорційно залежить від його вживання. Це було зроблено для слів, які є характерними для новин, які ми промаркували як позитивні (Рис. 18 -19) та негативні (Рис. 20 - 21). Також наведено word cloud для лематизованого варіанту повідомлень.

2.3 Прогнозування реакцій на Телеграм-повідомлення

Для переведення повідомлень у числові вектори були застосовані методи:

– “Торба слів”;
– TF-IDF;
– Glove;
– вивід мережі BERT.

Рисунок 18 — Word cloud часто використовуваних слів в позитивних повідомленнях без використання лематизації

Рисунок 19 — Word cloud часто використовуваних слів в позитивних повідомленнях з використанням лематизації

Рисунок 20 — Word cloud часто використовуваних слів в негативних повідомленнях без використання лематизації

Рисунок 21 — Word cloud часто використовуваних слів в негативних повідомленнях з використанням лематизації

Назва	F-міра без видалення шумових слів	F-міра з видаленням шумових слів	F-міра з використанням лематизації
nb_cv	0.843	0.841	0.825
nb_tf_idf	0.856	0.853	0.846
svc_cv	0.856	0.853	0.858
svc_tf_idf	0.855	0.850	0.858

Таблиця 1 — Таблиця F-мір для наївного баєса та методу опорних векторів

Для прогнозування реакцій були використані:

- Наївний баєсівський класифікатор;
- Метод опорних векторів;
- LSTM;
- GRU;
- Нейронну мережу на базі BERT.

2.3.1 Наївний баєсівський класифікатор та метод опорних векторів

Для наївного баєсівського класифікатора та методу опорних векторів навчання проводилось на трьох версіях датасету: з основною обробкою, описаною раніше, з додатковою лематизацією та з додатковим видвленням шумових слів. Векторизація слів для цих алгоритмів була проведена за методами “Торба слів” та TF-IDF.

Отримані F-міри на випробувальному датасеті для кожної моделі вказані в Таб. 1:

У Таб. 1 такі використовуються наступні скорочення:

- префікси в назвах: **nb** — наївний баєсівський класифікатор; **svc** — метод опорних векторів;
- суфікси в назвах: **cv** — переведення тексту в числовий вид за допомогою ”торби слів”; **tf_idf** — переведення тексту в числовий вид за допомогою методики TF-IDF.

Можна побачити, що лематизація погіршила точність моделей, що використовують наївний баєсівський класифікатор, особливо при використанні "торби слів" як інструменту переведення тексту в число. Видалення шумових слів має незначний вплив на точність моделей. Найкраща модель з отриманих - метод опорних векторів, тренований на лематизованих повідомленнях, має F-міру 0.858.

2.3.2 LSTM і GRU

Для LSTM та GRU були використані вкладення Glove. Вкладення взяті з сайту спільноти lang-uk ([31]), яка займається підтримкою і розвитком проектів по збору українських корпусів. Всього було було протестовано 4 види вкладень, які поділені за характером та регістром тексту, на якому вони були навчені(в дужках вказані скорочення, які використані в таблиці порівнянь для позначення конкретного виду вкладення):

- Характер даних:
  - новини(news);
  - уберкорпус — збірка текстів з українських періодичних видань загальним розміром більше 6 GB(ubercorpus).
- Регістр:
  - без зміни регістру джерела(cased);
  - з переведенням всіх слів у нижній регістр(uncased).

На базі даних вкладень та архітектур LSTM і GRU було побудовано нейромережеві моделі, програмну реалізацію та архітектуру яких показано на Рис. 22-23. На них вказано реалізацію моделей на базі архітектури LSTM. Для моделей на базі архітектури GRU архітектура така ж, з відповідною заміною блоків LSTM на блоки GRU та зміненими значеннями регуляризації. У Таб. 2 вказано F-міри результати тестування моделей на різних вкладеннях на випробувальному датасеті.

Бачимо невелику різницю в точності моделей в залежності від використаних вкладень. Також найкраща отримана F-міра — 0.889.

```
import tensorflow_addons as tfa

model = Sequential()
#Non-trainable embeddidng layer
model.add(Embedding(max_features, output_dim=embed_size, weights=[embedding_matrix],
                    input_length=maxlen, trainable=False))
#LSTM
model.add(LSTM(units=128 , return_sequences = True , recurrent_dropout = 0.2 ,
               dropout = 0.15))
model.add(LSTM(units=64, recurrent_dropout = 0.2 , dropout = 0.17))
model.add(Dense(units = 32 , activation = 'relu'))
model.add(Dense(1, activation='sigmoid'))
model.compile(optimizer=keras.optimizers.Adam(learning_rate = 0.005),
              loss='binary_crossentropy',
              metrics=['accuracy', tfa.metrics.F1Score(num_classes = 1, threshold = 0.5)])
```

Рисунок 22 — Код для створення моделі LSTM

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, 300, 300)          6000000

 lstm (LSTM)                 (None, 300, 128)          219648

 lstm_1 (LSTM)               (None, 64)                49408

 dense (Dense)               (None, 32)                2080

 dense_1 (Dense)             (None, 1)                 33

=================================================================
Total params: 6,271,169
Trainable params: 271,169
Non-trainable params: 6,000,000
_________________________________________________________________
```

Рисунок 23 — Архітектура моделі LSTM

Назва	news_uncased	news_cased	ubercorpus_uncased	ubercorpus_cased
LSTM	0.889	0.883	0.885	0.883
GRU	0.885	0.886	0.887	0.881

Таблиця 2 — Таблиця F-мір для LSTM та GRU

```
def __init__(self, hidden_size1, hidden_size2, num_classes):
    super(BertClassifier, self).__init__()
    self.base_m = base_model
    self.dropout = nn.Dropout(0.55)
    self.fc1 = nn.Linear(hidden_size1, hidden_size2)
    self.fc2 = nn.Linear(hidden_size2, num_classes)
    self.relu = nn.ReLU()
    self.softmax = nn.LogSoftmax(dim=1)
```

Рисунок 24 — Складові нейронної мережі класифікації

```
def forward(self, input_id, mask):
    seq_output = self.base_m(input_id, attention_mask=mask, return_dict=False)

    text_emb = self.mean_pooling(seq_output, mask)

    final_layer = self.fc1(text_emb)

    final_layer = self.relu(final_layer)

    final_layer = self.dropout(final_layer)

    final_layer = self.fc2(final_layer)

    final_layer = self.softmax(final_layer)

    return final_layer
```

Рисунок 25 — Порядок обчислення виводу

2.3.3 BERT

Було обрано дві моделі моделі BERT, які натреновані на великих обсягах даних та є у відкритому доступі:

- bert-base-cased(bert-base) — модель, що тренована на збірці англомовних книг BookCorpus та статтях англомовної Вікіпедії
- multi-cased-bert-base(bert-uk) — натренована на всіх статтях Вікіпедії для кожної мови, а також на датасеті SQuAD 2.0 для української мови.

Вхідний текст передається в натреновану модель BERT, на базі вихідних даних якої класифікацію робить нейронна мережа, складові якої вказані на Рис. 24. Порядок обчислення виводу вказаний на Рис. 25.

Розглянемо детальніше процес роботи моделі. На вхід моделі класифікації даються масиви input_id та attention_mask, які передають моделі BERT. Архітектура BERT передбачає існування пронумерованого словника, в яко-

```
sample_text = 'оголошена повітряна тривога будь ласка перейдіть до укриттів'
tokens = tokenizer.tokenize(sample_text)
token_ids = tokenizer.convert_tokens_to_ids(tokens)

print(f'Original sentence: {sample_text}')
print(f'Tokenized sentence: {tokens}')
print(f'Token IDs: {token_ids}')
```

```
Original sentence: оголошена повітряна тривога будь ласка перейдіть до укриттів
Tokenized sentence: ['о', '##гол', '##ош', '##ена', 'повітря', '##на', 'три', '##во', '##га', 'будь', 'ла',
Token IDs: [555, 85952, 43155, 12868, 90321, 10409, 13251, 15275, 11347, 38092, 26670, 12184, 61381, 36694,
```

Рисунок 26 — Приклад токенізації

```
def mean_pooling(self, model_output, attention_mask):
    token_embeddings = model_output[0]  #First element of model_output contains all token embeddings
    input_mask_expanded = attention_mask.unsqueeze(-1).expand(token_embeddings.size()).float()

    return torch.sum(token_embeddings * input_mask_expanded, 1) / torch.clamp(input_mask_expanded.sum(1), min=1e-9)
```

Рисунок 27 — Реалізація функції mean_pooling

му записані найчастіше вживані слова та частини слів. Після надання тексту для обробки моделі, кожне слово з тексту відображається в номер даного слова в словнику. Якщо такого слова в словнику немає, воно розбивається на наявні в словнику частини. В результаті ми отримуємо масив номерів слів у словнику, який і є масивом input_id. Даний процес називається токенізацією, а пара "слово в словнику — його номер— токеном. Приклад процесу показаний на Рис. 26. Також, моделі BERT приймають фіксовану довжину масиву input_id, у нашому випадку це 512 токенів. В той же час, вхідна послідовність може бути як і меншого, так і більшого розміру. У випадку, коли вхідна послідовність довша 512 токенів, надлишкові токени просто видаляються. У випадку, коли коротша - в послідовність додаються така кількість спеціальних токенів PAD, щоб вона мала довжину 512 токенів. Масив attention_mask позначає одиницями ті токени, які мають сенс і нулями токени PAD.

Після обробки вхідної послідовності за допопмогою BERT, ми отримуємо вкладення для всіх токенів — seq_output. За допомогою агрегувальної функції mean_pooling, реалізація якої показана на Рис. 27, знаходиться середнє значення між вкладеннями. Дана операція має на меті об'єднати значення всіх токенів, щоб в результаті отримати вкладення всієї послідовності.

Отримане вкладення передається в нейронну мережу, яка на виході за допомогою функції Softmax дає передбачення щодо класу повідомлення — позитивний чи негативний. Ціль навчання — на базі виводу моделі BERT навчити нейронну мережу правильно передбачати тип повідомлення.

В результаті були отримані значення F-мір:

– bert-uk - 0.813

– bert-base - 0.819

2.4 Висновки до розділу 2

З хмар слів видно, що слова, які самі по собі мають негативне забарвлення, опиняються в повідомленнях з позитивним значенням. Це вказує на існування випадків, де наївний баєсівський класифікатор з використанням "торби слів" може не впоратись із класифікацією та підтверджується зменшенням точності наївного баєсівського класифікатора при тренуванні на лематизованому наборі даних.

Бачимо, що отримана F-міра для алгоритму LSTM з використанням Glove на 0.03 більша, ніж та, що отримана за допомогою методу опорних векторів з використанням лематизації.

Також бачимо, що точність BERT-моделей є нижче показників як і класичних моделей наївного баєсівського класифікатора та методу опорних векторів, так і рекурентних нейронних мереж LSTM та GRU. Це вказує на необхідність додаткового тренування моделей BERT для української мови для кращого відображення її властивостей.

ВИСНОВКИ

У даній роботі представлені результати дослідження можливостей алгоритмів інтелектуального аналізу повідомлень та новин в Телеграм. Показано, що певні особливості цього медіа дозволяють ширше використовувати можливості вже існуючих методик збирання даних та алгоритмів обробки природної мови.

Також варто відмітити, що дана робота присвячена саме аналізу української природної мови, для якої на даний момент проведено не так багато досліджень в порівнянні з іншими, наприклад, англійською чи російською. Тому дана робота базувалась не тільки вже на наявних наборах даних (наприклад, словник лематизації слів, список шумових слів), але і сама є доволі перспективним кроком вперед в цьому напрямку.

Основною задачею для подальших досліджень є створення більш точних моделей для оцінки тонального звучання тексту, що може допомогти в написанні текстів в повсякденному житті, а також за допомогою моделі проводити аналітику повідомлень звичайних користувачів для оцінки громадського настрою.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Mondal A., Dey M., Das D., Nagpal S., Garda K. Chatbot: An automated conversation system for the educational domain. *2018 International Joint Symposium on Artificial Intelligence and Natural Language Processing (iSAI-NLP)*. 2018. P. 1–5.

2. Gunawan T. S., Babiker A. B. F., Ismail N., Effendi M. R. *Development of Intelligent Telegram Chatbot Using Natural Language Processing. In 2021 7th International Conference on Wireless and Telematics (ICWT)*. 2021. P. 1–5

3. Karimpour D., Chahooki M. A. Z., Hashemi A. User recommendation based on Hybrid filtering in Telegram messenger. *26th International Computer Conference, Computer Society of Iran (CSICC)*. 2021. P. 1–7.

4. Hashemi A., Zare Chahooki M. A. Telegram group quality measurement by user behavior analysis. *Social Network Analysis and Mining*. 2019. 9(1). P. 1–12.

5. Karimpour D., Zare Chahooki M. A., Hashemi, A. User recommendation in Telegram messenger by graph analysis and mathematical modeling of users' behavior. *Journal of Information and Communication Technology*. 2021. 49(49). 151.

6. Eichstaedt J. C., Smith R. J., Merchant R. M., Ungar L. H., Crutchley P., Preo?iuc-Pietro D., Schwartz H. A. Facebook language predicts depression in medical records. *Proceedings of the National Academy of Sciences*. 2018. 115(44). P. 11203–11208.

7. Kachamas P., Akkaradamrongrat S., Sinthupinyo S., Chandrachai A. Application of artificial intelligent in the prediction of consumer behavior from Facebook posts analysis. *International Journal of Machine Learning and Computing*. 2019. 9(1). P. 91–97.

8. Han B., Cook P., Baldwin T. Text-based twitter user geolocation prediction. *Journal of Artificial Intelligence Research*. 2014. 49. P. 451–500.

9. Jordan S. E., Hovet S. E., Fung I. C. H., Liang H., Fu K. W., Tse Z. T. H. Using Twitter for public health surveillance from monitoring and prediction to public response. *Data*. 2018. 4(1). P. 6.

10. Essien A., Petrounias I., Sampaio P., Sampaio S. A deep-learning model for urban traffic flow prediction with traffic events mined from twitter. *World Wide Web*. 2021. 24(4). P. 1345–1368.

11. Shevchuk I.M., Nakonechnyi O.G., Kapustian O.A., Kosukha O.Yu., Loseva M.V. Intellectual analysis of reactions to news based on data from Telegram channels. *XXXVII International Conference PROBLEMS OF DECISION MAKING UNDER UNCERTAINTIES (PDMU-2022)*, November 23 - 25, 2022, Sheki-Lankaran, Republic of Azerbaijan. P. 103–104.

12. Косуха О.Ю., Шевчук Ю.М. Розробка системи для сентиментального аналізу новин на основі даних Телеграм каналів. *Матеріали XXI Міжнародної науково-практичної конференції «Шевченківська весна - 2023»*, 14 квітня 2023 р., м. Київ, Україна. С.90–91.

13. Наконечний О.Г., Капустян О.А., Шевчук Ю.М., Лосєва М.В., Косуха О.Ю. Інтелектуальна система аналізу реакцій на новини на основі даних Телеграм- каналів. *Вісник Київського національного університету імені Тараса Шевченка. Серія: фізико-математичні науки.* 2022. No3. с.55—61.

14. Telegram APIs. Режим доступу: https://core.telegram.org/

15. Телеграм-канал ТСН. Режим доступу: https://t.me/TCH_channel

16. Код програмної реалізації. Режим доступу: https://github.com/ KosukhaOlexandr/ reactions_prediction/blob/main/clear_dataset.py

17. Mosteller F., Wallace D. L. Inference and disputed authorship: The Federalist. *Stanford Univ Center for the Study*. 2007.

18. Козак Є. Б. Принципи впровадження моделей машинного навчання у сфері інтелектуального обслуговування промислового обладнання. *Таврійський науковий вісник. Серія: Технічні науки*. 2021. (3). С. 19–28.

19. Білецький Т. П., Федасюк Д. В. Прогнозування дефектів у програмному забезпеченні алгоритмами глибинного навчання CNN та RNN. *Науковий вісник НЛТУ України.* 2021. 31(2). С. 114–120.

20. Ahmad F., Tang X. W., Qiu J. N., Wroblewski P., Ahmad M., Jamil I. Prediction of slope stability using Tree Augmented Naive-Bayes classifier: Modeling and performance evaluation. *Math. Biosci. Eng.* 2022. 19. P. 4526–4546.

21. Kewsuwun N., Kajornkasirat S. A sentiment analysis model of agritech startup on Facebook comments using naive Bayes classifier. *International Journal of Electrical & Computer Engineering* (2088-8708). 2022. 12(3).

22. Cortes C., Vapnik V. Support-Vector Networks. *Machine Leaming.* 1995. 20. P.273–297.

23. Jose C., Goyal P., Aggrwal P., Varma M. Local deep kernel learning for efficient non-linear svm prediction. *In International conference on machine learning.* 2013. P. 486–494.

24. Покідін Д. Економетрична модель Національного банку України для оцінки кредитного ризику банку та альтернативний метод опорних векторів. *Вісник Національного банку України.* 2015. 234. С. 53.

25. Верлань А. І., Олексій А. О. Огляд та порівняння методів машинного навчання для розпізнавання гідроакустичних сигналів. *Інфокомунікаційні та комп'ютерні технології.* 2022. 1(03). С. 296–306.

26. Ramasamy L. K., Kadry S., Nam Y., Meqdad M. N. Performance analysis of sentiments in Twitter dataset using SVM models. *Int. J. Electr. Comput. Eng.* 2021. 11(3). P. 2275–2284.

27. Pennington J., Socher R., Manning C. Glove: Global vectors for word representation. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP).* 2014.

28. Cho K., van Merrienboer B., Bahdanau, D. Bengio, Y. On the properties of neural machine translation: Encoder–decoder approaches. *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation.* 2014.

29. Qi P., Zhang Y., Zhang Y., Bolton J., Manning C. D. Stanza: A Python natural language processing toolkit for many human languages. *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*. 2020.

30. Hochreiter S., Schmidhuber J. Long short-term memory. *Neural Computation* 1997. 9(8). P. 1735–1780.

31. Векторні вкладення слів. Режим доступу: https://lang.org.ua/uk/models/

32. Bahdanau D., Cho K., Bengio Y. Neural machine translation by jointly learning to align and translate. 2014.

33. Silver D., Huang A., Maddison C. et al. Mastering the game of Go with deep neural networks and tree search. *Nature*. 2016. 529, P. 484-489.

34. Luong Thang, et al. Effective Approaches to Attention-Based Neural Machine Translation. *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. 2015.

35. Vaswani A., Shazeer N., Parmar N., Uszkoreit J., Jones L., Gomez A. N., Kaiser L., Polosukhin I. Attention Is All You Need. 2017.

36. Devlin J., Chang M., Lee K., Toutanova K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. 2018.

37. Фреймворк Pyrogram. Режим доступу: https://docs.pyrogram.org/

38. Multi-word tokens. Режим доступу: https://universaldependencies.org/u/overview/tokenization.html

39. Kahane S., et al. Multi-word annotation in syntactic treebanks: Propositions for Universal Dependencies". *Proceedings of the 16th international conference on Treebanks and Linguistic Theories (TLT), Prague*. 2018.

40. Taylor A., et al. The Penn Treebank: An Overview. *Treebanks*. 2003. P. 5–22.