Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет інформаційно-аналітичних технологій та менеджменту
(повна назва)

Кафедра прикладної математики
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

Математичні моделі та методи класифікації повідомлень про стихійні лиха в соціальних мережах
(тема)

Виконав:
студент _2_ курсу, групи _ПМм-20-1
Житник О.В.
(прізвище, ініціали)
Спеціальність
113 Прикладна математика
(код і повна назва спеціальності)
Тип програми _освітньо-професійна
(освітньо-професійна або освітньо-наукова)
Освітня програма
Прикладна математика
(повна назва освітньої програми)
Керівник доц. Єсілевський В.С.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри ПМ _____ Тевяшев А.Д.
(підпис) (прізвище, ініціали)

2021 р.

Харківський національний університет радіоелектроніки

Факультет _інформаційно-аналітичних технологій та менеджменту_____
Кафедра _прикладної математики_____
Рівень вищої освіти _другий (магістерський)_____
Спеціальність _113 Прикладна математика_____
(код і повна назва)
Тип програми _освітньо-професійна_____
(освітньо-професійна або освітньо-наукова)
Освітня програма _Прикладна математика_____
(повна назва)

ЗАТВЕРДЖУЮ:

Зав. кафедри ПМ _____
(підпис)
“____” _____ 2021 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові _Житнику Олександру Володимировичу_____
(прізвище, ім’я, по батькові)

1. Тема роботи _математичні моделі та методи класифікації повідомлень про стихійні лиха в соціальних мережах_____

затверджена наказом по університету від 05 листопада 2021 р. № 1641 Ст

2. Термін подання студентом роботи до екзаменаційної комісії 10 грудня 2021 р.

3. Вихідні дані до роботи повідомлення у соціальних мережах

4. Перелік питань, що потрібно опрацювати в роботі
1. Аналіз предметної області
2. Вибір і обґрунтування методу розв’язання
3. Програмна реалізація
4. Результати обчислювального експерименту
5. Аналіз можливих застосувань

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій

1. Актуальність теми роботи
2. Постановка задачі
3. Аналіз предметної області
4. Метод чисельного аналізу
5. Результати обчислювального експерименту

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Підбір та вивчення технічної літератури за темою роботи	8 – 14 листопада 2021 р.	виконано
2	Вибір та обґрунтування методу	15 – 21 листопада 2021 р.	виконано
3	Розробка алгоритму і програми	22 – 28 листопада 2021 р.	виконано
4	Проведення аналітичних досліджень та розрахунків	29 листопада – 5 грудня 2021 р.	виконано
5	Робота над текстом пояснювальної записки	6 – 9 грудня 2021 р.	виконано
6	Представлення роботи на рецензію в ЕК	10 грудня 2021 р.	виконано

Дата видачі завдання _8_ листопада 2021 р.

Студент _____
(підпис)

Керівник роботи _____ доц. Єсілевський В.С.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка: 56 с., 28 рис., 1 дод., 19 джерел.

КЛАСИФІКАЦІЯ, ПОВІДОМЛЕННЯ, ВЕКТОРІЗАЦІЯ СЛІВ, МАШИННЕ НАВЧАННЯ, КОМІРКА, СТИХІЙНІ ЛИХА, WORDPIECE, LOGISTIC REGRETION, DICISION TREE , LSTM, GRU, BERT, RNN.

Об’єкт дослідження – текстові повідомлення у соціальних мережах.

Мета роботи – застосування методів машинного навчання та векторізації тексту для класифікації повідомлень у соціальних мережах.

Методи дослідження – представлення двунаправленного енкодера від трансформаторів, нейроні мережі, рекурентні нейронні мережи, керований рекурентний блок, комірка довгої та короткої пам’яті, трансформери, логістична регресія, випадкові ліси.

У роботі реалізовано покроковий алгоритм класифікації тексту на два класи, катастрофа чи ні. Було проведено аналіз можливих алгоритмів для вирішення поставленої проблеми, а саме, рекурентні нейронні мережі, керований рекурентний блок, комірка довгої та короткої пам’яті та представлення двунаправленного енкодера від трансформаторів. Алгоритм реалізовано на мові програмування Python 3 за допомогою бібліотеки Transformer та Keras, які дозволяють легко будувати нейронні мережі.

Також наведені приклади інших алгоритмів, які помилково класифікують текст. Запропонований спосіб розв’язання проблеми може використовуватися у різних сферах життя: фільтрація повідомлень у соціальних мережах, системи безпеки тощо.

.

ABSTRACT

Introductory note: 56 pages, 28 figures, 1 appendix, 19 sources.

CLASSIFICATION, MESSAGE, VECTORIZATION OF WORDS, MACHINE LEARNING, CELL, NATURAL DISASTERS, WORDPIECE, LOGISTIC REGRETION, DICISION TREE, LSTM, GRU, BERT, RNN.

Object of research – text messages on social networks.

Purpose of work – application of machine learning and vectorization of the text for the classification of messages on social networks.

Methods of research – representation of bidirectional encoder from transformers, neural networks, recurrent neural networks, gated recurrent unit, long short-term memory, transformers, logistic regression, random forest.

The step-by-step algorithm of text classification into two classes, catastrophe or not, is implemented in the work. An analysis of possible algorithms to solve this problem, namely, recurrent neural networks, gated recurrent unit, long short-term memory and the representation of a bidirectional encoder from transformers. The algorithm is implemented in the Python 3 programming language using the Transformer and Keras libraries, which allow you to easily build neural networks.

There are also examples of other algorithms that badly classify text. The proposed solution can be used in various spheres of life: filtering messages on social networks, security systems, etc.

ЗМІСТ

ВСТУП

Актуальність теми. Соціальні медіа забезпечують загальну платформу для обміну особистим досвідом та спілкування з іншими людьми. Люди часто публікують свій життєвий досвід, місцеві новини та події в соціальних мережах, щоб інформувати інших. Багато рятувальних агентств регулярно відстежують цей тип даних, щоб виявити катастрофу та зменшити ризик для життя. Однак люди не можуть вручну перевірити масу даних і виявити катастрофу у режимі реального часу [1]. З цією метою було запропоновано багато дослідницьких робіт, щоб представити слова у зрозумілих машиною репрезентаціях та застосувати методи машинного навчання для визначення змісту тексту. Попередні методи дослідження забезпечують єдине представлення слова з даного документа. Однак останній прогресивний метод контекстного представлення (BERT) створює різні вектори для одного слова у різних контекстах. Представлення BERT успішно використовуються в різних завданнях обробки природної мови (NLP), але немає конкретного аналізу того, наскільки ці представлення корисні в аналізі повідомлень типу катастрофи.

Мета і завдання кваліфікаційної роботи. Метою кваліфікаційної роботи є аналіз основних методів векторізації та вибір методу для класифікації тексту. Для досягнення поставленої мети необхідно виконати наступні завдання:

– провести огляд і аналіз сучасного стану задачі класифікації тексту;

– провести огляд сучасних методів обробки тексту;

– отримання нового представлення повідомлень, використовуючи BERT;

– побудувати нейрону мережу та натренувати на оброблених даних;

Об'єкт дослідження є текстові повідомлення у соціальних мережах.

Предмет дослідження є математична модель класифікації текстових повідомлень

Методи дослідження. У кваліфікаційній роботі використовуються методи векторізації слів за допомогою BERT та подальша класифікація за допомогою класичних методів машинного та глибокого навчання.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧ ДОСЛІДЖЕННЯ

1.1 Математичні моделі класифікації тексту

1.1.1 Класична рекурентна мережа (RNN)

Перед тим як перейти до рекурентних нейронних мереж, необхідно сказати, чому саме требо використовувати саме їх, а не логістичну регресію, багатошаровий персептрон чи метод опорних векторів. Ці моделі страждають від неможливості зберігати інформацію протягом всієї послідовності і обробляли кожен вхід незалежно. По суті, відсутність елемента пам'яті не дозволяє цим моделям добре виконувати завдання моделювання послідовності.

Рекурентні нейронні мережі (RNN) намагаються виправити цей недолік, запровадивши цикли в мережі [2], що дозволило зберегти інформацію.

Як показано на рис. 1.1 та у рівнянні (1.1), поточний прихований стан нейрона, можна моделювати як функцію прихованого стану попереднього нейрона h_{t-1}, поточного входу x_t, вагових матриць U, W і зсуву b. Ці ваги мережі потім оновлюються за допомогою навчального алгоритму під назвою Backpropagation Through Time (BPTT).

$$h_t = \phi\left(Wh_{t-1} + Ux_t + b\right), \tag{1.1}$$

де W – матриця вагів прихованого стану;

h – прихований стан;

U – матриця вагів системи;

ϕ – гіперболічний тангенс;

b – зсув.

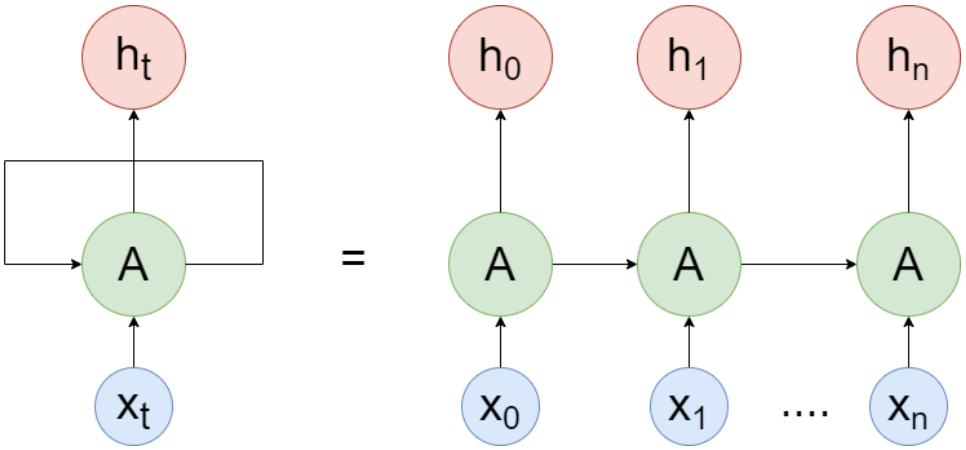

Рисунок 1.1 – Приклад рекурентної мережі

BPTT, по суті, є алгоритмом зворотного поширення з деякими модифікаціями. Мережа поширюється для кожного кроку часу – операція, яку часто називають розгортанням RNN. Параметри нейронної мережі залишаються незмінними протягом всієї операції розгортання RNN. Відповідні помилки щодо прогнозованого результату та основної істини потім обчислюються для кожного кроку часу. Потім розраховуються й накопичуються градієнти похибки щодо всіх параметрів за допомогою алгоритму зворотного поширення. Лише після завершення розгортання всі параметри RNN оновлюються за допомогою цього накопиченого градієнта помилки.

Класична RNNs була успішною в широкому спектрі завдань, включаючи розпізнавання мовлення, переклад і мовне моделювання. Незважаючи на свій початковий успіх, класичні RNN змогли моделювати лише короткострокові залежності. Їм не вдалося змоделювати довгострокові залежності, насамперед тому, що інформацію часто «забувають» після того, як активації блоку кілька разів помножувалися на невеликі числа.

Крім того, вони страждали від різних проблем під час навчання, таких як проблема зникаючого градієнта (градієнт помилки, який використовується для зменшення ваги до дуже низьких значень) і проблема градієнта, що вибухає. Таким чином, успішне навчання та застосування класичних RNN було складним завданням.

1.1.2 Комірка довгої та короткої пам’яті (LSTM)

Проблема «довгострокової» залежності, з якою стикалися попередні рекурентні нейронні мережі, була вирішена шляхом розробки особливого типу архітектури RNN, який називається LSTM (Long Short Term Memory) [3]. Вони були розроблені для відстеження інформації протягом тривалої кількості часових кроків. LSTM мають загальну ланцюжкову архітектуру, подібну до RNN, але суттєвою відмінністю є покращення внутрішньої структури вузла. Хоча вузол у RNN складається з одного нейронного шару, у LSTM є чотири шари, пов’язані однозначно, як показано на рис. 1.2.

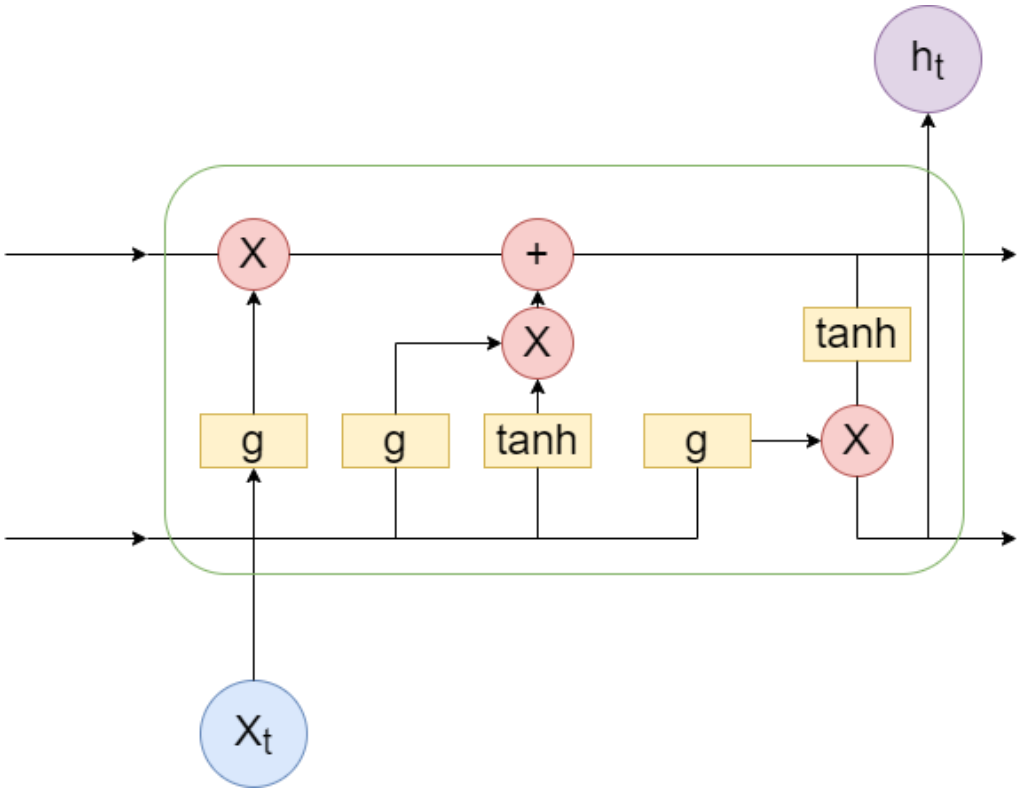

Рисунок 1.2 – Комірка LSTM

Ключовою особливістю LSTM є наявність носія інформації у верхній частині, відоме як «стан комірки». Виявляється корисним переносити інформацію на більші відстані, при цьому на кожному вузлу виконуються лише незначні лінійні операції. Можливість додавати або видаляти певну інформацію про стан

комірки в кожному вузлу, забезпечується структурами, які називаються комірками. LSTM має три такі комірки, кожен із яких складається із шару сигмовидної нейронної мережі та матричного множника. Використовуючи наступні рівняння (1.2) – (1.7), отримаємо усі необхідні розрахунки мережі.

$$f_t = \sigma\ W_f \cdot \left[h_{t-1}, x_t\right] + b_f\ , \tag{1.2}$$

$$i_i = \sigma\ W_i \cdot \left[h_{t-1}, x_t\right] + b_i\ , \tag{1.3}$$

$$\tilde{C}_t = \tanh\ W_C \cdot \left[h_{t-1}, x_t\right] + b_C\ , \tag{1.4}$$

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t, \tag{1.5}$$

$$o_t = \sigma\ W_o \cdot \left[h_{t-1}, x_t\right] + b_o\ , \tag{1.6}$$

$$h_t = o_t * \tanh\ C_t\ , \tag{1.7}$$

де f_t – комірка забуття;

i_t – вхідна комірки;

o_t – вихідна комірка;

W, b – ваги та зсув;

C_{t-1} – попередній стан комірки;

C_t – новий стан.

Перевага використання LSTM полягає в тому, що вони пропонують більше контролю в мережі, ніж звичайні рекурентні мережі. Система є більш складною і може зберігати інформацію протягом більш тривалого часу. Однак додані комірки призводять до більших вимог до обчислень, і, отже, LSTM, як правило, повільніші.

1.1.3 Керований рекуретний блок (GRU)

Керований рекурентний блок [4] або GRU, як на рис. 1.3, є спрощений варіант LSTM, призначений для зменшення проблем з обчисленням. Комірка забуття та вхідна комірка у LSTM об’єднані в один «шлюз оновлення». Стан комірки та приховані стани також об’єднуються разом і обчислюються за допомогою одної «комірки скидання».

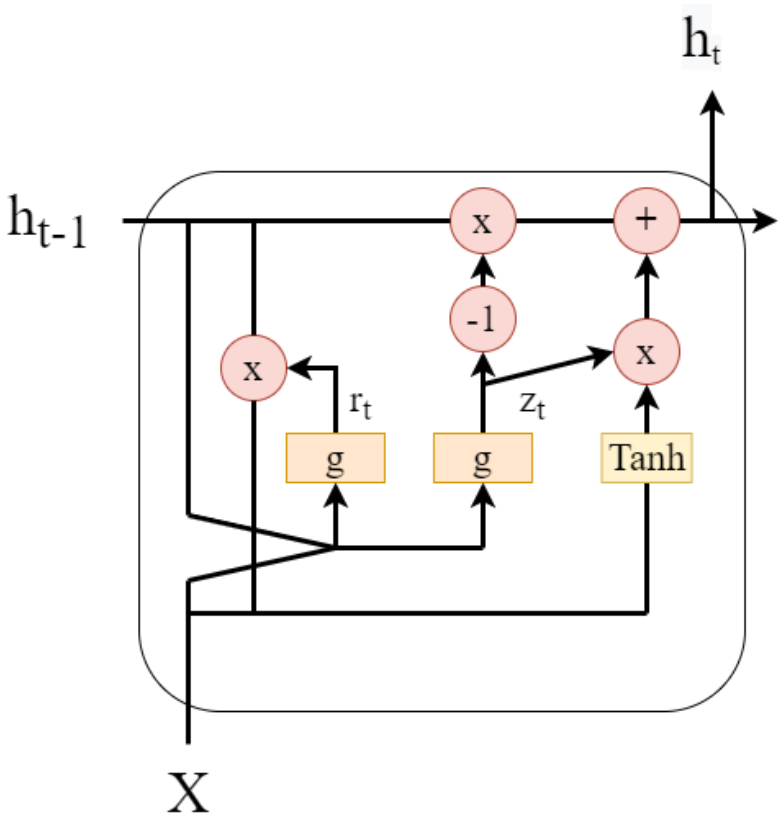

Рисунок 1.3 – Комірка GRU

Шлюз оновлення $z(t)$ відповідає за визначення обсягу попередньої інформації (попередніх кроків часу), яку необхідно передати у наступний стан. Тут x_t є вхідним вектором, який обслуговується в мережевому блоці. Ці значення множаться на матрицю вагів параметрів W_z. Значення t у h_{t-1} означає, що він містить інформацію попереднього блоку, і він помножений на його вагу. Далі значення цих параметрів додаються і передаються через сигмоїдну функцію активації. Стани комірок мають наступний вигляд:

$$z_t = \sigma\ W_z \cdot \left[h_{t-1}, x_t\right], \tag{1.8}$$

$$r_t = \sigma\ W_r \cdot \left[h_{t-1}, x_t\right], \tag{1.9}$$

$$\tilde{h}_t = \tanh\ W_o \cdot \left[r_t * h_{t-1}, x_t\right], \tag{1.10}$$

$$h_t = \ 1 - z_t\ \odot h_{t-1} + z_t \odot \tilde{h}_t. \tag{1.11}$$

Шлюз оновлення $z(t)$ відповідає за визначення обсягу попередньої інформації (попередніх кроків часу), яку необхідно передати у наступний стан. Тут x_t є вхідним вектором, який обслуговується в мережевому блоці. Ці значення множаться на матрицю вагів параметрів W_z. Значення t у h_{t-1} означає, що він містить інформацію попереднього блоку, і він помножений на його вагу. Далі значення цих параметрів додаються і передаються через сигмоїдну функцію активації.

Шлюз скидання r_t використовується у моделі, щоб вирішити, скільки інформації минулого потрібно знехтувати. Формула (1.9) така ж, як і шлюз оновлення.

Шлюз скидання зберігає відповідну інформацію з минулого кроку часу у новий стан. Потім він множить вхідний вектор і прихований стан на їх ваги, а далі обчислює по елементне множення між шлюзом скидання та раніше прихованим множником стану. Після підбиття підсумків до результатів застосовується функція нелінійної активації вищезазначених кроків, яка видає $\tilde{h}_t$.

Далі використовується шлюз оновлення h_t, який розраховується по формулі (1.11). Це векторне значення буде містити інформацію про поточний блок і передавати її до мережі. Він визначить, яку інформацію збирати.

Підводячи підсумок, GRU перевершило традиційний RNN. Якщо порівняти результати з LSTM, GRU мають перевагу в тому, що вони можуть керувати потоком інформації, не маючи явного блоку пам’яті, на відміну від LSTM. На тренування потрібно менше часу. Однак результати обох майже однакові.

Немає чіткої відповіді, який варіант виявився кращим. Зазвичай використовують обидва алгоритми і роблять висновок, який з них працює краще.

1.1.4 Класичні трансформери

Класичні трансформери є моделлю «від послідовності до послідовності» і складається з енкодера та декодера, кожен з яких є стеком L ідентичних блоків [5]. Уся архітектура представлена на рис 1.4.

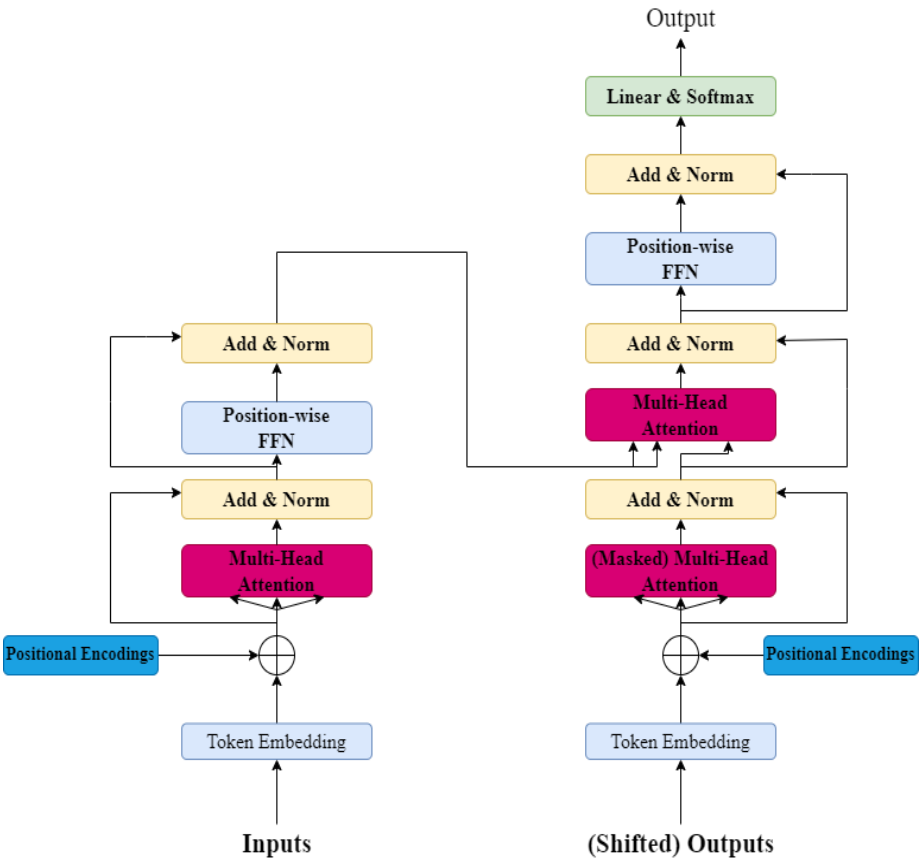

Рисунок 1.4 – Архітектура Трансформеру

Кожен блок енкодера в основному складається з модуля самоуваги та мережі з прямим поширенням зв’язком (FFN). Для побудови більш глибокої моделі навколо кожного модуля використовується залишкове з’єднання, за яким слідує модуль нормалізації. Порівняно з блоками енкодерів, блоки декодера до-

датково вставляють модулі перехресної уваги між модулями самоуваги та FFN. Крім того, модулі самоуваги в декодері адаптовані для запобігання відвідуванню кожної позиції наступних позицій.

У порівнянні з блоками енкодерів, блоки декодера додатково вставляють модулі перехресної уваги між модулями самоуваги. Крім того, модулі самоуваги в декодері пристосовані для запобігання переходу кожної позиції до наступних позицій

Енкодер: енкодер приймає вхідні дані, яке додається разом із позиційним кодуванням. Позиційне кодування дозволяє зберегти інформацію про позицію та порядок, та наведена у формулі 1.12.

$$PE_{(pos,2i)} = \sin\left(\frac{pos}{10000^{\left(\frac{2i}{d_{\text{mod}}}\right)}}\right), \tag{1.12}$$

де pos – місце слова у реченні.

Потім слідує залишкове з'єднання $R(x)$, визначене таким чином:

$$R(x) = LayerNorm(x + MultiHeadAttention(x)), \tag{1.13}$$

де x – це значення вхідних даних, додане з позиційним кодуванням.

Таким чином, ми можемо ефективно кодувати семантичну та пов’язану з позицією інформацію, використовуючи вхідне та позиційне кодування

Transformer використовує механізм уваги з Query–Key–Value (Запитом–Ключом–Значенням). Матричні представлення запитів $Q \in \mathbb{R}^{N \times D_k}$, ключі $K \in \mathbb{R}^{M \times D_k}$, значення $V \in \mathbb{R}^{M \times D_v}$, масштабований добуток, який використовує Transformer, визначається як,

$$Attention(Q,K,V) = softmax\left(\frac{QK^T}{\sqrt{D_k}}\right)V = AV, \quad (1.14)$$

де N та M – довжина запита та ключа,

D_k – розміри ключів,

A – матриця уваги.

Як показано вище у формулі 1.14, *Attention* приймає три вектори як вхідні дані — ключ, значення та запит. Як показано, ми виконуємо середнє зважене значення вектора V. Добутки запитів і ключів діляться на $\sqrt{D_k}$, щоб полегшити проблему зникнення градієнта функції *softmax*. Далі ми розглянемо основні частини моделі.

Декодер: майже ідентичний енкодеру, але використовує маскований модуль самоуваги. На відміну від звичайного модулю самоуваги, маскований модуль самоуваги отримує по одному значенню за раз.

На основі механізму уваги та без використання будь-якого механізму повторення, трансформатор фактично витіснив архітектури повторюваних нейронних мереж – LSTM, GRU тощо – як найсучасніші в кількох завданнях NLP [6]. Він використовується для широкого спектру завдань, включаючи класифікацію текста.

1.2 Огляд методів класифікації повідомлень

Для класифікації повідомлень можна використовувати різні методо машинного навчання, але необхідно обрати модель, яка буде відповідати реаліям, та виконувати поставлену задачу.

В першу чергу, буде розглянуто класичні методи машинного навчання. Розглянемо переваги та недоліки різних моделей та методів векторізації, зробимо висновки що до їх використання.

Далі буде розглянено статті у яких використовувалися класичні моделі машинного навчання з використанням попередньої обробки тексту.

У статті [14], автор використовував технологію для векторізації тексту, а саме CountVectorizer. Алгоритм простий, він підраховує кількість появлень у різних документів унікальних слів. Тим самим, буде отримані слова, які використовують у повідомленнях більш усього.

Далі було запропоновано використовувати два алгоритми машинного навчання для класифікації, логістичну регресію та k-найближчих сусідів. Можна зробити висновок що KNN не впорався при обраних параметрах, де кількість сусідів було 7,та досягає лише 0.6 точність, що зовсім не задовільно. А логістична регресія отримала результат 0.8, дуже не погано, але можна і краще. Автор не проводив пошук оптимальних параметрів, та не зробив увагу на незбалансовану вибірку.

Ще одна стаття [15] у якій використовуються класичні методи. С початку було проведено видалення різних символів та знаків, які не містять інформацію. Після цього, провели лематизацію тексту та видалення стоп-слів. У цьому випадку використовували TfidfVectorizer. Далі натреновувалася логістична регресія, яка досягла точності 0.8.

У статті [1], наведено різні підходи для розв’язку цієї проблеми. Як описують автори даної статті, вони використовують декілька способів векторізації тексту та різні моделі класифікації. Даний експеримент був поділений на три частини:

– класифікація тексту використовуючи векторне представлення BOW;

– класифікація тексту використовуючи Context-free embeddings;

– класифікація тексту використовуючи Context embeddings.

У першому експерименті використовувалися класичні моделі машинного навчання, такі як дерева прийняття рішення, випадкові ліси, логістична регресія. У другому використовувалися моделі векторізації тексту з останнім прошарком у вигляді Bi-LSTM. І останній експеримент з Bert та останнім прошарком у вигляді Bi-LSTM. Результати наведені на рис. 1.5

Можна побачити, класичні моделі не впорались з поставленою задачею, вони досягають точності 0.729, і не можуть дістати більше інформації. Якщо зробити модель біль складніше, обравши більш потужну модель, то результати стають кращими. Наприклад, Glove + Bi-LSTM, досягають більшої точності завдяки тому що інформація обробляється з двох сторін.

Model	**Train data**			**Test data**
	AUC	F1	Acc	Acc
BOW embeddings				
Decision tree	0.6320	0.5896	0.6273	0.6380
Random forest	0.8313	0.7320	0.7848	0.7042
Logistic regression	**0.8660**	**0.7443**	**0.7927**	**0.7293**
Context-free embeddings				
Skip-gram+Softmax	0.8281	0.7301	0.7769	0.7649
FastText+Softmax	0.8336	0.7231	0.7769	0.7826
GloVe+Softmax	0.8246	0.7323	0.7717	0.7827
Skip-gram+Bi-LSTM	0.8272	0.7440	0.7808	0.7775
FastText+Bi-LSTM	0.8327	0.7369	0.7817	0.7955
GloVe+Bi-LSTM	**0.8351**	**0.7500**	**0.7991**	**0.8093**
Contextual embeddings				
BERT+Softmax	0.8513	0.8254	0.8292	0.8250
BERT+Bi-LSTM	**0.8578**	**0.8316**	**0.8351**	**0.8308**

Рисунок 1.5 – Результати експерименту

Ще одна стаття [18] у яких використовують класичні моделі та бустінги. Результати обчислення наведені на рис. 1.6.

Classifier	Area Under Curve	Training time
Linear SVM	0.773577	4.100217
Logistic Regression	0.766635	0.268588
MNB	0.757180	0.004346
RidgeClassifier	0.751015	0.044368
Random Forest	0.744356	10.210122
catboost	0.738131	180.538903
xgboost	0.735154	11.151600
k-Nearest Neighbors	0.724199	0.002210
AdaBoost	0.723506	1.439533
Perceptron	0.703008	0.017310
Decision Tree	0.698416	2.244130
Gradient Boosting Classifier	0.681027	2.187549

Рисунок 1.6 – Результати обчислень класифікаторів

Хоча гіперпараметри не були оптимізовані, очищення тексту може зменшити шум у текстових наборах даних у вигляді стоп-слів, знаків пунктуації, чисел тощо. У реальних проблемах швидкість важливіша за точність. Тому, оскільки розмір даних збільшується, необхідно зосередитися на швидкості теж.

Можна зробити висновок, що класичні моделі не можуть впоратись з поставленою задачею, тому необхідно використовувати більш складні моделі.

Далі будуть розглянуті моделі з використанням LSTM, GRU та BERT.

У статі [16] автор використовував класичну LSTM для класифікації повідомлень. Як і у попередніх статтях, для векторізації повідомлень використалися CountVectorizer з попередньою обробкою. Результати наведені на рис. 1.7.

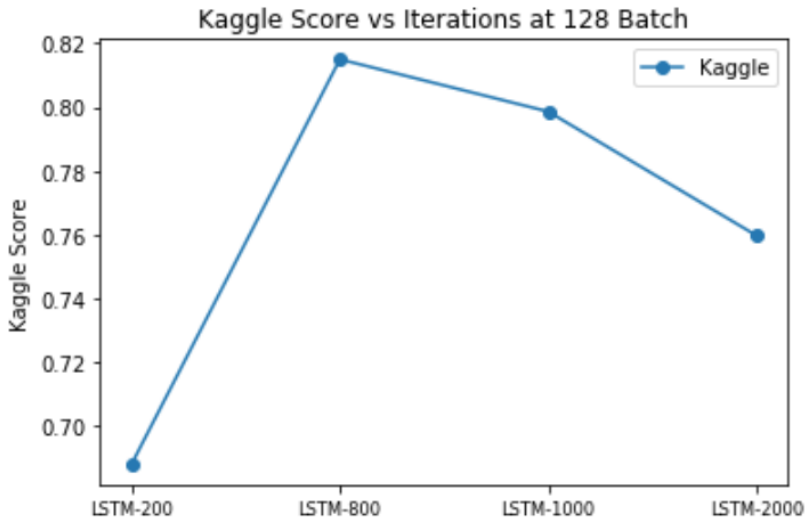

Рисунок 1.7 – Результати роботи LSTM при різної кількості ітерації

Для навчання моделі у даному випадку потребується 800 ітерації, і результат 0.81.

Далі автори статті [17], використовували для векторізації тексту не натреновану модель BERT. Також вони додали один прошарок нейронної мережи. Таким чином вони отримали наступний результат, на тренувальному наборі 0.95, на тестовому 0.81. Це перенавчання, але автори зробили висновок що це задовільний результат. Графік навчання моделі наведений на рис. 1.8.

Ідея використання BERT, як векторізацію повідомлень можна покращити,

для цього необхідно навчити його, а потим обробляти повідомлення.

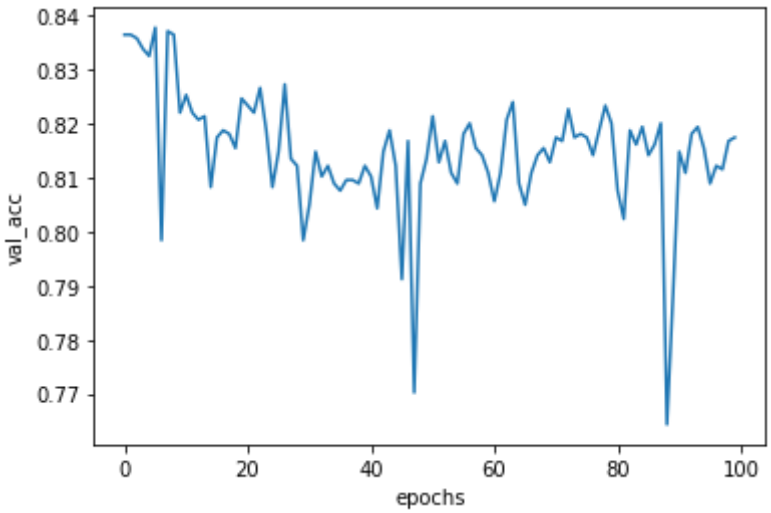

Рисунок 1.8 – Графік функції втрат

Автори статі [18], Різні прошарки на вихідних прошарках, та комбінують з різними параметрами на рис. 1.9.

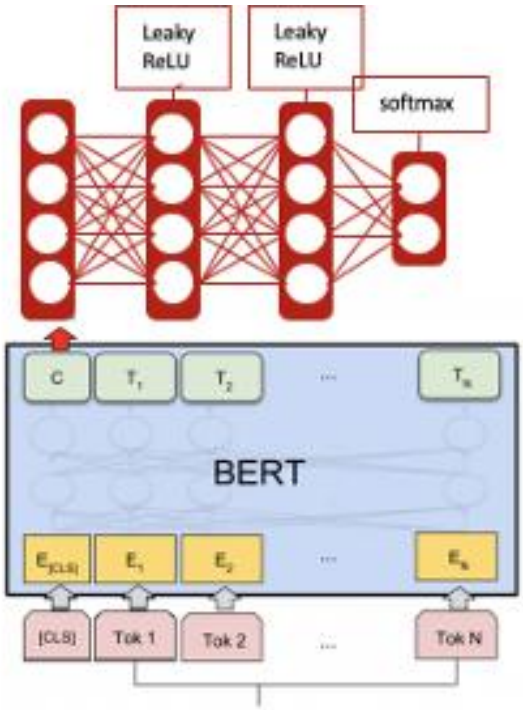

Рисунок 1.9 – Схема моделі у статі

Розглянувши статті, де описані кроки для класифікації тексту можна при-

йти до певних висновків. По-перше, класичні моделі не впораються з задачею, тому що методи векторізації достають мало інформації з тексту. По-друге, використовуючи BERT, можна покращити результати. Також необхідно правильно вирішувати проблему незбалансованих даних, це може покращити результат та усунути проблему оптимізації параметрів моделі.

1.3 Змістовна та формальна постановка задачі

Класифікація тексту є найважливішим завданням обробки природної мови. За останнє десятиліття спостерігається сплеск досліджень у цій галузі через безпрецедентний успіх глибокого навчання. Це досить складне завдання, необхідно передати усю суть речення або слів у вигляд, який зрозумілий для комп'ютера. За допомогою різних методів представлення слів у числовому вигляді можна досягти різних результатів.

Для отримання правильної класифікації тексту, було побудовано алгоритм за наступними кроками:

– отримання даних для класифікації;

– провести обробку даних та підготовити до тренування;

– тренування моделі BERT;

– отримання нового представлення повідомлень, використовуючи BERT;

– побудувати нейрону мережу та натренувати на нових даних;

– привести результати обчислення.

Нехай є повідомлення, яке містить інформацію про наявність або відсутність катастрофи. За допомогою обраного алгоритму, класифікувати це повідомлення

Нехай маємо повідомлення X та мітку його класу Y у наступному вигляді:

$$X = \{x_1, ..., x_n, x_{n+1}, ..., x_{n+l}\},$$

$$Y = \{y_1, ..., y_n\},$$

де $x_i \in R^{1 \cdot L_1}, 1 \le i \le n + l$ – векторне представлення повідомлення:

y_i – i-та мітка класу:

n – загальна кількість повідомлень.

Зауважимо, що для $\{x_{n+1}, ..., x_{n+l}\}$, повідомлення для яких немає відомого класу.

Нехай C – це загальна кількість класів, та $Y \in R^{n \cdot C}$ тоді матрицю класів можна представити у наступному вигляді:

$$Y_{ij} = \begin{cases} 1, \text{ if text } i \text{ belong to } j, 1 \le i \le l, \\ 0, \text{ if text } i \text{ does not belong to class } j, 1 \le i \le l. \end{cases}$$

Необхідно класифікувати $\{x_{n+1}, ..., x_{n+l}\}$ для яких невідомі класи.

1.4 Постановка задач дослідження

Метою написання кваліфікаційної роботи є застосування методів машинного навчання та векторізації тексту для класифікації повідомлень у соціальних мережах. Для досягнення поставленої мети треба виконати наступні етапи:

– ознайомлення із методами машинного навчання для класифікації повідомлень;

– ознайомлення із методами векторізації тексту;

– виконати програмну реалізацію для класифікації повідомлень;

– перевірити натреновану модель на тестових даних.

2 ВИБІР ТА ОБҐРУНТУВАННЯ МЕТОДУ РОЗВ’ЯЗАННЯ

2.1 Обробка природної мови NLP

Обробка природної мови (NLP) – наука про те, як змусити комп’ютери ефективно обробляти природний текст – нещодавно стала свідком швидкого прогресу завдяки збільшеній потужності обробки, даних і кращим алгоритмам.

Далі буде розглянуто основні задачі у яких використовується обробка природньої мови.

Вилучення інформації: задача вилучення явної або неявної інформації. Вихідні дані систем різняться, але часто витягнуті дані та зв'язки всередині них зберігаються в реляційних базах даних . Зазвичай витягується інформація включає названі сутності та відносини, події та їх учасників, тимчасову інформацію та кортежі фактів.

Розпізнавання іменованих об’єктів: задача яка автоматично ідентифікує названі сутності в тексті та класифікує їх за попередньо визначеними категоріями. Суб'єктами можуть бути імена людей, організацій, місцеположення, час, кількість, грошова оцінка, відсотки тощо.

Вилучення подій: виділення подій пов’язане з визначенням слів або фраз, які стосуються настання подій, а також учасників, таких як агенти, об’єкти, одержувачі та час їх виникнення. Виділення подій зазвичай має справу з чотирма під-задачами : визначення згадок про події або фраз, які описують події; визначення тригерів подій, які є основними словами (зазвичай дієсловам), які визначають настання подій; виявлення аргументів подій; і визначення ролі аргументів у подіях.

Вилучення зв’язків :інший важливий тип інформації, що витягується з тексту – це зв'язки. Це можуть бути присвійні, антонімічні чи синонімічні стосунки або більш природні, сімейні чи географічні стосунки.

Класифікація тексту: іншою класичною програмою НЛП є класифікація тексту або віднесення документів із вільним текстом до попередньо визначених

класів. Класифікація документів має безліч застосувань.

Генерація тексту: багато завдань НЛП вимагають створеннятексту. Підведення підсумків і машинний переклад перетворюють один текст в інший у послідовності в послідовність (seq2seq). Інші завдання, такі як створення субтитрів до зображень і відео, автоматичні звіти про погоду та спорт, перетворюють нетекстові дані в текст. Однак деякі завдання створюють текст без будь-яких вхідних даних для перетворення (або лише з невеликими обсягами, які використовуються як тема чи посібник). Ці завдання включають створення віршів, генерування жартів та оповідань.

Вилучення невеликого опису з тексту: знаходить елементи, що представляють інтерес у документах, щоб створити опис найважливішого змісту. Існує два основних типи узагальнення: екстрактивний і абстрактивний. Перший фокусується на виділенні, спрощенні, зміні порядку та об’єднанні речень, щоб передати важливу інформацію у документах за допомогою тексту, взятого безпосередньо з документів. Абстрактні резюме покладаються на вираження вмісту документів за допомогою абстракції в стилі створення, можливо, з використанням слів, яких ніколи не було в документах.

Відповідь на запитання: подібно до узагальнення та вилучення інформації, відповідь на запитання (QA) збирає відповідні слова, фрази або речення з документа. QA повертає цю інформацію узгодженим чином у відповідь на запит. Сучасні методи нагадують методи узагальнення.

Машинний переклад: квінтесенцією застосування НЛП. Він передбачає використання математичних та алгоритмічних прийомів для перекладу документів з однієї мови на іншу. Виконання ефективного перекладу по суті є обтяжливим навіть для людей, вимагаючи володіння такими галузями, як морфологія, синтаксис і семантика, а також вправне розуміння та розрізнення культурних особливостей обох мов (і асоційованих суспільств).

2.2 Векторізація тексту

У цьому пункті буде розглунено основні методи векторізації тексту для класифікації тексту.

OHE (One–hot Incoder): кожне слово в реченні представлено вектором [8]. Позиції входження слова у вектор надається одиниця рис. 2.1. Однак цей метод не є правильним підходом, коли ми хочемо провести аналіз настроїв або побудувати систему запитань-відповідей.

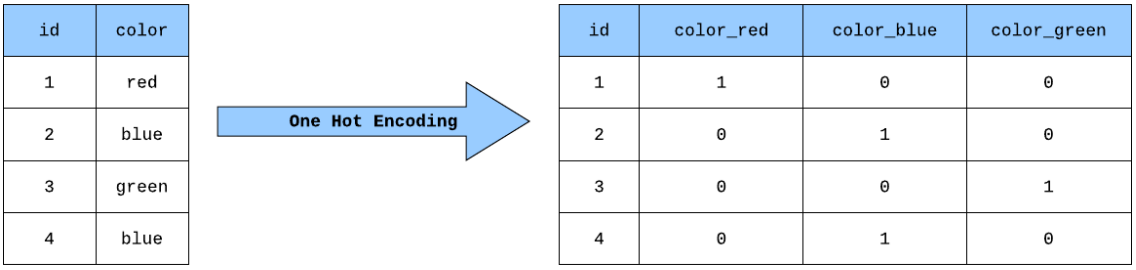

Рисунок 2.1 – Векторізація слів за допомогою OHE

Більше того, для великого корпусу ці вектори стають величезними і безглуздими.

CBOW (Continuous Bag of Words): часто використовується в програмах NLP [9]. CBOW намагається передбачити слово з огляду на його навколишній контекст рис 2.2, який зазвичай складається з кількох сусідніх слів. CBOW не залежить ні від послідовного порядку слів, ні від імовірнісних характеристик. Тому він зазвичай не використовується для мовного моделювання. Цю модель зазвичай навчають використовувати як попередньо навчену модель для більш складних завдань.

Альтернативою CBOW є зважений CBOW(WCBOW), в якому різні вектори отримують різну вагу, що відображає відносну важливість у контексті. Найпростішим прикладом може бути категоризація документа, де ознаками є слова, а ваги – це оцінки TF-IDF пов'язаних слів. Далі як векторне представлен-

ня слів, використовуються ваги моделі.

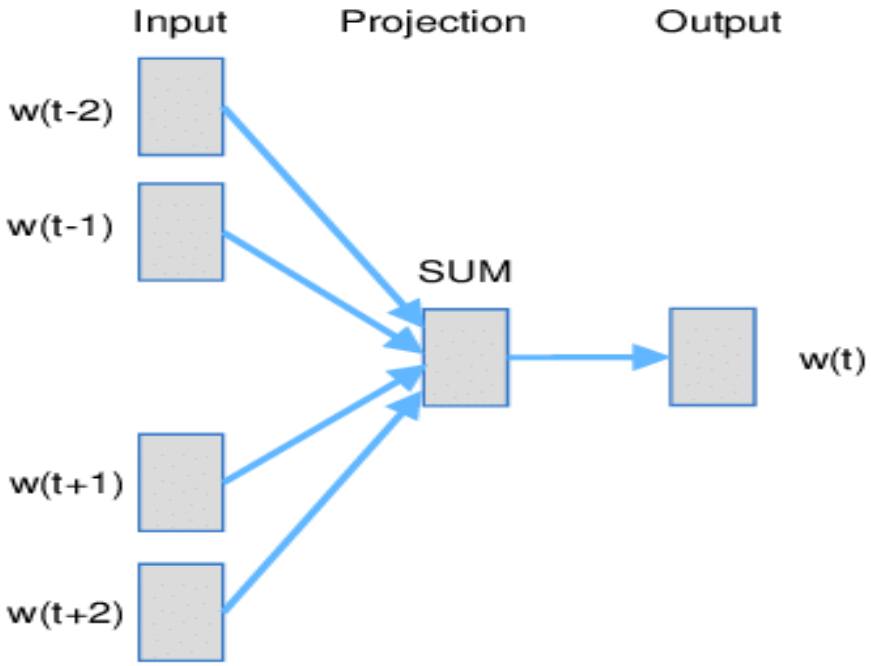

Рисунок 2.2 – Векторізація слів за допомогою CBOW

Skip-gram model: модель нейронної мережі skip-gram насправді проста в своїй основній формі [9]. Навчіть просту нейронну мережу з одним прихованим прошарком для виконання певного завдання. Натомість мета насправді полягає в тому, щоб просто дізнатися ваги прихованого шару – ці ваги насправді є «векторами слів», які намагаємося навчити. Зразок такої мережі наведено на рис. 2.3.

Отримавши певне слово в середині речення (вхідне слово), подивіться на слова поруч і виберіть одне навмання. Мережа скаже нам ймовірність того, що кожне слово в нашому словнику буде «ближнім словом», яке ми вибрали.

Імовірність виходу буде залежати від того, наскільки ймовірно знайти кожне словникове слово поблизу вхідного слова. Наприклад, якщо даси навченій мережі вхідне слово «сік», ймовірність виходу буде набагато вищою для таких слів, як «яблучний» і «смачний», ніж для непов’язаних слів, таких як «кит» і «бігати». Навчивши нейронну мережу робити це, передаючи їй пари слів, знайдені в наших навчальних документах.

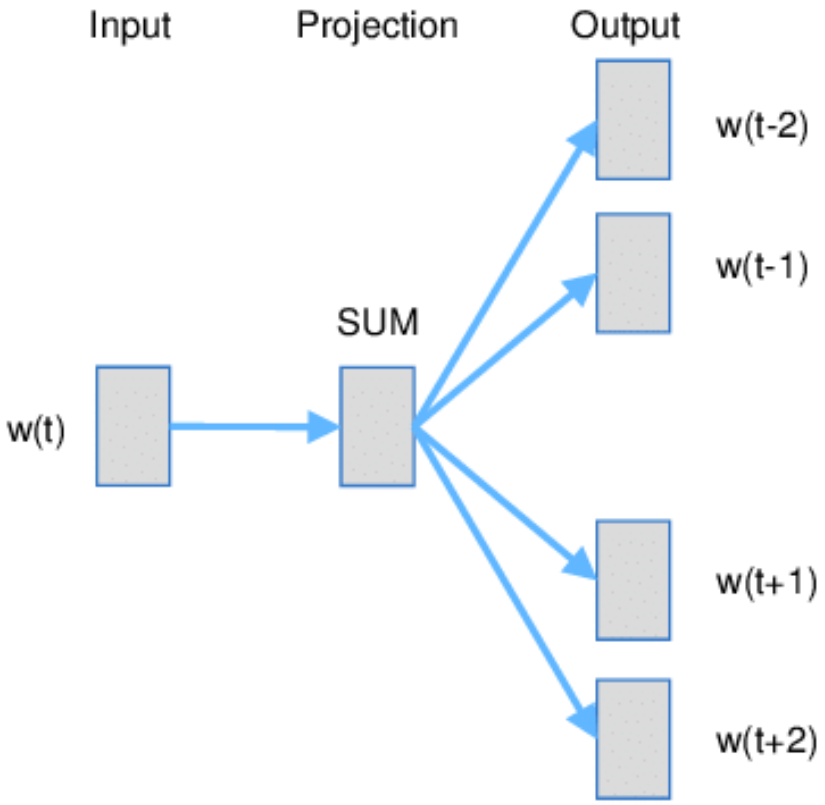

Рисунок 2.3 – Векторізація слів за допомогою Skip-gram model

GloVe: алгоритм навчання без учителя, для отримання векторних представлень слів [10]. Навчання виконується на основі агрегованої глобальної статистики спільного повторення слова з корпусу.

Модель GloVe навчається на ненульових записах глобальної матриці спільного зустрічання слова з іншим словом, яка відображає, як часто слова зустрічаються одне з одним у даному корпусі. Заповнення цієї матриці вимагає одного проходу через весь корпус для збору статистичних даних. Для великих корпусів ця перепустка може бути дорогою з точки зору обчислень, але це одноразова початкова вартість. Подальші ітерації навчання відбуваються набагато швидше, оскільки кількість ненульових записів матриці зазвичай набагато менша, ніж загальна кількість слів у корпусі.

Наприклад, розглянемо ймовірності спільного появи цільових слів лід і пар з різними пробними словами зі словника. Ось деякі фактичні ймовірності з 6 мільярдів слів на рис. 2.4.

Probability and Ratio	$k = solid$	$k = gas$	$k = water$	$k = fashion$
$P(k\|ice)$	1.9×10^{-4}	6.6×10^{-5}	3.0×10^{-3}	1.7×10^{-5}
$P(k\|steam)$	2.2×10^{-5}	7.8×10^{-4}	2.2×10^{-3}	1.8×10^{-5}
$P(k\|ice)/P(k\|steam)$	8.9	8.5×10^{-2}	1.36	0.96

Рисунок 2.4 – Зразок роботи GloVe, використовуючи два слова

FastText: розширення Word2Vec, запропоноване Facebook у 2016 році [11]. Замість того, щоб передавати окремі слова в нейронну мережу, FastText розбиває слова на кілька n-грамів (підслів). Наприклад, триграми для слова «apple» - це «app», «ppl» і «ple» (ігноруючи початок і закінчення меж слів). Векторне представлення слова для «apple» буде сумою всіх цих n-грамів. Після навчання нейронної мережі матиме векторне представлення слів для всіх n-грам з урахуванням навчального набору даних. Рідкісні слова тепер можуть бути належним чином представлені, оскільки дуже ймовірно, що деякі з їхніх n-грам також з’являються іншими словами.

WordPiece Embeddings: Векторне представлення Word Piece було розроблено для системи розпізнавання мовлення Google для азіатських мов, таких як корейська та японська [12]. У цих мовах є великий перелік символів, омонімів і небагато пробілів між словами. Відсутність або менша кількість пробілів означало, що для тексту потрібна сегментація. Однак сегментація призведе до створення великої кількості слів за межами словникового запасу (OOV) у моделі. Таким чином, представлення WordPiece було створено для автоматичного вивчення одиниць слів із великої кількості даних і не створює жодних OOV. Ця техніка використовується в BERT. OOV ігноруються в word2vec і GloVe, однак у FastText символові n-грамове подання слова компенсує OOV.

Далі буде використовуватися для векторного представлення слів WordPiece Embeddings, та яким чином він буде використовуватися.

2.3 Представлення двунаправленного енкодера від Трансформаторів

BERT (Bidirectional Encoder Representations from Transformers) – модель, опублікована дослідниками Google AI Language . Він викликав резонанс у спільноті машинного навчання, представивши найсучасніші результати в широкому спектрі завдань NLP.

Ключовою технічною інновацією BERT є застосування двунаправленного навчання Transformer, популярної моделі з використанням модуля уваги, до мовного моделювання. Це на відміну від попередніх зусиль, які розглядали послідовність тексту зліва направо або комбінували навчання зліва направо та справа наліво. Результати роботи показують, що мовна модель, яка навчається двосторонньою, може мати більш глибоке відчуття мовного контексту та потоку, ніж односпрямовані мовні моделі. У статті [13] дослідники докладно описують нову техніку під назвою Masked LM (MLM), яка дозволяє двонаправлене навчання в моделях, у яких раніше це було неможливо.

BERT використовує Transformer, який вивчає контекстні відносини між словами (або під словами) у тексті. У своїй звичайній формі Transformer включає два окремих механізму — енкодера, який зчитує введений текст, і декодер, який виробляє передбачення для виконання завдання. Оскільки метою BERT є створення мовної моделі, необхідний лише механізм енкодеру.

На відміну від моделей, які зчитують введений текст послідовно (зліва направо або справа наліво), енкодер Transformer зчитує всю послідовність слів одночасно. Тому він вважається двонаправленим, хоча точніше було б сказати, що він ненаправлений. Ця характеристика дозволяє моделі вивчати контекст слова на основі всього його оточення (ліворуч і праворуч від слова).

BERT використовує вбудовування WordPiece з 30 000 словників лексем. Перший маркер кожної послідовності завжди є маркером спеціальної класифікації ([CLS]). Остаточний прихований стан, відповідний цьому маркеру, використовується як представлення сукупної послідовності для завдань класифікації. Пари речень об’єднуються в одну послідовність. Речення диференціюються

за допомогою спеціальної лексеми ([SEP]) і шляхом додавання вивченого вбудовування до кожної лексеми, що вказує, чи належить вона до речення A чи речення B. Для даної лексеми її вхідне представлення будується шляхом підсумовування відповідного лексеми, відрізка, і вбудовування позицій.

На рис. 2.5 представлено як обробляються вхідні данні у модель. В першу чергу ми додаємо позиційне кодування, яке наведене у формулі (1.13).

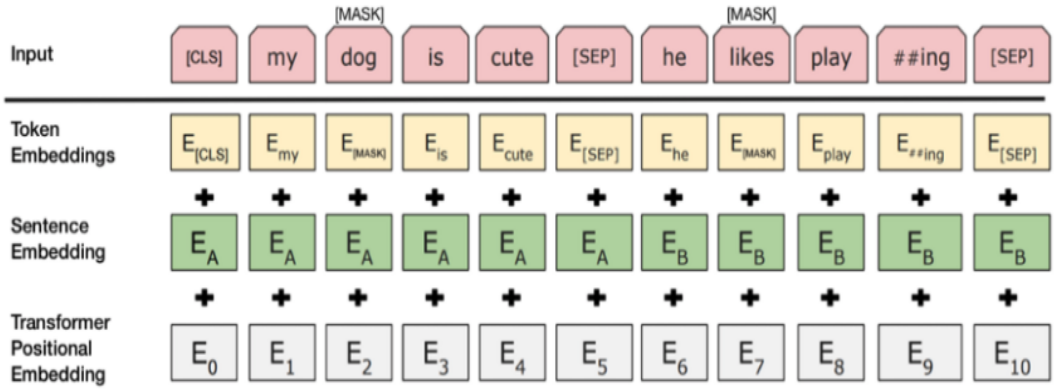

Рисунок 2.5 – Вхідні дані для BERT

Наступне, це Sentence Embeddings, який робиться для того, щоб відокремити класи речень. Bert буде розуміти, де починаються та закінчуються речення. Token embeddings: маркер [CLS] додається до вхідного слова на початку першого речення, а маркер [SEP] вставляється в кінці кожного речення. Також на цьому кроку, відбувається розбиття токену, як-от «playing» на «play» і «##ing».

BERT натренований на великому корпусі даних та на двох задачах. Перша, це класифікація пропущених слів та класифікація наступних речень. Для цього за допомогою маркеру [MASK], 10% замінювалися випадковими словами, та 10% не замінювалися новими словами, тобто замість [MASK] вставляються ті самі слова.

На виході з моделі отримується вектор 768 значень, які далі використовуються для класифікації.

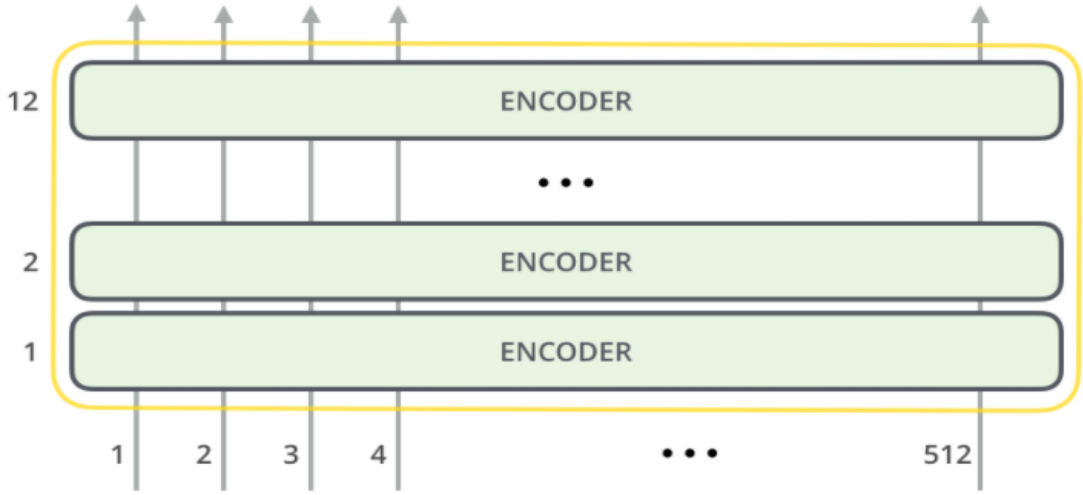

Рисунок 2.6 – Представлення моделі BERT

BERT повторює архітектуру енкодера Transformer, які йдуть послідовно та передають інформацію зліва на право та з право на ліво [14]. Повна модель BERT представлена на рис. 2.7

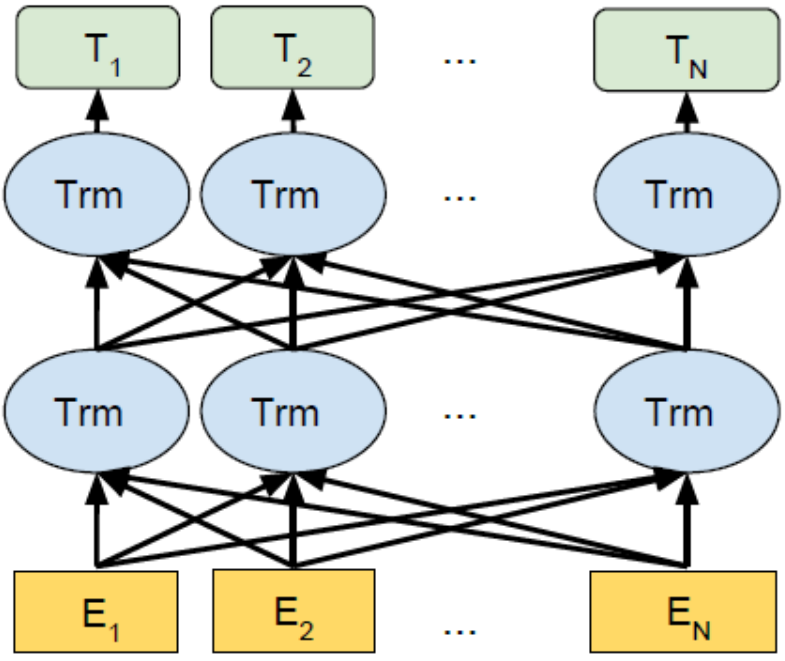

Рисунок 2.7 – Повна модель BERT

3 ПРОГРАМНА РЕАЛІЗАЦІЯ

3.1 Опис програмного середовища

Python є однією з найбільш поширених мов програмування та популярна серед розробників з різних причин, одна з яких полягає в тому, що він має неймовірно різноманітну колекцію бібліотек, в яких користувачі можуть встановлювати та використовувати. Переваги обраної мови програмування:

– дуже простий синтаксис: для людей, які тільки навчаються програмувати;

– логічний синтаксис: у порівнянні з багатьма іншими мовами, Python має простий синтаксис, який легко читати;

– підтримує основні структури даних (кортежі, списки, масиви, словники)

– підходить для різних платформ;

– широке застосування: використовується для розробки веб-додатків, ігор, зручний для оптимізації, математичних обчислень, для машинного навчання;

– необхідність у спеціалістах;

– висока кількість бібліотек для обробки та графічного зображення даних.

Python відрізняється суровою вимогою до написання коду (вимагає відступи), що є перевагою, тому що код зручніше читати.

Keras — це багаторівневий API нейронних мереж, написаний на Python, який може працювати поверх TensorFlow та інших фреймворків нижнього рівня. Він був розроблений з упором на те, щоб дозволити швидкі експерименти та забезпечити просте та швидке створення нейроних мереж (за рахунок зручності, модульності та розширюваності).

Найбільш вагомі причини для використання Keras випливають зі зручності використання. Код обробляє такі інструменти, як центральний процесор (CPU) і графічний процесор (GPU), що дозволяє оптимально використовувати їх. Він також має реалізацію функцій активації, оптимізаторів, метричних фор-

мул і процедур, необхідних для легкої обробки навчальних сесій. Надійна та проста у використанні безкоштовна бібліотека Python з відкритим кодом є найважливішим інструментом для розробки та оцінки моделей глибокого навчання.

HuggingFace — це стартап, орієнтований на NLP, із великою спільнотою з відкритим кодом, зокрема навколо бібліотеки Transformers. Transformers — це бібліотека на основі Python, яка надає API для використання добре відомий архітектури трансформаторів, таких як BERT, RoBERTa, GPT-2 або DistilBERT, які отримують найсучасніші результати для різноманітних завдань NLP, таких як класифікація тексту, вилучення інформації, відповіді на запитання та генерація тексту. Ці архітектури попередньо підготовлені з кількома наборами ваг.

3.2 Алгоритм розв’язання задачі класифікації повідомлень

Алгоритм для розв’язання проблеми класифікації повідомлень наведений на рис. 3.1.

Для початку необхідно загрузити та перевірити наявність даних у файлі. Після цього буде проведено аналіз набору даних на наявність якихось закономірностей або залежності. У залежності від висновків, буде проведена видалення стовпчиків, які містять найменшу кількість інформації. Після завершення обробки набору даних, буде проведені етап, з навчанням моделі BERT. Використовуючи векторне представлення за допомогою навченої моделі, будуть отримані нові дані. Після цього будуть використані різні моделі класичного машинного навчання для класифікації повідомлень. Також будуть представлені все необхідні рисунки, за якими можна буде підвести підсумки.

Алгоритм для розв’язання проблеми класифікації повідомлень наведений на рис. 3.1 та за 6 пунктами, які наведені нижче:

– отримання даних для класифікації;

– провести обробку даних та підготовити до тренування;

– тренування моделі BERT;

– отримання нового представлення повідомлень, використовуючи BERT;

– побудувати нейрону мережу та натренувати на нових даних;

– привести результати обчислення.

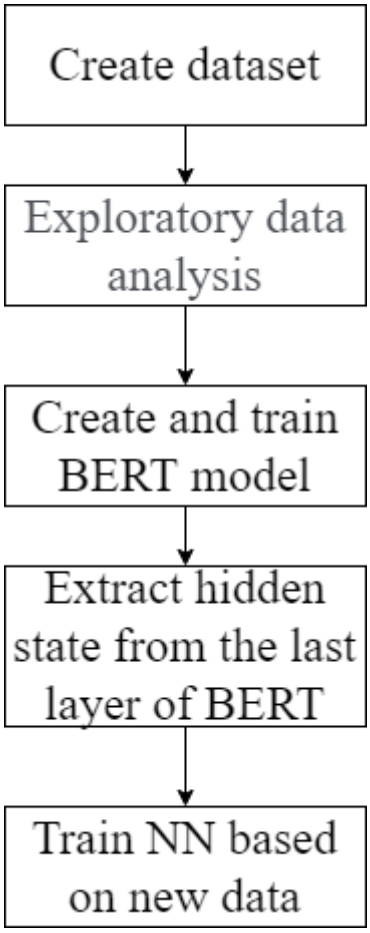

Рисунок 3.1 – Алгоритм класифікації тексту

3.3 Опис програми

Далі представлений список бібліотек. які будуть використану для обробки та обчислення експерименту:

–transformer: надає API для швидкого завантаження та використання попередньо навчених моделей для певного тексту;

– numpy: бібліотека мови Python, яка підтримує багатовимірні масиви та

матриці, разом з математичними функціями для операції над масивами;

– matplotlib.pyplot: бібліотека мови Python, яка дозволяє утворювати фігури, зони для малювання у фігурі, підписувати графіки, виводити результат на графік;

– tensorflow: відкрита програмна бібліотека для машинного навчання:

– pandas: бібліотека для роботи з даними:

– sklearn: бібліотека, яка використовувалася для розбиття на тренувальну та тестову вибірки.

Використовуючи обрані бібліотеки, опишемо покроковий алгоритм.

Спочатку завантажимо набір даних за допомогою бібліотеки pandas, та методу read_csv(). Після цього, маючи дані, проведемо первинній огляд даних та що вони містять. Для того, щоб навчити модель нам необхідно представити їх у вигляді якій розуміє BERT. Для цього використовується клас NewsGroupsDataset, який трансформує датафрейм у тензори.

Далі використовуючи бібліотеку transformer, завантажуємо модель BERT для навчання. Після цього кроку, необхідні кроки зроблено, починається навчання. Зберігаємо модель у файл, та використовуємо знову, для отримання нового набору даних, які будуть представлять векторне представлення повідомлень.

Будуємо нову нейрону мережу для класифікації речень використовуючи tensorflow у векторному вигляді, та виведемо результати навчання використовуючи бібліотеку matplotlib.pyplot .

Усі результати виводу програми, наведені у наступному розділі.

4 РЕЗУЛЬТАТИ ОБЧИСЛЮВАЛЬНОГО ЕКСПЕРЕМЕНТУ

У даній роботі використовувалися данні, які містять інформацію про стихійні лиха у соціальній мережі. У нас є 7613 спостережень, і необхідно спрогнозувати, чи стосується даний твіт про справжню катастрофу чи ні. Стовпчики, які містять інформацію та їх опис наведені нижче:

– text – текст, якій містить інформацію від користувача;

– location – місце, з якого було відправлене повідомлення (може бути пустим значенням);

– keyword – унікальне значення повідомлення;

– target – значення, яке повідомляє, чи йдеться у повідомленні про справжню катастрофу (1) чи ні (0).

Це п’ять різних стовпчиків, доступних у цьому наборі даних, але ми використовуємо лише стовпці text та target. Тепер давайте подивимося на усі стовпчики, які є у наборі даних рис 4.1.

	id	keyword	location	text	target
0	1	NaN	NaN	Our Deeds are the Reason of this #earthquake M...	1
1	4	NaN	NaN	Forest fire near La Ronge Sask. Canada	1
2	5	NaN	NaN	All residents asked to 'shelter in place' are ...	1
3	6	NaN	NaN	13,000 people receive #wildfires evacuation or...	1
4	7	NaN	NaN	Just got sent this photo from Ruby #Alaska as ...	1
...	...	...	...	...	...
7608	10869	NaN	NaN	Two giant cranes holding a bridge collapse int...	1
7609	10870	NaN	NaN	@aria_ahrary @TheTawniest The out of control w...	1
7610	10871	NaN	NaN	M1.94 [01:04 UTC]?5km S of Volcano Hawaii. htt...	1
7611	10872	NaN	NaN	Police investigating after an e-bike collided ...	1
7612	10873	NaN	NaN	The Latest: More Homes Razed by Northern Calif...	1

7613 rows × 5 columns

Рисунок 4.1 – Набір даних

Якщо візуалізувати кількість твітів про катастрофу до тих, що не є катас-

трофою, показано на рис. 4.2, зрозуміло, що набір даних не збалансований.

Рисунок 4.2 – Розподіл даних у тренувальному та тестовому наборі

Щоб вирішити цю проблему, ми будемо використовувати стратифіковану вибірку, де набір даних поділяється на однорідні підгрупи, які називаються стратами, і правильна кількість екземплярів відбирається з кожного шару, щоб гарантувати, що тестовий набір є репрезентативним для загальної сукупності. Ми бачимо, що в наборах даних поїздів і тестів майже 41% твітів про катастрофу.

Далі нам необхідно представити текст у вигляді, якій розуміє BERT. Для цього я використовую числове представлення, як описано у главі 2.3. Як приклад:

Forest fire near La Ronge Sask. Canada

[101, 3304, 1783, 1485, 2495, 187, 4553, 1162, 21718, 5276, 119, 1169, 7971, 102]

Після цього починаємо навчання моделі. У даному експерименті я використовував наступні параметри:

– max_length = 384, повідомлення, які більше цього значення, будуть об-

різатися;

– test_size = 0.25, 25% даних будуть використовуватися для перевірки моделі;

– num_train_epochs = 3, модель тричі пройде по всьому набору даних;

– batch_size = 8, оновлення параметрів буде здійснюватися після кожних 8 повідомлень.

На відеокарті GeForce GTX 1650 з 4 Gb операційної пам’яті навчання зайняло 13 хвилин. Точність на тестовому наборі досягла 0.82, що є гарним результатом, але використовуючи додатковий прийом, можна досягти ще більшого результату.

Отже, ми використали технологію, яку описали у главі 2.3. Далі ми використовуємо модель ще раз, але вже не для навчання. Останній прошарок нейроной мережі повертає для кожного повідомлення вектор, якій має розмірність 768 значень. Було отримані наступні дані на рис. 4.3.

	0	1	2	3	4	5	6	7	8	9	...
0	0.423426	-0.251528	0.282223	0.270210	0.258724	0.335305	0.064941	0.794778	0.420044	-0.202521	...
1	-0.014570	-0.093493	-0.390383	0.095072	-0.611734	0.977819	-0.165947	1.011086	0.771759	-0.983602	...
2	-0.038428	-0.017081	-0.326123	0.127117	-0.527579	0.923194	-0.267462	1.026882	0.722546	-1.067038	...
3	-0.256731	-0.074020	-0.487902	0.172744	-0.728080	1.003437	-0.290976	0.927803	0.650606	-1.011600	...
4	0.245798	-0.176952	-0.261151	0.021361	-0.246441	0.788786	0.011844	1.055458	0.672748	-0.803262	...
...	...	...	...	...	...	...	...	...	...	...	...
7608	-0.271265	-0.161847	-0.460418	0.242980	-0.730885	1.022845	-0.276768	0.947702	0.575634	-1.010137	...
7609	0.022558	-0.148551	-0.331165	0.136639	-0.519442	0.838553	-0.220106	1.030167	0.758880	-1.038242	...
7610	0.215609	-0.198080	-0.220306	0.105607	-0.203628	0.873986	0.033805	1.084941	0.672629	-0.806605	...
7611	-0.106945	-0.142927	-0.421893	0.176663	-0.584898	1.042966	-0.213914	1.031048	0.702504	-0.983345	...
7612	-0.233167	-0.110907	-0.501802	0.222161	-0.704949	0.959131	-0.250200	0.970461	0.602255	-1.012417	...

7613 rows × 768 columns

Рисунок 4.3 – Оброблений набір даних

Тепер можна візуалізувати дані. Для цього будемо використовувати TSNE, для представлення багатовимірного набору даних у двовимірний:

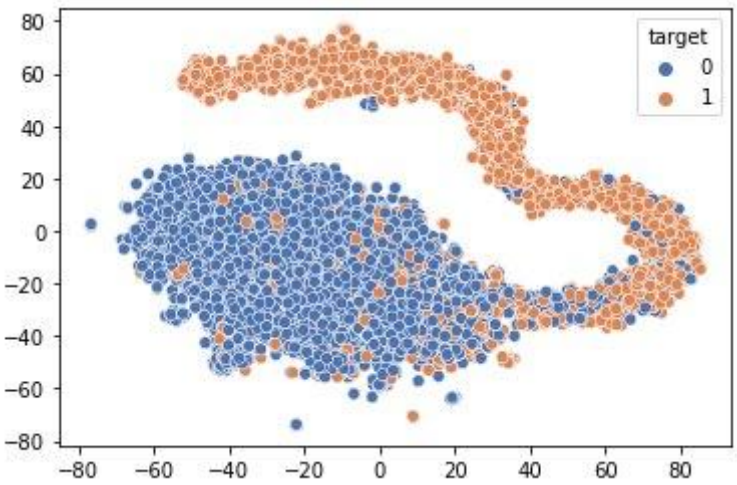

Рисунок 4.4 – Представлення даних у двовимірному просторі за допомогою TSNE

Вже можна зробити деякі висновки стосовно даних, які представлені на рис. 4.4. По-перше, є дві області, у яких розташовані два класи. По-друге, ці дві області можна розділити кривою. По-третє, якщо 2 двомірному просторі можна розділити дані на дві області, можна стверджувати, що у багатовимірному просторі їх теж можна розділити.

На цьому кроці, побудуємо нейронну мережу, яка буде вирішувати дану задачу. У роботі використовувалася нейрона мережа яка представлена на рис. 4.5. Також використовувалися наступні допоміжні функції та параметри:

– раннє зупинення = 10, якщо точність на тестовій вибірці не змінювалося 10 епох, то навчання припинялося:

– зберігання найкращої моделі, кожен раз, коли ми досягаємо найбільшої точності на тестовому наборі, ми зберігаємо модель.

– dropout – 0.1;

– units – 32;

– activation – relu та sigmoid на останньому прошарку;

– optimizer – метод оптимізації adam;

– loss – binary crossentropy;

– metrics – auc-roc та accuracy.

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 32)                24608
_________________________________________________________________
dropout (Dropout)            (None, 32)                0
_________________________________________________________________
dense_1 (Dense)              (None, 32)                1056
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 33
=================================================================
Total params: 25,697
Trainable params: 25,697
Non-trainable params: 0
```

Рисунок 4.5 – Архітектура нейроної мережи

У ході роботи було використовувані різні комбінації вузлів, з різними функціями активації. Також різна кількість прихованих прошарків, але результат не змінювався (у інших випадків було перенавчання).

Після навчання було отримано наступні результати, які досягають більшої точності, на рис. 4.6 – 4.8.

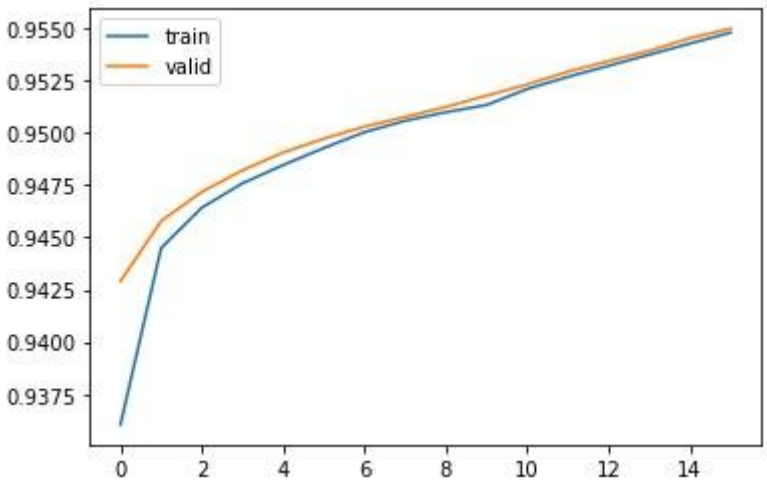

Рисунок 4.6 – Графік зміни auc_roc у залежності від епохи
на тренувальних та тестових вибірках

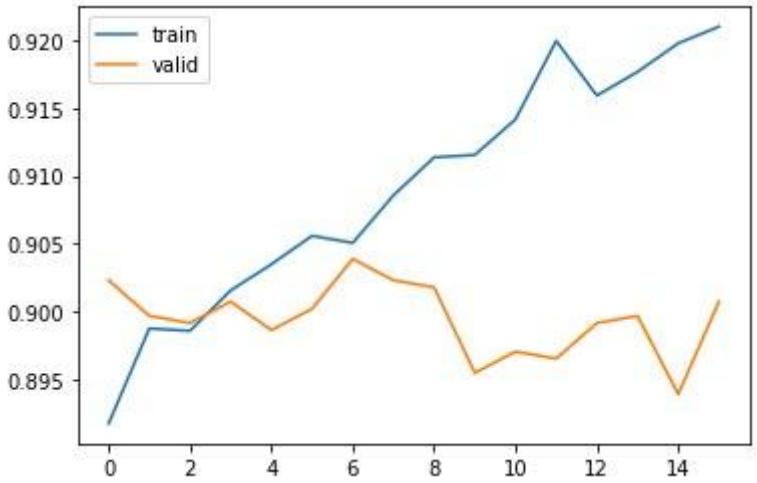

Рисунок 4.7 – Графік зміни точності у залежності від епохи
на тренувальних та тестових вибірках

Найкраща модель з найкращім результатом на тестовому наборі досягнена на 6 ітерації зі значенням 0.903. Якщо порівняти з звичайним BERT, результат був покращений на 8 пунктів, що показує правильність дії.

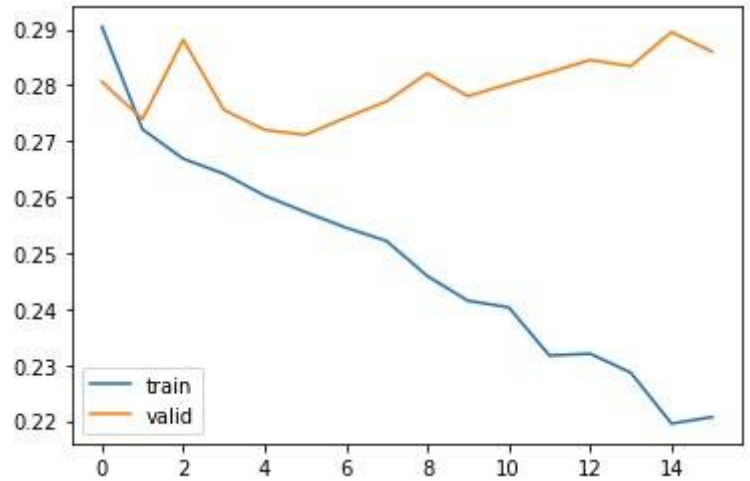

Рисунок 4.8 – Графік зміни похибки у залежності від епохи
на тренувальних та тестових вибірках

Проаналізуємо результат на тестовій вибірці, для цього побудуємо матрицю невідповідності на рис. 4.9.

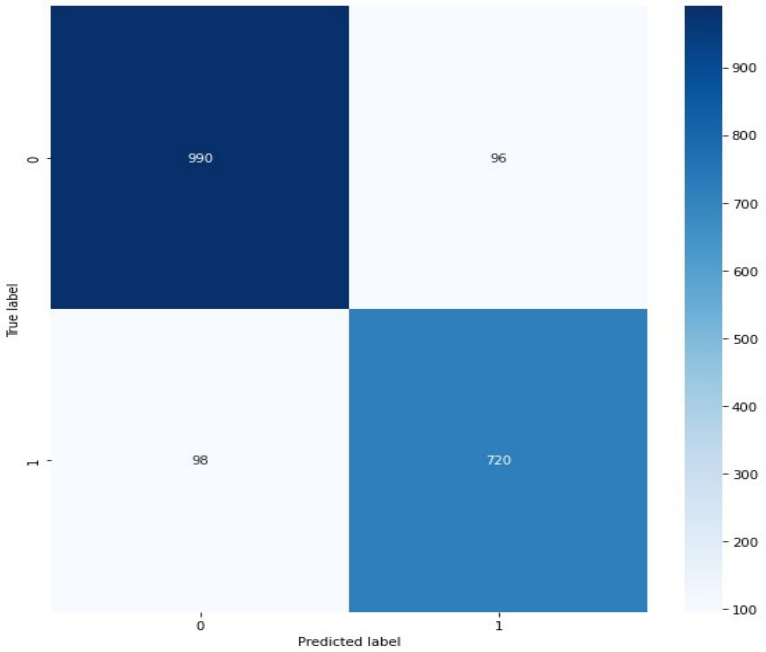

Рисунок 4.9 – Матриця невідповідності

Модель однаково помиляється на двох класах, але дуже добре класифікує.

Далі представлені на рис. 4.10, результаті роботи моделі на повідомленнях, які не бачила модель.

Text of Message	True	Predicted
EARTHQUAKE SAFETY LOS ANGELES	1	1
Just happened a terrible car crash	1	1
They'd probably still show more life than Arsenal did yesterday, eh? EH?	0	0
ACCIDENT - HIT AND RUN - COLD at 500 BLOCK OF SE VISTA TER GRESHAM	1	0

Рисунок 4.10 – Результати обчислювального експерименту

У деяких випадках класифікатор помиляється, але у загальному випадку ми маємо досить гарний результат. Також при аналізу набору даних, було виявлено декілька тестових зразків які були на різній мові. Як на приклад, foi no ROH Aftershock: Las Vegas procura no pirate bay que tem'.

Далі перевіряємо різні класифікатори. На рис. 4.12 приведено порівняння різних класифікаторів на навчальному наборі.

Classifier	Accuracy
BERT+RNN	0.80275
BERT+LSTM	0.82198
BERT+GRU	0.83198
BERT+NN	0.91250
BERT+LR	0.87104
BERT+RF	0.88932

Рисунок 4.12 – Результати роботи класифікаторів

Краще усіх працює комбінація BERT з NN. Я вважаю що це пов’язано з можливість розділити у багатовимірному просторі дані на два класи, з чим дана модель і впоралась.

Також результат роботи моделі був протестований на вибірці яку модель-не бачила та для котрих не відомі мітки класу, на платформі Kaggle [19]. Результат отриманий на перевірці, досяг відмітки 0.8225 точності та зайняв 184 місце з 902 завантажених результатів.

ВИСНОВКИ

У кваліфікаційній роботі було розглянуто проблему математичного моделювання та методів класифікації повідомлень про стихійні лиха в соціальних мережах. Було проведено огляд методів класифікації тексту, який показав що не потребує використання рекурентних нейронних мереж, які досягають невеликі результати.

У роботі використовувалося векторне представлення повідомлення за допомогою навченого BERT на даних, які необхідно класифікувати. Як наведено у результатах експерименту, на невеликому наборі даних, було отримано задовільні результати та проведено порівняння з іншими моделями. Класичні методи машинного навчання , такі як випадкові ліси та логістична регресія здобули задовільній результат, та досягли точності більше ніж рекурентні нейронні мережі. Як наведено на рис. 4.4, дані майже поділені на два класи у двомірному просторі, тому можлива класифікація класичними методами у багатовимірному просторі. Більш надійний результат надає класична нейронна мережа з одним прошарком, це завдяки більшої кількості параметрів у моделі. Беручи до уваги, кількість епох та точність на навчальній та тестовій вибірці рис. 4.6 –4.8, можна зробити висновок, що необхідно тільки пару епох для повного навчання моделі. Детальніше аналізуючи набір даних, було виявлено повідомлення які містять шум, тобто вони написані на різних мовах. Використовуючи результати отриманих від інших учасників, можна побачити, що результати розподілені від 0.820 до 0.845. Це показує, що отриманий результат відповідає можливостям отриманим даних.

Результати отриманого експерименту можна використовувати у рятувальних службах, сферах масової інформація, фільтрація повідомлень, класифікація стану людини по повідомленню. Можливість своєчасно відреагувати на можливу катастрофу, може врятувати життя.

У подальшому аналізу теми, можна буде зробити декілька покращень, а саме:

– використовувати біль складні моделі, такі як roBerta, Electra тощо;

– використовувати більшу вибірку;

– використовувати ансамблі моделей;

– використовувати інші комбінації параметрів нейронних мереж;

– використовувати додаткові ознаки у наборі даних;

– використовувати моделі, які розпізнають інші мови.

Сучасний світ генерує багато текстових даних, які необхідно розуміти та обробляти. Тому необхідно вдосконалювати методи векторізації тексту для більш легкого використання. Завдання, які використовують великий обсяг текстових даних, такі як соціальні мережі, будуть завжди актуальні.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Ashis K. C. Efficacy of BERT embeddings on predicting disaster from Twitter data. 2021. URL : https://arxiv.org/abs/2108.10698 (дата звернення: 26.11.2021).

2. Understanding LSTM Networks. URL : http://colah.github.io/posts/2015-08-Understanding-LSTMs/ (дата звернення: 28.11.2021).

3. Introduction to LSTM Units in RNN. URL : https://www.pluralsight.com-/guides/introduction-to-lstm-units-in-rnn (дата звернення: 27.11.2021).

4. LSTM versus GRU Units in RNN. URL : https://www.pluralsight.com /guides/lstm-versus-gru-units-in-rnn (дата звернення: 27.11.2021).

5. Aditya M., Pratik R. Evolution of Transfer Learning in Natural Language Processing. 2019 URL : https://arxiv.org/pdf/1910.07370.pdf (дата звернення: 28.11.2021).

6. Attention Is All You Need. URL : https://arxiv.org/abs/1706.03762 (дата звернення: 28.11.2021).

7. CrisisBench: Benchmarking Crisis-related Social Media Datasets for Humanitarian Information Processing. URL : https://arxiv.org/abs/2004.06774 (дата звернення: 03.12.2021).

8. Word Embeddings, WordPiece and Language-Agnostic BERT (LaBSE). https://medium.com/mlearning-ai/word-embeddings-wordpiece-and-language-agnostic-bert (дата звернення: 01.12.2021).

9. GloVe: Global Vectors for Word Representation. URL : https://nlp.stanford.edu/projects/glove/ (дата звернення: 03.12.2021).

10. Word2Vec and FastText Word Embedding with Gensim. URL : https://towardsdatascience.com/word-embedding-with-word2vec-and-fasttext (дата звернення: 03.12.2021).

11. Recent Trends in Deep Learning Based Natural Language Processing. URL : https://arxiv.org/abs/1708.02709v8 (дата звернення: 04.12.2021).

12. BERT: Pre-training of Deep Bidirectional Transformers for Language

Understanding. URL : https://arxiv.org/abs/1810.04805 (дата звернення: 03.12.2021).

13. The Illustrated BERT, ELMo, and co. (How NLP Cracked Transfer Learning). URL : https://jalammar.github.io/illustrated-bert/ (дата звернення: 04.12.2021).

14. Classification Model for Disaster-related Tweets. URL : https://blog.clairvoyantsoft.com/classification-model-for-disaster-related-tweets- (дата звернення: 05.12.2021).

15. Disaster Tweets Classification. URL : https://gnanaebiston.medium.com /disaster-tweets-classification (дата звернення: 05.12.2021).

16. Disaster-Tweets-Classification-Using-NLP. URL : https://github.com /harimanasa/ (дата звернення: 05.12.2021).

17. Real or Not? NLP with Disaster Tweets (Classification using Google BERT). URL : https://levelup.gitconnected.com/real-or-not-nlp-with-disaster-tweets-classify-cation-using-google-bert (дата звернення: 05.12.2021).

18 Tweets Classification with BERT in the Field of Disaster Management. URL : https://web.stanford.edu/class/archive/cs/cs224n/cs224n.1194/reports/custom/ 15785631 (дата звернення: 05.12.2021).

19. Natural Language Processing with Disaster Tweets. URL : https://www.kaggle.com/c/nlp-getting-started (дата звернення: 25.11.2021).