МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Харківський національний університет радіоелектроніки

Факультет _____Комп'ютерних наук_____

Кафедра _____Програмної інженерії_____

КВАЛІФІКАЦІЙНА РОБОТА

Пояснювальна записка

рівень вищої освіти - другий (магістерський)

Дослідження методів узагальнення текстів

Виконав:

Студент _2_ курсу, групи _____ІПЗм-19-2_____

_____Богданова А.О._____

Спеціальність 121-Інженерія програмного забезпечення

Тип програми _____освітньо-наукова_____

Керівник _____доцент Турута О.П._____

Допускається до захисту

Зав. Кафедри, проф. _____ З.В. Дудар

2021 р.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)

Кафедра Програмної інженерії
(повна назва)

Рівень вищої освіти другий (магістерський)

Спеціальність 121 – інженерія програмного забезпечення
(код і повна назва спеціальності)

Тип програми Освітньо-наукова
(освітньо-професійна або освітньо-наукова)

Освітня програма Інженерія програмного забезпечення
(повна назва)

ЗАТВЕРДЖУЮ:
Зав.кафедри _____
(підпис)
« ___ »_____ 2021р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студента *Богданової Аліни Олексіївни*
(прізвище, ім'я, по батькові)

1. Тема роботи *Дослідження методів узагальнення текстів*

затверджена наказом університету від *26.03.2021* № *34 Стз*
2. Термін подання роботи до екзаменаційної комісії *14 травня* 2021р.
3. Вихідні дані до роботи *методи обробки і аналізу текстових даних, нейромережеві підходи до вирішення задач, графові методи побудови узагальненя текстів, OS Windows, мова програмування Python, фреймворк Pytorch, середовище розробки Google Colaboratory*
4. Перелік питань, що потрібно опрацювати в роботі *мета роботи, аналіз проблемної галузі і постановка задачі, аналіз методів вирішення багатокритеріальних задач, аналіз методів нормалізації, розробка алгоритму ранжування, проектування програмного додатку.*
5. Перелік графічного матеріалу із зазначенням креслеників, схем, слайдів, ілюстрацій: *Актуальність роботи, підходи до вирішення задачі сумаризаціїб екстрактивні методи, метод оцінки якості узагальнення, алгоритм TextRank, порівняння результатів графових методів, застосування моделей машинного навчання, порівняння результатів методів машинного навчання, висновки.*

6. Консультанти розділів роботи

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата
Спецчастина	доц. Турута О.П.		

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1.	Аналіз предметної галузі	18 лютого 2021р.	виконано
2.	Аналіз методів роботи з текстовими даними	10 березня 2021р.	виконано
3.	Проектування експериментів з методів узагальнення текстів	17 березня 2021р.	виконано
4.	Реалізація методів узагальнення текстів	31 березня 2021р.	виконано
5.	Розробка демонстраційного додатку	19 квітня 2021р.	виконано
6.	Підготовка пояснювальної записки	06 травня 2021р.	виконано
7.	Підготовка презентації та доповіді	06 травня 2021р.	виконано
8.	Перевірка на академічний плагіат	07 травня 2021р.	виконано
9.	Нормоконтроль	07 травня 2021р.	виконано
10.	Рецензування	12 травня 2021р.	виконано
11.	Попередній захист	14 травня 2021р.	виконано
12.	Занесення диплома в електронний архів	14 травня 2021р.	виконано
13.	Допуск до захисту у зав. кафедри	14 травня 2021р.	виконано

Дата видачі завдання 25 січня 2021р.

Студент _____
(підпис)

Керівник роботи _____ доцент Турута О.П.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ / ABSTRACT

Кваліфікаційна робота магістра містить: 69 с., 10 рис., 8 табл., 46 джер.

NATURAL LANGUAGE PROCESSING, PYTHON, TEXT SUMMARIZATION, МАШИННЕ НАВЧАННЯ, РЕКУРЕНТНІ НЕЙРОННІ МЕРЕЖІ.

Метою роботи є дослідження, аналіз та реалізація методів узагальнення текстів. Ці методи надають можливість автоматизувати створення стислих переказів тексту для пришвидшення аналізу інформації та автоматизації рутинних задач.

Методи розробки базуються на інструментах та бібліотеках для аналізу даних, використовується мова програмування Python, а також бібліотеки обробки тексту NLTK та Spacy. Також використовується фраемоврк Pytorch та репозиторій моделей машинного навчання Pytorch-transformers.

В результаті роботи пропонується метод створення узагальнених текстів, що досягає найвищої якості стиснення при найменших витратах інформації.

MACHINE LEARNING, NATURAL LANGUAGE PROCESSING, PYTHON, TEXT SUMMARIZATION, RECURRENT NEURAL NETWORKS.

The aim of this work is to research, analysis and implementation of text summarization methods. Such methods will give the ability to automatically create short text summaries which can speed up data analysis and automate routine tasks.

Implementation methods are based on data analysis and machine learning tools such as NLTK and Spacy libraries, Pytorch framework with Pytorch-transformers model zoo, Python programming language.

As a result, a method of text summarization is proposed which achieves the best compression quality with lowest information loss.

Я, Богданова Аліна Олексіївна, студент гр. ІПЗм-19-2, здобувач вищої освіти на другому (магістерському) рівні кафедри «Програмна інженерія», заявляю: моя кваліфікаційна робота на тему «Дослідження методів узагальнення текстів», що буде представлена в екзаменаційну комісію для публічного захисту, виконана самостійно, в ній не містяться елементи плагіату і вона може бути опублікована в електронному архіві відкритого доступу ElAr KhNURE. Всі запозичення з друкованих та електронних джерел мають відповідні посилання.

Я ознайомлена з діючим положенням «Про протидію академічному плагіату в ХНУРЕ», згідно з яким виявлення плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту та застосування дисциплінарних заходів.

ЗМІСТ

ВСТУП

На сьогодення ми живемо у світі в якому відбувається вибух у кількості даних, що знаходяться у цифровому просторі. Велика кількість з них є неструктурованими текстовими даними. Незважаючи на те що ми маємо швидкий доступ до величезної кількості такої інформації більша її частина залишається для нас недоступною. Об’єм текстових документів може бути настільки великим, що люди часто виявляються перевантаженими цими даними та інформацію та не спроможні швидко та ефективно користуватися нею, використовувати її для своїх потреб.

Узагальнення тексту - це техніка для створення короткого та точного переказу певного тексту, яка повністю зберігатє суть та всю корисну інформацію, що міститься в ньому та не втрачаючє загального значення. Гарний суммаризатор викидає всю надмірну, незначну інформацію, яка не передає суттєвого значення але залишає ключові ідеї тексту за важливі факти. Здійснення узагальнення текстів може поліпшити читабельність документів, скоротити час, витрачений на дослідження інформації, і дозволити єфективно використовувати велику кількість інформації в певній галузі.

За останні роки в сфері штучного інтелекта та зокрема глибокого навчання ствся значний прорив. Ендрю Нг зі Стенфорда назвав машинне навчання новою електроенергією [1], а Microsoft і Google змінили свої бізнес-стратегії, щоб стати передовими компаніями які використовують штучний інтелект [2].

Глибоке навчання - це один із підрозділів машинного навчання, в якому для вирішення задач використовуються глибокі нейронні мережі. Він став популярним завдяки успіху методів вирішення таких проблем, як класифікація зображень (маркування зображення на основі візуального змісту) та розпізнавання мови (перетворення звуків у текст).

Одна із сфер застосування штучного інтелекту та глибоких нейронних мереж - обробка природної мови. Ця галузь не зазнала такого швидкого прориву з глибоким навчанням як, наприклад, обробка зображень. Але це змінилося за останні 2 - 3 роки [3]. Наразі існують моделі, що вирішують найскладніші задачі цієї області з доволі прийнятною якістю: голосовий пошук, інтелектуальні помічники та чат-боти стають все більш розповсюдженими елементами сучасних технологій та більше не дивують користувачів.

Алгоритми машинного навчання можна навчити розуміти документи та визначати які розділи, речення чи фрази передають важливі факти та інформацію. Це є одним з ключових елементів вирішення задачі узагальнення тексту, наряду з витягуванням семантичного змісту з тексту та задачею генерації природної мови [4].

Задача узагальнення тексту має багато варіацій за своєю специфікою та знаходить застосування у багатьох сферах: генерування заголовків новин, запис протоколів засідання, пошук інформації, підведення підсумків лекції для студентів та викладачів, написання списків задач за текстом, тощо. Створення методу для рішення задачі узагальнення текстів дозволить ефективніше використовувати текстові дані та суттєво зменшить час витрачений на обробку інформації що не важлива в поточному контексті чи повторює вже відоми концепти.

1 АНАЛІЗ СТАНУ ДОСЛІДЖЕНЬ

1.1 Застосування методів машинного навчання для обробки тексту

Обробка природної мови - це сфера застосування штучного інтелекту, яка займається обробкою, аналізом та генеруванням данних природньої мови. Спочатку для цього використовувались статистичні методи аналізу та сфера застосування називалась Комп'ютерна лінгвістика. За останні кілька років сфера машинного навчання зробила великий крок вперед, з’явились нові методи обробки даних які знайшли своє застосування і в сфері аналізу природної мови [5]. Зокрема, глибокі нейронні мережі демонструють найкращу якість роботи на більшості задач.

Виходячи з наявних даних та типу дослідницького питання науковець обирає тип навчання який буде використовуватися для вирішення задачі. Існує 3 основні групи алгоритмів навчання:

- навчання з учителем. Особливістю цього методу є те що алгоритм навчається на розмаркованому наборі даних. Тобто використовуються правильні відповіді для кожного прикладу, які алгоритм може використовувати для оцінки своєї точності на наборі даних. Більшість задач машинного навчання потрапляють до цієї категорії;
- навчання без учителя. В даному типі навчання немає правильних відповідей для вхідних даних, алгоритм намагається виявити закономірності в даних самостійного, шляхом вилучення ознак та шаблонів;
- навчання з підкріпленням. В цьому типі алгоритм працює у середовище та намагається знайти оптимальний спосіб досягнення певної мети. Коли агент (алгоритм) здійснює дії, спрямовані на досягнення мети, він отримує винагороду. Загальна мета - навчитися передбачати найкращий наступний крок для отримання найбільної остаточної винагороди.

Задачі сумаризації тексту може бути формулювана через усі 3 категорії, але найбільш розповсюдженим, канонічним способом є формулювання її як навчання з учителем [6]. Саме таке формулювання буде використовуватися в роботі.

Задачі навчання з учителем поділяються на декілька категорій:

– задачі класифікації, в яких алгоритму потрібно передбачити дискретне значення, тобто знайти приналежність вхідного значення до первого классу. Наприклад, подана на вхід фотографія яблука чи апельсина;
– задачі регресії, в яких алгоритму потрібно передбачити значення з неперервного проміжку. Наприклад, передбачити за яку ціну можна продати дом маючи його характеристики (кількість спален, площину у метрах, кількість злочинів у районі за останній рік, тощо).

Задача сумаризації тексту може бути подана як задача класифікації з учителем та без. Такі її формулювання буде описано далі у роботі.

1.2 Застосування нейронних мереж для вирішення задачі узагальнення тексту

Нейронна мережа - це набір алгоритмів машинного навчання, призначений для розпізнавання закономірностей в складних даних. Глибокі мережі можуть автоматично вивчати подання з таких даних як зображення чи текст, без введення ручних правил кодування [7]. На сьогодні глибокі нейронні мережі добре вирішувати широкий спектр задач, серед яких:

– розпізнавання людської мови;
– обробка та генерування звуків;
– пошук інформації;
– створення фейкових фото \ відео;
– перенос стилю, обличчя, емоцій між зображеннями або відео.

Нейронні мережі стали state-of-the-art підходом та широко використовуються для вирішення задач, де потрібне семантичне розуміння даних чи генерування данних, які не можуть будти закодовані формальними правилами [8]. Саме тому використання їх для задачи узагальнення тексту є доцільним та ефективним.

Елементами, з яких будується нейронна мережа називають нейрони. Вони мають ідентичну структуру та принцип роботи - вони перетворюють вхідні дані, застосовуючи нелінійну функцію до зваженої суми входів.

Проміжні виходи одного шару застосовуються як вхідні дані для наступного шару. Таким чином інформація подана на вхід нейронній мережі потрапляє через всі шари до самого останнього, де робиться передбачення. Загальна схема нейрона зображені на рисунку 1.1.

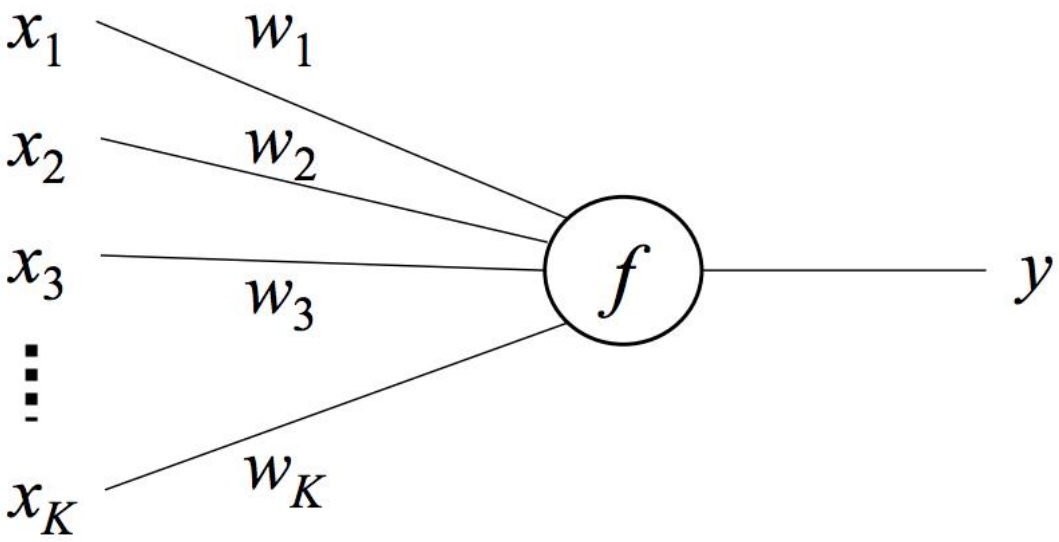


Рисунок 1.1 – Загальна схема нейрона

На етапі тренування змінюються ваги моделі (або ще називають параметри мережі), щоб мінімізувати різницю між прогнозами моделі та реальними значеннями. На прикладі нейрона на рисунку 1 ваги моделі відповідають позначенням w1, w2 … wk.

Зазвичай нейрони згруповані в шари. Кожен нейрон в шарі отримує однакові вхідні значення для своїх перетворень. В процесі навчання модель вчиться виділяти нелінійні ознаки з даних та на останніх шарах поєднує їх, щоб зробити прогноз. Тому вважають, що початкові шари моделі є екстракторами фіч (прикладами фіч можуть біти визначення чи є речення питанням, чи є в ньому пряма мова, що говориться в реченні про котиків), а на останніх шарах модель приймає остаточне рішення щодо прогнозу (ці шари називають “головою” мережі).

Текстові дані відрізняються від табличних даних чи зображень тим, що в них є послідовність. Тобто порядок слів у реченні є важливим, на відміну від таблиць де стовпці часто є незалежними один від одного чи картинок, де порядок обробки пікселів не має значення.

Рекурентні нейронні мережі (RNN) - це один з різновидів нейронних мереж, особливо корисний для обробки послідовних даних, таких як звук, дані часових рядів (наприклад, показники датчиків) або природня мова [9]. Загальна схема рекурентного нейрона наведено на рисунку 1.2.

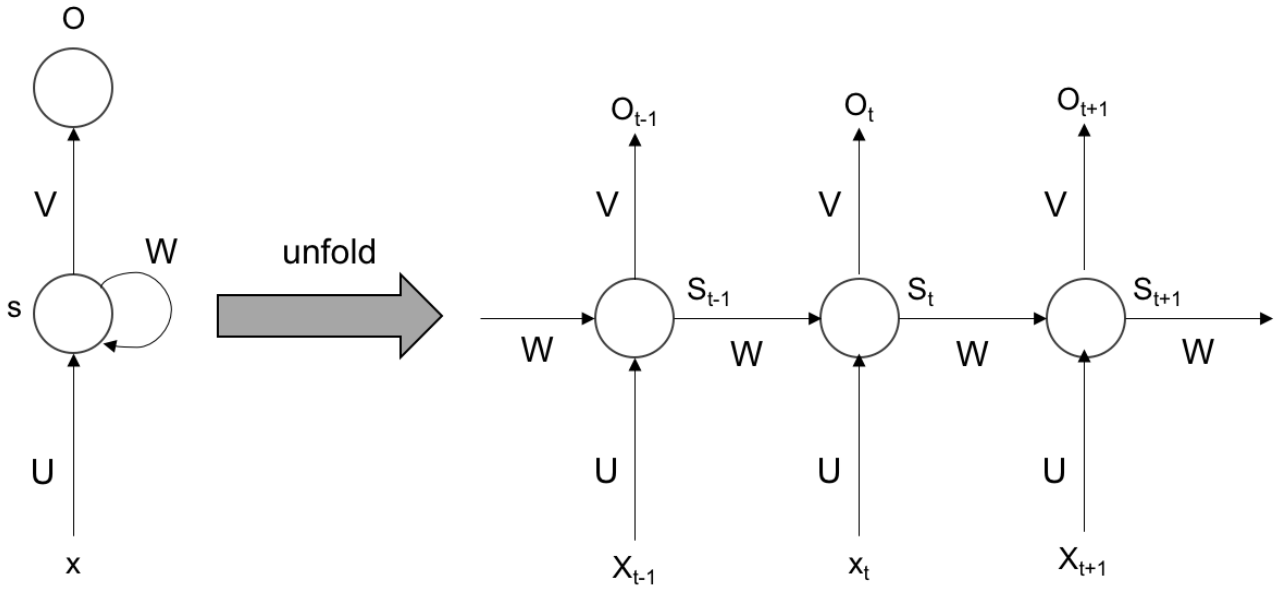


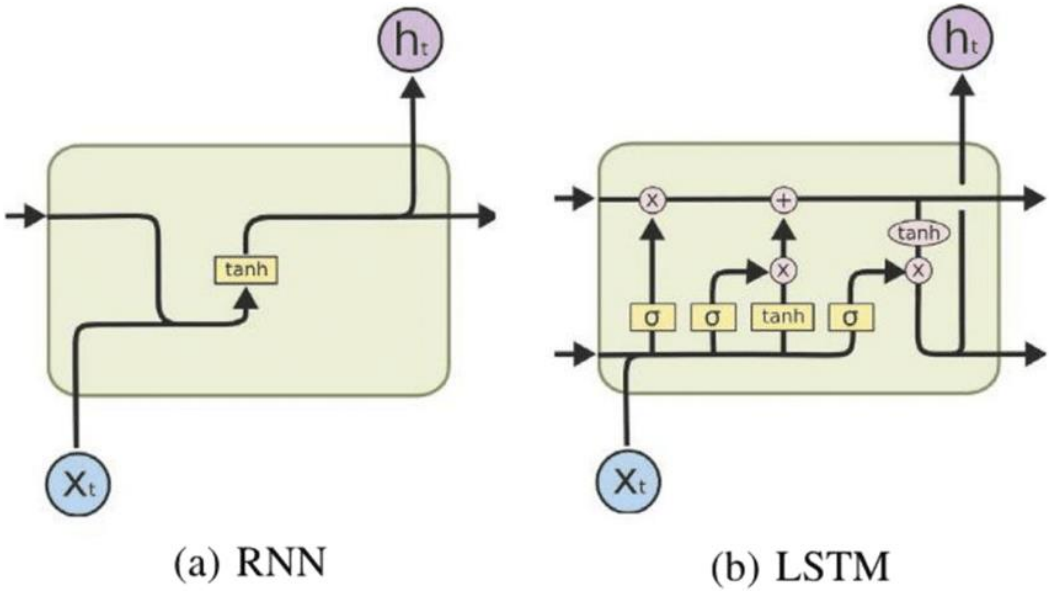
Рисунок 1.2 – Схема роботи рекурентного нейрона

Рекурентні нейронні мережі відрізняються від звичайних повнозв’язних мереж тим, що вони включають цикл зворотного зв'язку та передають поточний стан моделі для розрахунку її наступного стану [10].

У мережі є деякий внутрішній стан, який оновлюється після прочитання чергового елемента вхідної послідовності. За ідеєю, цей внутрішній стан має містити в собі інформацію про всі раніше обрацьовані елементи послідовності.

Але через те що на кожному рекурентному кроці при обчисленні внутрішнього стану мережі застосовується функція активації при достатньо довгих послідовностях рекурентні нейрони стають дуже схильні до проблеми затухаючого градієнта. Як наслідок, неможливо створити достатньо “довгі” моделі та навчити їх на існуючі задачі.

Цю проблему вирішують LSTM шари. В них більш складна структура ніж RNN, їх особливістю є те, що вони керують інформацією яка надходить з попереднього кроку [11]. Інформація може зберегтися в нейроні, передатися далі на опрацювання в наступний нейрон чи забутися. LSTM послідовності дозволяють опрацьовувати ціпі з даних довжиною до 1000, що є достатньо великим значенням для більшості задач [12]. Порівняння будови RNN та LSTM нейронів наведено на рисунку 1.3.



(a) RNN (b) LSTM

Рисунок 1.3 – Порівняння RNN та LSTM нейронів

Рекурентні мережі можуть мати один або декілька виходів в залежності від задачі яку вони вирішують. Наприклад, один вихід мають нейронні мережі що беруть на вхід текст та прогнозують 1 значення. Це можуть бути задачі сентимент аналізу, класифікації тексту, тощо. Такі мережі мають назву багато-до-одного. Мережі, що беруть на вхід послідовність та трансформують її в іншу послідовність, наприклад машинний переклад чи сумаризація тексту називають багато-до-багатьох мережами (many-to-many). Схема організації нейронів в такому випадку наведена на рисунку 1.4.

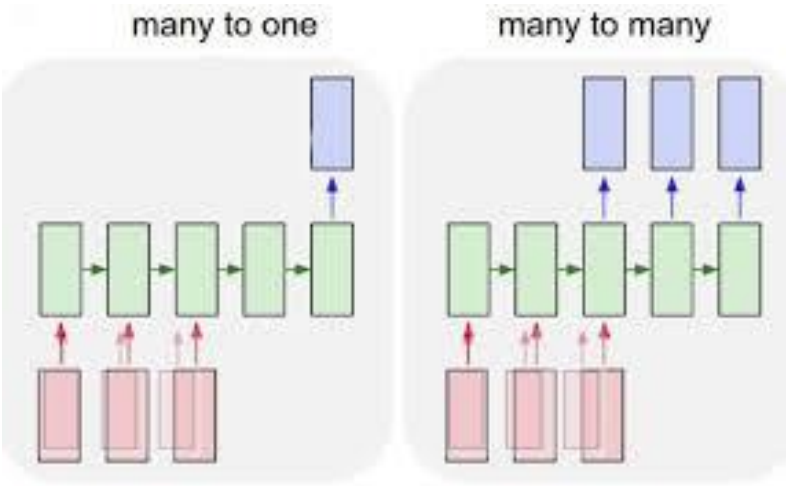


Рисунок 1.4 – Схема організації нейронів в мережі

Прикладом мережі що бере на вхід вектор та генерує вектор на виході є Sequence to Sequence моделі, які скорочено часто називають seq2seq моделями. Задача суммаризації тексту може вирішуватись seq2seq нейронними мережами, адже і вихідна інформація і результат є послідовностями, а саме текстами [13]. Загальна архітектура seq2seq моделей є такою: на початку мережі знаходиться елемент що кодує вхідну послідовність в вектор меншого розміру. Потім цей вектор бере на вхід елемент що декодує, який трансформує його до фінальної відповіді мережі [14]. Схема загальної архітектури seq2seq моделей наведено на рисунку 1.5

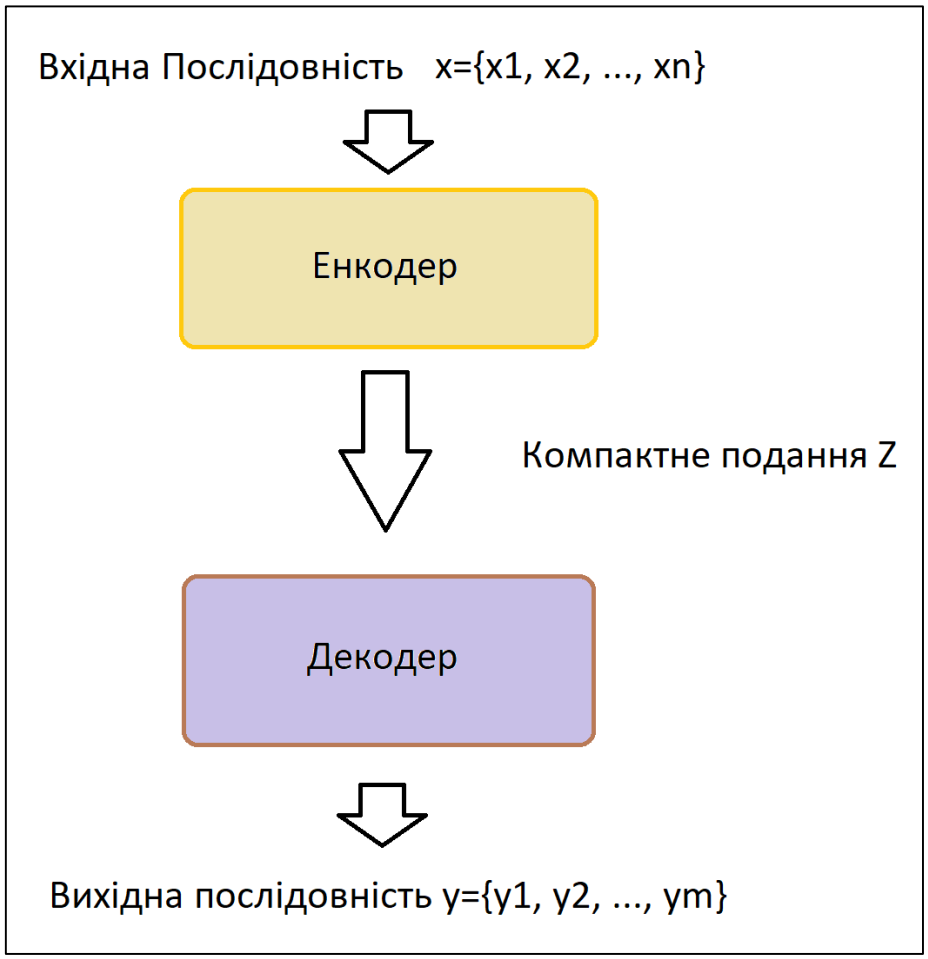


Рисунок 1.5 – Архітектура seq2seq мереж

Елементи що кодують і декодують можуть бути RNN чи LSMT шари. Вектор меншого розміру, який бере на вхід декодер є компактним поданнях вхідної послідовності, його ще називають проміжним представленням вихідного тексту [15].

Нейронні мережі, які працюють з рекурентними шарами мають вагомий недолік. На практиці в таких моделях слова, що знаходяться в кінці речення мають більший вплив на результат оскільки початкова інформація вже багато разів пройшла через рекурентні перетворення. Важлива інформація може міститися на початку чи в середині речення, тому є важливим вміння зберегти її до кінця обчислень мережі.

Вищезазначену проблему вирішує механізм уваги (attention mechanism) [16]. Цей компонент - потужний механізм, розроблений для підвищення продуктивності рекурентних нейронних мереж на основі архітектури Encoder-Decoder. Він був

запропонований для покращення якості задач машинного перекладу, але його використання не обмежується лише цією задачею та наразі його використовують в багатьох задачах обробки природньої мови [17].

Основна його ідея полягає в тому, щоб розділити операції використання інформації (перетворення та передачу її далі по нейронній мережі) та оцінки значимості цієї інформації (які слова є більш важливими в поточному контексті). Так, модель може навчитися звертати увагу на слова які є більш значимими для результату, незалежно від їх позиції в реченні.

Реалізація механіму уваги складається з таких кроків:

– спочатку до вектору з ембедингом кожного слова застосовується перетворення, яке оцінює рівень релевантності відповідного елемента тексту. Таким перетворенням може бути, наприклад, нейромережа. Чим важливіше слово – тим більше значення релевантності. Важливим є те, що для кожного параметру використовується одна й та сама мережа (з тими самими параметрами). Навчання цієї мережі відбувається одночасно з навчанням іншими параметрами;
– наступним кроком є нормування оцінок релевантності так, щоб їх сума дорівнювала одиниці. Для цього застосувується функція активації софтмакс;
– далі оцінки релевантності на відповідні значення вхідних векторів з ембедингами перемножуються. Результуюча матриця буде мати таку саму розмірність що і вихідна матриця ембедингів слів, але значення векторів-ембедингів будуть враховувати контекст, в якому знаходиться слово.

Також механізм уваги дає можливість частково інтерпретувати рішення, прийняті нейромережею. Ми можемо зрозуміти, на які ділянки вхідних даних вона звернула більше уваги, а на які – менше та зрозуміти як вона приймала фінальне рішення. Це, безумовно, є великою перевагою для її використання.

1.3 Застосування мереж трансформерів для обробки тексту

Одним з найуспішніших застосувань механізму уваги є нейронні мережі, що мають загальну назву мережі-трансформери. На сьогодення це популярна архітектура для вирішення задач розуміння та генерування природної мови. Ідея моделі нейронної мережі, що заснована на цій архітектурі, з'явилася у 2017 році та відразу стала state-of-the-art підходом для вирішення багатьох задач, таких як машинний переклад, генерация текстів природної мови, генерація відповідей на питання та ін [18]. Багато підходів що з'явилися за останній рік так чи інакше будуються на основі оригінального Transformer.

Архітектура Transformer вирішує проблеми попередніх підходів з рекурентними мережами, найголовніша з яких - можливість паралельної обробки вхідних послідовностей. В попередніх архітектурах з використанням рекурентних шарів кожен стан обчислюється після того як був обчислений попередній - тобто сама природа обчислень є послідовною. Саме тому рекурентні мережі не отримали такого великого пришвидшення роботи з використанням графічних відеокарт як, наприклад, конволюційні нейронні мережі [19].

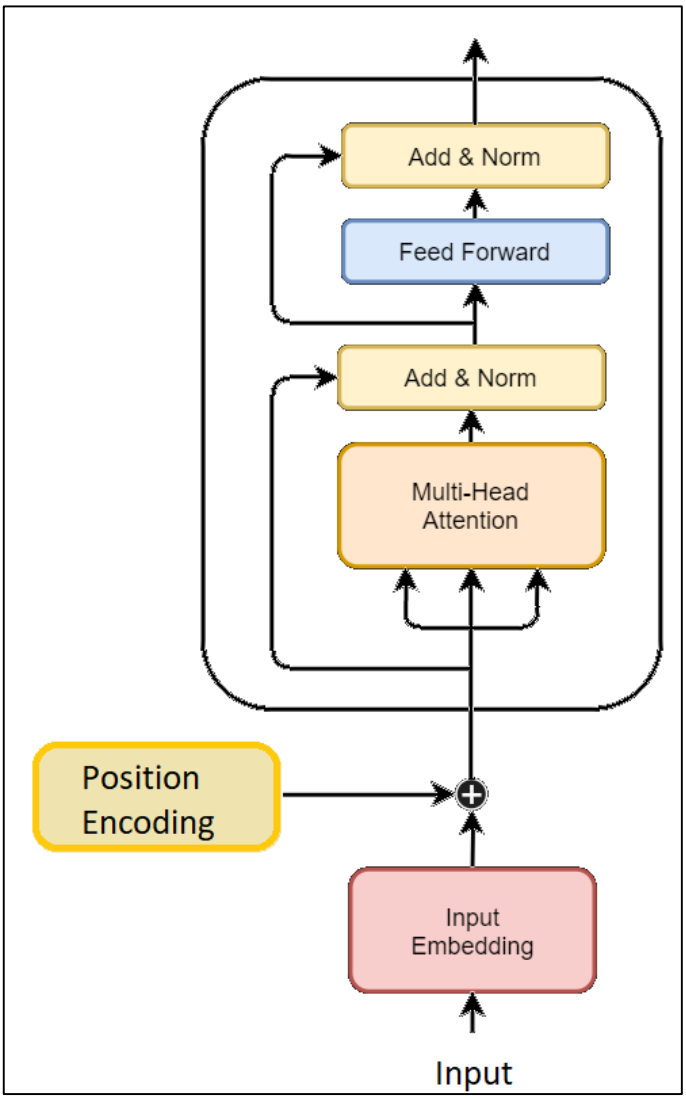
Ключовою особливістю архітектури є те що з мережі прибрали рекурентні шари повністю, залишивши лише механізми уваги та допоміжні шари: шари нормалізації даних та повнозв'язні шари. Стаття, в якій було запропоновано цю ідею має саме таку назву - Attention is all you need [20]. Таким чином за допомогою організації механізма уваги та прибирання послідовних шарів з архітектури стало можливо підтримувати паралельну обробку послідовностей та опрацьовувати усі токени одночасно, а не від токена до токена.

Моделі Transformer відносяться до категорії seq2seq моделей, тобто мають 2 частини: енкодер та декодер. Енкодер відповідає за перетворення вхідних даних у внутрішнє багатовимірне подання та складається декількох однакових блоків

кодування (в оригінальній архітектурі 6). Кожен блок складається з таких ключових компонентів:

- механізм уваги з кількома головами, що направлений на себе (self-attention block). Цей блок дозволяє для кожного токена (маркера) який ми подаємо в модель визначити які інші токени мають відношення до нього (тобто куди звертати увагу при читанні цієї лексеми / токена);
- повнозв’язний шар, який генерує 512-мірний вектор до кожного токена, що був поданий на вхід. Тобто відбувається перетворення інформації для подальшої її передачі в наступний блок енкодера;
- залишкові зв’язки, які є засобом боротьби з проблемою затухаючого градієнта в при тренуванні глибоких нейронних мереж [21]. Такі з'єднання дозволяють моделям ефективніше навчатися, оскільки градієнти можуть вільно протікати від кінця моделі до самого її початку;
- шари нормалізації. У цих шарах вихід блоку уваги або повнозв’язного шара додається до залишку що прийшов з залишкового зв’язка та нормалізується. Така процедура допомагає моделям ефективніше навчатися.

Загальна схема блоку кодування наведена на рисунку 1.6.



Рикунок 1.6 – Загальна схема енкодера трансформера

Для роботи моделі потрібна така вхідна інформація:

– ембедингн токенів вхідної послідовності. Зазвичай до кожної конкретної моделі трансформер існує свій ембединг, який навчається разом з основною мережею;

– вектор закодованої позиції слів. Він додається до вхідного вектору та модифікує його, додаючи інформацію про те де знаходиться слово у реченні.

Позиційне кодування потрібне для роботи трансформера адже токени більше не обробляються послідовно (як в рекурентних моделях), а інформація про позицію слова в реченні є корисною для аналізу. Зазвичай моделі вивчають загальне

впорядкування слів в реченні та розповсюджені словосполучення, бо це допомогає підвищити якість передбачення чи генерування тексту.

Задача енкодера в нейронній мережі – перевести вхідні дані до проміжного багатомірного представлення. Задача декодера – трансформувати це проміжне подання до передбачення. Загальна архітектура декодера дуже схожа на архітектуру енкодера – спочатку вхіні токени переводяться у вектор ембедингу та додається вектор позиційного кодування. Далі ця інформація проходить через декелька блоків декодеру, після яких відбувається її перетворення до вектора вірогідностей зустріти кожне з слів словаря на поточній позиції.

Мережі Transformers можуть використовувати окремо енкодер чи декодер мережі в залежності від задачі, яку вони вирішують. Наприклад, моделі автокодувальники (BERT, ALBERT, XLM) використовують тільки енкодер від оригінального Трансформера та застосовуються для вирішення задач на рівні токенів та речень. Наприклад, задача классифікації тексту або сентимент аналіз речень, пошук іменованих сутностей. Водночас, можелі можуть використовувати тільки декодер частину оригінального Трансформера та використовуватись для генерації тексту (називають авторегресійними моделями). Відомий праклад таких моделей – GPT від компанії OpenAI [22]. Також можливе використання повної версії, де зберігаються обидві частини можелі (і енкодер, і декодер) – повноцінні Sequence2Sequence моделі (наприклад BART чи T5). Вони викоритовуються для вирішення задач де вхідниим та віхідними даними є текст (або задача формалізована так, что на вхід та на вихід подається текстова інформація). Наприклад, модель T5 може вирішувати декілька задач обробки природньої мови, серед яких сумаризація чи сентимент аналіз [23]. Для цього перед вхідним реченням потрібно додати спеціальний токен, що дать інформацію моделі про те що їй потрібно зробити: «TL;DR» для сумаризації чи «translate» для машинного перекладу.

1.4 Підготовка текстових даних

Моделі машинного навчання не можуть брати на вхід безпосередньо текстові дані. Для їх використання необхідно закодувати іх як числа. Існує декілька способів це зробити. Зазвичай спочатку текст розбивають на речення та слова та потім кодують кожне слово окремо. Якість прогнозів може відрізнятися в залежності від способу кодування вхідного тексту, тому розглянемо кілька найпоширеніших способів це зробити.

Найпростіший спосіб - one-hot encoding. Його ідея в тому, що з вхідного документа будується словар, де для кожного слова яке зустрілося є унікальний числовий ідентифікатор. Потім кожне слово в документі представляється як вектор де на позиції номера слова стоїть одиниця, а всі інші - нулі. Таким чином відстань між будь-якими двома словами є рівною.

Іншим варіантом є TfIdf перетворення (розшифровується Text Frequency - Inverse Document Frequency). Для кожного слова в тексті розраховується коефіцієнт, який відображає наскільки важливим є конкретне слова для поточного документа в рамках колекції. Загальна формула для розрахунку значення TfIDF:

$$w_{i,j} = tf_{i,j} \cdot \log(\frac{N}{df_i})$$

де $w_{i,j}$ – значення TfIdf для слова *i* в документі *j*;

$tf_{i,j}$ – частота зустрічання слова *i* в документі *j*;

df_i – кількість документів, в складі яких є слово *i*;

df_i– загальна кількість документів.

Документом може вважатися одно чи декілька речень. Чим вище значення TFIDF - тим більш репрезентативним є слово для поточного документа. Наприклад,

якщо слово розповсюджене у корпусі (наприклад прийменники) - вони не несуть багато корисної інформації для моделі. Їх значення буде низким адже обернена частота документа буде близька до одиниці, а її логарифм до нуля. Навпроти, якщо слово зустрічається не часто та присутнє тільки в документах певної групи (наприклад слово “купувати” в задачі класифікації спаму) - його значення його частоти буде високим (адже часто воно зустрічається декілька разів в одному документі), загальне значення оберненої частоти документів значно більше 1 та, як наслідок, значення її логарифму буде додатнім числом.

Перевагою таких методів є їх простота використання - не потрібно ніяких додаткових ресурсів, використовуються лише статистики текста. Але вони мають вагомий недолік - результатом перетворення є вектор, розмір якого дорівнює довжині словаря. Це впливає і на розмір моделі, і на швидкість тренування та будування прогнозу

Метод що вирішує цю проблему - використання претренованих ембедингів. Ембединг - це пространство та функція подання до цього пространства, яке має особливість що елементи з близьким значенням знаходяться близько один до одного [24]. Тобто слова близькі за значенням мають меншу відстань один від одного ніж слова далекі за значенням.

Функція переводу до такого пространства часто є нейронною мережею та знаходиться у вільному доступі для використання. Найпоширенішими є декілька:

– word2vec - був розроблений в компанії Google в та є стандартом для використання попередньо вивченого ембедінгу [25];
– glove (Global Vectors For Word Representation) - цей эмбеддинг навчено на основі агрегованої статистики спільного виникнення слів [26]. Його використання дає виграш у якості на деяких корпусах.

Ще одна цікава властивість ембедингів - вони мають вектора “відношення” слів. Тобто наприклад відношення “Країна - Столиця” буде рівне для всіх таких прикладів,

чи відношення однакових іменників різного роду. Схематично ідея наведена на рисунку 1.7.

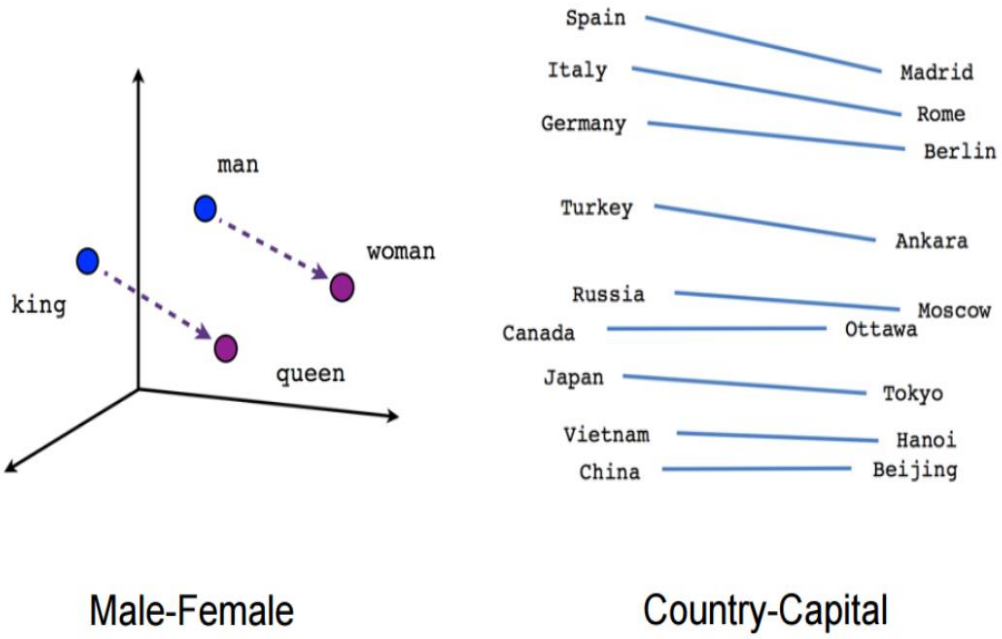


Рисунок 1.7 – Відношення у пространстві емебедингів

Саме такий підхід до подання слів та документів можна вважати одним із ключових проривів глибокого навчання у складних задачах обробки природної мови. Такі перетворення можуть бути корисні для розширення ключових слів / пошуку, семантичного пошуку по тексту та пошуку інформації.

Особливостями перелічених методів є то що вони кодують саме слова. Але значення слів майже завжди залежить від контексту речення. Тому має сенс кодувати не скільки саме слово – а скоріше сенс, який воно має в конкретному випадку.

Наразі існують ембедингі які переводять цілі речення у векторний вид, звертаючи увагу на їх зміст. Зазвичай для цього використовують ембедингі, що були навчені одночастно з відповідними мережами трансформерами.

2 ПОСТАНОВКА ЗАДАЧІ

2.1 Задача сумаризації текстів

Задача суммаризації тексту полягає у тому, щоб створити стиснення фрагменту початкового тексту до короткої версії, одночасно зберігаючи ключові ідеї та інформаційні елементи тексту, його значення.

Існує 2 типи створення стислих текстів: екстрактивний метод та генеративний (абстрактний) метод.

Техніки екстрактивного узагальнення тексту передбачають витягування ключових фраз чи речень з вихідного документа та комбінування їх разом. Рішення щодо включення фрази до результату проводиться відповідно до визначеної метрики без внесення змін до початкової фрази. Переваги цього метода:

- простота використання, адже метод не потребує великих обчислювальних ресурсів для генерування тексту;
- можливість порівняння отриманого результату стистенного тексту із запропонованою правильною відповіддю, наявність чітких метрик;
- гарантована граматична коректність результата;
- не потребує великого лінгвістичного аналізу, може виконуватись автоматично;
- можливе використання для різних мов.

До недоліків методу можемо віднести неможливість отримати нові, перефразовані речення. Усі фрази узагальнення є частинами початкового тексту.

Іншим методом створення узагальнень є генеративні (або ще називають абстрактивні) методи. Ці техніки спрямовані на створення узагальнення шляхом переформулювання речень з оригінального тексту з використанням передових методів генерації природної мови. Таким чином фрази з тесту-узагальнення не є

частинами оригінального документа, але охоплюють основну інформацію, що міститься в матеріалі та є семантично зв’язаними.

До переваг цього метода відноситься:

– створення більш цікавих текстів завдяки перефразуванню речень;
– краща якість узагальнення. Алгоритм може викидати деталі речень чи фраз які не є суттєвими, в результаті отримуючи більш стислий текст.

Серед недоліків можемо виділити:

– необхідність великої кількості обчислювальних ресурсів для роботи (генерація тексту виконується великими мовними моделями);
– для генерування правдоподібних текстів що складаються з кількох речень потрібно вирішувати додаткові задачі, такі як семантична репрезентація та перестановка слів;
– неможливість автоматичної оцінки якості узагальнення тому що є безліч вірних способів пеерфразувати текст. Зазвичай для оцінки якості абстрактивної сіммаризації використовують людей;
– можливе використання лише для мов, для яких існує модель-генератор придатної якості у вільному доступі (наразі доступні близько 20 найпоширеніших мов).

Серед перелічених недоліків є декілька істотних, а саме необхідність обчислювальних ресурсів для роботи генеративних моделей та неможливість автоматичної оцінки якості результата. Тому для подальшого аналізу розглянемо лише методи екстрактивної сумаризації.

2.2 Методи екстрактивної сумаризації

Основна ідея для екстрактивної суммаризації – розбити текст та речення чи фрази та для кожної текстової сутності визначити, чи варто її додавати в узагальнення. Цю задачу можна формалізувати декількома способами.

У випадку наявності розмічених паралельних корпусів з узагальнюючими тектами (текст в якому для кожної сутності відомо чи повинна вона потрапити в узагальнення) задачу можна формалізувати як задачу навчання з учителем – бінарна классифікація. В такому разі використовуються класифікаційні моделі. Це можуть бути невеликі моделі, такі як логістична регресія, або попередньо навчені моделі трансформери. Так як останні показують значну перевагу при роботі з текстами для порівняння ефективності підходу оберемо саме їх.

Для обробки тексту перед застосуванням моделей машинного навчання текст переводять в векторну форму: розбивають на речення чи фрази та за допомогою ембедингів переводять у векторну форму. Результуючими передбаченнями буде вектор, довжина якого дорівнює кількості текстових сутностей на вході. Значеннями будуть числа що відповідають релевонтності поточного речення. Для формування узагальнення потрібно взяти декілька найбільш релевантних речень. (Конкретна кількість буде визначена максимізацією метрики).

Важливо перед переводом моделі застосувати нормалізацію данних. Цей єтап складається з декількох кроків:

- очистка тексту від нетекстових символів (знаки пунктуації, символи, смайли, тощо);
- видалення стоп-слів. Стоп слова – це термін, що позначає найчастіше вживані слова в мові, які найчастіше не додають багато сенсу реченню та дуже часто використовуються (прийменники, частки, вигуки). Наприклад для українськох мови це слова: «під», «від», «б», «адже»;

– стеммінг чи лемматизація. Це процедура переводу слова до початкової форми. Це робиться з метою поднання різних лексичних форм одного й того самого слова до єдиного вектора, що позначає саме семантичний зміст. Наприклад, прибирається число та рід іменників, часто віднімається їх закінчення від дієслів. Наприклад, слова «бачити», «побачиш», «бачила» будуть приведені до однієї форми.

Загальна схема архітектури екстрактивної суммаризації як задачі бінарної классифікації наведена на рисунку 2.1.

Рисунок 2.1 – Загальна схема роботи екстрактивної суммаризації як задачі бінарної классифікації

Задачу екстрактивної сумаризації можна сформулювати як задачу навчання без учителя. В такому випадку наявність паралельного розміченого корпусу не є

обов’язковою, що є великою перевагою при вирішенні задач для непопулярних мов. Алгоритми, що показали себе досить добре для задачі сумаризації текстів – графові алгоритми. Для їх застосування, так само як і з задачами навчання з учителем, текст спочатку розбивають на речення чи фрази та очищують та нормалізують. Далі його переводять до графу, вершинами якого є текстові сутності (фрази чи речення), а ребрами показані зв’язки між ними. Так, ребра можуть відповідати семантичним зв’язкам у реченні, лексичним відносинам чи показумати семантичну схожість речень. Функція, що закодована у ребрах обирається відповідо до задачі, що вирішується.

Отже, задачею данної магістерської роботи є дослідити ефективність алгоритмів екстрактивної суммаризації як бінарної классифікації (навчання з учителем) та виконання сумаризації графовими методами (навчання без учителя).

2.3 Метод оцінки якості суммаризації

Для оцінки якості екстрактивної суммаризації найчастіше використовується метрика ROUGE (Recall-Oriented Understudy for Gisting Evaluation) [27]. Вона порівнює згенерований узагальнений текст з референсними документами (найчастіше ці документи створені людиною), тому може використовуватись і в інших задачах обробки природної мови, такі як машинний переклад. Ця метрика оперує поняттями точності передбачення, покриття передбачення та метрики, що узагальнює 2 вищезазначені показники – F-міра. Ці показники широко використовуються для оцінки якості передбачення в задачах класифікації, тому розглянемо їх більш докладно.

Для обчислення метрик нам знадобляться умовні позначення передбачень моделей машинного навчання. Їх умовно можна розділити на ті що були передбачені вірно та помилково, та розділити на ті що передбачили позитивний та негативний клас. Загалом, 4 типи:

– вірно-позитивні (TP – True Positive) - що позначають, що були вірно передбачено позитивний клас;
– невірно-позитивні (FP – False Positive) - що позначають, що правильний клас приклада був негативний, а модель передбачила позитивний (і це є помилкою передбачення);
– вірно-негативні (TN – True Negative) - що позначають, що було вірно передбачено негативний клас;
– невірно-негативні (FN – False Negative) - що позначають, що правильний клас приклада - позитивний, а модель передбачила негативний (і це є помилкою передбачення).

Описані вище помилки можна описати такою схемою на рисунку 2.2:

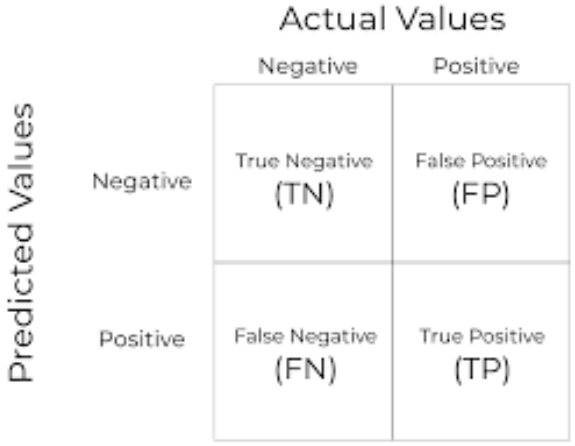


Рисунок 2.2 – Класифікація помилок моделі машинного навчання

Точність передбачення (позначається P – precision) - це відношення кількості правильно передбачених позитивних прикладів к загальній кількості позитивних

передбачень. Ця метрика дає оцінку відношення правильності передбачення позитивного класу. Точність можна обчислити за формулою:

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

де TP – кількість вірно-позитивних передбачень;

FP – кількість невірно-позитивних передбачень.

Покриття передачення (позначається R – recall) - це відношення правильно передбачених позитивних прикладів к загальній кількості позитивних прикладів. Тобто ця метрика позначає відношення знайдених позитивних прикладів з вибірки. Покриття можна обчислити за формулою:

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

де TP – кількість вірно-позитивних передбачень;

FN – кількість невірно-позитивних передбачень.

Ці 2 метрики окремо дають оцінку якості передбачення, але не завжди зручно оперувати двома числами при виборі найкращого результата. Для об’єднання цих двох метрик використовується їх середнє гармонічне, що має назву F-міра та обчислюється за формулою:

$$F = \frac{2PR}{P + R}$$

де P – точність передачбення;

R – покриття передбачення.

Метрика ROUGE визначає точність, покриття та F-міру між двома текстами та має декілька варіацій в залежності від того, які тестові сутності використовуються при виконанні оцінки:

– метрика ROUGE-1 використовує окремі токени для оцінки. Позитивними вважаються токени що повинні потрапити у узагальнення, негативними - ті що не повинні бути в узагальненні;
– метрика ROUGE-2 обчислюється за такою самою схемою, як і ROUGE-1, але використовуються біграми замість окремих токенів. Біграми - це послідовність з двох токенів, що стоять поряд у реченні. Наприклад, у словосполученні “обробка природної мови” є 2 біграми: “обробка природної” та “природної мови”;
– метрика ROUGE-L використовує найдовшу загальну підпослідовність для оцінки якості узагальнення. Позитивними вважаються токени що відносяться до найдовшої загальної підпослідовності між передбаченим узагальненням та правильною відповіддю. Використовується рідше ніж ROUGE-1 та ROUGE-2 [28].

Для оцінки якості узагальнення в роботі пропонується використання всіх трьох перелічених метрик, оскільки разом вони дають більш повну картину якості ствоерної сумаризації.

3 ОПИС ПРОВЕДЕНИХ ДОСЛІДЖЕНЬ

3.1 Особливості реалізації графових методів

Алгоритм ранжування графів - це алгоритми, що обчислюють важливості вершин у графі на основі інформації про його структуру. Такі алгоритми добре себе показали на різних застосуваннях задачі ранжування [29]. Найпростіші імплементації використовують ступінь вершини як її ранк, або максимальну вагу ребра яке виходить з неї як її релевантність. Однак, такі алгоритми не показують високої якості при вирішенні задач.

Один з алгоритмів, що ранжує вершини в графі та показую досить гарний результат має назву PageRank. Раніше він використовувався для ранжування веб сторінок у пошуковій системі Google [30]. Цей алгоритм визначає числове значення для важливості поточного документа до пошукового запиту серед набору документів, що йому подібні.

Особливістю PageRank є те що для розрахунку важливості веб сторінки він використовує зв’язки цього документа з іншими, тобто інформацію з яких сторінок є посилання на цей документ, та на які сторінки посилається цей документ. Коли одна вершина посилається на іншу, це, по суті, додає певну кількість “голосів” для цієї вершини. Чим більша кількість голосів має вершина, тим вище значення рангу. Важливість вершини також визначає скільки “голосів” впливу вона має на вершини, на які вона посилається, і ця інформація також враховується в процесі ранжування [31].

Самі значення PageRank розраховується рекурсивно. Це ітеративний алгоритм, яких працює “до сходження”. Після того як всім вершинам визначено ранг - потрібно відсортувати їх у зворотному порядку. Так, найбільш важливі та релевантні вершии (веб сторінки) потрабпляють вище у пошуковій видачі.

Формула, за якою розраховується релевантність поточної вершини:

$$PR(x) = \frac{1-\lambda}{N} + \lambda \sum_{y \to x} \frac{PR(y)}{out(y)}$$

де *PR(X)* – поточний ранк вершини X;

N – загальна кількість вершин в графі;

λ – коефіцієнт, що зазвичай дорівнює 0.85;

out(Y) – кількість вихідних вершин з вершини Y.

На початку всі вершини мають рівний ранк, що дорівнює 1 / N. Далі ітеративно обчислюються нові ранки для кожної вешини поки не буде зроблено максимальна дозволена кількість ітерацій, або поки алгоритм не зійдеться. Критерій сходження алгоритму PageRank:

$$|R(t+1) - R(t)| < \varepsilon$$

де *R(t)* – вектор ранків всіх вершин графа на етапі t;

ε – константа, що дорівнює 10^{-6}.

Цей алгоритм має адаптування для роботи з текстовими графами - TextRank, який є працює за такою самою схемою як і оригінальний алгоритм [32]. Вершинами графа є текстові структури (речення, слова, абзаци, документи, тощо). Ребрами текстового графа є залежності або зв’язки цих текстових сутностей між собою. Наприклад, схожість за сенсом, лексичні чи семантичні відносини, довжина найдовшої загальної підпослідовності, тощо.

Для застосування TextRank потрібно спочатку перевести текст природної мови до форми текстового графа [33]. Загальний алгоритм переводу тексту до графу складається з таких етапів:

– визначити, які текстові сутності найкраще підходять для вирішення задачі;

– виділити ці сутності з тексту та додати як вершини графу;

– визначити відношення, яке зв’язує текстові сутності між собою та які будуть закодовані ребрами графу;

– обчислити відношення для всіх необхідних пар вершин графу.

Для задачі узагальнення тексту найкраще підходять такі сутності як речення та фрази. Перевагою вилучення речень є те, що фінальний текст є граматично коректним, адже при з'єднанні фраз до речень можуть виникнути помилки. Перевагою вилучення фраз є більша атомарність при обчисленнях: інформація в різних реченнях може дублюватися, що приведе до її дублювання в результуючому узагальнені.

Існує багато способів оцінити відстань між векторами та кожен зі способів може бути оптимальним для вирішення конкретної задачі. Найбільш популярними варіантами в сфері машинного навчання є евклідова відстань, манхетенська відстань, та косинусна відстань.

Евклідова відстань визначається формулою:

$$d(p,q) = \sqrt{\sum_{k=1}^{n}(p_k - q_k)^2}$$

де *d(x, y)* – відстань між x та y;

p, q – вектори, між якими обчислюється відстань;

n – довжина обох векторів;

k – індекс.

Евклідова відстань є найбільш інтуітивною метрикою сереж запропонованих та визначає відстань між двома точками.

Манхатетенська відстань, яку ще називають метрикою житлових кварталів – пов’язана з особливістю будови житлових кварталів на Манхетені. В разі двомірного простору вона визначає відстань між двома точками за умови, що дозволено рухатися лише паралельно осям координат. Загальна формула для обчислення:

$$d(p,q) = \sum_{k=1}^{n} |p_k - q_k|$$

де *d(x, y)* – відстань між x та y;

p, q – вектори, між якими обчислюється відстань;

n – довжина обох векторів;

k – індекс.

Але найбільш популярна та найчастіше застосовується в для тектових задач в сфері машинного навчання метрика косинусної відстані. Загальна формула для обчислення:

$$d(p,q) = \frac{p \cdot q}{\|p\|\|q\|} = \frac{\sum_{k=1}^{n} p_k q_k}{\sqrt{\sum_{k=1}^{n} (p_k)^2}\sqrt{\sum_{k=1}^{n} (q_k)^2}}$$

де *d(x, y)* – відстань між x та y;

p, q – вектори, між якими обчислюється відстань;

n – довжина обох векторів;

k – індекс.

Ця метрика використовує вугол між двома векторами як міру їх відстані. Так, однонаправлені вектори будуть мати нульову відстань.

Оригінальна метрика відстані алгоритму TextRank використовує таку формулу відстані між текстовими сутностями:

$$d(S,V) = \frac{|\{w_k | w_k \in S \& w_k \in V\}|}{\log(|S|) + \log(|V|)}$$

де *d(x, y)* – відстань між x та y;

w_k – слово з індексом k;

S, V – текстові сутності, між якими обчислюється відстань.

Порівняємо ефективність перелічених метрик відстані для вирішення задачі екстрактивної сумаризації графовими методами.

Також проведемо експеримент з використанням різних ембедингів для порівняння семантичної схожості текстових сутностей замість TF-IDF векторів, що використовуються в оригінальному алгоритмі TextRank. Використаємо такі кодування:

– вектори TF-IDF текстових сутностей;

– вектори ембедингів, отриманих з переднавченої моделі ALBERT;

– вектори ембедингів, отриманих з переднавченої моделі DistillBERT.

Для реалізації алгоритму TextRank та усіх етапів підготовки тексту використаємо бібліотеку для обробки природньої мови SpaCy [34] на мові програмування Python. Для роботи з нейронними мережами, зокрема з моделями трансформерами використаємо фраемворк для тензорних обчислень PyTorch [35] та репозиторій претренованих моделей pytorch-transformers [36], де зберігаються різні натреновані моделі в вільному доступі.

3.2 Особливості реалізації сумаризації як бінарної класифікації

Для реалізації екстрактивної сумаризації як задачі навчання з учителем використамо поточний state-of-the-art підхід з використанням моделей з сімейства Transformer, а саме моделі DistillBERT [37] та ALBERT [38]. Ці моделі відносяться до категорії автокодуючих моделей, тобто вони складаються лише з енкодер блоку оригінального трансформера.

BERT – одна з найперших моделей трансформерів, що завдяки своїй архітектурі здатна навчатися на різні задачі обробки природньої мови (раніше для цього було потрібно робити модіфікації в архітектурі) [39]. Завдяки цьому вона швидко стала

відомою та популярною. На сьогодні вона досі є state-of-the-art підходом для ряда задач.

Для того щоб опрацьовувати задачі різного роду (на рівні речень це можуть бути задачі сентимент аналіза чи перефразування чи на рівні окремих токенів, як наприклад виявлення іменованих сутностей) в моделі BERT вхідні дані повинні подаватися в спеціальному форматі. Так, наприклад, на вхід може подаватися одно чи 2 речення, що розділені спеціальним токеном [SEP]. Це потрібно для вирішення таких задач як, наприклад, Question-Answering. Тоді одним з речень буде питання, а іншим – відповідь на нього. Тому надалі в термінології будемо використовувати термін «вхідна полідовність», яка позначає довільний проміжок тексту що подається на вхід моделі. Це може бути фраза, 1 чи 2 речення упаковані разом. Також, на початку вхідної послідовності потрібно додати спеціальний токен [CLS]. Саме він після інференсу моделі буде зберігати векторну семантичну репрезентацію вхідної послідовності (ембединг), що може далі використовуватися для, наприклад, задач классифікації, сентимент аналізу чи семантичного пошуку. Загальна схема структури вхідної послідовності для мережі BERT наведено на рисунку 3.1

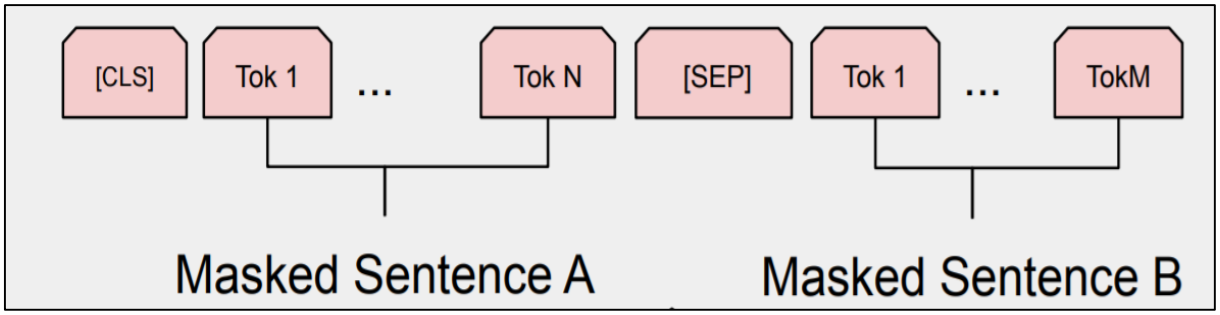


Рисунок 3.1 – Загальна схема структури вхідної послідовності для мережі BERT

Для будування вхідної послідовності використаємо BERTTokenizer, що міститься в бібліотеці Pytorch-Transformers разом з самою моделлю. Кожено токен він спочатку нормалізує до початкової форми, а далі будує семантичной ембединг

використовуючи підхід WordPiece [40]. Перед поданням в модель до вектора кожного токена додається закодований вектор позиції. Це робиться для того щоб зберегти інформацію про порядок слів у реченні [41].

Модель ALBERT – A Lightweight BERT – це модель подібна BERT за архітектурою, але має ряд змін що зменшують її розмір не витрачаючи в якості виконання задач [42]. Ключові відмінності моделі ALBERT від моделі BERT:

– зменшений розмір вектора представлення між блоками. Автори припускають, що початкові вектори ембедингів є контекто-незалежними, тому потребують більше простору для кодування вхідної послідовності;
– шаринг параметрів між блоками нейронної мережі для меншення загального об’єму моделі;
– незначна модифікація лосс функції: замість подавання на вхід двох випадкових речень з тексту та передбачення чи стоять вони поряд у тексті (передбачення «наступності» речення) подаються 2 речення що гарантовано стоять поряд, але можливо переставленні місцями. Задача моделі – передбачити, чи подані речення в прямому порядку. Такий лосс називається передбачення «впорядкованості» речень.

Модель DistillBERT має в 2 рази менше параметрів ніж оригінальний BERT за рахуток зменшення глибини мережі (6 енкодер блоків замість 12). Модель була навчена за методом переносу знань (knowledge distillation), тобто її вчили передачувати такі самі відповіді та вірогідності як і оригінальна, велика модель, щоб отім використовувати її замість великої моделі BERT мачи ті самі відповіді але викоритовуючи меньше ресурсів.

При використанні моделей трансформерів застосовується ширико відомий прийом переносу навчання (transfer learning). Його ідея в тому що навчання мережі починається не з випадково ініціалізованих параметрів, а з претренованого стану моделі на деякому іншому датасеті та можливо навіть на іншій задачі. Зазвичай, цей датасет є набагато більший і модель здатна вивчити більш генералізовані паттерни.

Цей підхід працює, бо всередині моделей нейрони згруповані в шари. Вважається, що початкові шари моделі працюють на визначення фіч, що будуть далі використовуватися для прийняття рішення. Чим раніше шар стоїть до початку мережі - тим більш низькорівневі фічі він витягає. Наприклад, в задачі класифікації тварин по фотографії один з останніх шарів що витягає фічі може працювати з досить високорівневими ознаками, такими як голова кота чи текстура шерсті кролика. Шар, що стоїть перед ним витягає менш конкретні, але все ще зрозумілі фічі, такі як око, плавник чи копито. Найперші шари нейронної мережі зазвичай важко зрозуміти, адже вони вивчають найпростіші закономірності – перехід кольору від світлого до темного вертикально чи горизонтально, або пляма певного розміру у деякій частині рецептивного поля. Використовуючи вже працездатні параметри для ініціалізації мережі можна здачно пришвидчити процес навчання, адже найперші шари не зазнають значної зміні параметрів . А також забезпечети кращу сходимість нейронної мережі та іноді навіть досягти кращих результатів, порівнюючи з випадково ініціалізованою мережею.

Також значною перевагою прийому переноса навчання є те, що переднавчання проводиться на значно більшому датасені ніж зазвичай є в наявності для навчання конкретній задачі. Тому, завдяки прийому переноса навчання, модель лише трохи адаптує ваги під вирішення поточної задачі, а не не пененавчається на поточний маленький датасет.

На практиці такий підхід працює дуже добре як в задачах машинного зору, де претренування проводиться на таких великих датасетах як Imagenet [43] або MS COCO [44], так і на задачах обробки природньої мови, де претренування моделей проводиться на задачі машинного переклада чи генерування природньої мови.

Моделі трансформери зазвичай йдуть у декількох розмірах, що відрізняються шириною векторів що проходять між блоками, кількістю блоків, розміром ембедингів що подається на вхід. Так, більші моделі зазвичай демонструють кращу якість вирішення задач але водночас потребують більше ресурсів для роботи.

Для експериментів будемо використовувати такі претреновані моделі:

– модель DistillBERT;

– модель MobileBERT.

Моделі були обрані з розрахунку модливості їх донавчання для задачі узагальнення текстів, а саме розмірів цих моделей та доступних ресурсів на платформі Google Collaboratory [45].

Для порівняння в якості класифікаторів оберемо звичайний лінійний классифікатор та LSTM шар, якому передамо ембединги отримані з моделі трансформера.

Для роботи з нейронними мережами, так само використаємо фраемворк для тензорних обчислень PyTorch та репозиторій претренованих моделей pytorch-transformers.

4 АНАЛІЗ РЕЗУЛЬТАТІВ

4.1 Функції відстані в графових методах

Однією з ідей покращення якості текстової суммаризації було використання інших функцій схожесті текстових сутностей. Різні функції дають різну внутрішню структуру графа та можуть приводити до кращого результату, в порівнянні з оргінальною функцією схожесті речень.

Тестування функцій схожесті будемо проводити на TF-IDF векторах вихідних речень, бо саме такий спосіб аналізу було запропоновано в оигінальному підході. Для цього використаємо класс TfidfVectorizer з бібліотеки scikit-learn. Цей клас автоматично сформує словник зі слів, що використовуються в тексті. В статті є посилання на те що автори викоритовували обмеження на розмір словника 30000 слів, тож зазначимо параметр max_features=30000. Кожен документ був стиснений до 20% від його початкового розміру.

Порівняння результатів використанні різних функцій відстані наведено в таблиці 4.1:

Таблиця 4.1 – Порівняння використання різних функцій відстані

Назва функції відстані	ROUGE-1	ROUGE-2	ROUGE-L
TextRank	0.3136	0.1419	0.287
Евклідова відстань	0.2945	0.1257	0.2561
Манхеттенська відстань	0.2888	0.1202	0.2493
Косинусна відстань	0.3156	0.1479	0.2935

Як бачимо, метрика відстані, що запропонована в оригінальній статті TextRank працює досить добре в порівнянні з такими популярними метриками як евклідова відстань чи манхеттенська відстань (в середньому на 0.02 та 0.025 умовні одиниці відповідно). Але косинусна відстань демонструє незначне покращення результату на на 0.03 – 0.06 умовні одиниці. Це може буди зв'язано з особливістю побудови векторів: оскільки косинусна відстань оперує саме векторами та вимірює вугол між ними (наскільки вони сонаправлені), а не оперує конкретними текстовими сутностями, як це робить метрика з TextRank – маємо змогу покращити результати роботи алгоритму.

Усі реалізації перелічених метрик мають однакову асимптотику та працюють досить швидко. Тому використання іншої функції відстані не дає перевагі в швидкодії.

4.2 Використання векторів ембедингів в графових методах

Порівняємо якість суммаризації при використанні векторів ембедингів замість TF-IDF векторів, що запропоновані в ориганільній статті TextRank. Для цього використаємо моделі ALBERTlarge, ALBERTxlarge та DistillBERT з бібліотеки pytorch-transformers. Були обрані саме такі моделі, бо вони потребують відносно меншого об'єму пам'яті, що необхідна для їх роботи ніж інші моделі трансформери, та водночас ці моделі демонструють гарну якість на вирішенні багатьох задач природньої мови. Характеристики моделей в порівнянні з моделлю BERT наведено в таблиці 4.2.

Таблиця 4.2 – Основні характеристики моделей

Характеристика	BERTbase	ALBERTlarge	ALBERTxlarge	DistillBERT
Розмір	110M	18M	60M	66M
Заявлена метрика якостіі в порівнянні з BERTbase	-	+3%	+5%	-3%
Швидкодія	1x	0.36x (повільніше)	0.12x (швидше)	4x (швидше)
Розмір датасету на етапі претренування	16Gb	16Gb	16Gb	16Gb
Розмір вектора ембедингу	768	1024	2048	768

Тож використовуючи ці моделі маємо змогу перевірити як впливає розмір моделі на якість суммаризації: чи буде більша модель давати кращі ембединги, що описують семантичний зміст речення і чи впливає це на метрики що оцінюють репрезентативність узагальнення.

Для отримання ембедингів завантажимо моделі та виконаємо інференс для кожного речення з датасету. Вектор, що знаходиться на першому місці (він відповідає спеціальному токену [CLS]) буде зберігати необхідний вектор. В реалізації моделей в бібліотеці трансформери можемо отримати цей вектор через параметр last_hidden_state в об’єкті з результатами роботи моделі. Далі використаємо його так само, як ми використовували TF-IDF вектори в первому експерименті.

Використовуючи косинусну функцію відстані, що показала себе найкраще в попередньому експерименті, порівняємо якість суммаризації. Результати експериментів з використанням різних моделей наведено в таблиці 4.3.

Таблиця 4.3 – Результати використання різних моделей

	ROUGE-1	ROUGE-2	ROUGE-L
ALBERTlarge	0.3839	0.1551	0.3469
ALBERTxlarge	0.3985	0.1634	0.3590
DistillBERT	0.3819	0.1518	0.3393

Як бачимо, використання ембеденгів з претренованих моделей транформерів одразу дає якість на декілька порядків вище ніж TF-IDF вектори. А самі вектори значно меншої розмірності: 30000 для TF-IDF та 768 – 2024 для ембеденгів трансформерів. Це пов’язано з тим, що TF-IDF вираховуються виключно зі статистичних параметрів тексту: наскільки часто слово вживано у тексті та як часто зустрічаються тексти, те присутнє поточне слово. Речення можуть мати схожі TF-IDF вектори але мати координально різні значення, тому що для побудови вектора ніяк не використовується контекст та порядок, в якому ці слова знаходяться. На відміну від TF-IDF, моделі трансформери викоритсовують інформацію про контекст та кодують саме семантичний змість вхідної послідовності. Тому при їх порівнянні маємо значно кращі результати.

Модель ALBERTxlarge показала трохи кращий результат ніж ALBERTlarge і це очіковано. ALBERTxlarge більша за ALBERTlarge як за загальним розміром (60M та 18M параметрів відповідно), так і за розміром внутрішнього представлення (2048 та 1024). Більша модель здатна вивчити більше закономірностей та виявити складніші залежності, їй достатньо виразності для того щоб ці залежності знайти. Всі інші характеристики моделей однакові: ті самі дані, той самий пайплайн навчання. Тому кращий результат ALBERTxlarge збігається з загальним правилом з машинного навчання – що більше модель – то кращий результат.

Моделі ALBERTlarge та DistillBERT показали приблизно однакові результати (ALBERTlarge на 0.005 умовних одиниць вище за DistillBERT). Обидві моделі були створені з метою пришвидшити оригінальну архітектуру BERT та зробити її менш

ресурсовитратною. Однак для цього використовувались різні прийоми: для ALBERT використовувася шаринг параметрів мережі (зберігаючи загальну кількість операцій), тоді як модель DistillBERT була отримана через процедуру дистиляції оригінальної моделі BERT та має в 2 рази меньше параметрів. За умови практично однакової якості виконання задачі сумаризації модель ALBERTlarge є кращою через те що потребує меньше ресурсів для роботи.

Загалом зміна TF-IDF вектора на ембединг з моделі-трансформера та заміна функції відстані з оригінально запропонованої на косинусну відстань дає приріст метрики в сердньому на 0.085 для ROUGE-1, 0.022 для ROUGE-2 та 0.072 для ROUGE-L. Загальне порівняння запропонованих підходів наведено в таблиці N. Метрика якості MEAN ROUGE інколи використовується науковцями для порівняння підходів та дорівнює середньому арифметичному ROUGE-1, ROUGE-2 та ROUGE-L. ЇЇ можна обчислити за формулою:

$$M\_ROUGE = \frac{ROUGE\_1 + ROUGE\_2 + ROUGE\_L}{3}$$

де ROUGE_1 – значення метрики ROUGE-1;

ROUGE_2 – значення метрики ROUGE-2;

ROUGE_L – значення метрики ROUGE-L.

При оцінці необхідної оперативної пам’яті для роботи моделі не враховується розмір текстового документа, що проходить оцінку адже при використанні будь якого з методів його розмір не змінюється. До уваги береться розмір моделі та словаря, що використовується для корректної роботи метода.

Порівняння якості виконання сумаризації з узагальненою метрикою MEAN ROUGE та об’ємом необхідної оперативної пам’яті наведено в таблиці 4.4.

Таблиця 4.4 – Порівняння ресурсів при використанні різних підходів

Підхід, що оцінюється	MEAN ROUGE	Оперативна пам’ять
TextRank	0.2475	<10МБ
TextRank + cosine distance	0.2523	<10МБ
TextRank + ALBERTlarge embedding + cosine distance	0.2953	144МБ
TextRank + ALBERTxlarge embedding + cosine distance	0.3069	480МБ

Отже, за умови наявності ресурсів на обробку тексту моделями тренсформерами краще використовувати саме їх, бо якість текстової сумаризації покращується при заміні векторів TF-IDF на вектори ембединги. Якщо ресурсів на опрацювання нейронними мережами немає – варто лише змінити функцію відстані на косинусну відстань. Ця зміна не несе ніяких змін в кількості ресурсів, необхідних для роботи метода та в той самий час дає невеликой приріст в якості.

4.3 Вирішення суммаризації як бінарної классифікації

Для порівняння вирішення задачі сумаризації як задачі бінарної классифікації було обрано моделі DistillBERT та MobileBERT. Це достатньо популярні моделі, що найчастіше використовуються в умовах обмеженості обчислювальних ресурсів адже вони є меншими порівняно з іншими моделями транформерами. Порівняння цих моделей за ключовими характеристиками наведена в таблиці 4.5

Таблиця 4.5 – порівняння моделей DistilBERT та MobileBERT

	DistillBERT	MobileBERT
Розмір	66M	25M
Заявлена метрика якостіі в порівнянні з DistillBERT	-	+ 0.5 - 1%
Швидкодія	1x	1x
Розмір вектора ембедингу	768	512

Для донавчання моделей на задачу текст сумаризації використаємо версії моделей, що мають голову для текстової классифікації. Для реалізації простого лінійного класифікатора додамо 1 лінійний шар після ембедингів з сигмоідальною функцією активації для отримання вірогідності на виході моделі. Тобто передбачення моделі буде йти за формулою:

$$y = \sigma(WE + b)$$

де W – параметр ваг лінійного шару, що навчається;

B – параметр баясу лінійного шару, що навчається;

E – ембединги, отримані з останніх шарів моделі трансформера;

σ – сигмоідальна функція активації.

Для реалізації LSTM класифікатора викристаємо клас LSTM з бібліотеки Pytorch для якого вкажемо розмірність вихідного вектора 1 – для классифікації, а всі інші параметри залишимо без змін.

Порівняльна таблиця результатів при використанні MobileBERT моделі наведена в таблиці 4.6.

Таблиця 4.6 – Результати експериментів для моделі MobileBERT

	ROUGE-1	ROUGE-2	ROUGE-L
Лінійний класифікатор	0.4225	0.1921	0.3861
LSTM	0.422	0.1916	0.3859

Як бачимо, при використанні лінійного класифікатора, що використовує лише інформацію з [CLS] токена результати сумаризації є кращими ніж при використанні LSTM шара для класифікації. Це може буде пов’язано з особливостями кодування інформації: LSTM шар викоритсовує виходи зі всіх токенів вхідної послідовності, тоду як лінійний шар використовує лише 1 вектор ембедингу вхідної послідовності. Тобто цей токен ембедингу речення трохи краще кодує семантичний зміст речення ніж LSTM шар. Також можливою причиною гірших результатів є недолік саме рекурентрих шарів, а саме приділяння більше уваги останнім частинам послідовності. Найважливіший вектор [CLS] стоїть на початку вхідної послідовності, тому його вплив на фінальне рішення може бути меньше ніж вплив останніх токенів.

Порівняння результатів при використанні DistillBERT моделі наведено в таблиці 4.7.

Таблиця 4.7 – Результати експериментів для моделі DistilBERT

	ROUGE-1	ROUGE-2	ROUGE-L
Лінійний класифікатор	0.4258	0.1943	0.3882
LSTM	0.4232	0.2004	0.3871

Так само бачимо, що використання лінійного класифікатора дає дещо кращі результати порівняно з LSTM щаром (різниця в середньому 0.15 умовних одиниць).

Тому можемо зробити вистоновок, що при використанні підхода з бінарною классифікацією лінійні классифіційні шари працюють краще за рекурентні, тому варто використовувати саме їх.

Порівняємо якості классифікації при використанні лінійного классифікатора з різними векторами ембедингів. Ниже приведена таблиця 4.8 з порівнянням моделей DistillBERT та MobileBERT. Для тренування використовувалось середовище розробки Google Colaboratory з увімкнено GPU NVIDIA P100 [46].

Таблиця 4.8 – Порівняння якості та ресурсів необхідних для роботи

	Mean ROUGE	Час тренування
DistillBERT	0.3361	25 хв \ епоха
MobileBERT	0.3335	13 хв \ епоха

Бачимо, що використання моделі DistillBERT дає кращі результати за MobileBERT, але потребує більше часу на тренування (в 2 рази: 25 хвилин на епоху порівняно з 13 хвилинами). Більший час на тренування пов’язаний з розміром моделі: більше параметрів потребують оновлення, більше операції займає прогін данних по моделі. Та як наслідок, краща якість сумарицзації.

Отже, при вирішенні задачі сумаризації як бінарної класифікації кращий результат досягається при використанні лінійного класифікатора над вектором ембедингу вхідного речення. Використання більших моеделей дає незначний приріст в якості узагальнення, але потребує більше часу на тренування. Тому якщо є обчислювальні ресурси та час на тренування – вартно використовувати саме їх.

ВИСНОВКИ

В ході роботи було вивчено та проаналізовано сучасній підхід до обробки, аналізу та перетворення текстової інформації за допомогою методів машинного навчання та, зокрема, нейронних мереж. Розглянуті такі аспекти як формулювання задач, обробка текстової інформації як послідовності слів, методи кодування тексту та його трансформації до числового вектора за допомого статистичних підходів та ембедингів. Розглянуті переваги та недоліки цих методів.

Розглянуті методи глибокого навчання для роботи з текстами, а саме рекурентні нейронні мережі, механізм уваги, моделі трансформери, розглянуті принципи їх роботи та галузі застосування.

Була поставлена задача дослідження – випробувати екстрактивні методи створення узагальнень для текстів, а саме формулювання стоверння суммаризації як задачі навчання з учителем та задачі навчання без учителя. Були запропоновані методи тестування цих підходів та метрики для автоматичного вимірювання якості узагальнення, а саме ROUGE-1, ROUGE-2 та ROUGE-L. Вони можуть використовуватися як самостійно, так і в комбінації з людською оцінкою якості створеної текстової сумаризації.

Були поставлені експерименти для порівняння якості створення узагальнення, а саме використання графових методів для аналізу тексту та використання різних функцій відстані та заміни запропонованих TF-IDF векторів на ембединги з моделей трансформерів. За результатами експериментів косинусна метрика відстані показала кращий результат без додаткових витрат на ресурси, тому при використанні графових методів для створення узагальнення текстів ватро використовувати її замість функції, що запропонована в оригінальному підході TextRank.

Використання ембедингів з моделей трансформерів дало значний приріст в якості узагальнення, але водночас таке рішення потрубує більше обчислювальних ресурсів.

Для перевірки якості створення текстового узагальнення як рішення задачі бінарної классифікації було порівняно використання різних моделей трансформерів та декілька варіантів архітектури голови нейронної мережі. За результатами експериментів використання лінійного классифікатора дає кращий результат ніж використання повноцінного LSTM шара як классифікаційнї частини. Беручи до уваги те що лінійний класиікатор є швидшим за LSTM шар (він робить меньше операцій для отримання результату) – рекомендовано його використання.

Порівнюючи якість суммаризації при використанні різних моделей трансформерів DistillBERT показав дещо кращий результат за MobileBERT, але обидів моделі є доволі маленькими (за кількістю параметрів) порівняно з всіма моделями з сімейства трансформер, тому використання більших за розміром моделей може дати значний приріст в якості створення текстової суммаризації.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Інтерв'ю Єндрю Нг. URL: https://www.wipo.int/wipo_magazine/en/2019/03/article_0001.html (дата звернення: 22.03.2021)
2. Концепція розвитку штучного інтелекту від компанії Google. URL: https://ai.google/ (дата звернення: 01.05.2021)
3. D. Deng. : Deep Learning in Natural Languge Processing. : Springer, 2020. 350 с. С.145–147
4. A. Yerokhin ; O. Turuta ; A. Babii ; A. Nechyporenko. Intelligent information system of heterogeneous medical data analysis // 12th 55 International Scientific and Technical Conference on Computer Sciences and Information Technologies (CSIT), 2017
5. Николенко С., Кадурин А., Архангельская Е. Глубокое обучение. : Питер, 2017. 481 с. С. 20–21.
6. Vishal Gupta and Gurpreet Singh Lehal. 2010. A survey of text summarization extractive techniques. Journal of Emerging Technologies in Web Intelligence 2, 3 (2010), 258–268
7. Y.Bengio. Deep Learning. : MIT, 2015. 537 с. С. 30–37.
8. M. Hermans, B. Schrauwen. Training and analysing deep recurrent neural networks.: Advances in Neural Information Processing Systems, 2013. 320 с. С. 190–198
9. H. Sak, A. Senior, F. Beaufays. Long Short-Term Memory Recurrent Neural Network Architectures for Large Scale Acoustic Modeling. URL: https://static.googleusercontent.com/media/research.google.com/en//pubs/archive/43905.pdf (Дата звернення: 15.03.2021)
10. F. Chollet, Deep Learning with Python. : O'Reilly, 2017. с. С. 120-125

11. A. Graves. Supervised Sequence Labelling with Recurrent Neural Networks. : O'Reilly, 2015. 348 c. C. 233-237
12. I.Sutskever, O.Vinyals, Q.V.Le. Sequence to Sequence learning with Neural Networks: arXiv preprint arXiv:1409.3215
13. Yerokhin, A. ; Nechyporenko, A. ; Babii, A. ; Turuta, O. A new intelligencebased approach for rhinomanometric data processing // IEEE 36th International Conference on Electronics and Nanotechnology, ELNANO, 2016
14. E. Jobson, A. Gutiérrez. Abstractive Text Summarization using Attentive Sequence-to-Sequence RNNs. URL: https://web.stanford.edu/class/archive/cs/cs224n/cs224n.1174/reports/2749095.pdf (Дата звернення: 14.03.2021)
15. Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. CoRR, abs/1409.0473, 2014
16. Alex Graves. Generating sequences with recurrent neural networks. arXiv preprint arXiv:1308.0850, 2013
17. Transformers is state-of-the-art model. URL: https://www.analyticsvidhya.com/blog/2019/06/understanding-transformers-nlp-state-of-the-art-models/ (дата звернення: 04.04.2021)
18. A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, N. Gomez, L. Kaiser,I. Polosukhin. Attention is all you need. : Advances in Neural Information Processing Systems, 2017. c 6000–6010
19. E. Stevens, L. Antiga, T. Viehmann. Deep learning with PyTorch. : Manning, 2020. 451 c. cc 252-254
20. X. Lim G. Xhang, H. Huang, et al. Performance analysis of GPU-based Convolutional Neural Networks. : 45th International Conference on Parallel Processing, 2016. Cc 1023 - 1030
21. T. Brown, B. Mann, N. Ryder et al. Language Models are Few-Shot Learners. Archive preprint arXiv:2005.14165, 2020

22. Exploring Transfer Learning with T5. URL: https://ai.googleblog.com/2020/02/exploring-transfer-learning-with-t5.html (дата звернення: 03.04.2021)
23. Improved Embeddings with Easy Positive Triplet Mining [Електронний ресурс] URL: https://www.groundai.com/project/improved-embeddings-with-easypositive-triplet-mining/1 (дата звернення: 22.04.2020)
24. T.Milkov, K.Chen, J, Dean. Efficient Estimation of Word Representations in vector space. URL: https://arxiv.org/pdf/1301.3781.pdf (дата звернення: 21.03.2021)
25. Glove: Global Vectors for Word Representations. URL: https://nlp.stanford.edu/projects/glove/ (дата звернення: 21.03.2021)
26. C. Lin. ROUGE: A Package for automatic evaluation of Summries. : ACL Antology, 2004
27. Лекція «Алгоритми НЛП, що засновані на обробці графів». URL: https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-864-advanced-natural-language-processing-fall-2005/lecture-notes/lec18.pdf (дата звернення: 15.03.2021)
28. How Search algorithms work. URL: https://www.google.com/search/howsearchworks/algorithms/ (дата звернення:)
29. Yerokhin, A. ; Nechyporenko, A. ; Babii, A. ; Turuta, O. Usage of F-transform to finding informative parameters of rhinomanometric signals // Proceedings of the International Conference on Computer Sciences and Information Technologies, CSIT 2015, 2015, pp. 129–132
30. S. Brin, L. Page. The anatomy of a large-scale hypertextual web search engine. : Computer networks and ISDN systems, 1998. Cc 107–117
31. R. Mihaleca, P. Tarau. TextRank: Bringing Order into Texts: Proceedings of the 25st International Conference on Computational Linguistics, 2011.
32. Yerokhin A., Turuta O., Babii A., Nechyporenko A. Usage of phase space diagram to finding significant features of rhinomanometric signals // 2016 XIth International

Scientific and Technical Conference Computer Sciences and Information Technologies (CSIT). : IEEE, 2016. С. 70–72.

33. Бібліолека для застосування НЛП Spacy. URL: https://spacy.io/ (дата звернення: 02.04.2021)
34. Фраемворк для побудови нейронніх мереж PyTorch. URL: https://pytorch.org/ (дата звернення: 02.04.2021)
35. Бібліотека з реалізаціями моделей Трансформерів Hugging Face. URL https://huggingface.co/ (дата звернення: 02.04.2021)
36. V. Sanh, L. Debut, J. Chaumond, T. Wolf. DistilBERT, a distilled version of BERT: 5th Workshop on Energy Efficient Machine Learning and Cognitive Computing – NeurIPS, 2019 cc 248-254
37. Z. Lan, M. Chen, S. Goodman. ALBERT: A Lite BERT for self-supervised lerning of language representations. : ICLR Conference, 2020. cc 1245-1255
38. J. Devlin, M. Chang, K. Lee, K. Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. : Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics, 2019 cc 2390-2400
39. Y. Wu, M. Schuster, Z. Chen, Q. Le, M. Norouzi. Google's neural machine translation system: Bridging the gap between human and machine translation. arXiv preprint arXiv:1609.08144
40. O. Melamud, J. Goldberger, I Dagan. context2vec: Learning generic context embedding with bidirectional LSTM. In CoNLL, 2016 cc 240-245.
41. Yerokhin, A., Semenets, V., Nechyporenko, A., Turuta, O., Babii, A. F-transform 3D Point Cloud Filtering Algorithm // Proceedings of the 2018 IEEE 2nd International Conference on Data Stream Mining and Processing, DSMP 2018, pp. 524–527
42. Датасет Imagenet для порівняння якості задач комп'ютерного зору. URL: https://www.image-net.org/ (дата звернення: 05.04.2021)
43. Датасет ms coco. URL: https://cocodataset.org/ (дата звернення: 02.04.2021)

44. Платформа Google Colaboratory для проведення експериментів. URL: https://colab.research.google.com/ (дата звернення: 05.04.2021)
45. Бібліотека для реалізації алгоритмів машинного навчання scikit-learn. URL: https://scikit-learn.org/stable/ (дата звернення: 07.04.2021)
46. Характеристики відеокарти NVIDIA P100. URL: https://www.nvidia.com/en-us/data-center/tesla-p100/ (дата звернення: 11.04.2021)