Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)

Кафедра Програмної інженерії
(повна назва)

АТЕСТАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти – другий (магістерський)

Дослідження методів аналізу тональності тексту з використанням мереж-трансформерів

(тема)

Виконала: студент 2 курсу, групи ІПЗм-18-3
Шуляк С.М.
(прізвище, ініціали)

спеціальності 121 – Інженерія програмного забезпечення
(код і повна назва спеціальності)

Освітньо-наукової програми
(тип програми)

Інженерія програмного забезпечення
(повна назва освітньої програми)

Керівник доц., к.т.н Валенда Н.А.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри, проф. _____ З.В.Дудар

2020 р.

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ

Факультет Комп'ютерних наук
Кафедра Програмної інженерії
Рівень вищої освіти – другий (магістерський)

Спеціальність 121 – Інженерія програмного забезпечення
(код і повна назва)
Тип програми освітньо-наукова програма

Освітня програма Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ:

Зав. кафедри _____

(підпис)

«_____»_____ 20 ___ р.

ЗАВДАННЯ

НА АТЕСТАЦІЙНУ РОБОТУ

студентові Шуляку Сергію Михайловичу
(прізвище, ім'я, по батькові)

1. Тема роботи Дослідження методів аналізу тональності тексту з використанням мереж-трансформерів
затверджена наказом університету від "___"_____20___р. № _____

2. Термін подання студентом роботи до екзаменаційної комісії 20 травня 2020 р.

3. Вихідні дані до роботи *методи аналізу та класифікації тональності тексту з використанням мереж-трансформерів, пояснювальна записка. Використовувати ОС macOS, хмарне середовище для досліджень Google Colab*

4. Перелік питань, що потрібно опрацювати в роботі *мета роботи, аналіз проблемної галузі і постановка задачі, огляд методи аналізу та класифікації тональності тексту з використанням мереж-трансформерів*

5 Консультанти розділів роботи

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата
Спецчастина	доц., к.т.н. Валенда Н.А.		

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка *
1.	Аналіз предметної галузі	28 січня 2020 р.	виконано
2.	Огляд існуючих методів аналізу тональності тексту	20 лютого 2020 р.	виконано
3.	Огляд методів ULMFiT, BERT та XLNet	10 березня 2020 р.	виконано
4.	Підготовка пояснювальної записки	22 березня 2020 р.	виконано
5.	Спецчастина	24 квітня 2020 р.	виконано
6.	Підготовка презентації та доповіді	29 квітня 2020 р.	виконано
7.	Нормоконтроль, рецензування	10 травня 2020 р.	виконано
8.	Занесення диплома в електронний архів	15 травня 2020 р.	виконано
9.	Попередній захист	17 травня 2020 р.	виконано
10.	Допуск до захисту у зав. кафедри	20 травня 2020 р.	виконано

Дата видачі завдання _____ _____ 2020 р.

Студент _____
(підпис)

Керівник роботи _____ доц. Валенда Н.А.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ / ABSTRACT

Атестаційна робота магістра містить: 85 с., 47 рис., 2 табл., 27 джерел.

МАШИННЕ НАВЧАННЯ, АНАЛІЗ ДАНИХ, МЕТОДИ МАШИННОГО НАВЧАННЯ, ТОНАЛЬНІСТЬ ТЕКСТУ, МЕРЕЖІ-ТРАНСФОРМЕРИ.

Об’єкт дослідження – аналіз даних для виявлення думки споживачів про продукцію інтернет-магазинів. Ціллю роботи є дослідження методів аналізу тональності тексту з використанням мереж-трансформерів. Методи дослідження полягають у теоретичному дослідженні існуючих рішень задачі багатокласової класифікації сентиментів у тексті, а саме у відгуках споживачів про товари та експериментальному порівнянні моделей ULMFiT, BERT та XLNet.

В результаті роботи було виявлено найбільш придатну для вирішення вищевказаної задачі за показником точності. Результати роботи можуть бути використані для подальшого дослідження розглянутих мовних моделей для вирішення поставленої задачі.

MACHINE LEARNING, DATA MINING, MACHINE LEARNING METHODS, TRANSFORMER NETWORKS.

The object of research is the analysis of data to identify consumer opinions about the products of online stores. The aim of the work is to study the methods of text tone analysis using transformer networks.

The research methods are a theoretical study of existing solutions to the problem of multiclass classification of sentiments in the text, namely consumer feedback on products and experimental comparison of ULMFiT, BERT and XLNet models.

As a result, the work was found to be the most suitable for solving the above problem in terms of accuracy. The results of the work can be used for further study of the considered language models to solve the problem.

ЗМІСТ

ВСТУП

Сучасні технології дозволяють автоматизувати процеси, необхідною умовою яких раніше була наявність спеціаліста, що відбирав дані і вручну проводив попередню обробку для подальшого аналізу і побудови моделей прогнозування.

Не є виключенням і аналіз відгуків споживачів магазинів, що є необхідною частиною маркетингового аналізу з метою поліпшення якості обслуговування клієнтів. Оскільки це відповідає саме задачам класифікації та прогнозу, актуальним є використання різних методів обробки природної мови, що відноситься до машинного навчання. Для вирішення задачі класифікації та прогнозування виконуються процес, що містить два етапи: створення моделі на навчальній вибірці та використання цієї моделі для передбачення значення залежної змінної на нових даних. Так, як сучасні інформаційні системи працюють в умовах великих обсягів даних, необхідною умовою є використання ефективних алгоритмів аналізу та прогнозування. Також ці алгоритми мають давати точний результат. В області NLP (natural language processing) на сьогодні одним з найефективніших методів є використання нейронних мереж-трансформерів. Цей метод заснований на тонкій настройці (fine-tuning) моделі за допомогою мережі-трансформера і подальшому використанні класифікатора, який здійснює аналіз тональності тексту.

Робота пов’язана з програмою досліджень кафедри ПІ, так як кафедра займається дослідженням таких напрямків, як семантичний аналіз зображень, розробка моделей, методів і алгоритмів розпізнавання для біометричних систем, а також знання-орієнтовані технології класифікації, діагностики і прогнозування ситуацій [1]. Таким чином тема роботи є спорідненою до цих напрямків.

Метою роботи є пошук найбільш точного способу серед існуючих методів аналізу тональності тексту серед підходів з використанням моделей ULMFiT (Universal Language Model Fine-tuning for Text Classification), BERT (Bidirectional Encoder Representations from Transformers) та XLNet для обробки відгуків

споживачів про товари магазину. Задачами дослідження є проведення експериментів спрямованих на виявлення моделі з найбільшим показником точності результату на наборі даних, що містить відгуки користувачів інтернет-магазину Amazon, а також розробка веб-застосунку, що надаватиме можливість побачити результати аналізу довільного відгуку за допомогою трьох моделей.

Об’єктом дослідження є виявлення думки споживачів про товари інтернет магазинів.

Предметом дослідження є методи аналізу тональності тексту з використанням мереж-трансформерів.

Методи дослідження, які були використані під час роботи включають в себе теоретичне дослідження трьох моделей: XLNet, BERT та ULMFiT для розуміння актуальних досягнень у досліджуваному напрямку, а також практичне дослідження у вигляді експериментів з цими моделями для аналізу та порівняння їх придатності до використання у аналізі тональності тексту, а саме класифікації відгуків споживачів товарів інтернет магазину.

Елементом наукової новизни даної роботи є систематизація попередніх знань щодо використання вище вказаних моделей для сентимент-аналізу в певному прикладному застосуванні.

Практичне значення одержаних результатів підтверджує придатність розглянутих моделей для вирішення задачі багатокласової класифікації сентиментів для набору даних, що складається з відгуків про товари.

Частина роботи, пов’язана з дослідженням моделі BERT та архітектури нейронної мережі з архітектурою «трансформер» опублікована у збірнику тез з IX Международної науково-практичної конференції у м. Софія, Болгарія [2].

1 ОГЛЯД НАУКОВОЇ І ПАТЕНТНОЇ ЛІТЕРАТУРИ

1.1 Історія обробки природних мов (NLP)

Історія обробки природних мов почалася в 30-х роках XX сторіччя з наукових праць на тему машинного перекладу тексту таких авторів, як Жорж Артсруні та Петро Троянський. Їх праці були пов’язані з двостороннім перекладом між двома мовами та базовими правилами роботи з граматичними ролями.

У 1957 році Ноам Хомський розробив правила універсальної граматики [3]. У 1969 р. Роджер Шенк ввів концептуальну теорію залежності для розуміння природної мови [4]. Ця модель, частково під впливом роботи Сідней Ламб, широко використовувалася студентами Єльського університету Шенка, такими як Роберт Віленський, Венді Ленерт та Джанет Колоднер. У 1970 році Вільям А. Вудс запровадив розширену мережу переходів (ATN) для відображення природних мов [5]. Замість правил фразової структури ATN використовували еквівалентний набір кінцевих автоматів, які використовували рекурсивно. ATH та їх більш загальний формат під назвою "узагальнені ATH".

У 80-х роках, завдяки розвитку обчислювальних потужностей, для задач NLP почали використовувати алгоритми машинного навчання. До цього моменту використовавили великі набори рукописних правил. Перші алгоритми які використовували були дерева рішень, але з плином часу стали переходити до методів, що базуються на статистичних моделях. Замість чітких правил дерев рішень, статистичні моделі надають більш гнучкі рішення, що базуються на ймовірнісних висновках з певних ознак слів. Завдяки цьому статистичні моделі дають кращі результати, коли необхідно обробляти незнайомий текст з можливими помилками та можуть бути інтегровані у більші системи, виконуючи певну підзадачу.

Великий внесок в розвиток NLP зробила корпорація IBM сумісно з Європейським Союзом та владою Канади.

Сучасні тенденції в обробці спостерігаються в напрямку використання алгоритмів машинного навчання без вчителя або з частковим наглядом. Цей спосіб навчання моделей є більш складним, ніж навчання з вчителем на розмічених вхідних даних, однак через те що ж можливість використовувати нескінченну кількість даних для навчання з відкритих джерел, він є більш перспективним. На відміну від навчання з учителем на певному наборі даних, це дає змогу створювати біль загальновживані моделі.

1.2 Огляд сучасних методів аналізу тональності тексту

Сучасні методи сентимент-аналізу поділяються на чотири класи:

– методи, засновані на правилах і словниках;
– машинне навчання з вчителем;
– машинне навчання без вчителя;
– теоретико-графові моделі.

1.2.1 Методи, засновані на правилах і словниках

Методи, що засновані на правилах і словниках використовують тональні словники, які містять слова, до яких заздалегідь приписано їх емоційну характеристику, та певні семантичні правила що описують зв’язок цих слів. При аналізі тексту до слова ставляться у відповідність до словникових і отримують певне значення емоції, після чого можна обчислити суму цих значень і зробити висновок про позитивність або негативність тексту базуючись на певній шкалі. Правила співставлення слів можуть базуватися на підстановці їх у регулярні

вирази, або бути спеціальними правилами співставлення слів у фразі або реченні, що впливатиме на їх спільну тональність [6].

Головною проблемою такого підходу є складність створення цих словників, оскільки ваги цих слів мають відповідати стилю та тематиці предметної галузі до якої відноситься оброблюваний текст, так як усі слова що позначають певну якість предмета або явища можуть мати як негативний так і позитивний настрій, відносно від того, до якого слова воно відноситься у фразі або реченні. Наприклад, слово «маленький» матиме позитивне значення відносно слова «телефон», але негативне відносно слова «телевізор». Набір правил також необхідно підбирати адекватно предметній області, що також є трудомісткою задачею. Існує декілька підходів, що дозволяють автоматизувати цей процес, але вони не вирішують цю проблему цілком.

1.2.2 Методи машинного навчання з учителем

Методи машинного навчання з учителем засновані на тому, що класифікатор навчають на заздалегідь розмічених даних, після чого використовують цю модель для обробки нових даних. Ці методи дозволяють створити робочу модель значно швидше, оскільки не існує необхідності задавати ваги для кожного слова і прописувати всі правила їх взаємодії, хоча і існує необхідність у наявності розмічених даних для навчання класифікатора. Машинне навчання з учителем дає кращий результат на рівні цілих документів, є більш стійким до непередбачуваних конструкцій та краще масштабується, ніж метод зі словниками та правилами.

Недоліками машинного навчання з учителем є те, що процес створення навчальної вибірки є трудомістким, отриману модель дуже важко відлагоджувати, оскільки вона є чорним ящиком, на відміну від набору правил, які зрозумілі людині.

Такі моделі також показують поганий результат на рівні речень і виконують лише поверхневий аналіз, не надаючи можливості для більш глибокого розуміння тексту.

1.2.3 Методи машинного навчання без вчителя

Машинне навчання без вчителя відрізняється від попереднього підходу тим, що етап навчання моделі виконується в процесі її роботи, а не заздалегідь на навчальній вибірці даних. Це дозволяє скоротити витрати на етап підготовки даних, так як машина розмітить їх у процесі. Крім цього, модель може самостійно визначити необхідні ознаки в тексті, що можуть бути використані для класифікації, які є неочевидні для людини. Останні дослідження в області обробки природних мов пов’язані саме із методами з цього напрямку і демонструють найкращі результати точності, близькі до 96%, наприклад XLNet, BERT та ULMFiT.

Недоліками цього підходу є складність створення та налагодження моделі, оскільки вона не залежить від вибірки тренувальних даних, а також є необхідним аналіз адекватності результатів робочої моделі зі сторони експерта в певній галузі.

1.2.4 Метод з використанням теоретико-графових моделей

Метод з використанням теоретико-графових моделей базується на припущенні, що слова в тексті не є рівнозначними і мають певну вагу, по-різному впливаючи на загальну тональність тексту. Цей метод складається з таких етапів: побудова спеціального графу, ранжування вершин, класифікація слів та обчислення результату. При цьому, використовується словник з тональностями слів оброблюваного тексту, з певними значеннями, наприклад «позитивний»,

«нейтральний» та негативний [7]. Обчислення тональності тексту виконується шляхом сумування усіх позитивних слів з урахуванням їх ваги, аналогічне сумування усіх негативних слів, після чого відбувається ділення значення позитивної оцінки тональності на негативну. Результат і буде відображувати поточну оцінку тональності тексту. Зазвичай висновок про тональність робиться на основі відхилення результату від одиниці, якщо значення більше одиниці, то текст вважається позитивним, якщо менше – негативним.

Недоліки цього методу аналогічні першому розглянутому методу зі словниками, однак цей метод дає кращий результат, оскільки слова мають пріоритети, але необхідність складання словника залишається, однак відсутні витрати на створення великого набору правил співвідношення слів.

1.3 Огляд актуальних наукових робіт

Як було написано раніше, сучасні тенденції розвитку обробки природних мов розвиваються у напрямку машинного навчання без вчителя, що підтверджується останніми науковими роботами з цієї галузі. Далі буде розглянуто три роботи, пов’язані з глибинним навчанням: ULMFiT, BERT, XLNet.

1.3.1 ULMFit

ULMFiT (Universal Language Model Fine-tuning for Text Classification, універсальна мовна модель точного налаштування для класифікації тексту) – це метод трансферного навчання, що дозволяє ефективно проводити попереднє тренування моделі для задач обробки природних мов.

Індуктивне трансферне навчання мало великий вплив на комп'ютерний зір (CV), де прикладні моделі для виявлення, класифікації та сегментації об'єктів не створюють з чистого аркуша, а налагоджують та підстроюють існуючі моделі, що були розроблені на загальновживаних наборах даних, таких як ImageNet, MS-COCO та інших [8].

До розробки цього методу використовувалися підходи глибинного навчання, коли розробка моделей проводилася з нуля, що потребувало багато часу і ресурсів та великі набори даних для навчання.

Дослідження в NLP були зосереджені переважно на трансдуктивному трансфері. Індуктивний трансфер для тонкої настройки попередньо підготовлених вбудовування слів (ембедингів) – проста техніка трансферу, яка націлена лише на перший шар моделі, мала великий вплив на практику і використовується в більшості найсучасніших моделей.

Підходи, розроблені останнім часом, що поєднують ембединги, похідні з інших прикладних задач із введенням даних у різні шари, навчають основну модель для конкретної задачі з нуля і трактують отримані ембединги як фіксовані параметри, що обмежує корисність цих ембедингів [9].

Таким чином, дослідження ULMFiT спрямоване на перенесення підходів, використаних в області задач комп’ютерного зору, в область обробки природної мови і розробці нових практик налагодження моделей під ці задачі.

ULMFiT складається з трьох етапів (див. рис. 1):

– мовна модель навчається на текстах загальної області знань для захоплення загальних особливостей мови в різних шарах;
– повна мовна модель адаптується на даних для цільової задачі, використовуючи дискримінаційну настройку та швидкість навчання за трикутною формою для вивчення певних особливостей завдання;
– класифікатор налаштовується на цільову задачу, використовуючи поступове розморожування для того, щоб зберегти базові шари моделі та адаптувати верхні шари під конкретну задачу.

Цей метод є універсальним, так як він:

– працює для різних задач, відмінних розміром, типами та кількістю міток;

– використовує один архітектурний та навчальний процес;

– не потребує ручної розробки та попередньої обробки ознак;

– не потребує специфічних для домену документів та міток.

У роботі над ULMFiT автори також використали для налагодження коефіцієнтів моделі метод дискримінаційної тонкої настройки. Формулу цієї настройки коефіцієнтів наведено нижче.

$$\theta^l = \theta^l - \eta^l \cdot \nabla_{\theta^l}(\theta), \quad (1)$$

де θ^l –параметри моделі, η – рейт навчання, $\nabla\theta\, J\,(\theta)$ – градієнт залежно від цільової функції моделі.

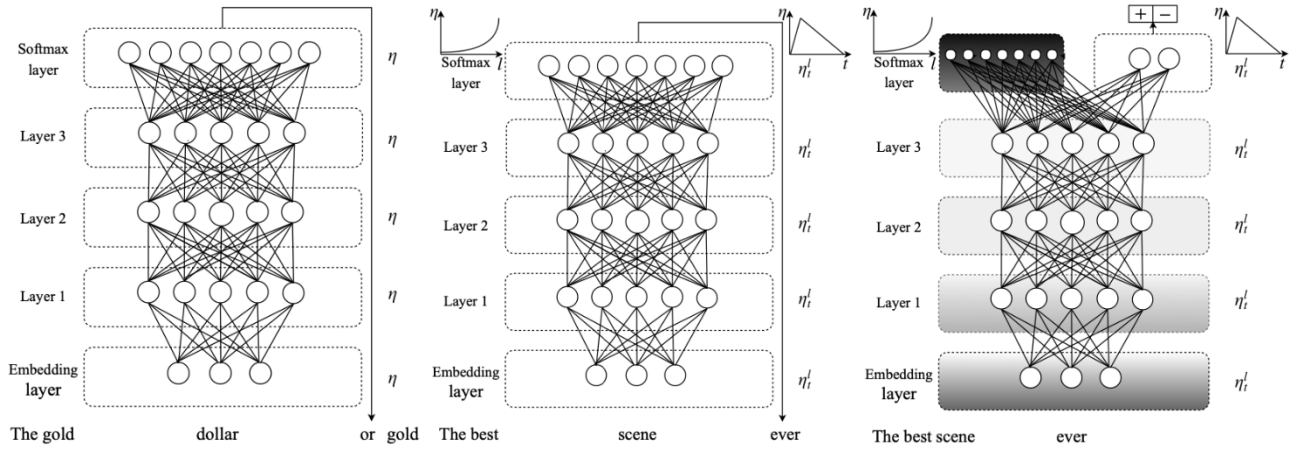


Рисунок 1 – Етапи методу ULMFiT

Для контролю коефіцієнта навчання використовується STLR (slanted triangular learning rates). Для адаптації параметрів моделі до особливостей поточної задачі модель на початку навчання швидко переходить до певної області параметрів, а потім лише підлаштовує ці параметри. При використанні одного коефіцієнта навчання (learning rate) протягом всього процесу навчання не

підходить для забезпечення такої поведінки [10]. У цьому випадку підходять STLR, які спочатку лінійно збільшують коефіцієнт навчання, а потім лінійно зменшують його (див. рис. 2).

Для тонкої настройки класифікатора, мовна модель, отримана з попереднього навчання, доповнюється двома додатковими лінійними блоками. Аналогічно стандартним практикам для класифікаторів з області комп'ютерного зору, кожен блок використовує пакетну нормалізацію та випадання, з функцією активації ReLU для проміжного шару та функцією активації softmax, що дозволяє отримати ймовірний розподіл на цільові класи в останньому шарі. Перший лінійний шар приймає на вхід об'єднаний останній прихований шар. Єдиними шарами, навчання яких проводиться з чистого аркуша є лише шари специфічні для конкретної задачі.

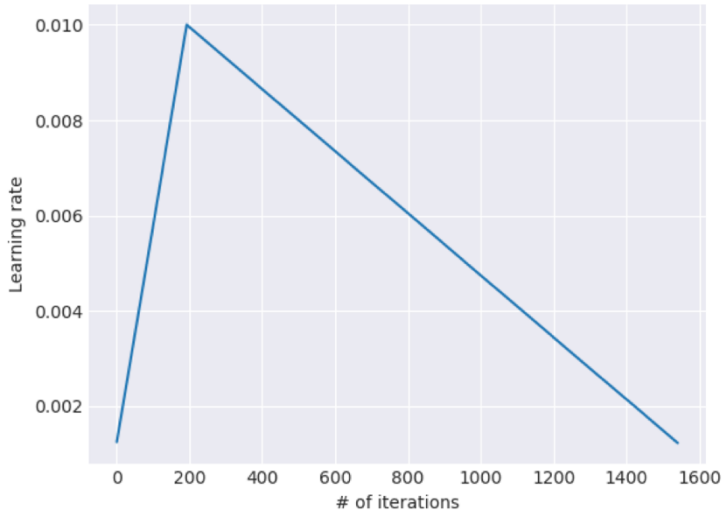


Рисунок 2 – Похилі трикутні коефіцієнти навчання

Останнім етапом тонкої настройки моделі є поступове розморожування шарів. Якщо налаштовувати усі шари одразу, існує великий ризик загубити результати навчання базових шарів. Щоб цього не було, налаштування відбувається поступово, починаючи з останнього, який відповідає найменш загальним знанням. В кожній епосі розморожується один шар і усі розморожені шари налагоджуються до зближення на останній ітерації [11].

Метод було протестовано на шістьох широко вивчених наборах даних, з різною кількістю документів та їх розміром, що використовуються в сучасній класифікації тексту. В якості задач класифікації було обрано класифікацію тематики, сентимент-аналіз та класифікація питань. Порівняння рівнів помилки наведено на рисунку 3.

	Model	Test
IMDb	CoVe (McCann et al., 2017)	8.2
	oh-LSTM (Johnson and Zhang, 2016)	5.9
	Virtual (Miyato et al., 2016)	5.9
	ULMFiT (ours)	**4.6**

Рисунок 3 – Порівняння рівня помилки ULMFiT з іншими моделями

В результаті, отриманий метод тонкої настройки моделі для задач NLP демонструє кращий результат, ніж попередні роботи, з рівнем помилки меншим на 18-24 відсотки на більших об’ємах даних [12].

1.3.2 BERT

BERT (Bidirectional Encoder Representations from Transformers) – мовна модель, розроблена корпорацією Google як продовження ідеї використання нейронних мереж з архітектурою «трансформер» для передтренування мовних моделей на великих обсягах текстових даних, що давало значну перевагу у виконанні задач обробки природних мов, порівняно з попередніми методами.

Вперше цю архітектуру було використано у OpenAI для задач обробки природних мов. Розроблена мережа називається GPT [13] і є однонаправленою, на

відміну від BERT, яка є двонаправленою. Модель від OpenAI мала деякий ряд недоліків, що наведені нижче:

– великі вимоги до обчислень: підхід вимагає дорогого етапу попередньої підготовки – 1 місяць на 8 графічних процесорах. Порівняно з іншими роботами, ця модель є значно більшою і використовує більше обчислень та пам'яті (37-шарова (12 блокова) архітектура трансформерів і використовує послідовності до 512 лексем для тренування;
– обмеженість та упередженість пізнання: текст, наявний в Інтернеті, не містить точних та повних даних про світ. Через що, модель не може повноцінно навчатися адекватно цим даним, оскільки нерівномірний розподіл даних призводить до того, що модель починає його експлуатувати;
– крихке узагальнення: GPT має покращену ефективність у широкому спектрі задач, але поступається сучасним моделям глибинного навчання у сфері обробки природних мов в умовах систематичної, змагальної (adversarial) або поза-розподільної оцінки.

BERT, на відміну від GPT, має здатність навчати мовні моделі на основі всього набору слів, тобто виконує двонаправлене навчання (див. рис. 4). В GPT навчання виконується класичним способом в одному напрямку (зліва направо, чи справа наліво).

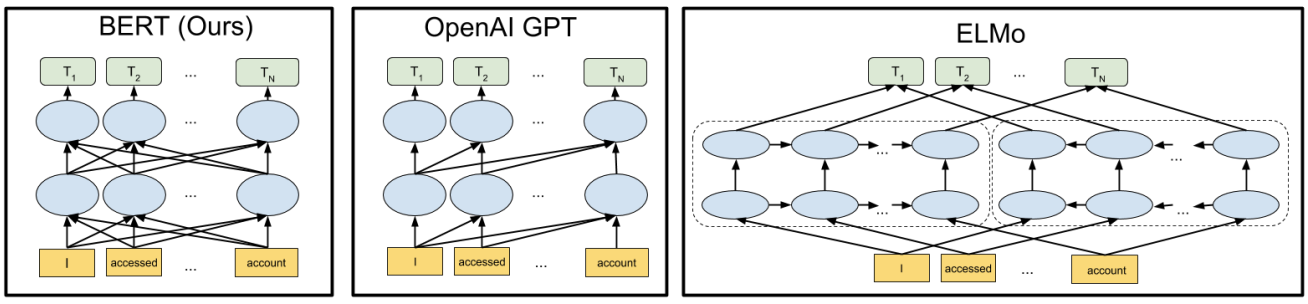


Рисунок 4 – Порівняння архітектури BERT, GPT та ELMo

BERT дозволяє мовній моделі розуміти контекст слова, базуючись на оточуючих словах, при чому не тільки безпосередньо сусідніх [14]. Наприклад, однонаправлена модель у реченні «Я отримав доступ до акаунта в банку» буде співставляти слово «акаунт» з «я отримав доступ до», але не буде брати до уваги слово «в банку», що не дає повного розуміння запиту. BERT у цьому випадку буде співставляти слово «акаунт» з цілим контекстом – «я отримав доступ до … в банку», що дає можливість знати повний контекст і сутність цього запиту.

BERT може вчитися моделювати зв’язки між реченнями у тексті, попередньо навчаючись на простих задачах, які можуть бути сформовані з будь-якого тексту. Наприклад, така задача може формулюватися наступним чином: розглядаючи два речення A і B, чи є B фактичним продовженням попереднього речення A, чи є випадковим реченням у тексті і не пов’язано з A. Приклад наведено на рисунку 5.

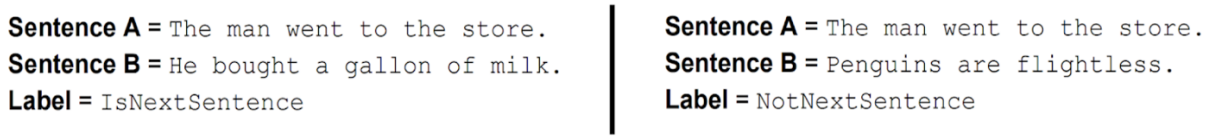


Рисунок 5 – Приклад розпізнавання зв’язків між реченнями

Для розуміння основної складової моделі BERT далі описано роботу мережі за архітектурою «трансформер». Приклад наведено на рисунку 6.

Мережа в такому вигляді складається з енкодеру з шаром багатоголосової уваги (multi-head attention) та декодеру.

Енкодер отримує на вхід слова і видає метаданні, що відповідають цим словам для подальшого використання у декодері. Кожне слово паралельно проходить через шари енкодеру, де частина – повнозв’язні шари (fully-connected layers), а деякі – скорочені (shortcut connections).

Елементом, який відрізняє цю архітектуру від згорткової нейронної мережі або рекурентної нейронної мережі є шар багатоголосової уваги (mutli-head attention). Цей шар дає можливість кожному вхідному вектору взаємодіяти з

іншими через механізм уваги (attention mechanism), а не використовувати схований стан (hidden state), як це відбувається в RNN (recursive neural network) або сусідніх слів в CNN (convolutional neural network). Структура цього шару наведена на рисунку 7.

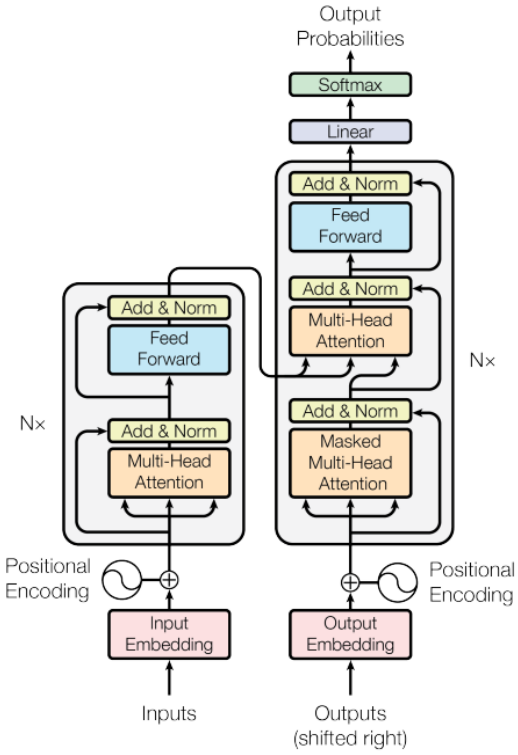


Рисунок 6 – Архітектура мережі-трансформера

В загальному випадку увага (attention) – обчислення середньозваженого середнього на виході попереднього шару. Цей метод використовувався у мовних моделях, що мають повторення, наприклад у LSTM (Long short-term memory), та у згорткових моделях, наприклад QRNN або quasi-RNN). У роботі «Attention Is All You Need» [15] зазначається, що для гарних результатів додаткові компоненти не потрібні, і увага (attention) – це все, що потрібно. Цей підхід має досить помітні вимоги до пам'яті, але повністю паралельний, добре масштабується та є відносно простим.

На вхід шару подаються вектори Q і декілька пар K і V. Зазвичай K і V це один і той же вектор. Кожен з них перетворюється лінійним перетворенням зі здатністю навчатися, обчислюється скалярний добуток Q з усіма K по черзі. Результат скалярних добутків через функцію softmax. Отримуються ваги, з якими усі вектори V сумуються в один вектор.

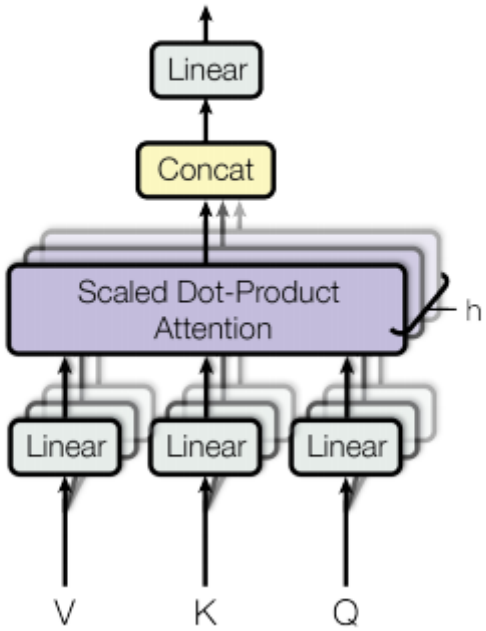


Рисунок 7 – Структура шару multi-head attention

Векторів уваги (attention) одночасно тренується декілька (h), після чого результат конкатенується і проходить через лінійне перетворення і передається далі у енкодері. Завдяки тому що на виході з цього блоку отримуються вектори однакового розміру їх можна використати в мережі декілька разів.

Однією з ознак кожного слова є позиційне кодування (positional encoding), завдяки чому є здатність «звертати увагу» на його контекст – сусідні слова в реченні.

Декодер запускається на одне слово, отримує на вхід попереднє слово і повинен видаляє наступне.

У декодера є два типи використання багатоголосової уваги (multi-head attention):

– звернення до векторів минулих декодованих слів;
– звернення до виходу енкодера. В цьому випадку Q – це вхідний вектор декодеру, а пари K-V – це фінальні метадані енкодера, де один і той же вектор йде в якості і K, і V, але проходять через різні перетворення.

Процес повторюється кілька разів, де результат одного блоку передається наступному. Над результатом виконується функція softmax для отримання ймовірностей слів. Після цього виконується семплювання результатом якого буде наступне слово у реченні. Результат передається на вхід декодеру поки не закінчиться речення [16].

Попереднє навчання моделі BERT складається з двох задач:

– створення маскованої мовної моделі (mask language model)
– передбачення наступного речення

Створення маскованої мовної моделі полягає у тому, що у кожній послідовності маскують 15% токенів випадковим чином через спеціальний маркер «[MASK]». Спеціальний маркер ніколи не зустрінеться під час тонкої настройки, якщо не використовувати деякі евристичні техніки, що використовуються у BERT:

– з вірогідністю 0.8 обрані слова замінюються на [MASK];
– з вірогідністю 0.1 обрані слова замінюються випадковим словом;
– з вірогідністю 0.1 слово не замінюється.

Модель буде передбачати тільки пропущені слова (замінені на «[MASK]»), але не буде мати інформацію про власне замінені слова, через що розмірність результату роботи моделі буде становити лише 15% від розмірності вхідної послідовності.

Для розуміння зв’язків між реченнями, що може бути використано для задач, в яких це необхідно, наприклад відповіді на запитання чи умовиводу, BERT вводить ще одне додатковий етап для підготовки бінарного класифікатора, який

необхідний для визначення, чи є одне речення похідним від іншого. Таким чином розподіл зв’язків між парами речень буде наступним:

– у 50% випадків, речення B слідує за реченням A;

– у 50% випадків, речення B не слідує за реченням A.

Після обробки цього набору даних модель виводить мітку (бінарну), яка позначає, чи є речення B похідним від речення A.

Ці два етапи можливі для будь-якого текстового корпусу, головне щоб він був на одній мові, тому можливі набори даних не мають обмежень.

Після попереднього навчання відбувається етап введення ембедингів (embeddings), який складається за таких кроків (див. рис. 8):

– токенизація за допомогою WordPiece;

– введення ембедингів до сегментів;

– введення позиційних ембедингів.

Токенизаційна модель WordPiece вперше була використана для сегментації японських або корейських слів. Слова цих мов можна додатково розділити на більш дрібні одиниці, для більш ефективної обробки невідомі або просто рідкісних слів.

Введення ембедингів до сегментів відбувається наступним чином: якщо вхідна послідовність має два речення, вони будуть мати ембединги для речення A і для речення B відповідно та розділені спеціальним токеном «[SEP]». Якщо речення лише одне, то будуть використовуватись лише ембединги з речення A.

Позиційні ембединги не задаються вручну, а виводяться під час процесу навчання.

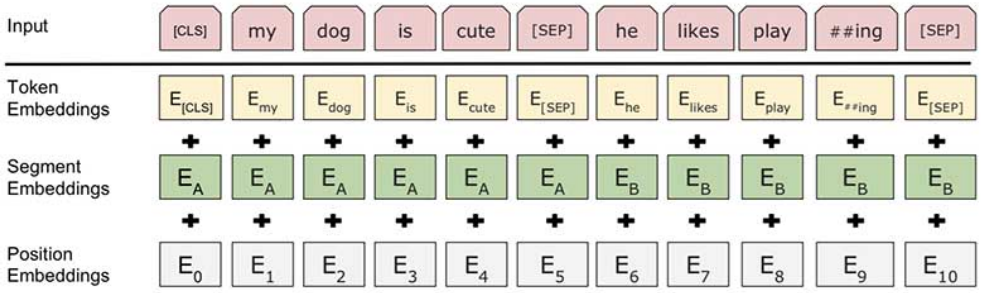


Рисунок 8 – Структура вхідних даних BERT

Як видно на рисунку першим елементом послідовності є «[CLS]», який пізніше буде використовуватися у більш прикладних задачах.

Наступним етапом є тонке налаштування моделі на конкретну задачу. Як і у випадку з GPT, BERT потребує лише невелику кількість додаткових параметрів для підлаштування під специфічну задачу, при чому процес попереднього тренування та тонкого налаштування не відрізняються архітектурно (див. рис. 9).

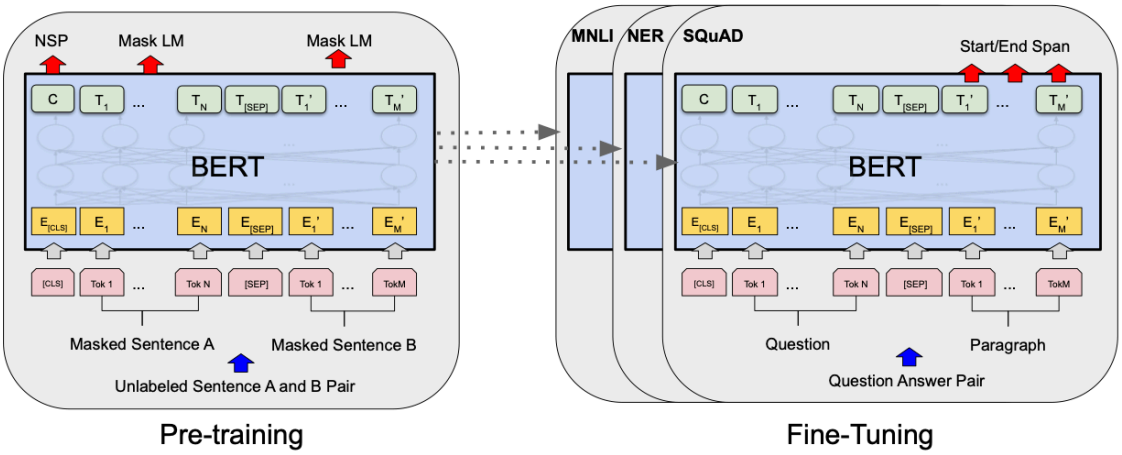


Рисунок 9 – Порівняння процесів попереднього навчання та налаштування

Експериментальні дослідження потребували 1 години на хмарному TPU, або кілька годин на GPU, що є дешевшою операцією, відносно попереднього навчання.

В результаті роботи науковців з Google було розроблено дві аналогічні моделі, різні за розміром: $BERT_{BASE}$ та $BERT_{LARGE}$:

– $BERT_{BASE}$ має наступні розміри: L = 12, H = 768, A = 12, кількість параметрів = 110 мільйонів;

– $BERT_{LARGE}$ має наступні розміри: L = 24, H = 1024, A = 16, загальні параметри = 340 мільйонів.

L означає кількість шарів (блоків мережі-трансформеру), H – розмірність схованого шару, A – кількість елементів уваги (attention).

У роботі наводять результати порівняння цих двох моделей з BiLSTM+ELMo+Attn (state-of-the-art рішення на той момент) та GPT у тесті GLUE (загальна оцінка розуміння мови), які наведені на рисунку 10.

System	MNLI-(m/mm) 392k	QQP 363k	QNLI 108k	SST-2 67k	CoLA 8.5k	STS-B 5.7k	MRPC 3.5k	RTE 2.5k	**Average** -
Pre-OpenAI SOTA	80.6/80.1	66.1	82.3	93.2	35.0	81.0	86.0	61.7	74.0
BiLSTM+ELMo+Attn	76.4/76.1	64.8	79.8	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	87.4	91.3	45.4	80.0	82.3	56.0	75.1
$BERT_{BASE}$	84.6/83.4	71.2	90.5	93.5	52.1	85.8	88.9	66.4	79.6
$BERT_{LARGE}$	**86.7/85.9**	**72.1**	**92.7**	**94.9**	**60.5**	**86.5**	**89.3**	**70.1**	**82.1**

Рисунок 10 – Результати порівняння моделей BERT з іншими рішеннями

Як можна побачити з результатів, моделі BERT дають кращу точність, ніж існуючі на той момент рішення, демонструючи приріст на 4,5% та 7,0% відповідно для $BERT_{BASE}$ та $BERT_{LARGE}$ при виконанні різних задач (MNLI, QQP, QNLI, SST-2, CoLA, STS-B, MRPC, RTE) відповідно до тесту GLUE.

В результаті роботи автори зробили внесок в подальшому узагальненні висновків щодо трансферного навчання з мовними моделями, продемонстрували що попереднє навчання моделі є важливою частиною систем розуміння мови. Завдяки цьому підходу можливе використання моделей глибокого навчання з невеликими обчислювальними ресурсами. Розроблена глибока двонаправлена архітектура дозволяє одній попередньо тренованій моделі бути використаною для вирішення широкого спектру задач обробки природних мов.

1.3.3 XLNet

XLNet – узагальнений метод авторегресивного попереднього навчання, який базується на BERT-моделі, тому не є по суті новою моделлю.

Авторегресивна мовна модель використовує контекстне слово у процесі передбачення слова, що йде після нього [17]. В рамках XLNet контекстне слово обмежене двома напрямками – прямим та оберненим (вперед або назад у послідовності).

Головною перевагою таких моделей є те, що вони гарно підходять для задач, пов’язаних з генерацію нових послідовностей, оскільки генерація нового контексту зазвичай пов’язана саме з прямим напрямком, але є й недолік, оскільки ці моделі можуть працювати лише в одному напрямку одночасно, тому відпадають позитивні ефекти використання обох напрямків одразу.

BERT у свою черго позиціонується як автоенкодуюча мовна модель, завдяки чому вона має можливість використовувати контекст в обох напрямках і з більшою точністю передбачати слова у середині послідовності, а також має властивість реконструювати речення, базуючись на контексті. Недолік цього підходу полягає в тому, що спеціальний маскуючий токен, що використовується під час попереднього навчання буде відсутній на етапі тонкого налаштування, що може призводити до певних невідповідностей, а також виключається залежність між передбаченими словами, оскільки вони трактуються як незалежні.

Мета створення XLNet полягає у розробці такої мовної авторегресійної моделі, яка могла б навчатися з обох напрямків контексту одночасно, оминаючи недоліки, пов’язані з відсутністю зв’язків між маскованими словами як у моделі BERT.

Як і мовні моделі розглянуть раніше, XLNet складається з двох фаз:

– попереднє навчання;

– тонке налаштування.

Головною особливість цієї моделі є те, що у першій фазі присутній процес моделювання мови перестановок (permutation language modeling) [18]. На рисунку 11 наведені варіанти прогнозування частини послідовності x_3 у послідовності x, використовуючи різні перестановки.

Ідея AR на основі перестановки була досліджена раніше, однак XLNet відрізняється своїм підходом.

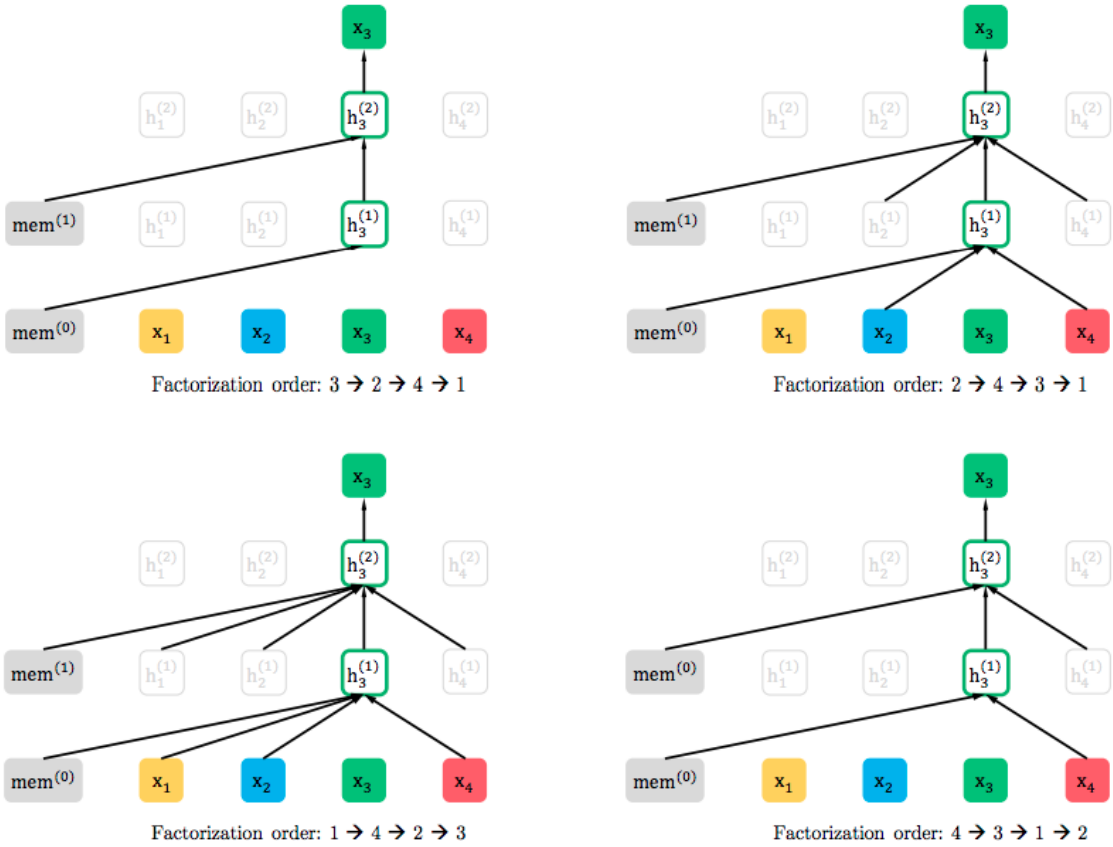


Рисунок 11 – Моделювання мови перестановок у XLNet

Перша відмінність полягає у тому, що попередні моделі покращують щільність оцінки через підготовку «невпорядкованого» індуктивного відхилення в моделі, а XLNet дозволяє мовній AR-моделі вивчати двонаправлені контексти. Для побудови правильного розподілу передбачення, що орієнтоване на кінцеву ціль, технічно для побудови дійсного розподілу прогнозування, орієнтованого на ціль, XLNet додає до прихованого стану цільову позицію через двопоточну увагу (attention). Попередні AR-моделі залежать від неявної інформованості про позицію, що властива їх архітектурам MLP (багатошаровий перцептрон). Слід зазначити, що порядок слів у вхідній послідовності не є випадковим, а лише може бути переставлений обмеженою кількістю перестановок.

XLNet запозичує ідеї від NADE (метод оцінки нейронного авторегресивного розподілу) [19] і пропонує метод моделювання мови перестановок, яка зберігає позитивні сторони мовних AR-моделей та надає можливість використовувати двосторонні контексти. Для послідовності X довжини T існує T! різних перестановок для виконання правильної авторегресивної факторизації. Таким чином, якщо параметри моделі спільні для всіх послідовностей факторизації, тому модель має навчитися отримувати інформацію з обох сторін цільового слова в послідовності.

Нехай Z_T є сукупністю всіх можливих перестановок послідовності індексу T, [1, 2, ..., T]. Для позначення t-го елемента та перших t − 1 елементів перестановки z ∈ Z_T використовуємо z_t і z < t. Тоді мета моделювання мови перестановок може бути позначена формулою (2).

$$\max_{\theta} \ \mathbb{E}_{\mathbf{z}\sim Z_T}\left[\sum_{t=1}^{T} log\ p_\theta\left(x_{z_t}\middle|\mathrm{x}_{\mathrm{z}_{<t}}\right)\right], \tag{2}$$

де Z_T – набір перестановок, p_θ – функція схожості, x – текстова послідовність, z – порядок факторизації.

Важливою частиною моделі XLNet є механізм двопотокової самоуваги (Two-Stream Self-Attention). Ідея цілеорієнтованих представлень виключає неоднозначність в передбаченні цілей. Також запропоновано зафіксуватися на цільовій позиції Z_t для отримання інформації з контексту $x_{z<t}$ за допомогою уваги (attention). Для того, щоб така параметризація працювала є дві вимоги, що є суперечливими для стандартної архітектури мережі-трансформера. Перша вимога – передбачення x_z, $g_0(x_{z<t}, z_t)$ має використовувати лише позицію z_t, а не зміст x_{zt}, в іншому випадку мета стає тривіальною. Друга вимога – для передбачення інших токенів x_{zj}, де $j > t$, $g_0(x_{z<t}, z_t)$ має кодувати зміст x_{zt} щоб надавати повну інформацію з контексту [20].

Для вирішення цих проблем використовуються два набори прихованих представлень замість одного:

– представлення контенту h_{zt}, що виконує роль схожу на роль прихованих станів у «трансформері». Це представлення кодує як контекст, так і x_{zt};

– представлення запиту $g_0(x_{z<t}, z_t)$ має доступ лише до контекстуальної інформації $x_{z<t}$ та позиції z_t без даних про зміст x_{zt}.

Два потоку уваги описуються формулою (3), наведеною нижче.

$$\begin{aligned} g_{z_t}^{(m)} &\leftarrow \text{Attention}\left(\text{Q} = g_{z_t}^{(m-1)}, \text{KV} = \mathbf{h}_{z<t}^{(m-1)}; \theta\right), \text{(потік запиту)} \\ h_{z_t}^{(m)} &\leftarrow \text{Attention}\left(\text{Q} = h_{z_t}^{(m-1)}, \text{KV} = \mathbf{h}_{z<t}^{(m-1)}; \theta\right), \text{(потік змісту)} \end{aligned}, \quad (3)$$

де $g_{z_t}^{(m)}$ – представлення запиту, $h_{z_t}^{(m)}$ – представлення контенту, Q – запит, K – ключ, V – значення в операції уваги

Архітектура двопотокового самоуваження для цільових уявлень (Two-Stream Self-Attention for Target-Aware Representations) подана на рисунку 12.

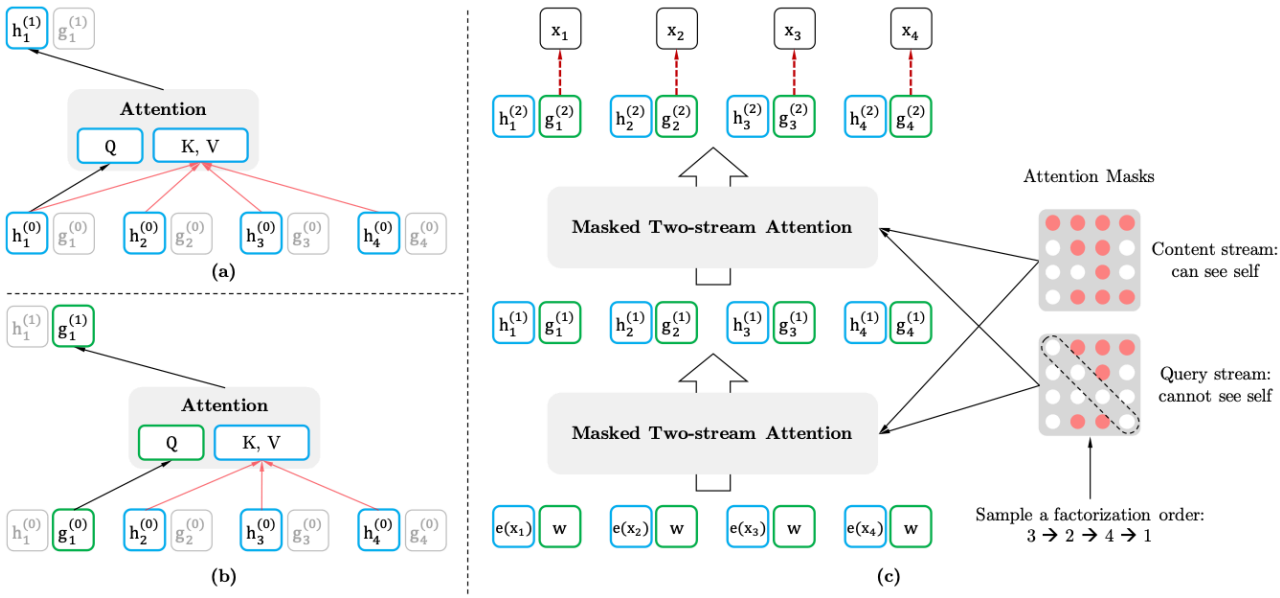


Рисунок 12 – Зображення процесу двопотокової самоуваги

Частина (a) рисунку зображує увагу для потоку змісту, частина (b) зображує увагу для потоку запиту, що не містить інформації про контент. (c) зображує процес навчання моделювання мови перестановок за допомогою двопоточної уваги.

В результаті експериментів автори отримали результат кращий за попередні рішення з використанням GPT, BERT та RoBERTa, що наведені на рисунку 13.

Model	MNLI	QNLI	QQP	RTE	SST-2	MRPC	CoLA	STS-B	WNLI
Single-task single models on dev									
BERT [2]	86.6/-	92.3	91.3	70.4	93.2	88.0	60.6	90.0	-
RoBERTa [21]	90.2/90.2	94.7	92.2	**86.6**	96.4	**90.9**	68.0	92.4	-
XLNet	**90.8/90.8**	**94.9**	**92.3**	85.9	**97.0**	90.8	**69.0**	**92.5**	-
Multi-task ensembles on test (from leaderboard as of Oct 28, 2019)									
MT-DNN* [20]	87.9/87.4	96.0	89.9	86.3	96.5	92.7	68.4	91.1	89.0
RoBERTa* [21]	90.8/90.2	98.9	90.2	88.2	96.7	92.3	67.8	92.2	89.0
XLNet*	**90.9/90.9**†	**99.0**†	**90.4**†	**88.5**	**97.1**†	**92.9**	**70.2**	**93.0**	**92.5**

Рисунок 13 – Результати тесту GLUE для BERT, RoBERTa та XLNet

Як видно з результатів тесту, XLNet дає більшу точність на 4,2% ніж BERT та на 0.6% ніж RoBERTa.

Таким чином, в результаті роботи над XLNet було розроблено узагальнений метод попереднього навчання AR, який використовує моделювання мови перестановок для поєднання переваг методів авторегресії та автоенкодингу. Нейронна архітектура XLNet розроблена для безшовної роботи з метою AR, включаючи інтеграцію Transformer-XL. Також було спроектовано механізм двопотокової уваги. В результати, отримана модель XLNet є значним вдосконаленням, порівняно з попередніми моделями (BERT, RoBERTa і GPT).

1.4 Постановка задачі дослідження

Звертаючи увагу на розглянуті вище роботи, де моделі тестувалися у загальних тестах (GLUE) для широкого спектру задач, виникає питання як будуть поводитися ці моделі на конкретній задачі аналізу тональності тексту для виявлення думки споживачів товарів інтернет-магазинів, а саме на наборі даних Amazon product data. Наукова задача даної роботи полягає в тому, щоб виявити найкращу модель для вирішення цієї задачі, шляхом налаштування трьох моделей (ULMFiT, BERT та XLNet) для роботи з відгуками споживачів інтернет-магазину Amazon і проведення практичних експериментів з кожною моделлю на одному наборі даних. Це дозволить виявити найбільш придатну для цієї задачі модель, спираючись на метрику точності (accuracy), і в майбутньому використовувати її в інтернет-магазинах для дослідження голосу споживачів (voice of customer) з метою вдосконалення маркетингових компаній, наприклад .

2 ОПИС ПРОВЕДЕНИХ ТЕОРЕТИЧНИХ ДОСЛІДЖЕНЬ

2.1 Вибір напрямку дослідження

В якості напрямку дослідження було обрано аналіз трьох моделей, що є найкращими для використання у задачах, пов'язаних обробкою природних мов, з метою з’ясування їх придатності для класифікації відгуків, що є підзадачею аналізу тональності тексту. В результаті дослідження існуючих наукових праць не було знайдено робіт пов’язаних з дослідженням і порівнянням моделей BERT, XLNet та ULMFiT для аналізу тональності відгуків споживачів товарів.

Методом дослідження є аналіз існуючих робіт пов’язаних з сентимент-аналізом, де використовуються вище зазначені моделі, а також експерименти на наборі даних з відгуками про товари від користувачів інтернет-магазину Amazon [21].

В якості експерименту було проведено класифікацію текстів відгуків за п’ятьма класами, що відповідають оцінкам від однієї до п’яти зірок з використанням BERT, XLNet та ULMFiT без тонкої настройки на конкретний датасет для порівняння моделей у базовому стані. Для проведено класифікацію і порівняно з очікуваними результатами для розрахунку метрик, таких як точність (accuracy) та інших.

2.2 Результати теоретичного дослідження існуючих робіт

Перед початком експериментів було досліджено існуючі результати використання моделей BERT, XLNet та ULMFiT специфічно для аналізу тональності тексту, а саме його багатокласової класифікації.

2.2.1 Дослідження сентимент-аналізу з використанням BERT

Під час дослідження було розглянуто роботу, пов'язану з використанням BERT для аналізу тональності тексту відгуків про фільми [22]. Автори роботи використовували BERT з простою методикою тонкого налаштування моделі на набір даних для отримання найкращого результату.

В якості робочого набору даних було використано SST (Stanford Sentiment Treebank), що являє собою один з найбільш вживаних наборів даних серед відкритих джерел для роботи з багатокласовою класифікацією текстів, а саме аналізом тональності з розділенням на 5 класів: «дуже негативно», «негативно», «нейтрально», «позитивно», «дуже позитивно». У наборі міститься 11855 відгуків про фільми з бази Rotten Tomatoes, що складаються з одного речення. Також у наборі містяться частини цих речень, розділені за допомогою спеціального парсера, що створює деревоподібну структуру речень, де коренем я ціле речення, а листя – частини цього речення, для яких вказано певний сентимент (див. рис. 14).

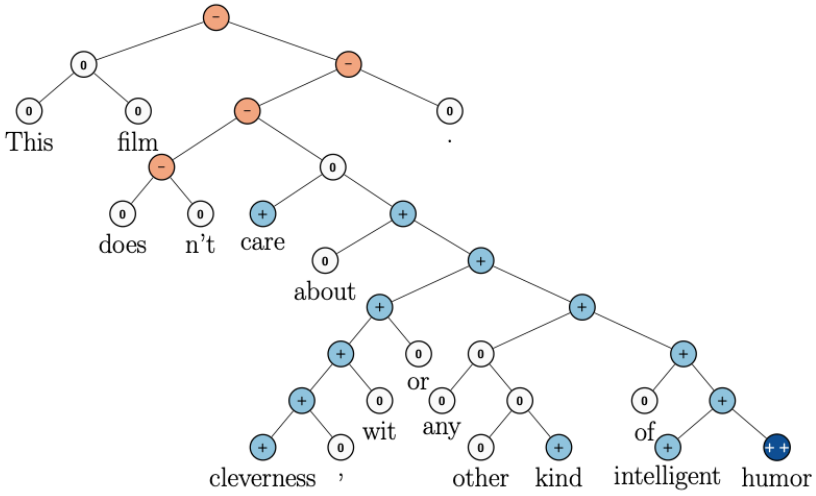


Рисунок 14 – Структура набору даних SST

Таким чином, сумарно, цей датасет містить 215,154 унікальних текстів різної довжини і має вищенаведену структуру.

Методика дослідження полягає у класифікації відгуків з в наведеного вище набору даних і складається з чотирьох етапів (див. рис. 15), не включаючи подання речення на вхід моделі і отримання мітки класу на виході:

– попередня обробка вхідного відгуку (WordPiece) [23];
– введення BERT-специфічних ембедингів у токени відгуку;
– виключення незначущих нейронів з нейронної мережі (dropout);
– класифікація за допомогою softmax.

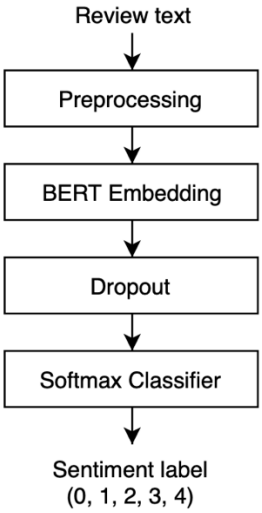


Рисунок 15 – Візуальний вигляд етапів обробки відгуків

Перший етап пов’язаний з попередньою обробки вхідної послідовності перед подачею на вхід до нейронної мережі полягає у канонікалізації, токенизації та додавання спеціальних токенів «[CLS]» та «[SEP]» до послідовностей (див. рис. 16).

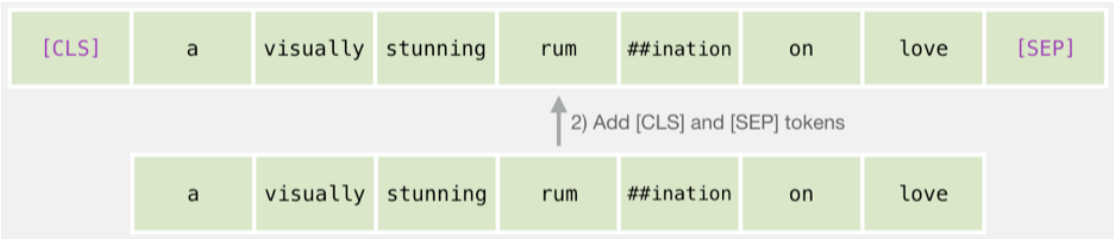


Рисунок 16 – Додавання BERT-специфічних токенів-маркерів

Канонікалізація має на увазі видаляення усіх цифр, наголосів та знаків пунктуації з речення, а також конвертацію усіх літер у рядкові.

Токенізація полягає у обробці тексту спеціальним токенізатором Word-Piece, який розбиває слова на морфеми: префікс, корінь та суфікс для більш точної обробки слів, що можуть не зустрітись у виборці для навчання, але будуть присутні у вибірці для тестування роботи моделі.

Додавання спеціальних лексем – етап що полягає у вставці спеціальних токенів-маркерів, специфічних для моделі BERT – «[CLS]» та «[SEP]». Маркер «[CLS]» вставляється перед початком речення, а «[SEP]» – після останнього слова в реченні.

Етап виключення незначущих нейронів з нейронної мережі (dropout) – це зміна структури нейронів таким чином, що деякі нейрони виключаються з мережі з певною однаковою вірогідністю. Виключення нейрона полягає в тому що при будь-якому вході він повертає 0 в якості результату і не мають впливу на процес оберненого поширення похибки (backpropagation). Це виключає проблему надмірного навчання (overfitting), так як ці нейрони не будуть впливати на процес адаптації інших нейронів, з якими вони пов'язані.

За умови, що лінійна проекція вхідного d_i-мірного вектору x на d_h-мірний простір вихідних значень виражена формулою (4):

$$h(x) = xW + b, \tag{4}$$

Функція активації позначена, як $a(h)$, тоді застосування процедури виключення нейронів може бути представлена як змінена функція активації, що виражена формулою (5).

$$f(h) = D \odot a(h), \tag{5}$$

де $D = (X_1, \cdots, X_{d_h})$ – d_h-мірний вектор випадкових значень X_i, що розподілені за законом Бернуллі, $a(h)$ – функція активації.

Етап класифікації за допомогою softmax представляє собою повнозв'язний шар нейронної мережі з відповідною функцією активації (6).

Обирається вихідний вузол з найбільшою вірогідністю в якості передбачуваної мітки моделі для відповідної вхідної послідовності.

$$\sigma(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}, \text{ де } i = 1, \dots, K, \tag{6}$$

де $\mathbf{z} = (z_1, \dots, z_K) \in \mathbb{R}^K$ є проміжним результатом шару softmax, що називається «logits», e^{z_i} – елемент вектору параметрів.

Під час експериментів автори порівнювали показник точності отриманої моделі BERT (accuracy) з іншими моделями, такими, як RNN, RNTN, LSTM та CNN (див. рис. 17).

Model	SST-2	
	All	Root
Avg word vectors [9]	85.1	80.1
RNN [8]	86.1	82.4
RNTN [9]	87.6	85.4
Paragraph vectors [2]	–	87.8
LSTM [10]	–	84.9
BiLSTM [10]	–	87.5
CNN [11]	–	87.2
$\text{BERT}_{\text{BASE}}$	94.0	91.2
$\text{BERT}_{\text{LARGE}}$	**94.7**	**93.1**

Рисунок 17 – Порівняння метрики точності отриманої BERT-моделі

Як можна побачити, отримана авторами модель демонструє найкращий результат з різницею в 6,4% для BERT base та 7,1% для BERT large. Таким чином

автори довели, що з мінімальними змінами модель BERT придатна для аналізу тональності тексту, а саме класифікації за 5 класами [24].

2.2.2 Дослідження сентимент-аналізу з використанням ULMFiT

Під час дослідження також було розглянуто роботу, де модель ULMFiT порівнюється з іншими методами рішення поставленої задачі – класифікація сентиментів на наборі даних Amazon Customer Reviews, що містить відгуки споживачів про товари інтернет-магазину Amazon. Автори роботи порівнювали метрику точності (accuracy) для BOW, CNN, HCNN, HAN та ULMFiT на вищевказаному наборі даних з різними варіантами попередньої обробки.

З набору даних було використано 10 тисяч відгуків для навчання, та 35 тисяч для тестування. Кожну одиницю даних (речення) було токенізовано за допомогою UDPipe [25]. Як і у випадку з дослідженням BERT у попередньому підпункту, у вибірці міститься 5 класів: : «дуже негативно», «негативно», «нейтрально», «позитивно», «дуже позитивно».

Було досліджено вплив розміру вибірки на точність класифікації. Найбільша втрата точності спостерігалася для CNN, що можна побачити на рисунку 18.

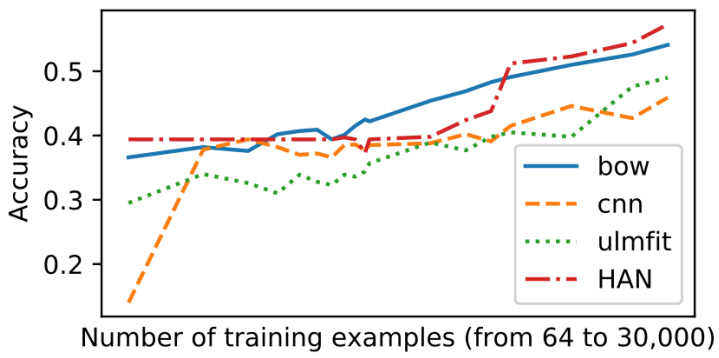


Рисунок 18 – Залежність точності від розміру набору даних

Фільтрація об’єктивних речень (без сентиментальної складової) з набору даних не проводилася, оскільки підходи останніх років дозволяють моделям самостійно відрізняти доцільність використання певних речень для навчання,

Дослідження впливу розміру речення на точність класифікації також показало, що HAN значно краще класифікує великі речення, ніж ULMFiT.

Тонке налаштування мовної моделі ULMFiT було виконано згідно з запропонованими методиками, використовуючи похилі трикутні коефіцієнти навчання та дискримінативне тонке налаштування.

В результаті роботи автори встановили протиріччя у тому, що в оригінальній роботі, пов’язаній з дослідженням ULMFiT, ця модель дає найбільшу точність на багатьох задачах обробки природних мов [26].

2.2.3 Дослідження сентимент-аналізу з використанням XLNet

Під час дослідження було розглянуто роботу, де модель XLNet використовується для класифікації відгуків з набору даних Amazon Customer Reviews. Автори роботи порівнювали метрику точності (accuracy) для наступних моделей: DAmSDA, CNN-aux, AMN, HATN, HANP, BERT та XLNet (див. рис. 19).

В результаті тестування, було з’ясовано що BERT і XLNet мають значну перевагу перед усіма іншими методами.

Таким чином, BERT дає більшу точність класифікації, використовуючи близько 300 тренувальних, що в 20 разів менше даних, використаних для попередніх рішень. XLNet перевершує попередні методи, якщо провести тонку настройку на 50 зразках з навчальної вибірки, що приблизно в 120 разів менше даних. Це каже про те, що використання попередньо тренованих моделей, що використовують нейронні мережі з архітектурою «трансформер», краще підходять

для аналізу тональності тексту, оскільки вони є високоадаптивними у відношенні захоплення контексту на невеликій кількості зразків з навчальної вибірки.

Source	Target	DAmSDA	CNN-aux	AMN	HATN	HANP	**BERT**	**XLNet**
Books	DVD	86.12%	84.42%	85.62%	87.07%	88.12%	92.49%	**95.10%**
	Electronics	79.02%	80.63%	80.55%	85.75%	85.81%	93.13%	**95.92%**
	Kitchen	81.05%	83.38%	81.88%	87.03%	88.91%	94.08%	**96.54%**
	Video	84.98%	84.43%	87.25%	87.80%	89.21%	91.75%	**94.54%**
DVD	Books	85.17%	83.07%	84.53%	87.78%	89.18%	93.67%	**95.68%**
	Electronics	76.17%	80.35%	80.42%	86.32%	86.87%	93.25%	**95.17%**
	Kitchen	82.60%	81.68%	81.67%	87.47%	88.54%	94.15%	**96.42%**
	Video	83.80%	85.87%	87.40%	89.12%	91.25%	93.88%	**95.82%**
Electronics	Books	79.92%	77.38%	77.52%	84.03%	85.67%	91.83%	**93.56%**
	DVD	82.63%	79.07%	80.53%	84.32%	85.29%	89.93%	**91.99%**
	Kitchen	85.80%	87.15%	87.83%	90.08%	91.08%	95.37%	**96.79%**
	Video	81.70%	78.78%	82.12%	84.18%	85.96%	89.33%	**91.79%**
Kitchen	Books	80.55%	78.47%	79.05%	84.88%	85.04%	91.74%	**95.29%**
	DVD	82.18%	79.07%	79.50%	84.72%	86.47%	90.34%	**94.44%**
	Electronics	88.00%	86.73%	86.68%	89.33%	90.43%	94.82%	**96.46%**
	Video	81.47%	78.82%	82.15%	84.85%	85.93%	89.82%	**94.31%**
Video	Books	83.00%	81.48%	83.50%	87.10%	88.94%	93.05%	**95.31%**
	DVD	85.90%	85.25%	86.88%	87.90%	88.54%	93.32%	**95.60%**
	Electronics	77.67%	82.32%	79.68%	85.98%	86.11%	92.87%	**95.71%**
	Kitchen	79.52%	81.28%	80.98%	86.45%	87.21%	93.35%	**96.11%**
Average		82.36%	81.98%	82.79%	86.61%	87.76%	92.61%	**95.13%**

Рисунок 19 – Порівняння XLNet з попередніми методами класифікації

У порівнянні, XLNet має краще показники точності, ніж BERT, на будь-якому розмірі навчальної вибірки, що можна побачити на рисунку 20.

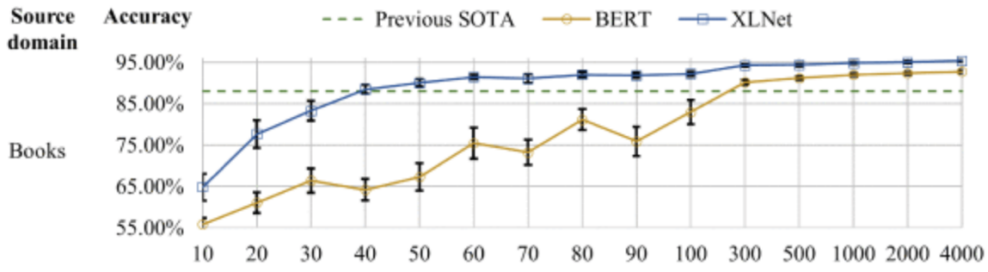


Рисунок 20 – Графіки залежності точності від розміру навчальної вибірки

XLNet є більш ефективним, ніж BERT при проведенні тестування, витрачаючи менше часу на 10%.

З іншого боку, для XLNet необхідно набагато більше обчислювальних потужностей при навчанні, таки чином XLNet працює на 20% повільніше за BERT

при кількості кроків навчання рівній 3000. Це відбувається через механізм повтору сегментів для фіксації контекстних залежностей у документах, що мають довжину більшу за максимальну. Під час тестування цей механізм скорочує час виконання XLNet менш, ніж BERT [27].

Таким чином, в результаті роботи автори довели, що XLNet є більш ефективним ніж BERT на задачах, пов’язаних з сентимент-аналізом відгуків, оскільки він дає кращі показники точності при використанні меншої кількості зразків для тренування і здатен використовувати ширший контекст, використовуючи можливі віддалені залежності. При цьому XLNet потребує більше ресурсів під час навчання, але робить цей етап швидше за BERT.

3 ОПИС ПРОВЕДЕНИХ ЕКСПЕРИМЕНТАЛЬНИХ ДОСЛІДЖЕНЬ

Метою експериментального дослідження є класифікація текстів відгуків за п’ятьма класами, що відповідають оцінкам від однієї до п’яти зірок з використанням ULMFiT (LSTM), BERT, XLNet та без тонкої настройки на конкретний датасет для порівняння моделей у базовому стані. Для цього проведено класифікацію і порівняно з очікуваними результатами для розрахунку метрики точності (accuracy).

Беручи до уваги результати теоретичного дослідження має сенс висунути гіпотезу, що серед розглянутих моделей найкраща точність має бути у моделі на базі XLNet, оскільки вона демонструє найкращий результат у більшості задач пов’язаних з обробкою природних мов, а точніше, для аналізу тональності тексту.

3.1 Опис набору даних

Як було вказано у пункті 2.1, в якості набору даних використовується Amazon Customer Reviews (категорія електроніка), що містить 16,8 мільйонів відгуків.

Розподіл даних за сентиментами від «дуже негативно» до «дуже позитивно» (5 градацій) наведено на рисунку 21.

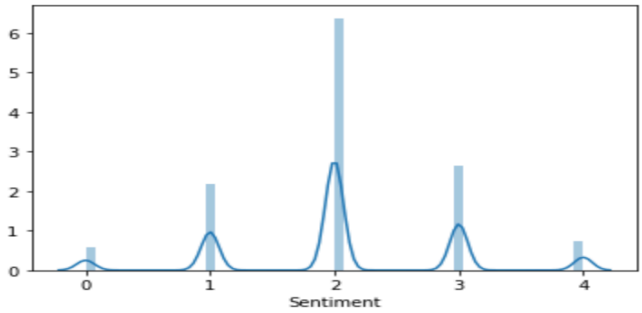


Рисунок 21 – Графік розподілу відгуків за класами у навчальній вибірці

З цього набору даних було обрано 124 тисячі відгуків в якості навчальної вибірки та 31 тисяча для тестування.

Кількісний розподіл навчальної вибірки наведено на рисунку 22. Як можна побачити, зразків з нейтральними відгуками значно більше, ніж в усіх інших класах.

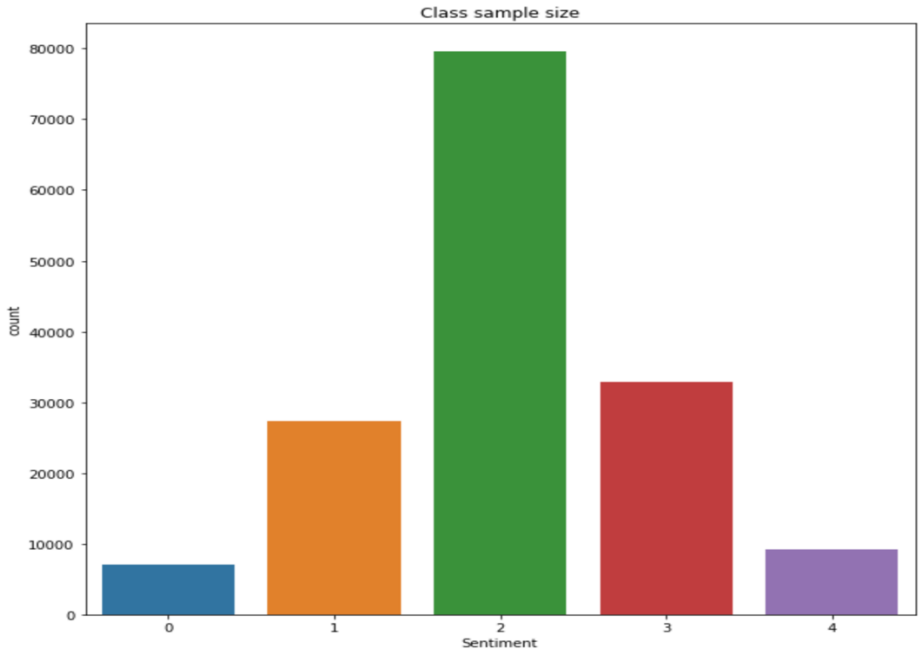


Рисунок 22 – Кількісний розподіл зразків за класами

Для оптимізації вхідних даних для навчання проведено попередню обробку, що буде описано далі.

Дані, представленні у табличному вигляді наведені на рисунку 23.

Phrase	Sentiment
that never bothers to hand viewers a suitcase ...	2
confining color to Liyan 's backyard	2
that even its target audience talked all the w...	1

Рисунок 23 – Табличний вигляд набору даних

З набору даних використовуються 2 елементи: текст та прив’язаний до нього сентимент (що позначає клас від «дуже негативно» до «дуже позитивно»).

3.2 Попередня обробка даних

Зазвичай від якості даних дуже сильно залежить кінцевий результат якості моделі, тому вхідні текстові дані було оптимізовано. Для цього виконані наступні кроки:

– видалено посилання з речень;

– приведено усі літери до малих;

– видалено усі незначущі частини речень (стоп-слова).

Прикладом цієї обробки є перетворення речення «They are familiar with the item» на «they familiar item». Функція для оптимізації тексту наведена на рисунку 24.

```
#Text Pre-processing
def text_cleaning(sentence):
    letters = re.sub("[^a-zA-Z]"," ",sentence)
    ht = re.sub(r'http\S+', '',letters)
    mention = re.sub(r'@\w+', '', ht)
    p = re.sub(r'[^\w\s]','',mention)
    words = p.lower().split()
    stops = set(stopwords.words("english"))
    meaningful_words = [w for w in words if not w in stops]
    return( " ".join(meaningful_words))
df_dplr['text_clean']=df_dplr['Phrase'].apply(lambda x: text_cleaning(x))
```

Рисунок 24 – Функція фільтрації тексту

Для кожної моделі також було виконано додаткові кроки для виконання специфічної попередньої обробки даних під кожну модель, що описано у наступних підрозділах.

3.3 Розробка моделі з ULMFiT (на основі LSTM)

Оскільки ULMFiT є методом тонкої настройки моделей, для експерименту його було вирішено використати з моделлю LSTM (Long short-term memory).

Додатковим етапами попередньої обробки вхідних даних для цієї моделі є токенизація та додання відступів (padding). Код, використаний для токенизації та введення відступів наведено на рисунку 25.

```
from keras.preprocessing.text import Tokenizer

#Tokenizer
tk = Tokenizer(num_words=max_words)
tk.fit_on_texts(X_train)
X_train_tk = tk.texts_to_sequences(X_train)
X_test_tk = tk.texts_to_sequences(X_test)

#Padding sequences
X_train_pad = sequence.pad_sequences(X_train_tk, maxlen=max_len)
X_test_pad = sequence.pad_sequences(X_test_tk, maxlen = max_len)
```

Рисунок 25 – Код для токенизації та введення відступів

Також ця модель є чутливою до збалансованості набору даних, тому було виконано розрахунок вагових коефіцієнтів класів (див. рис. 26).

```
def get_weight(y):
    class_weight_current =  cw.compute_class_weight('balanced', np.unique(y), y)
    return class_weight_current
class_weight = get_weight(Y_train.flatten())
```

Рисунок 26 – Розрахунок вагових коефіцієнтів для класів

Базовий вигляд моделі LSTM без використання ULMFiT наведено на рисунку 27.

Як можна побачити зі структури, міститься шар ембедингу, прихований шар LSTM та шар Dense для виводу вірогідності певного сентименту для вхідного тексту.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_1 (Embedding)      (None, 216, 128)          1917824
_________________________________________________________________
lstm_1 (LSTM)                (None, 128)               131584
_________________________________________________________________
dense_1 (Dense)              (None, 5)                 645
=================================================================
Total params: 2,050,053
Trainable params: 2,050,053
Non-trainable params: 0
_________________________________________________________________
```

Рисунок 27 – Структура моделі без ULMFiT

Було виконано навчання цієї моделі протягом 10 епох з покращенням точності з 0.57 до 0.65.

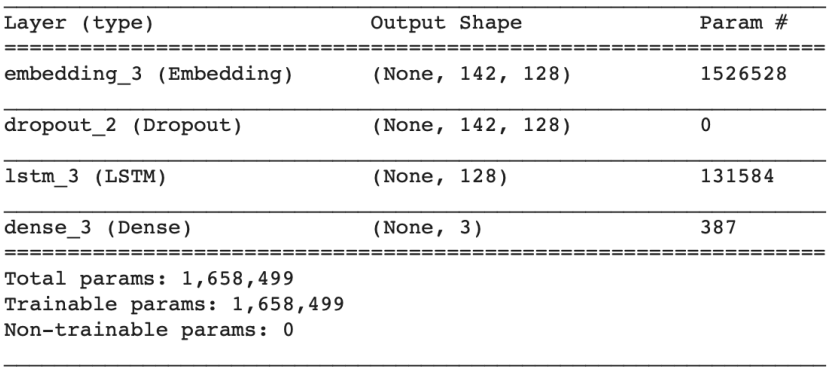
Після цього до моделі було додано шар Dropout для оптимізації, що зображено на рисунку 28.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_2 (Embedding)      (None, 142, 128)          1526528
_________________________________________________________________
lstm_2 (LSTM)                (None, 128)               131584
_________________________________________________________________
dropout_1 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_2 (Dense)              (None, 3)                 387
=================================================================
Total params: 1,658,499
Trainable params: 1,658,499
Non-trainable params: 0
_________________________________________________________________
```

Рисунок 28 – Модель з доданим шаром Dropout

Було проведено навчання протягом 10 епох з покращенням точності з 0.65 до 0.82. Таким чином видно, що додавання шару Dropout дозволяє покращити точність моделі. В даному випадку це покращення складає 17%.

Наступним кроком оптимізації є додавання регуляризатора до шару LSTM (див. рис. 29).



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_3 (Embedding)      (None, 142, 128)          1526528
_________________________________________________________________
dropout_2 (Dropout)          (None, 142, 128)          0
_________________________________________________________________
lstm_3 (LSTM)                (None, 128)               131584
_________________________________________________________________
dense_3 (Dense)              (None, 3)                 387
=================================================================
Total params: 1,658,499
Trainable params: 1,658,499
Non-trainable params: 0
_________________________________________________________________
```

Рисунок 29 – Модель з регуляризатором в LSTM

Після цього також було проведено навчання протягом 10 епох. Після цього можна порівняти отримані показники точності для трьох моделей LSTM.

Результати порівняння моделей LSTM без ULMFiT та з ним наведені у таблиці 1.

Таблиця 1 – Порівняння точності моделей LSTM з ULMFiT

Назва моделі	Точність
LSTM	0.57
LSTM + Dropout	0.65
LSTM + Regularizer + Dropout	0.82

Таким чином, можна побачити, що застосування ULMFiT з мовними моделями дає помітне покращення точності цих моделей. У даному випадку було отримано приріст точності на 8% в порівняно з базовою моделлю. Для подальшого порівняння буде використано показник точності моделі LSTM + Regularizer + Dropout.

3.4 Розробка моделі на основі BERT

Для можливості використання BERT вхідні данні необхідно додатково токенізувати наступним чином:

– перед початком кожного речення вставити токен «[CLS]»;

– вставити токени слів речення;

– додати токен «[SEP]»

– додати відступ (padding).

В результаті тензор вхідної послідовності буде виглядати так, як показану на рисунку 30.

```
[CLS] id : 101
[SEP] id : 102
[PAD] id : 0
Batch shape :  torch.Size([16, 80])
tensor([[ 101, 1011, 1048,  ..., 2095, 1012,  102],
        [ 101, 2031, 2000,  ...,    0,    0,    0],
        [ 101, 2065, 2017,  ...,    0,    0,    0],
        ...,
        [ 101, 1005, 1055,  ...,    0,    0,    0],
        [ 101, 2004, 2035,  ...,    0,    0,    0],
        [ 101, 3904, 1997,  ...,    0,    0,    0]])
```

Рисунок 30 – Тензор вхідної послідовності

Як можна побачити, токени «[CLS]», «[SEP]» та «[PAD]» були вставлені на відповідні місця.

В якості попередньо тренованої моделі BERT, що взята за основу, було використано bert-base-uncased.

Вихідна конфігурація моделі наведена на рисунку 31.

Ця модель містить шар енкодингу, 12 шарів-трансформерів (BERT) та вихідний шар класифікатору (див. рис. 32).

Під час тренування моделі було застосовано методи похилого трикутного коефіцієнту навчання, дискримінативного коефіцієнту навчання та поступового розморожування шарів моделі.


```
BertConfig {
  "architectures": [
    "BertForMaskedLM"
  ],
  "attention_probs_dropout_prob": 0.1,
  "hidden_act": "gelu",
  "hidden_dropout_prob": 0.1,
  "hidden_size": 768,
  "id2label": {
    "0": "LABEL_0",
    "1": "LABEL_1",
    "2": "LABEL_2",
    "3": "LABEL_3",
    "4": "LABEL_4"
  },
  "initializer_range": 0.02,
  "intermediate_size": 3072,
  "label2id": {
    "LABEL_0": 0,
    "LABEL_1": 1,
    "LABEL_2": 2,
    "LABEL_3": 3,
    "LABEL_4": 4
  },
  "layer_norm_eps": 1e-12,
  "max_position_embeddings": 512,
  "model_type": "bert",
  "num_attention_heads": 12,
  "num_hidden_layers": 12,
  "pad_token_id": 0,
  "type_vocab_size": 2,
  "vocab_size": 30522
}
```


Рисунок 31 – Параметри моделі BERT

Першим етапом навчання є заморожування останнього шару моделі, пошук оптимального коефіцієнту навчання (див. рис. 33) та навчання моделі (див. рис. 34).

```
[
 learner.model.transformer.bert.embeddings,
 learner.model.transformer.bert.encoder.layer[0],
 learner.model.transformer.bert.encoder.layer[1],
 learner.model.transformer.bert.encoder.layer[2],
 learner.model.transformer.bert.encoder.layer[3],
 learner.model.transformer.bert.encoder.layer[4],
 learner.model.transformer.bert.encoder.layer[5],
 learner.model.transformer.bert.encoder.layer[6],
 learner.model.transformer.bert.encoder.layer[7],
 learner.model.transformer.bert.encoder.layer[8],
 learner.model.transformer.bert.encoder.layer[9],
 learner.model.transformer.bert.encoder.layer[10],
 learner.model.transformer.bert.encoder.layer[11],
 learner.model.transformer.bert.pooler
]
```

Рисунок 32 – Структура моделі BERT

На рисунку 35 наведено графік залежності втрати (loss) від коефіцієнту навчання (learning rate).

CustomTransformerModel

Layer (type)	Output Shape	Param #	Trainable
Linear	[80, 768]	2,360,064	False
LayerNorm	[80, 768]	1,536	False
Dropout	[80, 768]	0	False
Linear	[768]	590,592	True

Рисунок 33 – Модель із замороженими шарами

На рисунку 33 можна побачити, що усі попередні шари заморожені, крім останнього.

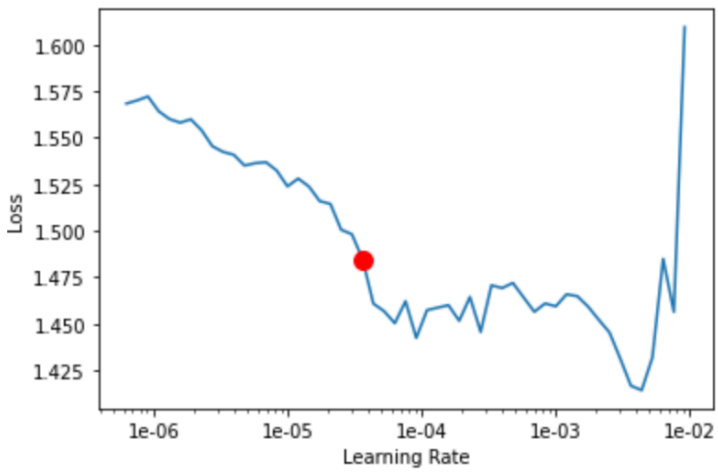


Рисунок 34 – Пошук оптимального коефіцієнту навчання

Виходячи з графіка, наведеного вище, було обрано коефіцієнт 1e-04. Після цього було запущено перший етап навчання нейронної мережі. На рисунку 36 наведено графік залежності train loss від кількості оброблених батчів.

Наступним кроком є повторення попередніх дій, але розморожується наступний з кінця шар моделі.

Як можна побачити на графіку мінімізації похибки (див. рис. 36), похибка вже менша ніж на попередньому етапі.

epoch	train_loss	valid_loss	accuracy	error_rate	time
0	0.939589	0.914978	0.621876	0.378124	1:00:08

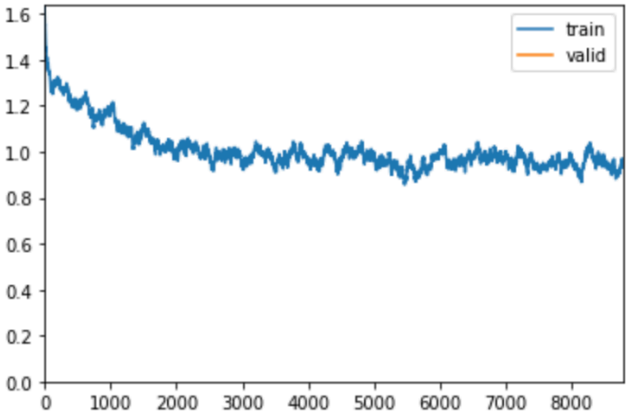


Рисунок 35 – Графік залежності похибки від кількості батчів на 1 етапі

Таким чином помітно зниження похибки з 0.93 до 0.85.

epoch	train_loss	valid_loss	accuracy	error_rate	time
0	0.856935	0.859312	0.644047	0.355953	1:20:33

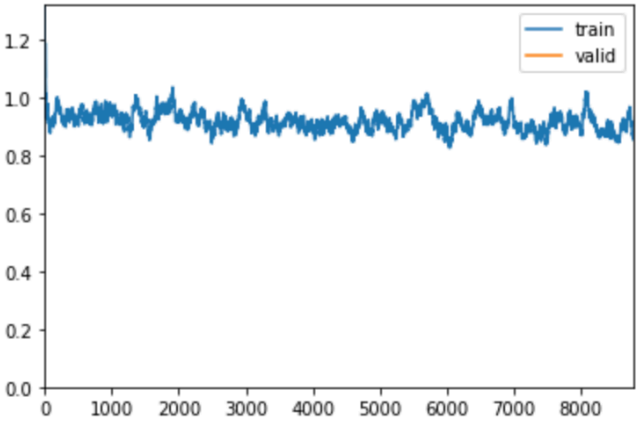


Рисунок 36 – Графік залежності похибки від кількості батчів на 2 етапі

Поступове розмороження відбувається далі і розморожується третій з кінця шар. Після цього виконується навчання. Графік похибки наведено на рисунку 37. Як видно, похибка зменшилася з 0.85 до 0.82.

Після оптимізації останніх двох шарів моделі виконується завершальне навчання моделі.

epoch	train_loss	valid_loss	accuracy	error_rate	time
0	0.824566	0.827368	0.655581	0.344419	1:35:07

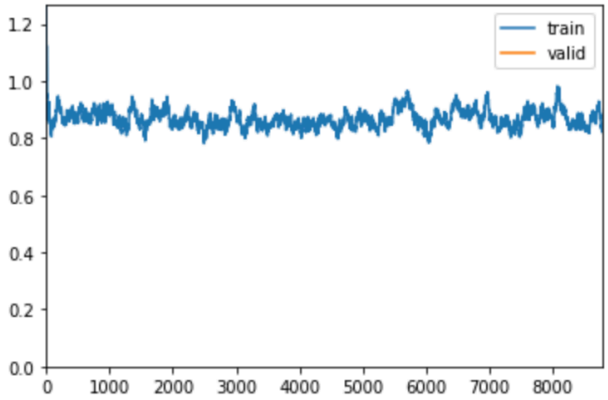


Рисунок 37 – Графік залежності похибки від кількості батчів на 3 етапі

Після повторного навчання рівень похибки зменшився з 0.82 до 0.63. Це можна побачити на рисунку 38.

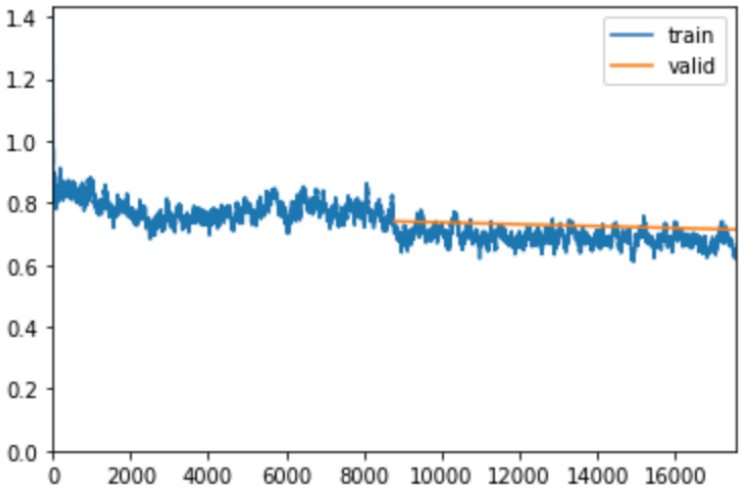


Рисунок 38 – Графік залежності похибки на фінальному етапі

В результаті, отримана модель на базі BERT має точність 0.87. Цей показник буде використано для порівняння з іншими моделями.

3.5 Розробка моделі на основі XLNet

Для можливості використання XLNet вхідні данні також треба токенізувати, але, на відміну від BERT, це робиться інакше.

– перед початком кожного речення вставити відступ – токен «[PAD]»;

– вставити токени слів речення;

– додати токен «[SEP]»

– додати токен «[CLS]».

В результаті тензор вхідної послідовності буде виглядати так, як показану на рисунку 39.

```
[CLS] id : 3
[SEP] id : 4
[PAD] id : 5
Batch shape :  torch.Size([16, 94])
tensor([[   17,     9,     9,  ...,    13,     4,     3],
        [    5,     5,     5,  ...,     9,     4,     3],
        [    5,     5,     5,  ...,  1615,     4,     3],
        ...,
        [    5,     5,     5,  ..., 12330,     4,     3],
        [    5,     5,     5,  ...,   680,     4,     3],
        [    5,     5,     5,  ...,     9,     4,     3]])
```

Рисунок 39 – Тензор вхідної послідовності

Як можна побачити, токени та «[PAD]», «[SEP]» та «[CLS]» були вставлені на відповідні місця.

В якості попередньо тренованої моделі XLNet, що взята за основу, було використано xlnet-base-cased.

Вихідна конфігурація моделі наведена на рисунку 40.

Ця модель містить шар енкодингу, 12 шарів-трансформерів (XLNet) та вихідний шар класифікатору (див. рис. 41).

Під час тренування моделі було застосовано методи похилого трикутного коефіцієнту навчання, дискримінативного коефіцієнту навчання та поступового розморожування шарів моделі.

Першим етапом навчання є заморожування останнього шару моделі (див. рис. 42), пошук оптимального коефіцієнту навчання (див. рис. 43) та навчання моделі.


```
XLNetConfig {
  "architectures": [
    "XLNetLMHeadModel"
  ],
  "attn_type": "bi",
  "bi_data": false,
  "bos_token_id": 1,
  "clamp_len": -1,
  "d_head": 64,
  "d_inner": 3072,
  "d_model": 768,
  "dropout": 0.1,
  "end_n_top": 5,
  "eos_token_id": 2,
  "ff_activation": "gelu",
  "id2label": {
    "0": "LABEL_0",
    "1": "LABEL_1",
    "2": "LABEL_2",
    "3": "LABEL_3",
    "4": "LABEL_4"
  },
  "initializer_range": 0.02,
  "label2id": {
    "LABEL_0": 0,
    "LABEL_1": 1,
    "LABEL_2": 2,
    "LABEL_3": 3,
    "LABEL_4": 4
  },
```


Рисунок 40 – Параметри моделі XLNet

```
[
  learner.model.transformer.transformer.word_embedding,
  learner.model.transformer.transformer.layer[0],
  learner.model.transformer.transformer.layer[1],
  learner.model.transformer.transformer.layer[2],
  learner.model.transformer.transformer.layer[3],
  learner.model.transformer.transformer.layer[4],
  learner.model.transformer.transformer.layer[5],
  learner.model.transformer.transformer.layer[6],
  learner.model.transformer.transformer.layer[7],
  learner.model.transformer.transformer.layer[8],
  learner.model.transformer.transformer.layer[9],
  learner.model.transformer.transformer.layer[10],
  learner.model.transformer.transformer.layer[11],
  learner.model.transformer.sequence_summary
 ]
```

Рисунок 41 – Структура моделі XLNet

На рисунку 44 наведено графік залежності втрати (loss) від коефіцієнту навчання (learning rate).

CustomTransformerModel

Layer (type)	Output Shape	Param #	Trainable
Linear	[1, 768]	2,360,064	False
Dropout	[1, 768]	0	False
Dropout	[1, 768]	0	False
Linear	[768]	590,592	True

Рисунок 42 – Модель із замороженими шарами

На рисунку 42 можна побачити, що усі попередні шари заморожені, крім останнього.

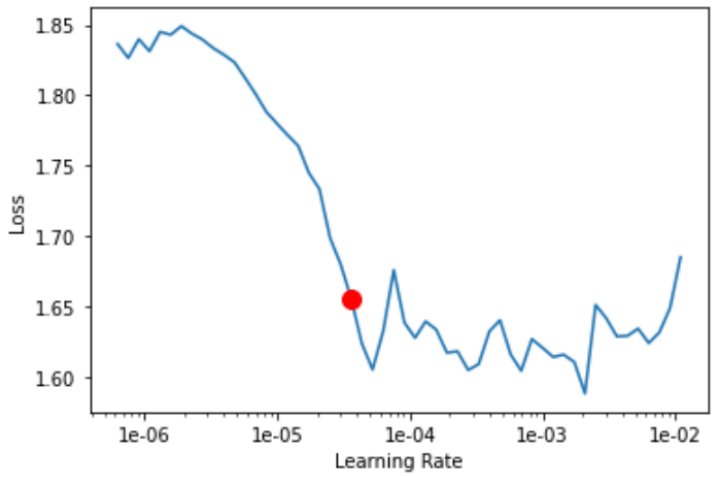


Рисунок 43 – Пошук оптимального коефіцієнту навчання

Виходячи з графіка, наведеного вище, було обрано коефіцієнт 1e-04. Після цього було запущено перший етап навчання нейронної мережі. На рисунку 45 наведено графік залежності train loss від кількості оброблених батчів.

Наступним кроком є повторення попередніх дій, але розморожується наступний з кінця шар моделі.

Як можна побачити на графіку мінімізації похибки (див. рис. 45), похибка менша ніж на попередньому етапі.

epoch	train_loss	valid_loss	accuracy	error_rate	time
0	1.135981	1.120599	0.546713	0.453287	1:57:24

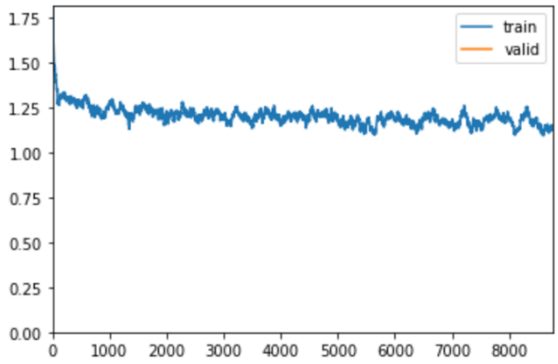


Рисунок 44 – Графік залежності похибки від кількості батчів на 1 етапі

Таким чином помітно зниження похибки з 1.13 до 1.05.

epoch	train_loss	valid_loss	accuracy	error_rate	time
0	1.052435	1.015872	0.589837	0.410163	1:51:16

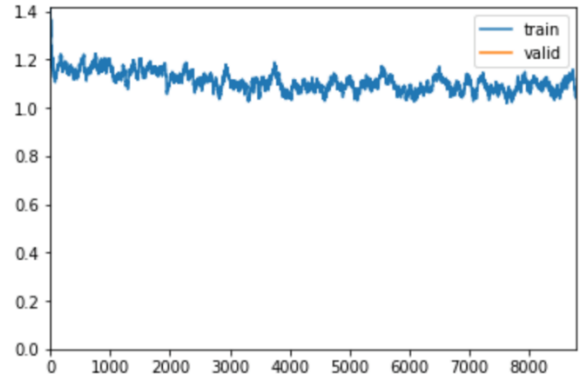


Рисунок 45 – Графік залежності похибки від кількості батчів на 2 етапі

Поступове розмороження відбувається далі і розморожується третій з кінця шар. Після цього виконується навчання. Графік похибки наведено на рисунку 46. Як видно, похибка зменшилася з 1.05 до 1.00.

Після оптимізації останніх трьох шарів моделі виконується завершальне навчання моделі.

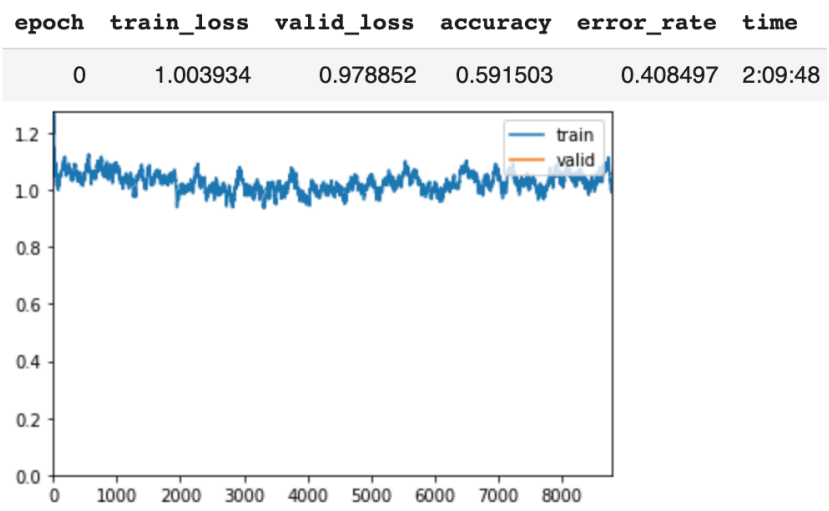


Рисунок 46 – Графік залежності похибки від кількості батчів на 3 етапі

Після повторного навчання рівень похибки зменшився з 1.0 до 0.73. Це можна побачити на рисунку 47.

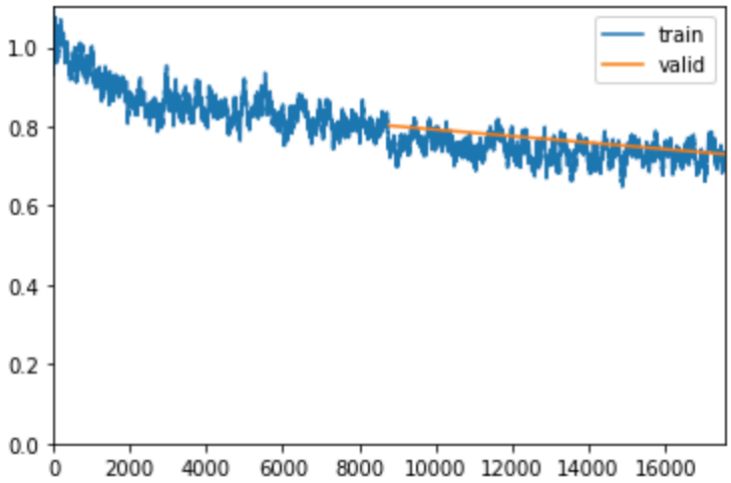


Рисунок 47 – Графік залежності похибки на фінальному етапі

В результаті, отримана модель на базі XLNet має точність 0.91. Цей показник буде використано для порівняння з іншими моделями.

3.5 Порівняння отриманих моделей

Оскільки метою роботи є пошук моделі, найбільш оптимальної за показником точності для вирішення задачі багатокласової класифікації сентиментів у тексті відгуків, далі буде порівнюватися саме цей показник, а також загальний час, витрачений на навчання та тестування кожної з моделей.

Отримані моделі було протестовано но тестовій вибірці з 31 тисячі відгуків користувачів.

Результати порівняння моделей наведено у таблиці 2.

Таблиця 2 – Порівняння ULMFiT (LSTM), BERT та XLNet

Модель	Точність	Час навчання	Час тестування
ULMFiT (LSTM)	0.82	1 год. 3 хв.	16 с.
BERT	0.87	27 хв. 34 с.	12 с.
XLNet	0.91	34 хв. 11 с.	10 с.

Як видно з наведених результатів, найвищий показник точності показує модель на основі XLNet, що на 4% краще ніж BERT та на 8% краще ніж LSTM з ULMFiT.

Час навчання XLNet не є найменшим, так як він є на 6 хв. 23 с. більшим за час навчання моделі на основі BERT, просте час тестування є меншим, що відповідає результатам теоретичних досліджень у підрозділі 2.2.2, пов’язаній з XLNet. Також видно, що моделі на основі мереж-трансформерів мають помітно менший час навчання, порівняно з LSTM-моделлю, що пояснюється високою

здатністю цієї архітектури до розпаралелювання, що також відповідає дослідженню с вихідних робіт про BERT та XLNet.

Як можна побачити з таблиці, час тестування XLNet менший за такий для BERT, що також відповідає висновкам у підрозділі 2.2.3.

За результатом дослідження можна зробити висновок, що гіпотезу про те, що модель на основі XLNet буде мати найкращий результат доведено. Таким чином, можна стверджувати, що моделі на основі XLNet є оптимальними для сентимент-аналізу відгуків.

Подальший шлях розвитку досліджень полягає у тонкій настройці кожної з моделей для максимальної адаптації під конкретний датасет, а також порівняння точності класифікації сентиментів цих моделей.

ВИСНОВКИ

В результаті роботи було проведено аналіз існуючих рішень для обробки природних мов, а саме пов’язаних з задачею багатокласової класифікації тексту за сентиментами.

Було проведено огляд існуючих наукових робіт, пов’язаних з найкращими підходами до обробки природних мов. Було розглянуто такі методи:

– ULMFiT – метод тонкої настройки мовних моделей;
– BERT – попередньо тренована модель від Google;
– XLNet – модель, що демонструє state-of-the-art результати на більшості завдань з обробки природних мов.

Було проаналізовано існуючі рішення, пов’язані з вирішенням задачі аналізу тональності тексту за допомогою вищевказаних методів. В результаті аналізу було висунуту гіпотезу, що XLNet буде найбільш придатним методом для вирішення поставленої задачі: сентимент-аналіз відгуків споживачів про товари в інтернет магазинах.

Для доказу цієї гіпотези було проведено експериментальне дослідження. В результаті експериментального дослідження було розроблено такі моделі на базі ULMFiT, BERT та XLNet:

– LSTM з використанням методу ULMFiT;
– BERT (bert-base), натренована з виконанням поступового розмороження;
– XLNet (xlnet-base), натренована з поступовим розмороженням.

Кожну модель було натреновано на наборі даних, що містить 124 тисячі відгуків споживачів з пов'язаним з відгуком сентиментом – числовою міткою, що позначає рівень задоволеності користувача у градації «дуже негативно», «негативно», «нейтрально», «позитивно», «дуже позитивно».

В результаті аналізу розроблених моделей було проведено їх порівняння за такими метриками, як точність (accuracy), час тренування та час класифікації тестової вибірки (валідації).

Під час порівняння моделей було з'ясовано, що XLNet має найвищі показники точності серед трьох моделей (на 4% краще за BERT і на 8% краще за LSTM з ULMFiT), але потребує більших обчислювальних потужностей для процесу тренування. Таким чином гіпотезу було підтверджено.

Подальшим розвитком роботи може бути вивчення процесу тонкої настройки (fine-tuning) моделі на базі XLNet для отримання вищого рівню точності на використаному наборі даних, або адаптація її для потреб певного інтернет магазину, який зможе використовувати її для автоматизації сентимент-аналізу відгуків споживачів.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Опис кафедри програмної інженерії // ХНУРЕ. URL: https://nure.ua/department/kafedra-programnoyi-inzheneriyi-pi (дата звернення: 03.05.2020).
2. I. Ivanov. Analysis of the phaunistic composition of Ukraine. Topical issues of the development of modern science // Abstracts of the 9th International scientific and practical conference. Софія, Болгарія: ACCENT, 2020. С. 21-27.
3. SEM1A5 A brief history of NLP. URL: https://www.cs.bham.ac.uk/~pjh/sem1a5/pt1/pt1_history.html (дата звернення: 04.05.2020).
4. Schank R. A conceptual dependency parser for natural language // Proceedings of the conference on Computational linguistics. – 1969. – С. 1-3.
5. Woods, W. Transition Network Grammars for Natural Language Analysis // Communications of the ACM. – 1970. – С. 591-606.
6. Валенда Н.А., Самофалов Л.Д. Функционально-семантический анализ конструкций естественного языка на основе унификации // Системи обробки інформації. – 2016. – №7 (144). – С. 79-82.
7. Четвериков Г. Г. Формалізація принципів побудови універсальних k-значних структур мовних систем штучного інтелекту // Доповіді НАН України. – 2001. – №. 1. – С. 41.
8. Razavian Ali S., Azizpour H., Sul-livan J., Carlsson S. Cnn features off- the-shelf: an astounding baseline for recognition // In Proceedings of the IEEE conference on computer vision and pattern recognition. – 2014. – С. 806-813.
9. Peters M., Ammar W., Bhagavat-ula C., Power R. Semi-supervised sequence tagging with bidirectional language models // In Proceedings of ACL. – 2017. – С. 2.
10. Smith L. Cyclical learning rates for training neural networks. In Applications of Computer Vision (WACV) // IEEE Winter Conference. – 2017. – С. 464–472.

11. Yosinski J., Clune J., Bengio Y., Lipson H. How transferable are features in deep neuralnetworks? // Advances in neural information processing systems. – 2014. – С 3320-3328.
12. Howard J., Ruder S. Universal Language Model Fine-tuning for Text Classification. – 2018. – С. 2.
13. Radford A., Narasimhan K., Salimans T., Sutskever I. Improving language understanding with unsupervised learning // Technical report. – 2018. – С. 4.
14. Devlin J., Chang M., Lee K., Toutanova K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. – 2018. – С. 3.
15. Vaswani A., Shazeer N., Parmar N., Uszkoreit J., Jones L., Gomez A., Kaiser L., Polosukhin I. Attention is all you need // Advances in Neural Information Processing Systems. – 2017. – С. 6000-6010.
16. Open Sourcing BERT: State-of-the-Art Pre-training for Natural Language Processing // Google AI Blog. URL: https://ai.googleblog.com/2018/11/open-sourcing-bert-state-of-art-pre.html (дата звернення: 22.03.2020).
17. Dai A., Le Q. Semi-supervised sequence learning // Advances in neural information processing systems. – 2015. – С. 3079-3087.
18. Dai Z., Yang Z., Yang Y., Cohen W., Carbonell J., Le Q., Salakhutdinov R. Transformer-xl: Attentive language models beyond a fixed-length context // arXiv preprint. – 2019. – С. 3.
19. Uria B., Côté M., Gregor K., Murray I., Larochelle H.. Neural autoregressive distribution estimation // The Journal of Machine Learning Research. – 2016. – №17 (1). – С. 7184-7220.
20. Yang Z., Dai Z. Yang Y., Carbonell J., Salakhutdinov R., Le Q. XLNet: Generalized Autoregressive Pretraining for Language Understanding. – 2019. – С. 2.
21. Amazon product data. URL: http://jmcauley.ucsd.edu/data/amazon (дата звернення: 05.05.2020)
22. Maas A. L., Daly R. E., Pham P. T., Huang D., Ng A. Y., Potts C. Learning word vectors for sentiment analysis // Proceedings of the 49th Annual Meeting of the

Association for Computational Linguistics: Human Language Technologies. – 2011. – C. 142-150.

23. Tai K. S., Socher R., Manning C. D. Improved semantic representations from tree-structured long short-term memory networks // Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing. – 2015. – C. 1556-1566.
24. Munikar M., Shakya S., Shrestha A. Fine-grained Sentiment Classification using BERT. – 2019. – C. 2.
25. Straka M., Straková J. Tokenizing, POS tagging, lemmatizing and parsing UD 2.0 with UDPipe // Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies, Vancouver. – 2017. – C. 43.
26. Barnes J., Ravishankar V., Øvrelid L., Velldal E. Hierarchical models vs. transfer learning for document-level sentiment classification. – 2020. – C. 3.
27. Myagmar B., Li J., Kimura S. Cross-Domain Sentiment Classification With Bidirectional Contextualized Transformer Language Models // IEEE Access. – 2019. – № 7. – C. 163219-163230.