Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет _____Комп'ютерних наук_____
(повна назва)

Кафедра _____Інформаційних управляючих систем_____
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти _____другий (магістерський)_____

_____Дослідження використання мовних моделей_____
_____для тестування ІТ-проєктів_____
(тема)

Виконав:
студент _2_ курсу, групи _УПГІТм-22-1_____
____ Альошкін Олексій Андрійович_____
(прізвище, ім'я, по батькові)
Спеціальність ____122 Комп'ютерні_____
_____науки_____
(код і повна назва спеціальності)
Тип програми ____освітньо-наукова_____
(освітньо-професійна або освітньо-наукова)
Освітня програма _Управління проєктами в галузі інформаційних технологій_____
(повна назва освітньої програми)

Керівник _проф. каф. ІУС Максим ЄВЛАНОВ
(посада, власне ім'я, прізвище)

Допускається до захисту

Зав. кафедри _____ ___Костянтин ПЕТРОВ__
(підпис) (власне ім'я, прізвище)

2024 р.

Харківський національний університет радіоелектроніки

Факультет _____Комп'ютерних наук_____

Кафедра _____Інформаційних управляючих систем_____

Рівень вищої освіти _____другий (магістерський)_____

Спеціальність _____122 Комп'ютерні науки_____
(код і повна назва)

Тип програми _____освітньо-наукова_____
(освітньо-професійна або освітньо-наукова)

Освітня програма_Управління проєктами в галузі інформаційних технологій
(повна назва)

ЗАТВЕРДЖУЮ:

Зав. кафедри _____
(підпис)

«_01__»__квітня__ 20 24_ р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові___Альошкіну Олексію Андрійовичу_____
(прізвище, ім'я, по батькові)

1. Тема роботи _____Дослідження використання мовних моделей_____
для тестування ІТ-проєктів _____
_____
_____

затверджена наказом університету від _01 квітня_2024_ р. № _258Cm__

2. Термін подання студентом роботи до екзаменаційної комісії 01.06.2024 р.

3. Вихідні дані до роботи мовні моделі, сучасні дослідження використання мовних моделей у тестуванні ІТ-проєктах._____
_____
_____
_____
_____
_____
_____
_____
_____

4. Перелік питань, що потрібно опрацювати в роботі огляд та аналіз сучасних мовних моделей для тестування ІТ-проєктів; дослідження мовних моделей для тестування ІТ-проєктів; адаптація мовнох моделі BERT для тестування ІТ-проєктів; апробувати результати досліджень, провести експерименти для оптимізації моделі.

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Огляд сучасного представлення тестування як сукупності робіт ІТ-проєкту	01.04.2024 – 05.04.2024	Виконано
2	Огляд та аналіз сучасних мовних моделей	06.04.2024 – 10.04.2024	Виконано
3	Інноваційна роль моделей обробки природної мови в тестуванні ІТ-проєктів	11.04.2024 – 12.04.2024	Виконано
4	Постановка задачі дослідження	13.04.2024 – 14.04.2024	Виконано
5	Дослідження основних різновидів мовних моделей	15.04.2024 – 17.04.2024	Виконано
6	Обґрунтування вибору мовної моделі для тестування ІТ-проєкту	18.04.2024 – 20.04.2024	Виконано
7	Подання вхідних даних у моделі BERT	21.04.2024 – 23.04.2024	Виконано
8	Параметри моделі BERT	23.04.2024 – 24.04.2024	Виконано
9	Процедура навчання моделі BERT	25.04.2024 – 26.04.2024	Виконано
10	Обробка даних для навчання моделі BERT	27.04.2024 – 28.04.2024	Виконано
11	Налаштування мовної моделі на кінцеві завдання	29.04.2024 – 30.04.2024	Виконано
12	Розробка плану проєкту, в рамках якого проходитиме практична- апробація методу	01.05.2024 – 02.05.2024	Виконано
13	Опис вхідних даних моделі	03.05.2024 – 04.05.2024	Виконано
14	Виконання експерементів та оптимізація моделі BERT для тестування ІТ-проєкту	05.05.2024 – 10.05.2024	Виконано
15	Оформлення пояснювальної записки	11.05.2024 – 16.05.2024	Виконано
16	Оформлення графічної частини та презентаційних матеріалів захисту	17.05.2024 – 18.05.2024	Виконано
17	Проведення передзахисту кваліфікаційної роботи	01.06.2024	Виконано
18	Захист кваліфікаційної роботи	04.06.2024	

Дата видачі завдання 01 квітня 2024 р.

Студент _____
(підпис)

Керівник роботи _____ проф. каф. ІУС Максим ЄВЛАНОВ
(підпис) (посада, власне ім’я, прізвище)

РЕФЕРАТ

Пояснювальна записка до кваліфікаційної роботи містить: 93 с., 4 розділи, 8 рис., 7 табл., 14 формул, 63 джерела, 1 додаток.

МОВНІ МОДЕЛІ, ОБРОБКА ПРИРОДНЬОЇ МОВИ, ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ, ІТ-ПРОЄКТИ, BERT.

У кваліфікаційній роботі виконано дослідження використання мовних моделей для тестування ІТ-проєктів. Здійснено глибокий огляд літературних джерел та наукових робіт, з метою виявлення актуальних підходів та тенденцій у сфері застосування мовних моделей у тестуванні програмного забезпечення. В результаті проведеної роботи вдалося ідентифікувати перспективні напрями для ефективного використання мовних моделей в процесі тестування ІТ-проєктів та оптимізації моделі.

Об'єктом дослідження в рамках магістерської кваліфікаційної роботи є мовні моделі.

Предметом дослідження є використання мовних моделей у процесі тестування ІТ-проєктів.

Методи дослідження: аналіз і систематизація існуючих дослідницьких матеріалів, дослідження та порівняння мовних моделей, практична апробація моделі на задачі у ІТ-проєкту.

Результати роботи:

– проведено аналітичний огляд літератури за темою роботи;
– досліджено мовні моделі для тестування ІТ-проєктів;
– обрано мовну модель для тестування ІТ-проєктів;
– апробовано результати дослідження.

ABSTRACT

The explanatory note to the qualification work contains: 93 pages, 4 sections, 8 figures, 7 tables, 14 formulas, 63 sources, 1 ann.

BERT, IT PROJECTS, LANGUAGE MODELS, NATURAL LANGUAGE PROCESSING, SOFTWARE TESTING.

The study investigates the use of language models for testing IT projects. A comprehensive review of literature sources and scientific works related to this topic has been conducted to identify current approaches and trends in the application of language models in software testing. As a result of the study, promising directions for the use of language models in the testing of IT projects have been identified, and recommendations for their effective implementation have been developed.

The object of research of the master's qualification study is language models.

The subject of the research is the use of language models in the process of testing IT projects.

Research methods: analysis and systematization of existing research materials, research and comparison of language models, practical testing of the model on an IT project task.

Work results:

– an analytical review of the literature on the topic of the work has been conducted;

– language models for testing IT projects have been investigated;

– a language model for testing IT projects has been selected;

– the results of the study have been validated.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

ЖЦ – Життєвий цикл

ІС – Інформаційна система

ПЗ – Програмне забезпечення

ШІ – Штучний інтелект

BERT – Bidirectional Encoder Representations from Transformers

CBOW – Continuous Bag of Words

COALS – Correlated Occurrence Analogue to Lexical Semantic

GLUE – General Language Understanding Evaluation

HAL – Hyperspace Analog to Language

LSA – Latent Semantic Analysis

MLM – Masked Language Modeling

MRPC – Microsoft Research Paraphrase Corpus

MultiNLI – The Multi-Genre Natural Language Inference

NLP – Natural Language Processing

NNLM – Neural Network Language Model

NSP – Next Sentence Prediction

QA – Quality Assurance

RTE – Recognizing Textual Entailment

SG – Skip-Gram

WNLI – Winograd NLI

ВСТУП

Розробка програмного забезпечення стає все більш складною і вимагає використання новітніх технологій та підходів. Використання мовних моделей для тестування IT-проєктів може значно полегшити процес тестування, а також підвищити його ефективність та якість. З огляду на поширення штучного інтелекту і машинного навчання, впровадження таких моделей в тестування програмного забезпечення визнається актуальним і має потенціал відкрити нові можливості для покращення процесів розробки програмного забезпечення.

Об'єктом дослідження є мовні моделі.

Предметом дослідження є використання мовних моделей у процесі тестування IT-проєктів.

Мета даного дослідження полягає у вивченні та аналізі можливостей використання мовних моделей у процесі тестування IT-проєктів.

Для досягнення цієї мети необхідно забезпечити вирішення наступних завдань:

- провести аналітичний огляд літератури за темою роботи;
- дослідити мовні моделі для тестування IT-проєктів;
- обрати мовну модель для тестування IT-проєктів;
- апробувати результати дослідження.

Дослідження використання мовних моделей у тестуванні IT-проєктів відкриває нові перспективи для застосування штучного інтелекту в сфері розробки програмного забезпечення. Досліджується можливість застосування та оптимізації засобів обробки природної мови (англ. Natural language processing, NLP) для тестування, що є інноваційним напрямком в області QA (англ. Quality Assurance). Результати дослідження можуть сприяти

покращенню процесів розробки програмного забезпечення та підвищенню ефективності тестування.

Впровадження таких моделей дозволить автоматизувати частину тестування, покращити його ефективність та знизити кількість помилок у програмах. Це може призвести до скорочення термінів випуску продукту на ринок, зменшення витрат на тестування та підвищення якості програмного забезпечення.

Кваліфікаційна робота виконується згідно з державними стандартами [1-2] та методичними вказівками щодо розробки та оформлення кваліфікаційної роботи другого (магістерського) рівня вищої освіти [3].

1 ОГЛЯД ТА АНАЛІЗ СУЧАСНИХ МОВНИХ МОДЕЛЕЙ ДЛЯ ТЕСТУВАННЯ ІТ-ПРОЄКТІВ

1.1 Огляд сучасного представлення тестування як сукупності робіт ІТ-проєкту

Відповідно до IEEE Std 829, тестування – це процес аналізу програмного забезпечення (ПЗ), спрямований на виявлення відмінностей між його реально існуючими та необхідними властивостями (дефект) та на оцінку властивостей ПЗ [1].

За ISO/IEC 12207 у життєвому циклі (ЖЦ) ПЗ визначені серед інших допоміжні процеси верифікації, атестації, спільного аналізу та аудиту [2]. Процес верифікації є процесом визначення того, що програмні продукти функціонують у повній відповідності до вимог або умов, реалізованих у попередніх роботах. Цей процес може включати аналіз, перевірку та випробування (тестування). Процес атестації є процесом визначення повноти відповідності встановлених вимог, створеної системи або програмного продукту їхньому функціональному призначенню. Процес спільного аналізу є процесом оцінки станів і, при необхідності, результатів робіт (продуктів) за проєктом. Процес аудиту є процесом визначення відповідності вимогам, планам та умовам договору. У сумі ці процеси і становлять те, що зазвичай називають тестуванням.

Тестування ґрунтується на тестових процедурах з конкретними вхідними даними, початковими умовами та очікуваним результатом, розробленими для певної мети, такою як перевірка окремої програми або верифікація відповідності на певну вимогу. Тестові процедури можуть перевіряти різні аспекти функціонування програми – від правильної роботи окремої функції до адекватного виконання бізнес-вимог.

При виконанні проєкту необхідно враховувати, відповідно до яких стандартів та вимог буде проводитися тестування продукту. Які інструментальні засоби будуть використовуватися для пошуку та документування знайдених дефектів. Якщо пам'ятати про тестування з самого початку виконання проєкту, тестування продукту, що розробляється, не доставить неприємних несподіванок. Отже, і якість продукту, швидше за все, буде досить високою.

Базовим стандартом розробки ПЗ є ISO/IEC 12207 – Systems and software engineering – Software Life Cycle Processes [5], в якому всі процеси ЖЦ ПЗ розділені на три групи (рис.1.1).

У табл. 1.1 наведено опис основних процесів ЖЦ ПЗ інформаційних систем (ІС) відповідно до ISO/IEC 12207.

Рисунок 1.1 – Процеси життєвого циклу відповідно до стандарту ISO/IEC 12207

Допоміжні процеси призначені для підтримки виконання основних процесів, забезпечення якості проєкту, організації верифікації та тестування ПЗ.

Організаційні процеси визначають дії і завдання замовників і розробників для управління процесами в ході проєкту.

Таблиця 1.1 – Зміст основних процесів ЖЦ ПЗ ІВ відповідно до ISO/IEC 12207

Процес (Виконавець)	Дії	Вхід	Результат
Купівля (Замовник)	– ініціювання; – підготовка вимог – заявки; – підготовка угоди; – контроль діяльності постачальника; – прийом ІС.	– рішення про початок; впровадження ІС; – результати дослідження діяльності замовника; – результати аналізу ринку ІС / тендера; – план поставки / – розробки; комплексний тест ІС.	– техніко-економічне обґрунтування впровадження ІС; – технічне завдання на ІС; – угода на постачання / – розробку; – акти приймання етапів роботи; – акт приймально-передавальних випробувань.
Поставки (Розробник ІС)	– ініціювання; – відповідь на замов- – лення; – підготовка угоди; – планування вико- – нання; – поставка ІС.	– технічне завдання на ІС; – рішення про участь у розробці; – результати тендеру; – технічне завдання на ІС; – план управління проєктом; – створена ІС та документація.	– рішення про участь у розробці; – комерційна пропозиція / конкурсна заявка; угода про постачання / розробки; – план управління проєктом; – реалізація / коригування; – акт приймально-передавальних випробувань.

Для підтримки практичного використання стандарту ISO/IEC 12207 розроблені такі технологічні документи:

– керівництво для ISO/IEC 12207 (ISO/IEC TR 24748-3: 2011 Systems and software engineering – Life cycle management – Part 3: Guide to the application of ISO/IEC 12207 (Software life cycle processes));

– керівництво по використанню ISO/IEC 12207 в управлінні проєктами (ISO/IEC TR 16326: 2009 Systems and software engineering – Life cycle processes – Project management).

Також опублікований стандарт на процеси життєвого циклу систем ISO/IEC 15288 Systems and software engineering – System life cycle processes [3], в розробці якого брали участь фахівці різних галузей: системної інженерії, програмування, управління якістю, людськими ресурсами, безпекою та ін.

Цей документ враховує практичний досвід створення систем в урядових, комерційних, військових та академічних організаціях і може бути застосований для широкого класу систем, але його основне призначення – підтримка створення комп'ютеризованих систем.

У стандарті ISO/IEC 15288 в структурі ЖЦ виділені групи процесів за видами діяльності [6]. Стадії створення системи, передбачені в стандарті ISO/IEC 15288, і основні результати, що мають бути досягнуті до моменту їх завершення, наведені в таблиці 1.2.

Таблиця 1.2 – Стадії створення систем за ISO/IEC 15288

№ п/п	Стадія	Опис
1	Формування концепції	Аналіз потреб, вибір концепції та проєктних рішень
2	Розробка	Проєктування системи
3	Реалізація	Створення системи
4	Експлуатація	Введення в експлуатацію і використання системи
5	Підтримка	Забезпечення функціонування системи
6	Зняття з експлуатації	Зупинка використання, демонтаж, архівування системи

Стандарти ISO/IEC 12207 та ISO/IEC 15288 мають єдину термінологію і розроблені таким чином, щоб могли використовуватися одночасно в проєкті.

Таким чином, тестування ПЗ визначається як процедура забезпечення якості програмних продуктів або послуг, що надаються замовникам організацією. Забезпечення якості фокусується на вдосконаленні процесу розробки ПЗ та робить його ефективним та дієвим відповідно до стандартів

якості, визначених для програмних продуктів. При цьому, тестування важливо, оскільки помилки ПЗ можуть бути дорогими або навіть небезпечними. Помилки програмного забезпечення потенційно можуть спричинити грошові та людські втрати.

1.2 Огляд та аналіз сучасних мовних моделей

Впровадження передавального навчання та навчених мовних моделей у NLP розсунуло межі розуміння та генерації мови, і наразі це дуже популярні методи при обробці природної мови.

Але нині існує суперечка щодо дослідницької цінності величезних попередньо навчених мовних моделей, що займають таблиці лідерів. Хоча багато експертів зі штучного інтелекту погоджуються із твердженням Анни Роджерс про те, що отримання сучасних результатів просто за допомогою більшої кількості даних та обчислювальних потужностей не є новиною для досліджень, інші лідери думок щодо NLP вказують на деякі позитивні моменти в поточній тенденції, наприклад, в можливості побачити основні обмеження поточної парадигми.

У будь-якому випадку, останні вдосконалення мовних моделей NLP, схоже, зумовлені не тільки значним збільшенням обчислювальних можливостей, але також відкриттям винахідливих способів полегшення моделей при збереженні високої продуктивності.

Щоб розібратися в останніх проривах у мовному моделюванні, проведемо аналіз популярних мовних моделей, впроваджених протягом останніх кількох років.

BERT (англ. Bidirectional Encoder Representations from Transformers, двоспрямовані кодувальні представлення з Трансформерів) — модель,

призначена для попередньої підготовки глибоких двоспрямованих представлень шляхом спільного кондиціонування як лівого, так і правого контексту у всіх сторонах. Як результат, заздалегідь навчені представлення BERT можуть бути відрегульовані лише за допомогою одного додаткового рівня виведення, щоб створити найсучасніші моделі для широкого кола завдань, таких як відповідь на запитання та мовний висновок, без істотних модифікацій архітектури, специфічних для конкретного завдання [7].

BERT концептуально проста та емпірично потужна. Вона отримує нові найсучасніші результати по одинадцяти завданням NLP, включаючи досягнення бенчмарку GLUE (General Language Understanding Evaluation) до 80,4% (абсолютне покращення на 7,6%), точності MultiNLI (The Multi-Genre Natural Language Inference) до 86,7 % (абсолютне покращення на 5,6%) та проходження тесту F1 SQuAD v1.1 на 93,2 % (абсолютне покращення на 1,5%), перевершуючи результати людини на 2,0% [58].

Команда Google AI зробила ультрасучасну модель для NLP. Її конструкція дозволяє моделі розглядати контекст як з лівої, так і з правої сторони кожного слова. Будучи концептуально простою, BERT отримує нові найсучасніші результати в одинадцяти завданнях NLP, включаючи відповіді на запитання, розпізнавання іменних сутностей та інші завдання, пов’язані із загальним розумінням мови.

BERT може надавати допомогу підприємствам із широким колом проблем з NLP, включаючи: чат-боти для кращої взаємодії з клієнтами; аналіз відгуків клієнтів; пошук відповідної інформації тощо.

GPT-2 – модель-трансформер, яка досягає найкращих результатів в 7 з 8 перевірених наборів даних моделювання мови, і підходить для WebText. WebText – метод контрольованого тренування моделей без явного нагляду на даних з мільйонів веб-сторінок. Це вказує на перспективний шлях до побудови систем обробки мови, які вчаться виконувати завдання на основі своїх природних демонстрацій. Команда OpenAI демонструє, що заздалегідь навчені

мовні моделі можуть використовуватися для вирішення подальших завдань без будь-яких змін параметрів або архітектури. Вони навчили дуже велику модель, трансформер із параметрами 1,5B (півтора мільярда), на великому та різноманітному наборі даних, що містить текст, викреслений з 45 мільйонів веб-сторінок. Модель формує послідовні абзаци тексту та досягає перспективних, конкурентних і найкращих результатів у найрізноманітніших завданнях.

Навчається дана модель на великому та різноманітному наборі даних:

– веб-сторінки, які фільтруються людьми;

– очищених текстах та з видаленням всіх документів Вікіпедії для мінімізації накладання навчальних та тестових наборів;

– з використанням отриманого набору даних WebText із трохи більше 8 мільйонами документів на загальну кількість 40 ГБ тексту.

GPT2 демонструє досить перспективні результати в плані відповідей на запитання, розуміння читання та перекладу. Що стосується практичних застосувань, модель GPT-2 без точного налаштування далеко не придатна для використання, але вона показує дуже перспективний напрямок досліджень.

Завдяки можливості моделювання двонаправлених контекстів, попередження підготовки на основі автоматичного кодування, як BERT, досягає кращих показників, ніж підходи до попередньої підготовки на основі авторегресивного моделювання мови. Однак, покладаючись на псування вхідних даних масками, BERT нехтує залежністю між маскованими позиціями. Існує альтернатива — модель XLNet. Дослідники з університету Карнегі Меллона та Google розробили модель XLNet для завдань з обробки природної мови, таких як розуміння читання, класифікація тексту, аналіз сентиментальності та інші. XLNet - це узагальнений метод авторегресивного преднавчання, який використовує найкращі результати як авторегресивного моделювання мови (наприклад, Transformer-XL), так і автокодування (наприклад, BERT), уникаючи їх обмежень. Експерименти демонструють, що

ця модель перевершує BERT і Transformer-XL і досягає найкращих результатів у 18 завданнях NLP [8].

XLNet може надавати допомогу компаніям із широким спектром проблем NLP, зокрема:

– чат-боти для підтримки клієнтів першої лінії або відповіді на запити щодо продуктів;

– аналіз сентиментальності для оцінки проінформованості та сприйняття бренду на основі відгуків клієнтів та соціальних мереж;

– пошук відповідної інформації в базах документів або в Інтернеті тощо.

Моделі обробки природної мови досягли значного прогресу завдяки впровадженню методів попередньої підготовки, але обчислювальні витрати на навчання ускладнили параметри реплікації та точного налаштування. Facebook AI та дослідники університету Вашингтона проаналізували навчання моделі двостороннього кодування Google від трансформерів (BERT) та виявлено кілька змін у навчальній процедурі, що покращують її ефективність. Зокрема, дослідники використали новий, більший набір даних для навчання, навчили модель набагато більшим ітераціям та видалили наступну послідовність прогнозування навчальної мети. Отримана в результаті оптимізована модель RoBERTa (надійно оптимізований підхід BERT) відповідає оцінкам нещодавно представленої моделі XLNet в бенчмарку GLUE [9].

Дана модель може використовуватися в бізнес-середовищі для широкого кола подальших завдань, включаючи системи діалогу, відповіді на запитання, класифікацію документів тощо.

Команда Google Research займається проблемою постійно зростаючого розміру попередньо навчених мовних моделей, що призводить до обмеження пам’яті, збільшення часу навчання та інколи несподівано погіршення продуктивності. Зокрема, вони ввели архітектуру Lite BERT (ALBERT). Дана модель включає два методи зменшення параметрів [10]:

– факторизована параметризація вбудовування, де розмір прихованих шарів відокремлюється від розміру вкладених словникових запасів шляхом розкладання великої матриці, що вкладає словниковий запас, на дві малі матриці;

– спільне використання параметрів між шарами, щоб запобігти зростанню кількості параметрів із глибиною мережі.

Ефективність роботи ALBERT додатково покращується шляхом введення самоконтрольованої втрати для прогнозування порядку речень для усунення обмежень BERT щодо узгодженості між реченнями.

ALBERT може в подальшому покращити свою продуктивність за допомогою важких прикладів майнінгу, ефективнішого навчання моделі та інших підходів. Ця модель може бути використана в бізнес-налаштуваннях для підвищення продуктивності широкого кола подальших завдань, включаючи продуктивність чат- ботів, аналіз сентиментальності, аналіз документів та класифікацію тексту.

Дослідницька група Alibaba запропонувала розширити BERT до нової мовної моделі StructBERT. Експерименти демонструють, що введена модель суттєво покращує сучасні результати з різних завдань із розуміння природної мови, включаючи аналіз сентиментальності та відповіді на запитання. Модель базується на архітектурі BERT з багатошаровою двонаправленою мережею трансформерів [10].

Як і інші попередньо навчені мовні моделі, StructBERT може допомагати компаніям у виконанні різноманітних завдань NLP, включаючи відповіді на запитання, аналіз сентиментальності, узагальнення документів тощо.

Дослідницька група Google запропонувала уніфікований підхід до передачі навчання в NLP з метою встановлення нового рівня технології в цій галузі. З цією метою вони запропонували розглядати кожну проблему NLP як проблему "перетворення тексту в текст". Така структура дозволяє використовувати одну і ту ж модель, ціль, процедуру навчання та процес

декодування для різних завдань, включаючи узагальнення, аналіз сентиментальності, відповіді на запитання та машинний переклад. Дослідники називають свою модель Text-to-Text Transfer Transformer (T5) [12] і навчають її на великому зборі даних, зібраних з Інтернету, для отримання найкращих результатів в ряді завдань NLP.

Для цієї моделі потрібно досліджувати методи для досягнення більш високої продуктивності за допомогою більш дешевих моделей, вивчати більш ефективні методи вилучення знань та досліджувати мовно-агностичні моделі. Незважаючи на те, що T5 має мільярди параметрів і може бути занадто важкою для застосування в бізнес-середовищі, представлені ідеї можуть бути використані для поліпшення ефективності різних завдань NLP, включаючи узагальнення, відповіді на питання та аналіз сентиментальності.

Дослідницька група OpenAI звернула увагу на той факт, що потреба у маркованому наборі даних для кожного нового мовного завдання обмежує застосовуваність мовних моделей. Враховуючи, що існує широкий діапазон можливих завдань, і часто важко зібрати великий маркований набір навчальних даних, дослідники запропонували альтернативне рішення, яке полягає в масштабуванні мовних моделей для поліпшення швидкодії. Вони перевірили своє рішення, навчивши 175 мільярдну авторегресивну параметричну модель, і назвали її GPT-3, та оцінили її ефективність у понад двох десятках завдань NLP. GPT-3 показує багатообіцяючі результати і навіть іноді перевершує сучасний рівень, досягнутий за допомогою відрегульованих моделей [13].

Модель GPT-3 використовує ту ж модель та архітектуру, що і GPT-2, включаючи модифіковану ініціалізацію, попередню нормалізацію та оборотну токенізацію. Однак, на відміну від GPT-2, вона використовує чергування щільних і локально смугастих розріджених моделей уваги в шарах трансформатора, як у розрідженому трансформері. Модель оцінюється за трьома різними параметрами:

– навчання за декілька разів, коли моделі дають кілька демонстрацій завдання (як правило, від 10 до 100) під час висновку, але не дозволяється самооновлення;

– навчання за один раз, коли дозволяється лише одна демонстрація, разом із описом завдання природньою мовою;

– без навчання, коли не дозволяється демонстрація, і модель має доступ лише до опису завдання природньою мовою.

Статті новин, створені моделлю GPT-3 із параметром 175B (мільярдів), важко відрізнити від реальних, згідно з оцінками людей (з точністю ~ 52%). Але потрібно працювати над цією моделлю, щоб зменшити великі моделі до менших розмірів для використання у реальних програмах, та підвищувати ефективність вибірки при тренуванні. Модель з параметром 175B важко застосувати до реальних бізнес- проблем через її непрактичні вимоги до ресурсів, але якщо дослідникам вдасться зменшити цю модель до реальних розмірів, її можна застосувати до широкого кола мовних завдань, включаючи відповіді на запитання та генерація копій реклами.

Завдання попередньої підготовки для таких популярних мовних моделей, як BERT та XLNet, передбачає маскування невеликої підмножини немаркованих введень, а потім навчання мережі для відновлення цих введень. Незважаючи на те, що він працює досить добре, цей підхід не є особливо ефективним для обробки даних, оскільки він навчається лише на невеликій частині токенів (зазвичай ~ 15%). В якості альтернативи дослідники зі Стенфордського університету та Google Brain запропонували нове завдання при підготовці, яке називається виявлення заміненим маркером. Представлений підхід називається ELECTRA (ефективне вивчення кодера, який точно класифікує заміну токенів): він дає змогу моделі навчатися з усіх вхідних токенів замість невеликої замаскованої підмножини; не є змагальним, оскільки генератор, що виробляє токени для заміни, навчається з

максимальною ймовірністю. Виявлення заміненим маркером означає, що деякі токени замінюються зразками з невеликої мережі генераторів [14].

Завдяки своїй обчислювальній ефективності модель ELECTRA може зробити застосування попередньо навчених кодерів тексту більш доступним для бізнесу.

Автори Microsoft Research запропонували модель DeBERTa з двома основними вдосконаленнями в порівнянні з BERT, а саме механізмами розкутої уваги та вдосконаленим декодером маски.

DeBERTa має два окремі вектори, що представляють зміст і позицію, а самоувага розраховується між усіма можливими парами, тобто вміст до вмісту, вміст до позиції, позиція до вмісту та позиція до позиції. Позиція до позиції самоуваги тривіально постійно дорівнює 1 і не має інформації, тому вона не обчислюється [15]. Автори припускають, що модель потребує інформації про абсолютну позицію, щоб зрозуміти синтаксичні відтінки, такі як характеристика суб'єкта-об'єкта. Отже, DeBERTa надає інформацію про абсолютну позицію разом з інформацією про відносну позицію. Вбудовування абсолютної позиції надається останньому шару декодера безпосередньо перед шаром softmax, який дає вихідні дані.

Для збільшення узагальнення як метод регуляризації використовується віртуальний змагальний алгоритм навчання, який називається інваріантним масштабуванням. Вбудовані слова порушуються незначною мірою і навчаються видавати такий самий результат, як і для невзбурених вбудованих слів. Слова вкладання векторів нормуються до стохастичних векторів (де сума елементів у векторі дорівнює 1), щоб бути інваріантним до кількості параметрів у моделі.

Розглянемо переваги та недоліки кожної з цих мовних моделей в контексті тестування ІТ-проєктів (таблиця 1.3).

Таблиця 1.3 – Переваги та недоліки мовних моделей в контексті тестування IT-проєктів

Модель	Переваги	Недоліки
BERT	Враховує контекст з обох сторін слова	Велика кількість параметрів
GPT-2	Здатність генерувати тексти без заданих завдань	Обмежена в різних завданнях обробки тексту
XLNet	Враховує контекст з обох сторін слова, без обмежень	Великі обчислювальні витрати
RoBERTa	Надійно оптимізований підхід до підготовки	Потребує великих обсягів даних для тренування
ALBERT	Має менше параметрів, зберігає точність	Обмежена точність порівняно з іншими моделями
StructBERT	Додає мовні структури для кращого розуміння тексту	Вимагає додаткового часу та ресурсів для тренування
T5	Універсальний підхід до обробки тексту	Великі обчислювальні витрати
GPT-3	Велика кількість параметрів, здатність швидко навчатися	Вимагає великих обчислювальних ресурсів
ELECTRA	Ефективніше використання обчислювальних ресурсів	Вимагає попередньої підготовки кодерів тексту
DeBERTa	Поліпшене декодування BERT	Вимагає додаткових обчислювальних ресурсів

На основі аналізу таблиці 1.3, можемо зробити наступні висновки:

Моделі BERT та її варіації (RoBERTa, ALBERT) мають великий потенціал для різних завдань обробки тексту, але вимагають значних обчислювальних ресурсів.

GPT-2 та GPT-3 підходять для генерації тексту, але можуть бути обмежені у точності та обчислювальних витратах.

Моделі XLNet, T5 та StructBERT враховують контекст з обох сторін слова, але вимагають значних обчислювальних витрат.

ELECTRA та DeBERTa надають покращене використання обчислювальних ресурсів, але можуть вимагати додаткової підготовки або ресурсів для тренування.

1.3 Інноваційна роль моделей обробки природної мови в тестуванні IT-проєктів

NLP – це підгалузь штучного інтелекту (ШІ), яка займається розробкою алгоритмів і моделей для обробки та аналізу людської мови. NLP поєднує в собі принципи комп'ютерних наук, математики, лінгвістики та когнітивної психології, щоб дати можливість машинам розуміти та взаємодіяти з людською мовою.

NLP охоплює широкий спектр завдань, включаючи розпізнавання мови, машинний переклад, аналіз настроїв, пошук інформації, класифікацію текстів, розпізнавання іменованих об'єктів, узагальнення текстів і відповіді на запитання. Ці завдання включають різні рівні лінгвістичного аналізу, від фонетики і морфології до синтаксису і семантики.

Однією з головних проблем NLP є неоднозначність і складність людської мови. Мова дуже залежить від контексту, і слова можуть мати кілька значень залежно від контексту, в якому вони використовуються. Крім того, мова може бути дуже мінливою, з різними діалектами, акцентами та регістрами. Системи NLP повинні бути здатні впоратися з цією мінливістю і неоднозначністю, щоб точно інтерпретувати і генерувати мову.

Системи NLP зазвичай використовують статистичні моделі та алгоритми машинного навчання для аналізу та обробки мовних даних. Ці моделі можна навчати на великих обсягах маркованих даних, що може підвищити їхню точність і продуктивність. Методи глибокого навчання, такі як нейронні мережі, також застосовуються в завданнях NLP, досягаючи найсучасніших результатів у таких завданнях, як машинний переклад і класифікація текстів.

В останні роки моделі NLP стали ключовим інструментом у багатьох сферах технологічної індустрії. В тестуванні IT-проєктів ці моделі також

знаходять широке застосування, переформовуючи традиційні підходи до тестування та забезпечуючи нові можливості для підвищення якості та ефективності процесу.

Хоча сфера тестування ПЗ набула значної популярності, залишаються десятки проблем, які не були ефективно вирішені. Наприклад, однією з таких проблем є автоматизована генерація модульних тестів. Незважаючи на різні підходи, включаючи методи на основі пошуку [11, 17], на основі обмежень [18] або на основі випадковостей [19] для створення набору модульних тестів, охоплення та значущість згенерованих тестів все ще далекі до задовільних [21]. Подібним чином, коли справа доходить до тестування мобільного GUI, існуючі дослідження з методами на основі випадковостей, методів на основі моделей і методів на основі навчання не можуть зрозуміти семантичну інформацію сторінки графічного інтерфейсу і часто не досягають повного охоплення. Враховуючи ці обмеження, зараз проводяться численні дослідницькі роботи з вивчення інноваційних методів, які можуть підвищити ефективність завдань тестування ПЗ, серед яких великі мовні моделі є найбільш перспективними. Великі мовні моделі, такі як BERT і GPT-3, зробили революцію в області NLP і ШІ. Ці моделі, спочатку пройшовши попередню підготовку з обширних корпусів, продемонстрували надзвичайну ефективність у широкому діапазоні завдань NLP, включаючи відповіді на запитання, машинний переклад і генерацію тексту [24-27]. В останні роки відбувся значний прогрес з появою моделей, здатних обробляти навіть більш масштабні набори даних. Це розширення розміру моделі призвело не тільки до підвищення продуктивності, але й відкрило нові можливості для застосування мовних моделей, як загального штучного інтелекту (ШІ). Такі моделі мають величезний потенціал для вирішення складних практичних завдань у таких областях, як генерація коду та художня творчості. Завдяки своїм розширеним можливостям мовні моделі змінили правила NLP та штучного інтелекту, а

також сприяють прогресу в інших сферах, таких як кодування та тестування ПЗ.

Мовні моделі використовувалися для різних завдань, пов’язаних із кодуванням, включаючи генерацію коду та рекомендації коду [28-31]. З одного боку, у тестуванні ПЗ існує багато завдань, пов’язаних із створенням коду, наприклад, створення модульного тесту [20], де очікується, що використання мовних моделей дасть хорошу продуктивність. З іншого боку, тестування ПЗ має унікальні характеристики, які відрізняють його від генерації коду. Наприклад, генерація коду в першу чергу зосереджується на створенні єдиного правильного фрагмента коду, тоді як тестування ПЗ часто вимагає створення різноманітних тестових вхідних даних для забезпечення кращого охоплення ПЗ, що тестується. Існування цих відмінностей створює нові виклики та можливості під час використання мовних моделей для ПЗ. Крім того, люди отримали вигоду від чудової продуктивності мовних моделей у задачах генерації та логічного висновку, що призвело до появи десятків нових практик, які використовують мовні моделі для тестування ПЗ [32].

Мовні моделі відіграють ключову роль у таких аспектах тестування ІТ-проєктів:

– автоматичне створення тестових даних. Мовні моделі можуть бути використані для генерації реалістичних тестових даних, що дозволяє автоматизувати процес створення тестових наборів даних та зменшити залежність від ручної роботи;

– генерація тестових сценаріїв. Застосування мовних моделей дозволяє автоматично генерувати тестові сценарії на основі специфікацій проєкту, документації та інших вхідних даних. Це допомагає виявляти потенційні проблеми та вразливості в програмному забезпеченні ще на етапі розробки;

– аналіз та інтерпретація результатів. Мовні моделі можуть бути застосовані для автоматичного аналізу результатів тестування, виявлення

шаблонів, аномалій та важливих закономірностей, що допомагає розробникам і тестувальникам швидше та ефективніше реагувати на виявлені проблеми;

– підтримка мовної локалізації. Мовні моделі допомагають у виявленні та виправленні проблем, пов'язаних із локалізацією програмного забезпечення та його адаптацією до різних мовних середовищ.

Враховуючи ці аспекти, використання мовних моделей стає необхідним інструментом для підвищення якості та швидкості процесу тестування ІТ-проєктів. У дослідженні планується детально розглянути вплив мовних моделей на ефективність тестування та виявити їхні переваги та обмеження в конкретних сценаріях застосування.

1.4 Постановка задачі дослідження

Результати аналізу сучасних мовних моделей для тестування ІТ-проєктів дозволяють зробити такі висновки. По-перше, проведений огляд та аналіз різних мовних моделей виявив їхні переваги та обмеження у контексті тестування програмного забезпечення. По-друге, виявлено, що багато з цих моделей не враховують специфіку завдань тестування ІТ-проєктів, що може обмежити їхню ефективність. По-третє, результати також підтверджують потребу в подальших дослідженнях та розробці нових методів оптимізації мовних моделей для кращого використання у тестуванні ІТ-проєктів. Ці висновки дозволяють припустити, що для досягнення ефективного та точного тестування ІТ-проєктів необхідні подальші дослідження та розвиток нових підходів до використання мовних моделей.

Таке представлення визначає мету даного дослідження полягає у вивченні та аналізі можливостей використання мовних моделей у процесі

тестування ІТ-проєктів. Для досягнення цієї мети необхідно забезпечити вирішення наступних завдань:

– провести аналітичний огляд літератури за темою роботи, щоб отримати повністю обґрунтоване розуміння сучасних тенденцій та підходів до використання мовних моделей у тестуванні ІТ-проєктів;

– дослідити різні мовні моделі, їхні можливості та обмеження в контексті тестування ІТ-проєктів;

– вибрати найбільш підходящу мовну модель для тестування ІТ-проєктів, враховуючи її можливості, обмеження та конкретні вимоги дослідження;

– провести апробацію обраної мовної моделі, щоб оцінити її ефективність та придатність у конкретному контексті тестування ІТ-проєктів.

2 ДОСЛІДЖЕННЯ МОВНИХ МОДЕЛЕЙ ДЛЯ ТЕСТУВАННЯ ІТ-ПРОЄКТІВ

2.1 Дослідження основних різновидів мовних моделей

Протягом останнього десятиліття векторне представлення слів набули особливої значущості, ставши ключовим елементом множини систем NLP. Такий стан речей обумовлено характером методів NLP, що оперують природною мовою, яка, як правило, представлена у вигляді тексту. Структура тексту передбачає дроблення на більш малі одиниці – слова і символи, які піддаються прямому комп'ютерному сприйняттю. Це обумовлює необхідність у векторних представленнях слів, що надають можливість чисельного кодування текстового введення, доступного для читання, обробки та аналізу комп'ютерних програм.

Для формального представлення, нехай функція відображення слів у вектор буде позначено як [33]:

$$f : W \to R^n, \tag{2.1}$$

де f – функція відображення слів у вектор;

W – множина всіх слів природної мови;

R^n – простір векторів розмірності n.

Тоді для кожного слова w з W, його векторне представлення може бути визначене як:

$$f(w) \in R^n, \tag{2.2}$$

де $f(w)$ – векторне представлення слова w;

R^n – простір векторів розмірності n.

У межах практичного застосування є кілька підходів до векторизації слів. Ймовірно, найпростішим є метод «швидкого кодування» (one-hot encoding). Припустимо, що є словник, що складається з N слів. Цей метод полягає у привласненні кожному слову цілочисленного індексу i, де $i \in \{1,\ldots,N\}$. При такому відображенні слова в ціле число, кожне слово представляється у вигляді N-вимірюваного розрідженого вектора, заповненого в основному нулями, за винятком одного елемента на позиції, що відповідає індексу слова в словнику. Цей елемент набуває значення 1.

Вектори швидкого кодування є простим способом чисельного подання слів, проте з ними пов'язані дві важливі проблеми. По-перше, розмір вектора збільшується зі збільшенням розміру словника, що є очевидним недоліком з погляду розмірності ознак. Розширення кола ознак тягне у себе зростання кількості параметрів з метою оцінки, і навіть необхідності додаткової обчислювальної потужності. По-друге, швидке кодування не враховує схожість між словами. Набагато важливіше мати спосіб представлення, який би відображав лінгвістично значущі відносини між словами. Зважаючи на зазначені причини, one-hot кодування рідко використовується безпосередньо як векторне подання слів, а скоріше служить вхідними даними для більш складних методів векторизації слів.

Щоб вирішити проблему подібності, що виникає в контексті використання one-hot кодування, дослідники в галузі NLP вдалися до дистрибутивної гіпотези, представленої З.С. Харрісом [34]. Дистрибутивна гіпотеза передбачає, що з розуміння значення слова необхідно аналізувати його сусідів у реченні чи більш широкому контексті тексту. Вона виходить з спостереженні, що семантично близькі слова мають схожий контекст використання і найчастіше взаємозамінні в реченнях.

Подальше поширення цієї теорії забезпечив англійський лінгвіст Джон Р. Фірз, який стверджував: «You shall know a word by the company it keeps» [35], що можна перекласти як «По оточенню слова дізнаєшся його суть».

Сучасні методи векторного представлення слів базуються на цій гіпотезі, хоча механізми її реалізації можуть змінюватись. Основні підходи до моделювання векторних уявлень слів можна класифікувати на дві основні категорії:

– моделі на основі розрахунку (count-based models) використовують статистику по всьому корпусу, включаючи підрахунок слів та частоти, для створення уявлень слів. Математично ці моделі будують векторний простір слів на основі кількісних характеристик корпусу;

– моделі на основі прогнозування (prediction-based models) навчаються представленням, максимізуючи їхню прогнозувальну здатність. Тобто ці моделі прагнуть передбачити слово на основі його контексту чи навпаки. Тут логіка полягає у оптимізації функції втрат, яка оцінює здатність моделі прогнозувати контекст слова.

Останнім часом виникла третя категорія: контекстулізовані моделі (contextual models). Ці моделі застосовують унікальний підхід, що ґрунтується на використанні локального контексту для створення представлення слова.

Контекстуалізовані моделі покликані усунути одне з обмежень попередніх підходів, а саме те, що кожному слову в моделі відповідає єдине векторне представлення незалежно від контексту його вживання. На відміну від стандартних векторних уявлень, що надають статичне представлення слова, контекстні моделі генерують динамічні представлення. Іншими словами, векторне представлення слова може змінюватись в залежності від його контексту. Це дозволяє моделям точніше вловлювати нюанси природної мови, такі як полісемія – здатність слів мати кілька значень залежно від контексту.

Одним із найпопулярніших представників контекстних моделей є BERT, розроблений Google AI. BERT використовує механізми трансформера для кодування контекстної інформації слова, враховуючи при цьому інформацію

як праворуч, так і зліва від слова, що розглядається. Цей підхід дозволив BERT досягти визначних результатів у низці завдань NLP.

2.2 Розрахункові моделі

Моделі, засновані на підрахунку входжень (count-based models), формуються на основі матриці спільної слів. Ця матриця складається шляхом проходу великим корпусом текстових документів з метою підрахунку кількості спільних входжень двох або більше слів в набір даних. На практиці існують два типи спільної зустрічальності: за документами та вікнами.

Відомі моделі, що базуються на розрахунку входжень, включають Латентно-семантичний аналіз (Latent Semantic Analysis, LSA) [36], Гіперпросторовий аналог мови (англ. Hyperspace Analog to Language, HAL) [37], Корельований аналог лексичної семантики (англ. Correlated Occurrence Analogue to Lexical Semantics, COALS) [38] та Hellinger-PCA [39]. Незважаючи на те, що ці методи ефективно використовують глобальну статистичну інформацію, вони насамперед використовуються для виявлення подібностей слів і погано справляються із завданнями, що виходять за рамки одиничних смислів, що вказує на неоптимальну з точки зору підсумкової якості структуру векторного простору.

Говорячи про методи представлення мовних даних в автоматизованих структурах даних, не можна не згадати про n-грамні моделі. N-грама – це послідовність з n слів: біграма описує послідовність з двох слів, наприклад, «здайте домашнє», у той час як триграма описує послідовність із трьох слів, наприклад, «здайте домашнє завдання». Елементарна у своїй концепції порівняно з більш передовими нейронними мовними моделями, що базуються

на рекурентних нейронних мережах або трансформерах, n-грамна модель несе важливу цінність у розумінні базових принципів мовного моделювання.

Стосовно математичного аспекту, n-грамні моделі ґрунтуються на розрахунку частоти послідовностей з n слів у навчальному корпусі текстів. Основна ідея n-грамної моделі полягає у передбаченні ймовірності наступного слова на основі попередніх $n-1$ слів. У контексті біграмної моделі ($n=2$), ймовірність слова w за умови попереднього слова v можна обчислити по формулі:

$$P(w|v) = Count(v,w)/Count(v), \tag{2.3}$$

де $P(w|v)$ – ймовірність слова w за умови попереднього слова v;

$Count(v,w)$ – кількість випадків, коли слово w слідує за словом v;

$Count(v)$ – загальна кількість випадків.

Одна з значних переваг n-грамної моделі – її простота та операційна ефективність. Модель не вимагає витратних обчислювальних ресурсів і може бути навчена великих текстових корпусів. Однак є і ряд недоліків, найбільш помітними з яких є припущення про те, що ймовірність наступного слова залежить тільки від попередніх $n-1$ слів, що в контексті реального використання природної мови може бути не коректно, оскільки тема-рематична організація тексту зазвичай охоплює широкий контекст, ніж тільки $n-1$ попередніх слів. Також важливим недоліком є проблема «розсіювання даних», коли модель присвоює нульову ймовірність n-грам, які не були зустрінуті в навчальному корпусі, попри те, що в мовному розмаїтті вони можуть зустрітися.

Для подолання цієї проблеми використовуються методи згладжування, такі як згладжування Лапласа і згладжування Кнесера-Ней, які встановлюють

зв'язок між розподілом ймовірностей для n-грам, що зустрічаються в навчальному корпусі, і n-грамами, які в ньому не зустрічаються [40].

2.3 Прогнозувальні моделі

2.3.1 Нейронні мережі

Нейронна мовна модель (англ. Neural Network Language Model, NNLM) є інноваційним підходом, що поєднує процес навчання векторного подання слів зі створенням статистичної моделі мови [41]. Дана модель використовує багатошарову нейронну мережу прямого поширення, яка включає лінійний проекційний шар і нелінійний прихований шар.

Функція задається як композиція двох відображень, C та g відповідно до наступної формули:

$$f(i, w_{t-1}, \ldots, w_{t-n+1}) = g(i, C(w_{t-1}), \ldots, C(w_{t-n+1})), \quad (2.4)$$

де $f(i, w_{t-1}, \ldots, w_{t-n+1})$ – функція, що відображає слово і в вектор ознак;

w_1, w_2 – вектори ознак слів у контексті;

g – нелінійна функція активації;

$C(w_{t-1})$ – функція перетворення слова у вектор ознак;

і – індекс слова.

Відображення C відповідає функції перетворення слова у вектор ознак, що відповідає кожному слову у словнику.

Відображення g, в свою чергу, приймає послідовність векторів ознак слів у контексті, таких як $C(w_{t-n+1}), \ldots C(w_{t-1})$ відображає її на умовний розподіл ймовірностей за словами у словнику для передбачення наступного слова w_t.

Це досягається шляхом генерації вектора, де *i*-й елемент є оцінкою ймовірності завдяки застосуванню softmax-шару.

Процес навчання моделі полягає в оптимізації параметрів Θ, що здійснюється шляхом максимізації логарифму:

$$L = \frac{1}{T}\sum_{t} \log f\left(w_t, w_{t-1}, \ldots, w_{t-n+1}; \Theta\right), \tag{2.5}$$

де L – функція втрат;

Θ – вхідні параметри;

T – загальна кількість слів.

Основним недоліком таких мовних моделей, безумовно, є фінальний шар softmax, оскільки вартість обчислення softmax пропорційна числу слів у словнику, яке зазвичай становить сотні тисяч або мільйони. Отже, моделі NNLM є дуже ресурсоємними.

2.3.2 Модель «безперервного мішка слів»

У той час як мовна модель здатна орієнтуватися тільки на попередні слова для своїх прогнозувань (оскільки її ефективність оцінюється на основі здатності передбачати кожне наступне слово, враховуючи послідовність попередніх слів) модель, основною метою якої є створення точних векторних уявлень слів, може оминути це обмеження. Модель «безперервного мішка слів» (англ. Continuous Bag of Words, CBOW) використовує як n слів до, так і після цільового слова w_t для свого прогнозування (звідси і назва «безперервний», оскільки вона використовує безперервні представлення, порядок яких не має значення) [42]. Отже, мета навчання трохи відрізняється

від мети навчання NNLM, представленої у вищезгаданому рівнянні. Замість подачі n попередніх слів у модель вона отримує вікно з n слів навколо цільового слова w_t на кожному тимчасовому кроці t, так що функція втрат набуває вигляду:

$$L = \frac{1}{T}\sum_{t} \log p\left(w_t \mid w_{t-n}, \ldots, w_{t-1}, w_{t+1}, \ldots, w_{t+n}\right), \tag{2.6}$$

де L – функція втрат;

p – ймовірність передбачення контекстного слова;

w_t – цільове слово;

t – тимчасовий крок.

Крім того, архітектура моделі відрізняється від мережі прямого поширення тим, що прихований нелінійний шар видалений, що знижує обчислювальні витрати.

2.3.3 Модель Skip-Gram

Модель Skip-Gram (SG) дуже схожа на модель «безперервного мішка слів», але замість прогнозування поточного слова на основі контексту вона прогнозує оточуючі контекстні слова, виходячи з центрального слова w_t [43]. Отже, завдання Skip-Gram – максимізувати суму логарифмічних ймовірностей навколо цільового слова $w_t n$ слів зліва і праворуч від нього:

$$L = \frac{1}{T}\sum_{t=1}^{T} \sum_{-c<j<c} \log p\left(w_{t+j} \mid w_t\right), \tag{2.7}$$

де L – функція втрат;

p – ймовірність передбачення контекстного слова;

w_t – цільове слово;

t – тимчасовий крок;

c– розмір вікна контексту.

У Skip-Gram перехід від вхідного шару до прихованого шару працює аналогічно CBOW на рівні слова, як представлено в рівнянні вище. Іншими словами, відбувається копіювання та транспонування рядка з матриці переходу W, що асоціює входження з унікальним вхідним словом W_i. Різниця полягає у вихідному шарі, який замість видачі одного мультиноміального розподілу видає n мультиноміальних розподілів.

Модель Skip-Gram є основою другої реалізації знаменитого методу Word2Vec [42, 43], і навіть відомого FastText [44]. Особливістю FastText є те, що замість прямого навчання представленням слів він представляє кожне слово як N-граму символів і навчає подання для кожної N-грами символів, тому загальне векторне представлення слова є сумою цих уявлень N-грам. Головна перевага цього підходу полягає в тому, що він може обробляти рідкісні, що не спостерігаються в корпусі слова, що не вдається зробити Word2Vec.

На додаток до моделей CBOW та SG, були запропоновані додаткові стратегії для покращення швидкості та точності [43], які були так само революційними, як і самі моделі. Ці методи в основному фокусуються на поліпшенні обчислення підсумкового розподілу ймовірностей, який спочатку неефективно виконувався за допомогою функції softmax, як представлено вище. Альтернативи включають негативну вибірку (negative sampling) та ієрархічний softmax.

Негативна вибірка застосовується спрощенням функції втрат. Замість обчислення градієнта всіх слів у словнику, обчислюємо градієнт лише одного

позитивного прикладу (цільового слова) і невеликого числа негативних прикладів (слів, обраних випадковим чином, відмінним від цільового). Цей метод значно зменшує обчислювальну складність, особливо під час роботи з великими словниками.

Ієрархічний softmax використовує бінарне дерево (зазвичай дерево Хаффмана), у якому слова словника представлені як листя. Шлях від кореня до листя визначає кодування слова. Це дозволяє зменшити складність обчислення ймовірностей слів за допомогою softmax до логарифмічної.

Практичне використання Word2Vec досить різноманітне. Над векторними представленнями слів, отриманими за допомогою Word2Vec, можуть бути здійснені лінійні перетворення, щоб отримати, наприклад, єдиний нормалізований вектор для пропозиції або цілого тексту. Також дані вектора можуть бути зважені щодо частотності слова в корпусі або використовуватися як вхід для іншої моделі, зокрема, нейромережі або дерев рішень.

FastText, як і Word2Vec, можна навчати двома основними способами: Skip-Gram (з пропуском *n*-грам) та CBOW. Обидві моделі навчаються з урахуванням максимізації ймовірності контекстних слів кожного центрального слова. Слова FastText представляються не як окремий вектор, а як суму векторів символьних *n*-грам.

2.4 Контекстуалізовані моделі

Донедавна всі методи формування векторних уявлень слів упускали один важливий аспект, що дозволяє повноцінно відображати синтаксичні та семантичні параметри слів: локальний контекст. Усі поширені методи подання слів, такі як Word2Vec чи FastText, є контекстно-незалежними. Ці методи

стають здатними виявляти загальний (тобто найбільш поширений) контекст слів у їхніх векторних представленнях, спостерігаючи за тим, як це слово часто зустрічається у тому чи іншому контексті. Однак вони аж ніяк не можуть впоратися з полісемією, оскільки єдине слово з множинними значеннями завжди відображається в унікальному векторі. Таким чином, ці методи успішно справляються зі словами, що представляють унікальний концепт, але не можуть передати реальне значення слова, що неминуче залежить від його локального контексту.

Щоб вирішити цю проблему, дослідники дедалі більше цікавляться глибокими контекстуалізованими векторними представленнями слів. Ідея проста: токену присвоюється представлення, яке є функцією всього вхідного речення. Ранні роботи, присвячені навчанню контекстно-залежних уявлень, включали CoVe [45], ELMo [46] та ULMFiT [47], та всі вони ґрунтувалися на двонаправленій моделі LSTM для кодування контексту. Нещодавне введення архітектури Transformer [48] призвело до появи потужних попередньо навчених мовних моделей, таких як BERT [7], XLNet [8] та інших, які продемонстрували свою ефективність у широкому спектрі завдань NLP.

2.5 Модель BERT

BERT (Bidirectional Encoder Representations from Transformers), представлений дослідниками Google AI [7], є глибокою контекстуалізованою мовною моделлю. Ця архітектура розроблена для попереднього навчання глибоких двонаправлених вкладень слів із нерозміченого тексту із спільним обліком лівого та правого контексту. В результаті, попередньо навчена модель може бути доопрацьована за допомогою всього одного додаткового вихідного

шару, що дозволяє створити високоефективні моделі для широкого спектра завдань NLP, таких як класифікація тексту.

Механізм самоуваги (self-attention) є особливою формою механізму уваги [49], вперше представлену з моделлю Transformer [48]. Коротко, це механізм, що пов'язує різні позиції однієї послідовності з метою обчислення контекстуального подання кожного терміну цієї послідовності.

У більш формальному сенсі механізм самоуваги може бути визначений як перетворення, яке зіставляє вектор запиту і велику кількість пар ключ-значення на вихідні дані, причому запити, ключі, значення і результати всі представлені у вигляді векторів. Для більш конкретного опису, при заданій вхідній послідовності розміру N, механізм самоуваги виконує наступні операції для кожного елемента $i(i=1,\ldots,N)$ у наступній послідовності.

Вектори запиту (query vector) q_i та вектори ключа (key vector) k_j, обидва розмірності d_k, а також вектор значення v_i розмірності d_v обчислюються шляхом множення початкового векторного подання $x_i \in R^{d_{model}}$ елемента *i* на три матриці ваг $W^Q \in R^{d_{model} \times d_k}$, $W^K \in R^{d_{model} \times d_k}$ і $W^V \in R^{d_{model} \times d_k}$, які налаштовуються в процесі навчання моделі:

$$q_i = x_i \cdot W^Q, \tag{2.8}$$

де q_i – вектор запиту;

x_i – початкове векторне подання;

W^Q – матриця ваг.

$$k_i = x_i \cdot W^K, \tag{2.9}$$

де k_i – вектор ключа;

x_i – початкове векторне подання;

W^K – матриця ваг.

$$v_i = x_i \cdot W^V, \tag{2.10}$$

де v_i – вектор значень;

x_i – початкове векторне подання;

W^V – матриця ваг.

Кожному елементу і надається оцінка щодо всіх інших елементів у послідовності за допомогою скалярного відтворення його вектора запиту q_i з усіма векторами ключів k_j в послідовності:

$$s_{ij} = q_i k_j, j \in 1 \ldots N. \tag{2.11}$$

де s_{ij} – оцінка елементу щодо всіх інших елементів.

Дані оцінки потім необхідно пропустити через операцію Softmax, щоб привести їх до розподілу ваг, що підсумовують.

Потім відбувається операція перемноження кожного вектора значень з відповідною вагою:

$$v_{ij} = s_{ij} v_i, \tag{2.12}$$

де v_{ij} – зважений вектор.

Зважені вектори значень підсумовуються, отримуючи в результаті остаточний результат обчислення самоуваги.

На практиці функція самоуваги обчислюється паралельно, об'єднавши токени в матрицю $Q \in R^{N \times d_x}$. Відповідним чином значення запаковуються в матриці для пар ключ-значення: $K \in R^{N \times d_x}$ та $V \in R^{N \times d_x}$. Отже, вихідна матриця розраховується за такою формулою [61]:

$$Attention(Q,K,V) = soft\max\left(\frac{QK^T}{\sqrt{d_k}}\right)V, \quad (2.13)$$

де Q – матриця запитів;

K – матриця ключів;

V – матриця значень;

T – кількість токенів;

$\sqrt{d_k}$ – фактор масштабування для стабілізації градієнтів при великих значеннях.

Алгоритм роботи механізму самоуваги моделі BERT зображено на рисунку 2.1 [59].

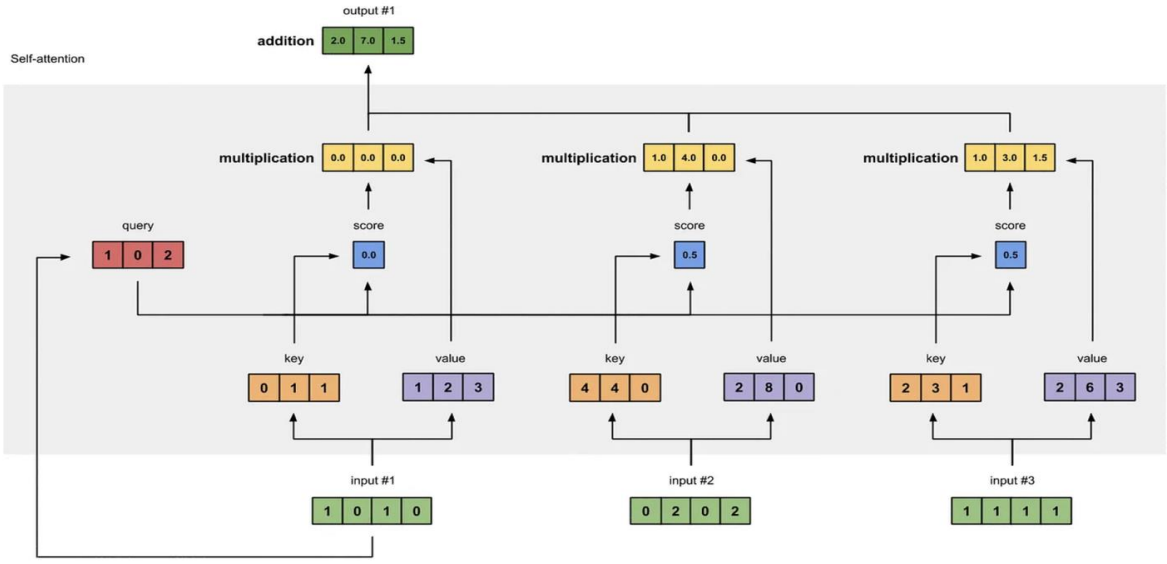

Рисунок 2.1 – Алгоритм роботи механізму самоуваги моделі BERT

Моделі на основі трансформера не обмежуються однією операцією самоуваги з ключами, значеннями та запитами певної розмірності. Вони включають ще один рівень складності, впроваджуючи концепцію «багатоголовкової» уваги (Multi-head Attention). По суті, цей підхід передбачає лінійну проекцію запитів, ключів і значень h разів з використанням різних

лінійних проекцій, що навчаються. Це забезпечує можливість зосередити увагу на різноманітній інформації з різних підпросторів уявлень на різних позиціях.

Так, функція самоуваги реалізується паралельно на різних проекціях матриць запиту, ключа і значення. В результаті формуються *h* різних вихідних матриць, відомих як «голови уваги». Ці голови уваги потім конкатенуються і проєктуються в інший підпростір представлення, що призводить до створення остаточної вихідної матриці «багатоголовкової» уваги.

Суть концепції «багатоголовкової» уваги полягає в тому, що кожна «голова» може спеціалізуватися на виявленні певного типу взаємозв'язків у даних, що робить цю модель загалом ефективнішою порівняно з моделлю, що використовує одну голову уваги. Ця структура дозволяє моделі більш повно і точно розуміти контекст та семантику тексту, що суттєво підвищує її продуктивність на різних завданнях NLP.

Зрештою, фінальний вихід після «багатоголовкової» уваги виглядає так:

$$MultiHead(Q,K,V) = Concat(Z_1,\ldots,Z_h)W^O, \qquad (2.14)$$

де $MultiHead(Q,K,V)$ – функція механізму багатоголової уваги;

Z – результат окремого механізму уваги;

W^0 – матриця ваг для об'єднання результатів.

Архітектура моделі BERT ґрунтується на принципі багатошарового двонаправленого трансформерного кодувальника. В основі BERT лежить стек із *L* ідентичних шарів трансформера. У кожному шарі є два типи підшарів.

Перший підшар реалізує механізм «багатоголовкової» уваги. Його основне завдання полягає в тому, щоб при кодуванні певного слова враховувати контекст, поданий іншими словами у послідовності.

Другий підшар являє собою позиційно-орієнтовану повнозв'язну пряму мережу, що застосовується до кожної позиції послідовності окремо, і включає

два лінійних перетворення. Розмірність вхідних та вихідних даних цього підшару складає d_{model}. Важливо відзначити, що цей підшар використовує функцію активації GELU, яка демонструє кращу ефективність порівняно зі стандартною ReLU у рамках трансформерного кодувальника.

Порівняння архітектури BERT на основі кодувальника проти архітектури перетворювача тексту в текст на основі кодувальника-декодера наведено на рисунку 2.2 [60].

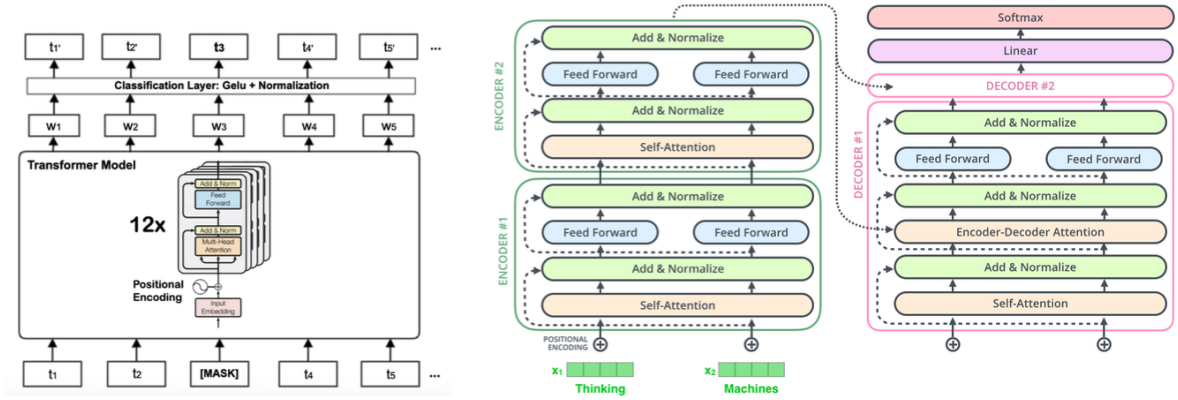

Рисунок 2.2 – Архітектура BERT на основі кодувальника (ліворуч) проти архітектури перетворювача тексту в текст на основі кодувальника-декодера (праворуч)

Варто згадати і про залишкові зв'язки (residual connections), які реалізуються в кожному шарі кодувальника. Вони використовуються навколо кожного з двох підшарів, після чого відбувається нормалізація шару. В результаті, вихід кожного підшару є $LayerNorm(x + Sublayer(x))$, де $Sublayer(x)$ представляє собою функцію, реалізовану всередині підшару.

Всі підшари в моделі BERT генерують вихідні дані однієї і тієї ж розмірності d_{model}, що полегшує процес залишкового зв'язування. Варто відзначити, що, незважаючи на те, що лінійні перетворення однакові для

різних позицій усередині одного підшару, модель BERT використовує різні параметри для різних шарів.

2.6 Обґрунтування вибору мовної моделі для тестування ІТ-проєкту

В сучасному цифровому світі, де роль програмного забезпечення стає все більш визначальною для успіху бізнесу та комфорту користувачів, ефективне тестування ІТ-проєктів стає надзвичайно важливим завданням. Зростаюча складність та об'єми програмного коду вимагають від тестувальників не лише високої кваліфікації, але й використання передових інструментів та методів для ефективного аналізу та перевірки функціональності програм.

У цьому контексті, роль мовних моделей для тестування ІТ-проєктів набуває особливого значення. Вибір відповідної моделі для конкретного проєкту впливає на його результативність, швидкість та надійність. В даній роботі було вирішено вибрати модель BERT для тестування ІТ-проєктів, в зв’язку з наступними моментами:

– здатність до роботи з текстом різного структурованості. Модель BERT відома своєю здатністю до роботи з текстом різної природи і структури, що робить її ефективним інструментом для тестування ІТ-проєктів, де можуть зустрічатися тексти різного характеру – від коротких описів функціоналу до довгих технічних документів;

– контекстуальне розуміння. BERT здатний до контекстуального розуміння тексту, що дозволяє моделі краще інтерпретувати сенс інформації, враховуючи його зв'язок з навколишнім контекстом. Це особливо важливо для тестування ІТ-проєктів, де розуміння контексту може бути ключовим для правильного та повністю охопленого тестування;

– широке застосування та підтримка. Модель BERT має широке застосування та підтримку у великій кількості областей, включаючи тестування програмного забезпечення. Вона є досить популярною серед дослідників та практиків у сфері NLP, що забезпечує доступність додаткових ресурсів, які можна використовувати для оптимізації та розвитку моделі для конкретних потреб проєкту;

– становлення стандартом. BERT стає де-факто стандартом у багатьох завданнях NLP, включаючи різноманітні аспекти тестування програмного забезпечення. Вибір цієї моделі може сприяти сумісності та порівняльній оцінці результатів з іншими дослідженнями та проєктами, що використовують аналогічні підходи;

– ефективність та результативність. Дослідження показують, що модель BERT демонструє високу ефективність та результативність у різних завданнях NLP. Її застосування для тестування IT-проєктів може підвищити швидкість та точність процесу тестування, що в свою чергу може призвести до покращення якості та надійності розроблюваного програмного забезпечення.

Таким чином, вибір моделі BERT для тестування IT-проєктів є обґрунтованим рішенням, яке може сприяти підвищенню якості, швидкості та надійності розроблюваного програмного забезпечення.

2.7 Висновки до другого розділу

Проведено аналіз основних різновидів мовних моделей для тестування IT-проєктів. Досліджено розрахункові та прогнозувальні моделі, такі як нейронні мережі, модель «безперервного мішка слів» та модель Skip-Gram. Також розглянуто контекстуалізовані моделі.

Розглянуті моделі відображають різні аспекти мовної обробки та можуть бути використані для вирішення різноманітних завдань у тестуванні програмного забезпечення. Важливо враховувати специфіку проєкту та вимоги до точності та швидкості обробки інформації при виборі підходящої моделі.

В результаті проведеного аналізу відзначено, що модель BERT виявилася особливо перспективною для застосування у тестуванні IT-проєктів. Її здатність до контекстуального аналізу тексту та широкий функціонал роблять її потужним інструментом для покращення процесів тестування програмного забезпечення.

3 АДАПТАЦІЯ МОВНОЇ МОДЕЛІ BERT ДЛЯ ТЕСТУВАННЯ ІТ-ПРОЄКТІВ

3.1 Подання вхідних даних

У тестуванні ІТ-проєктів, використання моделі BERT для обробки вхідних даних вимагає уважного підходу до токенізації та структурування текстової інформації. З огляду на різноманітність типів даних у цій галузі, вхідні дані можуть включати не лише тексти програмного коду, але й логи, документацію, звіти про тестування, технічні специфікації тощо.

Для ефективного використання моделі BERT, важливо правильно відобразити структуру вхідних даних у формат, придатний для обробки. Наприклад, тексти програмного коду можуть бути розділені на фрагменти по функціям або класах, щоб забезпечити збереження контексту та зв'язності. Логи та звіти можуть бути розбиті на окремі записи або події для аналізу впливу певних подій на функціонування системи [7].

Архітектура BERT перетворює вхідну послідовність слів у числові представлення, обмежуючись 512 токенами. Це початкове числове представлення створюється шляхом комбінування трьох різних видів вбудовувань: вбудовування токенів, сегментів та позицій [9].

Важливою стадією в цьому є токенізація слів за допомогою методу WordPiece. Суть цього методу полягає у створенні фіксованого словника, який включає окремі символи, підслів та цілі слова, що найбільш підходять для даного корпусу тексту. Процес токенізації починається з перевірки наявності цілого слова у словнику. Якщо слово у словнику не знайдено, воно розбивається на найбільші можливі підслів, що містяться у словнику. Якщо підслова немає у словнику, слово розщеплюється на окремі символи. Після обробки слово перетворюється на один або кілька токенів WordPiece. Ідентифікатори цих токенів потім використовуються для отримання

відповідних вбудов з навченої матриці вбудовування токенів. Словник, який використовує BERT, включає приблизно 30 000 слів і підслів англійської мови, що найчастіше використовуються, а також всі англійські символи і три спеціальні токені: [CLS], [SEP] і [MASK] (наведено на рисунку 3.1).

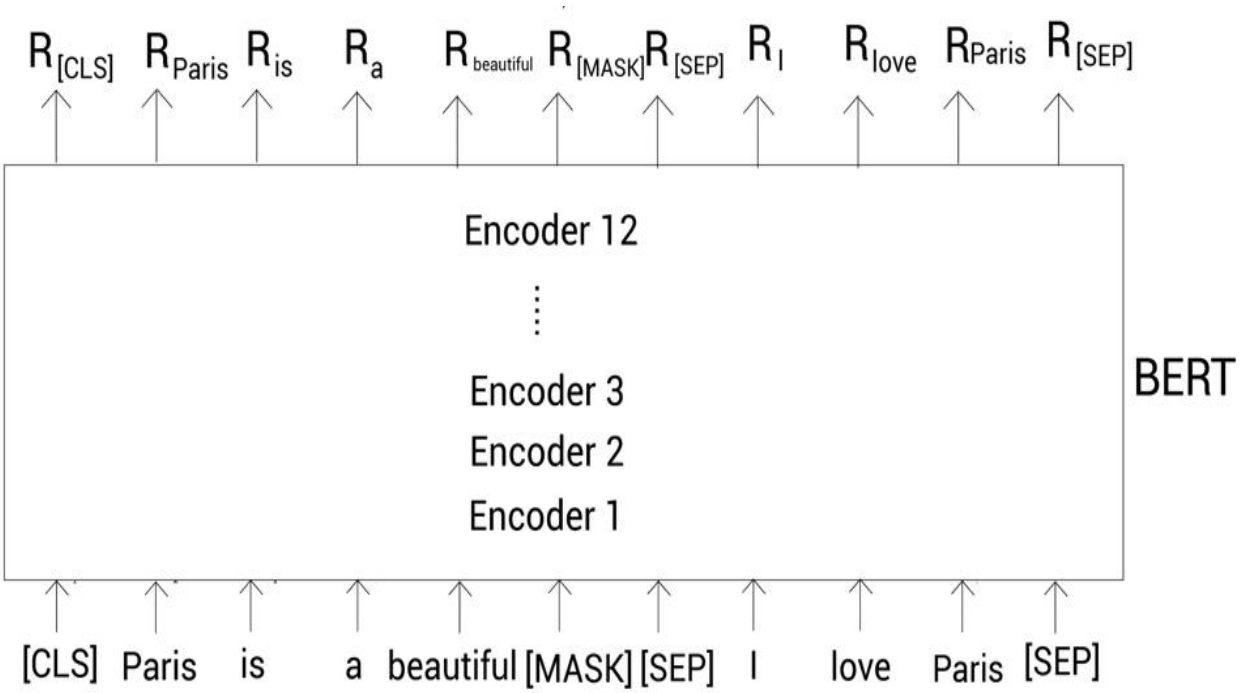

Рисунок 3.1 – Спеціальні токени моделі BERT

Коли BERT обробляє пари пропозицій, до кожного токену додається вбудовування сегмента, що навчається, яке вказує, чи належить токен пропозиції A або B [12].

Щоб впровадити інформацію про роль токенів у контексті їхньої позиції у вхідній послідовності, BERT використовує позиційні вбудовування. Вони розраховуються за допомогою синусоїдальних та косинусоїдальних функцій різних частот. Завдяки цьому уявленню модель здатна легко вчитися відносним позиціям, тому що для будь-якого фіксованого зміщення k, PE_{pos+k} може бути представлено як лінійна функція PE_{pos}.

Отже, ефективне подання вхідних даних для моделі BERT у тестуванні IT-проєктів вимагає комбінації технічного розуміння предметної області та вміння правильно структурувати та обробляти різноманітні тексти та дані.

3.2 Параметри моделі BERT

У контексті тестування IT-проєктів важливо уважно вибирати параметри моделі BERT з метою оптимізації її продуктивності та точності при обробці різноманітних типів даних, що характерні для цієї області. Архітектура BERT представлена у двох варіантах: базовому та розширеному. Базовий варіант містить близько 110 мільйонів параметрів, а розширений – 340 мільйонів параметрів [7]. Серед цих параметрів є:

– матриці вбудовування токенів, позицій та сегментів у вхідному шарі вбудовування;

– матриці ваг запитів, ключів і значень кожного підшару самоуваги;

– h означає кількість голів уваги;

– матриці проекцій виведення «багатоголовкової» уваги у кожному підшарі самоуваги;

– параметри прямого розповсюдження мережі в кожному підшарі прямого поширення;

– параметри залишкового з'єднання кожного шару.

Для моделі BERT-base використовується наступний набір параметрів:

– $d_{model} = 768$ (для BERT-large значення $d_{model} = 1024$);

– $d_{voc} = 30{,}522$ (для чутливого до регістру словника ($d_{voc} = 28{,}996$ для словника, нечутливого до регістру);

– $d_{context} = 512$.

Детальне врахування цих параметрів під час конфігурації моделі BERT для тестування IT-проєктів допомагає забезпечити оптимальну продуктивність та точність результатів у вимогливих умовах реального світу.

Для цього дослідження буде використано підтип моделі BERT – «bert-base-cased», який є меншою версією та містить 110 мільйонів параметрів, порівняно з оригінальними 340. Слово «cased» у назві означає чутливість до регістру, тобто модель розрізняє між різними комбінаціями малих та великих літер. Переваги вибору моделі на основі BERT включають:

– попереднє навчання моделі, що дозволяє використовувати її з лише додатковим налаштуванням для конкретної задачі;

– широке застосування та популярність у науковій спільноті, що призводить до наявності багатьох ресурсів та попередньо навчених моделей у відкритому доступі;

– висока точність та продуктивність в метриках NLP при достатніх зусиллях навчання та попередньої підготовки.

Недоліки вибору BERT включають:

– великі обчислювальні вимоги, що роблять її повільною для навчання та складною для розгортання в середовищах з обмеженими ресурсами;

– потреба великої кількості даних для навчання, що може бути проблемою, якщо доступ до даних обмежений;

– складність моделі, що може ускладнити розуміння та інтерпретацію її прогнозів.

3.3 Процедура навчання моделі BERT

У тестуванні IT-проєктів, навчання моделі BERT вимагає уважної уваги до деталей процесу навчання та вибору оптимальних стратегій для досягнення

високої точності та надійності передбачень. BERT навчається одночасно на двох завданнях: маскованого мовного моделювання (англ. Masked Language Modeling, MLM) та прогнозування наступного речення (англ. Next Sentence Prediction, NSP). Функція втрат при навчанні BERT складається із суми середніх ймовірностей для завдань MLM та NSP [13].

Завдання маскованого мовного моделювання відрізняється від класичного моделювання мови, яке передбачає наступне слово на основі попередніх. У MLM, навпаки, метою є прогнозування випадково замаскованих токенів у вхідній послідовності. MLM була обрана для навчання BERT замість традиційного моделювання мови, оскільки BERT використовує контекст обох сторін слова, що прогнозується. Якби використовувався стандартний підхід, кожне слово могло б побічно «бачити себе» у контексті, що зробило б прогнозування цільового слова тривіальним завданням. Замість цього, BERT використовується підхід, заснований на MLM, який також відомий як завдання Clozer. Алгоритм виконання завдання Cloze наведено на рисунку 3.2 [51].

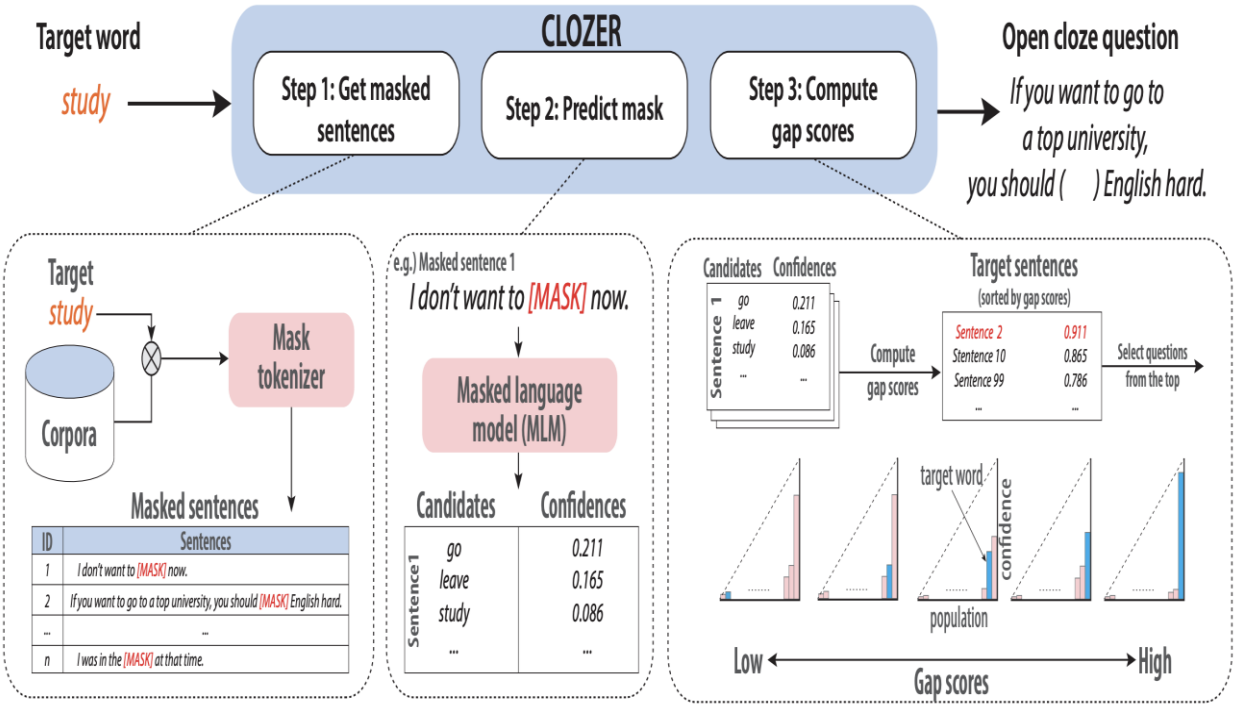

Рисунок 3.2 – Алгоритм виконання завдання Clozer

Механізм маскування в BERT працює наступним чином: BERT випадково вибирає 15% всіх токенів WordPiece в кожній тренувальній послідовності. Якщо обраний *i*-й токен, він замінюється на:

– токен [MASK] у 80% випадків;

– випадковий токен у 10% випадків;

– залишається незмінним у 10% випадків.

Такий підхід до маскування був обраний, щоб уникнути розриву між попереднім навчанням та донавчанням, оскільки токен [MASK] ніколи не зустрічається до фази донавчання.

Прогнозування наступного речення – це завдання бінарної класифікації, в якій модель отримує кілька речень та навчається прогнозувати, чи є друге речення у парі наступним реченням в оригінальному корпусі. Це завдання допомагає моделі покращити розуміння зв'язків між реченнями, що не досягається прямим моделюванням мови, але має важливе значення для багатьох наступних завдань, таких як відповіді на питання та логічні висновки природною мовою.

Завдання передбачення наступного речення може бути легко сформоване з будь-якого одномовного корпусу. Конкретно, при виборі речень A і B для кожного прикладу попереднього навчання, у 50% випадків B є реальним наступним реченням після A (маркується як IsNext), та в інші 50% випадків це випадкове речення з корпусу (маркується як NotNext). У цьому випадку підсумковий прихований вектор, відповідний токену [CLS], подається на вихідний softmax за двома можливими прогнозами [52].

Обрана модель є попередньо навченою та доступною у відкритому доступі разом з усіма параметрами. Вона базується на англійській Вікіпедії та BookCorpus, який містить приблизно 800 мільйонів слів тексту з 11000 книг. Ця модель була обрана через широкий спектр даних для підготовки та доступність даних BookCorpus у відкритому доступі. Однак, вона може містити упередженості та не точно відображати певні аспекти через своє

походження від даних, створених людьми. Також, модель bert-base-cased обмежена англійською мовою, що обмежує її застосування у мовно-специфічних завданнях [53].

Обрана модель є попередньо навченою, тому для експериментів потрібні набори даних для додаткової обробки. Одним з підходів є доробка (fine-tuning) моделі за допомогою спеціалізованих наборів даних, що дозволяє підготувати модель для конкретної задачі.

3.4 Обробка даних для навчання

Підготовка даних оригінального текстового корпусу проводиться у два основні етапи: високорівневий та низькорівневий етапи очищення. Перший етап фокусується на обробці зібраних документів, другий етап – на індивідуальних реченнях.

На етапі високорівневого очищення відбувається початковий відбір серед вихідних документів. В рамках цього процесу здійснюються такі операції:

– виправлення некоректно сформованих документів за допомогою відповідних бібліотек. Ці бібліотеки дозволяють виправляти помилки Unicode у тексті;

– видалення документів, що не належать до мови, власне вчать модель за допомогою спеціалізованих бібліотек. Ця операція дозволяє не вносити несподівані навчальні дані написані іноземними мовами, у модель, коли більшість вибірки відповідає іншому розподілу;

– видалення коротких документів, що містять менше 128 токенів, оскільки такі документи часто містять контактну інформацію, авторські права,

посилання або різні текстові символи, які не приносять цінної інформації для попереднього навчання моделі (деанонімізація).

Другий етап – низькорівневе очищення – займається видаленням або коригуванням окремих речень. Для цього кожен документ спочатку розбивається на речення за допомогою відповідних бібліотек. Потім на кожне окреме речення застосовуються такі операції:

– видалення послідовностей більше трьох поспіль ідучих спеціальних символів із певного списку;

– видалення чисел або спеціальних символів, що з'являються на початку речення;

– видалення речень довжиною менше 2 слів та понад 200 слів.

Попередня обробка вихідного корпусу призводить до створення очищеного набору даних. Цей набір даних потім розбивається на тренувальні, валідаційні та тестові набори у певних пропорціях.

3.5 Налаштування мовної моделі на кінцеві завдання

Існують дві основні стратегії для застосування попередньо вивчених мовних уявлень: тонке налаштування та метод на основі функцій.

Метод тонкого налаштування мінімально залежить від параметрів конкретного завдання. У цьому підході додатково навчаємо всі попередньо навчені параметри кінцевої задачі. Цей процес дуже ефективний і практично зручний, оскільки не вимагає значної перебудови чи модифікації основної моделі архітектури [54]. У контексті цього підходу, попередньо навчена модель використовується для отримання уявлень слів, які надалі служать вхідними даними для інших архітектур, специфічних для кожної конкретної задачі [7].

Для методу на основі функцій потребується адаптація архітектури базової моделі, більш тонкого налаштуватися на кінцеве завдання. У разі такого застосування, модель розширюється за рахунок включення додаткових специфічних для завдання компонентів, які посилюють представлення та роблять їх більш пристосованою до кінцевого завдання [55].

Обидва підходи знайшли широке застосування в різних сферах NLP, але тонке налаштування стало особливо популярною завдяки своїй простоті та універсальності. За допомогою двох цілей попереднього навчання BERT можна застосовувати до будь-яких завдань, пов'язаних з одиночними послідовностями та парними послідовностями. Для цього достатньо підключити до BERT специфічні для завдання вхідні та вихідні дані та додатково налаштувати всі параметри після кожної епохи.

Один із прикладів того, як BERT обробляє пари речень у процесі попереднього навчання, це використання речень A і B. Це відповідає кільком завданням високого рівня, таким як перифразування, слідування, відповідь на запитання та класифікація тексту або тегування послідовностей. Зрештою, вихідні дані моделі можуть бути використані для різних завдань NLP, чи то на рівні токена, наприклад, у задачах тегування послідовності чи відповіді на запитання, чи на рівні тексту, наприклад, завдання класифікації тексту, логічного слідування чи аналізу тональності [56].

Порівнюючи bert-base-cased і BERT, обидві моделі є високоточними та можуть бути налаштовані для різних задач NLP. BERT відстає у продуктивності через свій більший розмір та кількість параметрів, проте може бути точнішим завдяки інтенсивнішому навчанню. Таким чином, для практичної реалізації дослідження через обмежені ресурси обрано bert-base-cased.

У порівнянні з іншими моделями, такими як GPT і Transformer-XL, BERT розроблено для кодування двонаправленого контексту, що означає, що вона враховує як попередні, так і наступні слова в реченні чи документі. Це

дозволяє BERT глибше розуміти контекст тексту. Однак, у порівнянні з GPT або Transformer-XL, BERT менш підходить для завдань, що вимагають генерації тексту, оскільки вона призначена для кодування послідовностей слів [7].

Оскільки BERT не спеціалізується на створенні тексту з нуля, а замість цього призначений для попереднього навчання на великих обсягах текстових даних та налаштування під конкретні завдання NLP, такі як класифікація тексту, відповіді на запитання та розпізнавання іменованих сутностей. Щодо генерації тексту, модель може використовуватися з певними підказками або початковими послідовностями слів, а потім доповнювати їх додатковим текстом.

Якість згенерованого тексту залежить від кількох факторів, включаючи розмір та якість навчальних даних, які використовуються для попереднього навчання моделі, конкретні підказки, передані моделі, і складність мови та синтаксису, що використовуються в створеному тексті. Незважаючи на здатність BERT генерувати текст, він не завжди може створювати логічні чи граматично правильні речення, і згенерований текст часто потребує значної після обробки, щоб стати придатним для використання.

3.6 Висновки до третього розділу

У цьому розділі досліджено процес адаптації мовної моделі BERT для тестування ІТ проєктів. Таким чином, було проведено аналіз ефективності різних способів подання вхідних даних моделі BERT. Виявлено, що правильне форматування та передача даних можуть суттєво вплинути на якість та точність результатів. Досліджено вплив різних параметрів моделі BERT на її ефективність у контексті тестування ІТ проєктів. Встановлено, що оптимізація

параметрів може покращити результати та швидкість обробки даних. Вивчено процес навчання моделі BERT та виявлені можливості для оптимізації цього процесу. Розглянуто методи підвищення швидкості навчання та покращення якості отриманих моделей. В результаті аналізу проведеної обробки даних для навчання моделі BERT, було визначено ключові аспекти обробки, що впливають на ефективність та надійність результатів. Розглянуто процес налаштування моделі BERT під конкретні потреби тестування IT проєктів, і як наслідок виявлено можливості для індивідуальної оптимізації моделі з урахуванням конкретних вимог та характеристик проєкту.

В результаті проведеного дослідження виявлено ряд способів оптимізації та покращення використання мовної моделі BERT для тестування IT проєктів. Отримані результати можуть бути використані для подальших досліджень у цій області, зокрема для розробки нових методів та алгоритмів, що сприятимуть покращенню ефективності тестування та розробки програмного забезпечення.

4 АПРОБАЦІЯ РЕЗУЛЬТАТІВ ДОСЛІДЖЕНЬ

4.1 Розробка плану проєкту, в рамках якого проходитиме практична апробація методу

Проєкт в рамках якого проходить практична апробація розроблюється для компанії «Віртуальна Франція».

Мета цього проєкту — надати відвідувачам Франції унікальний та інтерактивний спосіб вивчення країни, допомагаючи їм ознайомитися з культурним спадщиною, пам'ятками, історією та традиціями. Інтерактивний путівник буде реалізовано у вигляді веб-сайту, який запропонує користувачам персоналізовані маршрути, рекомендації та інформацію про різноманітні місця.

У рамках цієї задачі, тестування відіграє ключову роль у забезпеченні якості інтерактивного путівника. Тестувальники працюють над перевіркою функціональності веб-сайту, переконуючись в його надійності, продуктивності та безпеці. Вони активно спілкуються з розробниками та дизайнерами, щоб забезпечити якісний та інтуїтивно зрозумілий користувацький досвід. Таким чином, тестування відіграє важливу роль у успішній реалізації бізнес-ідеї, допомагаючи задовольнити потреби та очікування цільової аудиторії.

На проєкті використовується модель життєвого циклу розробки Agile, що дозволяє гнучко реагувати на зміни вимог та оперативно впроваджувати поліпшення. У рамках цієї моделі тестування є неперервним та вбудованим процесом, який розпочинається на ранніх етапах розробки і триває до моменту випуску продукту. Основна мета фази тестування - переконатися, що програмне забезпечення відповідає всім вимогам та стандартам якості перед його випуском на ринок.

Нижче наведено структуру робіт цієї фази:

– планування тестування: на цьому етапі команда зосереджується на визначенні цілей тестування, аналізі вимог та оцінюванні роботи;

– розробка тестів: тестувальники розробляють детальні сценарії тестів, і створюють потрібні тестові дані;

– виконання тестування: на цьому етапі, тестувальники проводять тестування відповідно до розроблених тестових сценаріїв, відстежуючи їх хід, а також запускають автоматизовані тести за наявності;

– аналіз результатів тестів: тестувальники перевіряють результати тестів, виявляють проблеми, аналізують причини їх виникнення;

– звітування: тестувальники узагальнюють результати тестування і передають їх зацікавленим сторонам у вигляді звітів. Це дозволяє забезпечити відповідність продукту вимогам та виявити доцільність подальшої діяльності з проєктом;

– виправлення дефектів та повторне тестування: розробники виправляють виявлені під час тестування дефекти, після чого тестувальники проводять тести для переконання, що дефекти зникли;

– регресійне тестування: це процес перевірки продукту після внесення змін. Це допомагає повторно провести тести та пересвідчитися, що внесені зміни не призвели до нових проблем;

– оцінка процесу тестування: після завершення процесу тестування команда здійснює аналіз своєї роботи, спілкується з розробниками щодо виявлених проблем, проводиться фінальне звітування процесу тестування.

На рисунку 4.1 наведено діаграму Ганта етапу тестування проєкту.

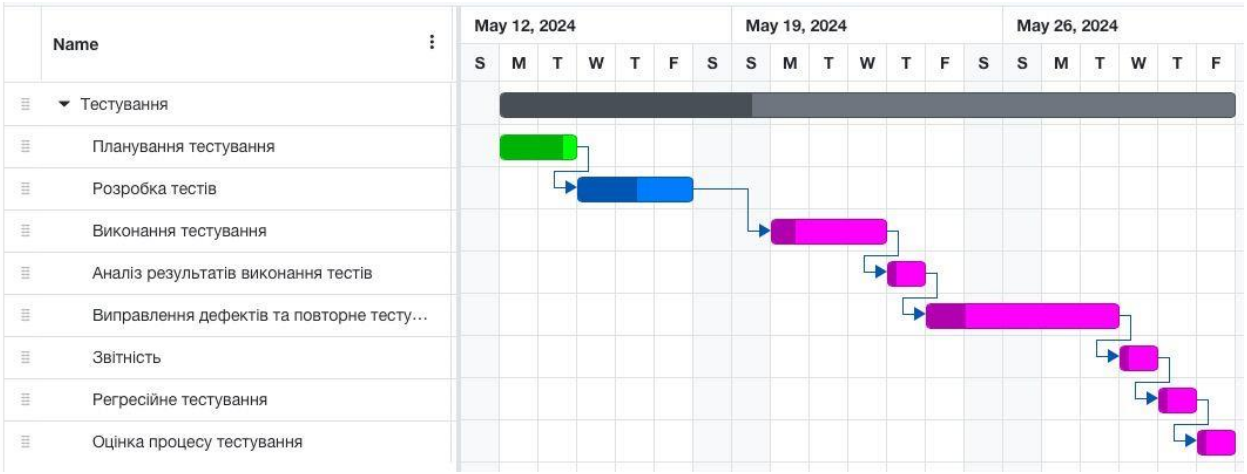

Рисунок 4.1 – Діаграма Ганта етапу тестування проєкту

4.2 Опис вхідних даних моделі

Для успішного використання мовної моделі важливо мати доступ до різноманітних наборів даних, які включають в себе інформацію про тестові сценарії, вимоги до програмного забезпечення, звіти про дефекти та іншу відповідну інформацію.

В якості наборів даних, які можна використовувати для цієї цілі, є:

– набір даних для тест-кейсів. Цей набір містить інформацію про різні тест-кейси, їх опис, передумови, кроки виконання та очікувані результати. Для навчання моделі це допоможе в розумінні структури та змісту тест-кейсів;

– набір даних звітів про дефекти. Цей набір містить інформацію про виявлені дефекти під час тестування, включаючи їх опис, кроки для відтворення, статус та іншу відповідну інформацію. Для моделі це допоможе в розпізнаванні та аналізі потенційних проблем у програмному забезпеченні;

– набір даних вимог до ПЗ. Цей набір містить вимоги до програмного забезпечення, включаючи функціональні та нефункціональні вимоги. Для

моделі це може служити для розуміння основних функцій та можливостей програмного забезпечення.

У використаному методі маскування, це може бути набір даних, де інформація про сценарії використання та вимоги до програмного забезпечення змішана з додатковими неважливими даними або шумом. Таке маскування допомагає оцінити ефективність моделі при роботі з реальними даними, які можуть бути неповними або неоднорідними.

Наведемо приклад вимог до функції рекомендацій віртуального турне у проєкті «Віртуальна Франція»:

– аналізувати інтереси користувачів до культурних відвідувань, природних місць та спортивних локацій на основі їх профілів;

– урахування активності користувачів у соціальних мережах для кращого розуміння їхніх переваг та рекомендацій, які найбільше відповідають їх інтересам;

– перевірка, чи рекомендації враховують години роботи пам'яток та об'єктів, доступних для віртуальних турів;

– забезпечення, що рекомендації віртуальних турів враховують особливі кліматичні умови та можливі обмеження доступу локацій.

База кейсу: користувач отримує рекомендацію віртуального турне, засновану на своїх інтересах, зазначених у їх профілі.

Вхідний текст: «Рекомендація віртуального турне для користувачів, зацікавлених у [MASK1]: перевірити що рекомендації враховують [MASK2].»

Застосовуючи маскування токену з BERT, ми можемо отримати готові тест кейси для перевірки:

– «Рекомендація віртуального турне для користувачів, зацікавлених у культурних відвідуваннях: перевірити що рекомендації враховують активність у соціальних мережах»;

– «Рекомендація віртуального турне для користувачів, зацікавлених у природних місця: перевірити що рекомендації враховують години роботи пам'яток»;

– «Рекомендація віртуального турне для користувачів, зацікавлених у спортивних місця: перевірити що рекомендації враховують особливості локацій за кліматичними умовами».

4.3 Виконання експерементів та оптимізація моделі BERT для тестування IT-проєкту

Для проведення експериментів у даному дослідженні використовується модель BERT як основна. Розглянемо підхід за використання маскованого токену. Передана підказка має вигляд: «The capital of France, [MASK] contains the Eiffel Tower.». У цьому прикладі, символ маски позначений словом «MASK» у дужках. Він може відрізнятись у різних моделях і повинен зчитуватись з відповідних налаштувань. Передаючи таку підказку до моделі та генеруючи 10 результатів з найбільшими ваговими коефіцієнтами, які відповідають за ймовірність використання певного токену, отримуємо наступний відсортований результат, починаючи з найбільш вагомого і закінчуючи найменш вагомим:

– The capital of France, Paris, contains the Eiffel Tower;
– The capital of France, Lyon, contains the Eiffel Tower;
– The capital of France, Strasbourg, contains the Eiffel Tower;
– The capital of France, Versailles, contains the Eiffel Tower;
– The capital of France, Toulouse, contains the Eiffel Tower;
– The capital of France, Brussels, contains the Eiffel Tower;
– The capital of France, Metz, contains the Eiffel Tower;

– The capital of France, Bordeaux, contains the Eiffel Tower;

– The capital of France, Lille, contains the Eiffel Tower;

– The capital of France, Orléans, contains the Eiffel Tower.

Розглянемо фрагмент початкового коду, який використовує моделі з бібліотеки transformers та використовує додаткові інструменти з torch (рис. 4.2).

Також проведемо оцінку моделі за допомогою методів, що були згадані під час теоретичних досліджень. Попередньо підготовлена модель залишиться незмінною – bert-base-cased, але для цього етапу буде використано інший набір даних – GLUE. Цей датасет складається з 9 частин, кожна з яких має своє призначення під час тестування моделі [57].

У зв'язку з обмеженими обчислювальними можливостями було вирішено здійснити тести на класифікацію текстів з використанням наступних частин набору даних: WNLI, RTE, MRPC.

Набір даних WNLI (Winograd NLI) базується на класичному тесті на розуміння природної мови Луїзи Віноград. Включає пари речень, анотовані з відповідями на запитання, чи можна однозначно визначити правильну відповідь з контексту. Це завдання вимагає розуміння контексту та семантики речень [57].

RTE (Recognizing Textual Entailment) складається з пар текстів, анотованих з вказівкою, чи один текст може бути виведений з іншого (припущення). Завдання полягає у визначенні, чи сенс одного речення випливає з іншого, або є ймовірним на основі контексту [57].

MRPC (Microsoft Research Paraphrase Corpus) містить пари речень, які були ретельно зібрані та анотовані як парафрази (речення зі схожим смислом, але різною формою). Завдання полягає у визначенні, чи є пара речень парафразами [57].

Результати представлені у табл. 4.1.

```
# Кількість результатів
limit = 10
# Підказка з місцем для згенерованого тексту на місці спеціального
# токену mask_token
text = "The capital of France, " + tokenizer.mask_token + ",
contains the Eiffel Tower."
input = tokenizer.encode_plus(text, return_tensors = "pt")
mask_index = torch.where(input["input_ids"][0] ==
tokenizer.mask_token_id)
output = model(**input)
logits = output.logits
softmax = F.softmax(logits, dim = -1)
mask_word = softmax[0, mask_index, :]
# Розрахунок найбільш вірогідних результатів
top = torch.topk(mask_word, limit, dim = 1)[1][0]
for token in top:
   word = tokenizer.decode([token])
   generated = text.replace(tokenizer.mask_token, word)
   print(generated)
```

Рисунок 4.2 – Приклад коду з використанням моделі з бібліотеки transformers та додаткових інструментів з torch

Таблиця 4.1 – Оцінка точності текстової класифікації

Фрагмент	Результати (точність, F1)	Витрачений час, хв
WNLI	65.1	0:34
RTE	66.43	2:06
MRPC	85.05, 89.54	01:47

Навчання виконується з використанням даних, організованих за допомогою ключа (seed), тому при використанні однакової версії бібліотеки PyTorch (2.0.0 на момент проведення дослідження), результати будуть однаковими. Однак складніше оцінити час навчання, оскільки він залежить від

графічного процесора і може значно відрізнятись як у гіршу, так і у кращу сторону. Вищезгадані результати було отримано з використанням 3060TI.

Також, як відзначають автори GLUE, для WNLI існує різниця в розподілі міток між навчальним та тестовим наборами даних, що може призвести до розбіжностей у оцінках для цієї категорії [57]. Поділ на навчальний та тестовий набори даних для WNLI є відповідним, але виявляється дещо суперечливим: коли два приклади містять однакове речення, це зазвичай означає, що вони мають протилежні мітки. Розподіл навчального та тестового наборів даних може містити спільні речення, тому якщо модель недоучена на навчальному наборі, це може призвести до погіршення результатів порівняно з випадковою точністю на WNLI у тестовому наборі даних. Крім того, тестовий набір має інший розподіл міток, ніж навчальний і валідаційний набори даних.

Для порівняння, наведемо приклад результатів для аналогічної моделі distilbert-base-uncased (відрізняється від bert-base-cased тим, що має менший розмір і, як виходить з назви, не чутлива до регістру тексту), які були взяті з таблиці результатів для оцінки за допомогою GLUE. Дані не містять інформації про витрачений час, але можна припустити, що ця модель має меншу кількість параметрів, тому ймовірно, що час навчання буде меншим, ніж в попередній таблиці.

Таблиця 4.2 – Точність моделі distilbert-base-uncased

Фрагмент	Результати (точність, F1)	Витрачений час, хв
WNLI	38.03	n/a
RTE	54.1	n/a
MRPC	87.6, 83.1	n/a

Під час процесу навчання було використано кілька методів для оптимізації моделі, одним з яких є метод деградації швидкості навчання. Цей підхід полягає у тренуванні мережі з високою швидкістю спочатку, а потім поступовому зменшенні швидкості навчання до досягнення локальних

мінімумів. Емпірично було підтверджено, що це сприяє оптимізації та узагальненню результатів.

На рис. 4.3 синьою лінією зображено процес навчання з постійною швидкістю. Кроки, які виконуються під час ітерацій до мінімумів, настільки шумні, що після деяких повторень видається, що модель блукає навколо мінімумів і, фактично, не збігається.

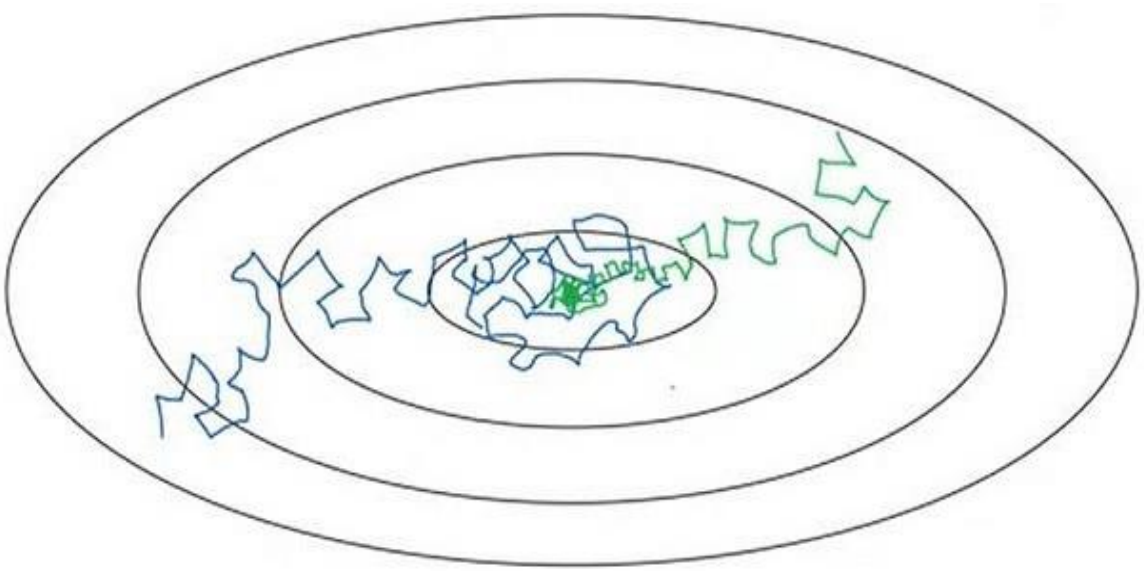

Рисунок 4.3 – Порівняння сталого (синій) та регресивного (зелений) темпу навчання

У той же час підхід зі зменшенням швидкості навчання (позначений зеленою лінією) в кінцевому підсумку коливається в більш тісному регіоні навколо мінімумів, не віддаляючись далеко від них. Початкова велика швидкість навчання забезпечує високу швидкість на початкових етапах, а зменшення швидкості навчання призводить до збільшення точності результатів, оскільки наближаємося до мінімальної швидкості навчання.

Ще одним важливим аспектом було корегування епохи моделі - гіперпараметра, який відповідає за кількість повторних тренувань. Щоб уникнути перенавчання мережі, важливо не встановлювати це значення

занадто великим. Таким чином, для фрагментів WNLI та RTE, які мають порівняно невеликий розмір, було встановлено значення 5, а для MPRC - 3.

Для порівняння результатів можна виконати інверсію параметру епохи під час навчання, щоб перевірити їх вплив на точність та час навчання моделі. Відповідно, для WNLI та RTE застосовувалося значення 3, а для більшого за розміром MPRC - 5. Отримані результати наведені у табл. 4.3.

Таблиця 4.3 – Оцінка точності з інвертованим параметром епохи

Фрагмент	Результати (точність, F1)	Витрачений час, хв
MRPC	85.05, 89.54	3:48
RTE	65.70	1:08
WNLI	33.8	0:20

Порівнюючи з попередніми даними (табл. 4.1), бачимо, що точність та F-міра для MRPC залишилися незмінними, але час навчання збільшився приблизно в 1,5 рази, що становить приблизно 2 хвилини. У випадку з меншими наборами WNLI та RTE, спостерігалася невелика знижка часу навчання, але також відбулося зменшення ефективності на 1% та 5% відповідно.

Такі значення пояснюються тим, що менші моделі мають менше параметрів, тому їм може знадобитися більше часу для тренування, щоб знайти оптимальне рішення. Крім того, менші моделі можуть бути більш схильними до перенавчання, і тривале тренування може допомогти зменшити цей ефект.

Ще одним важливим гіперпараметром є максимальна довжина послідовності навчання. Це обмеження встановлює розмір послідовностей, які модель може обробляти після токенізації. Для дослідження було обрано значення 128, оскільки очікувалося, що вхідні послідовності будуть короткими та нескладними. Важливо вибирати цей параметр обережно, оскільки він може впливати на здатність моделі фіксувати необхідну інформацію у вхідному тексті.

Обмеження кількості навчальних прикладів, які обробляються паралельно на пристрої під час тренування, є ще одним важливим гіперпараметром для оптимізації навчання на GPU. У поточному дослідженні було використано значення 32, але в разі нестачі ресурсів або помилок під час навчання це значення було зменшено до 16. Проведемо експериментальне порівняння цих двох значень у таблиці 4.4. Інші параметри, включаючи кількість епох, залишаються незмінними і відповідають значенням з таблиці 4.3 (5 для WNLI та RTE, 3 для MRPC).

Таблиця 4.4 – Порівняння швидкості навчання до максимальної кількості одночасно оброблюваних прикладів

Фрагмент	Попередні результати	Витрачений час, хв
MRPC	01:47	2:09
RTE	1:08	1:25
WNLI	0:20	0:35

Шляхом експерименту було підтверджено зниження швидкості навчання. Міри точності та F-міра не зазначені, оскільки зміни в параметрах не вплинули на них, тому дані з таблиці 4.1 залишаються актуальними для оновлених результатів.

Важливо враховувати, що кількість навчальних прикладів, оброблюваних паралельно, має відповідати максимальній довжині вхідної послідовності. Якщо послідовність має великий розмір, кількість паралельних прикладів має бути зменшена, щоб уникнути нестачі пам'яті на GPU.

Узагальнюючи, використання цих двох гіперпараметрів вимагає знаходження компромісу між швидкістю навчання та точністю результатів. Найкращим підходом є проведення експериментів з різними комбінаціями, щоб максимально використовувати ресурси відеокарти.

4.4 Висновки до четвертого розділу

У підрозділі 4.1 описано проєкт, в рамках якого проходить практична апробація моделі. Наведено структуру робіт фази тестування проєкту.

У підрозділі 4.2 описані вхідні дані для моделі, яка використовується для генерації тест-кейсів за допомогою моделі. В якості наборів даних використовуються тест-кейси, чек-лісти та вимоги до програмного забезпечення. Модель BERT є ефективним інструментом для роботи з цими даними, оскільки вона може аналізувати та розуміти семантику тексту, використовуючи різноманітну інформацію з наборів даних та структуру тест-кейсів.

У підрозділі 4.3 проведено експерименти з моделлю BERT та оптимізацією гіперпараметрів під час навчання моделі. Належні налаштування мовної моделі є одним з ключових чинників для досягнення високої ефективності при проведенні тестування в IT-проєктах, що дозволяє скоротити час навчання та покращити точність прогнозів моделі. Отримані результати дали можливість порівняти ефективність моделі в залежності від обраних гіперпараметрів і налаштувань.

ВИСНОВКИ

У результаті проведеного дослідження за темою «Дослідження використання мовних моделей для тестування ІТ-проєктів», були отримані наукові та практичні результати:

– проаналізовано сучасні мовні моделі та виявлено їх переваги та обмеження у контексті тестування програмного забезпечення;

– досліджено декілька видів мовних моделей, у результаті чого було визначено, що модель BERT є особливо перспективною для застосування в тестуванні ІТ-проєктів, завдяки своїй здатності до контекстуального аналізу тексту та широкому функціоналу;

– проаналізовано процес адаптації мовної моделі BERT для тестування ІТ-проєктів і виявлено способи оптимізації параметрів та процесу навчання моделі для досягнення кращих результатів та швидкості обробки даних;

– здійснено практичну апробацію моделі BERT на задачі ІТ-проєкту, проведено експерименти з оптимізацією гіперпараметрів під час навчання моделі і порівняно ефективність моделі в залежності від обраних гіперпараметрів і налаштувань.

Використання мовних моделей для тестування ІТ-проєктів має високу актуальність і може принести суттєві покращення в процеси тестування програмного забезпечення. Застосування таких моделей дозволяє автоматизувати значну частину завдань тестування, підвищити ефективність та зменшити кількість помилок у продуктах. Однак, для досягнення оптимальних результатів на практиці, необхідно продовжувати дослідження в галузі адаптації мовних моделей для тестування ІТ-проєктів, розробляти нові методи, що враховують специфіку та вимоги конкретних програмних продуктів.

Результати, отримані під час дослідження, було опубліковано у [62-63].

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. ДСТУ 3008:2015. Інформація та документація. Звіти у сфері науки і техніки. Структура і правила оформлювання. Чинний від 22.06.2015. Київ: ДП «УкрНДНЦ», 2016. 31 с.

2. ДСТУ 8302:2015. Інформація та документація. Бібліографічні посилання. Загальні положення та правила складання. Чинний від 04.03.2016. Київ: ДП «УкрНДНЦ», 2016. 20 с.

3. Методичні вказівки щодо розробки та оформлення кваліфікаційної роботи другого (магістерського) рівня вищої освіти за освітньо-науковою програмою «Управління проєктами в галузі інформаційних технологій» / Упоряд.: Петров К.Е., Левикін В.М., Чалий С.Ф., Євланов М.В., Міхнов Д.К., Міхнова А.В., Чала О.В. – Харків: ХНУРЕ, 2024. – 24 с.

4. IEEE Standard for Software and System Test Documentation. URL: https://standards.ieee.org/ieee/829/3787/ (дата звернення 29.04.2024).

5. ISO/IEC/IEEE 12207:2017. URL: https://www.iso.org/standard/43447.html (дата звернення 29.04.2024).

6. ISO/IEC/IEEE 15288:2023. URL: https://www.iso.org/standard/63711.html (дата звернення 29.04.2024).

7. Devlin J., Chang M. W., Lee K., Toutanova K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding //arXiv preprint arXiv:1810.04805. – 2018.

8. Yang, Z., Dai, Z., Yang, Y., Carbonell, J.G., Salakhutdinov, R., Le, Q.V. XLNet: Generalized Autoregressive Pretraining for Language Understanding. //arXiv preprint arXiv:1906.08237– 2019.

9. Liu Y., Ott M., Goyal N., Du J., Joshi M., Chen D., Levy O., Lewis M., Zettlemoyer L., Stoyanov V. RoBERTa: A Robustly Optimized BERT Pretraining Approach //arXiv preprint arXiv:1907.11692. – 2019.

10. Lan Z., Chen M., Goodman S., Gimpel K., Sharma P., Soricut R. ALBERT: A Lite BERT for Self-supervised Learning of Language Representations //arXiv preprint arXiv:1909.11942. – 2019.

11. Wang Y., Liu K., Liu J., Lyu Y., Xu Z., Wang S., Chen Y., Li W., Zhou S. StructBERT: Incorporating Language Structures into Pre-training for Deep Language Understanding //arXiv preprint arXiv:1908.04577. – 2019.

12. Raffel C., Shazeer N., Roberts A., Lee K., Narang S., Matena M., Zhou Y., Li W., Liu P. J. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer //arXiv preprint arXiv:1910.10683. – 2019.

13. Brown T. B., Mann B., Ryder N., Subbiah M., Kaplan J., Dhariwal P., Neelakantan A., Shyam P., Sastry G., Askell A., et al. Language Models are Few-Shot Learners //arXiv preprint arXiv:2005.14165. – 2020.

14. Clark K., Khandelwal U., Levy O., Manning C. D. ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators // arXiv preprint arXiv:2003.10555. – 2020.

15. He P., Liu X., Chen W., Gao J. DeBERTa: Decoding-enhanced BERT with Disentangled Attention //arXiv preprint arXiv:2006.03654. – 2020.

16. Harman M., McMinn P. A theoretical and empirical study of search-based testing: Local, global, and hybrid search //IEEE Transactions on Software Engineering. – 2009. – Т. 36. – №. 2. – С. 226-247.

17. Delgado-Pérez P. et al. INTEREVO-TR: Interactive Evolutionary Test Generation with Readability Assessment //IEEE Transactions on Software Engineering. – 2022.

18. Xiao X. et al. Characteristic studies of loop problems for structural test generation via symbolic execution //2013 28th IEEE/ACM International Conference on Automated Software Engineering (ASE). – IEEE, 2013. – С. 246-256.

19. Pacheco C. et al. Feedback-directed random test generation //29th International Conference on Software Engineering (ICSE'07). – IEEE, 2007. – С. 75-84.

20. Yuan Z. et al. No More Manual Tests? Evaluating and Improving ChatGPT for Unit Test Generation //arXiv preprint arXiv:2305.04207. – 2023.

21. Tang Y. et al. Chatgpt vs sbst: A comparative assessment of unit test suite generation //arXiv preprint arXiv:2307.00588. – 2023.

22. Pan M. et al. Reinforcement learning based curiosity-driven testing of Android applications //Proceedings of the 29th ACM SIGSOFT International Symposium on Software Testing and Analysis. – 2020. – С. 153-164.

23. Liu Z. et al. Make LLM a Testing Expert: Bringing Human-like Interaction to Mobile GUI Testing via Functionality-aware Decisions //arXiv preprint arXiv:2310.15780. – 2023.

24. Su T., Wang J., Su Z. Benchmarking automated gui testing for android against real-world bugs //Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering. – 2021. – С. 119-130.

25. Shanahan M. Talking about large language models //Communications of the ACM. – 2024. – Т. 67. – №. 2. – С. 68-79.

26. Zhao W. X. et al. A survey of large language models //arXiv preprint arXiv:2303.18223. – 2023.

27. Kojima T. et al. Large language models are zero-shot reasoners //Advances in neural information processing systems. – 2022. – Т. 35. – С. 22199-22213.

28. Wei J. et al. Chain-of-thought prompting elicits reasoning in large language models //Advances in Neural Information Processing Systems. – 2022. – Т. 35. – С. 24824-24837.

29. Li J. et al. Structured chain-of-thought prompting for code generation //arXiv preprint arXiv:2305.06599. – 2023.

30. Li J. et al. Skcoder: A sketch-based approach for automatic code generation //arXiv preprint arXiv:2302.06144. – 2023.

31. Li J. et al. Acecoder: Utilizing existing code to enhance code generation //arXiv preprint arXiv:2303.17780. – 2023.

32. Dong Y. et al. Self-collaboration Code Generation via ChatGPT //arXiv preprint arXiv:2304.07590. – 2023.

33. Deep Learning, NLP, and Representations. URL: https://colah.github.io/posts/2014-07-NLP-RNNs-Representations/ (дата звернення 29.02.2024).

34. Harris Z. S. Distributional structure //Word. – 1954. – Т. 10. – №. 2-3. – С. 146-162.

35. Firth J. A synopsis of linguistic theory, 1930-1955 //Studies in linguistic analysis. – 1957. – С. 10-32.

36. Landauer T. K., Dumais S. T. A solution to Plato's problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge //Psychological review. – 1997. – Т. 104. – №. 2. – С. 211.

37. Lund K., Burgess C. Producing high-dimensional semantic spaces from lexical co-occurrence //Behavior research methods, instruments, & computers. – 1996. – Т. 28. – №. 2. – С. 203-208.

38. Rohde D. L. T., Gonnerman L. M., Plaut D. C. An improved method for deriving word meaning from lexical co-occurrence //Cognitive Psychology. – 2004. – Т. 7. – С. 573-605.

39. Lebret R., Collobert R. Word emdeddings through hellinger PCA //arXiv preprint arXiv:1312.5542. – 2013.

40. Chen S. F., Goodman J. An empirical study of smoothing techniques for language modeling //Computer Speech & Language. – 1999. – Т. 13. – №. 4. – С. 359-394.

41. Bengio Y., Ducharme R., Vincent P. A neural probabilistic language model //Advances in neural information processing systems. – 2000. – Т. 13.

42. Mikolov T. Efficient estimation of word representations in vector space //arXiv preprint arXiv:1301.3781. – 2013.

43. Mikolov T. Distributed representations of words and phrases and their compositionality //Advances in neural information processing systems. – 2013. – T. 26.

44. Joulin A. Bag of tricks for efficient text classification //arXiv preprint arXiv:1607.01759. – 2016.

45. McCann B. Learned in translation: Contextualized word vectors //Advances in neural information processing systems. – 2017. – T. 30.

46. Sarzynska-Wawer J. et al. Detecting formal thought disorder by deep contextualized word representations //Psychiatry Research. – 2021. – T. 304. – C. 114135.

47. Howard J., Ruder S. Universal language model fine-tuning for text classification //arXiv preprint arXiv:1801.06146. – 2018.

48. Vaswani A. et al. Attention is all you need //Advances in neural information processing systems. – 2017. – T. 30.

49. Bahdanau D., Cho K., Bengio Y. Neural machine translation by jointly learning to align and translate //arXiv preprint arXiv:1409.0473. – 2014.

50. Radford A., Wu J., Child R., Luan D., Amodei D., Sutskever I. Language Models are Unsupervised Multitask Learners //arXiv preprint arXiv:1911.00172. – 2019.

51. Mask and Cloze: Automatic Open Cloze Question Generation using a Masked Language Model / S. Matsumori та ін. IEEE Access. 2023. С. 1. URL: https://doi.org/10.1109/access.2023.3239005 (дата звернення: 19.05.2024).

52. Pennington J., Socher R., Manning C. D. GloVe: Global Vectors for Word Representation //Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP). – 2014.

53. GitHub - soskek/bookcorpus: Crawl BookCorpus. GitHub. URL: https://github.com/soskek/bookcorpus (дата звернення: 19.05.2024).

54. Howard J., Ruder S. Universal Language Model Fine-tuning for Text Classification //arXiv preprint arXiv:1801.06146. – 2018.

55. Lample G., Ballesteros M., Subramanian S., Kawakami K., Dyer C. Neural Architectures for Named Entity Recognition //arXiv preprint arXiv:1603.01360. – 2016.

56. Diao Z., Sharma H., Yang R., Xu J. Efficient Fine-Tuning of BERT Models on the Edge //arXiv preprint arXiv:2104.06022. – 2021.

57. Wang A., Singh A., Michael J., Hill F., Levy O., Bowman S. R. GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding //arXiv preprint arXiv:1804.07461. – 2018.

58. Singh S. BERT Explained: State-of-the-art language model for NLP. Labellerr. URL: https://www.labellerr.com/blog/bert-explained-state-of-the-art-language-model-for-nlp/ (дата звернення: 11.05.2024).

59. Shivanandhan M. Understanding BERT: The Key to Advanced Language Models. LinkedIn: Log In or Sign Up. URL: https://www.linkedin.com/pulse/understanding-bert-key-advanced-language-models-m-shivanandhan-f6jtc/ (дата звернення: 11.05.2024).

60. Alammar J. The Illustrated Transformer. Visualizing machine learning one concept at a time. URL: https://jalammar.github.io/illustrated-transformer/ (дата звернення: 10.05.2024).

61. Md Fahim. BERT - In Depth Understanding. Kaggle: Your Machine Learning and Data Science Community. URL: https://www.kaggle.com/code/mdfahimreshm/bert-in-depth-understanding (дата звернення: 09.05.2024).

62. Левикін В.М., Діденко Д.О., Альошкін О.А. Метод формування заявок природною мовою на основі вдосконаленої моделі BERT. АСУ та прилади автоматики, 2024, Вип. 180. С. 55-71.

63. Альошкін О.А. Дослідження моделей використання засобівмашинного навчання для генерації тест-кейсів у ІТ-проєктах. 27-й Міжнародний молодіжний форум «Радіоелектроніка та молодь у XXI столітті». 2023. Т. 6. С. 224–225.