Міністерство освіти і науки Україна
Харківський національний університет радіоелектроніки

Факультет комп'ютерних наук
(повна назва)

Кафедра програмної інжеренії
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

Дослідження методів виправлення помилок в текстах
(тема)

Виконав:
студент(ка) _2_ курсу, групи ІПЗм-22-5

Чипіженко О. О.
(призвіще, ініціали)

Тип програми освітньо-науко

Керівник Турута О. П.
(посада, призвіще, ініціали)

Допускається до захисту
Зав. Кафедри _____ З. В. Дудар
(підпис) (призвіще, ініціали)

2024 р.

Харківський національний університет радіоелектроніки

Факультет комп'ютерних наук (або центр післядипломної освіти, або навчально-науковий центр заочної форми навчання)
Кафедра програмної інженерії
Рівень вищої освіти другий (магістерський)
Спеціальність 121 – Інженерія програмного забезпечення
Тип програми освітньо-наукова програма
Освітня програма Інженерія програмного забезпечення
(шифр і назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«___» _____ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Чипіженку Олексію Олександровичу
(прізвище, ім'я, по батькові)

1. Тема роботи «Дослідження методів виправлення помилок в текстах»
Затверджена наказом по університету від 29.03. 2024р. № 250 Ст
2. Термін подання студентом роботи до екзаменаційної комісії 20.06.2022
3. Вихідні дані до роботи досліджуваних методів GEC, мова програмування C#, Entity Framework, середовище розробки Visual Studio
4. Перелік питань, що потрібно опрацювати в роботі
мета роботи, аналіз предметної галузі і постановка задачі, огляд та аналіз літературних джерел з дослідження, практичне дослідження предметної області

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Аналіз предметної галузі та постановка задачі	23.01 – 14.02.24	*виконано*
2	Аналіз та вибір API для дослідження	15.02 – 24.02.24	*виконано*
3	Аналіз та моделювання предметної області	17.02 – 28.02.24	*виконано*
4	Планування експериментів	25.02 – 28.02.24	*виконано*
5	Програмна реалізація кожного з обраних для дослідження API	25.02 – 01.04.24	*виконано*
6	Експериментальні дослідження	02.04 – 20.04.24	*виконано*
7	Аналіз результатів експериментальних досліджень та розробка рекомендацій	20.04 – 23.04.24	*виконано*
8	Підготовка пояснювальної записки	01.04 – 26.04.24	*виконано*
9	Підготовка презентації та доповіді	26.05.04 – 14.06.24	*виконано*
10	Нормоконтроль	15.06.24	*виконано*
11	Рецензування	16.06.24	*виконано*
12	Занесення диплома в електронний архів	17.06.24	*виконано*
13	Попередній захист	17.06.24	*виконано*
14	Допуск до захисту у зав. кафедри	18.06.24	*виконано*

Дата видачі завдання 20 січня 2023р.

Студент _____ Чипіженко О. О
(підпис)

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

РЕФЕРАТ / ABSTRACT

Звіт з практики містить 48 стор., 8 рис., 4 табл., 8 джерел.

АВТОМАТИЗАЦІЯ, ВИПРАВЛЕННЯ ПОМИЛОК, ЕФЕКТИВНІСТЬ, МАШИННЕ НАВЧАННЯ, МОДЕЛІ, ОПТИМІЗАЦІЯ ПРОЦЕСУ, ЯКІСТЬ ТЕКСТУ.

Об’єкт дослідження – методи та стратегії виправлення помилок у текстах, що включають автоматизовані програмні засоби, правописні перевірки, контекстний аналіз та інші підходи, спрямовані на покращення якості та точності інформації в письмових матеріалах.

Мета роботи – аналіз та порівняння методів виправлення помилок в текстах з метою визначення їх ефективності, переваг та недоліків. Метод рішення – проведення комплексного аналізу доступних методів виправлення помилок у текстах, включаючи, автоматизовані програмні засоби, правописні перевірки, аналіз контексту та інші підходи. На основі отриманих результатів обирається найбільш оптимальний метод чи їх комбінація для виправлення помилок у текстах.

Результат роботи – завершений аналіз, що буде відображати ключові висновки щодо ефективності різних методів виправлення помилок у текстах.

ERROR CORRECTION, AUTOMATION, MODELS, TEXT QUALITY, PROCESS OPTIMIZATION, EFFICIENCY, MACHINE LEARNING

The object of research is methods and strategies for correcting errors in texts, including automated software tools, spell checks, contextual analysis, and other

approaches aimed at improving the quality and accuracy of information in written materials.

The purpose of the work is to analyze and compare methods of correcting errors in texts in order to determine their effectiveness, advantages and disadvantages. The

solution method is to carry out a comprehensive analysis of available methods of correcting errors in texts, including automated software tools, spell checkers, context analysis and other approaches. Based on the obtained results, the most optimal method or their combination is chosen for correcting errors in the texts. Output – The completed analysis will reflect the key performance findings for the various error correction methods in texts.

Я, Чипіженко Олексій Олександрович, студент гр. ІПЗм-22-5, здобувач вищої освіти на другому (магістерському) рівні кафедри «Програмна інженерія», заявляю: Моя кваліфікаційна робота , виконана самостійно, в ній не містяться елементи плагіату і вона може бути опублікована в електронному архіві відкритого доступу ElAr KhNURE. Усі запозичення з друкованих та електронних джерел мають відповідні посилання.

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

ЗМІСТ

Вступ.....7
1 Аналіз предметної області.....8
1.1 Загальні відомості про Natural Language Processing.....8
1.2 Основні етапи Natural Language Processing.....8
1.3 Проблематика яку опрацьовує Natural Language Processing.....13
1.4 Grammatical error correction завдання та підходи його вирішення.....14
1.5 Основні характеристики та різновиди GEC.....15
1.6 Постановка задачі 16
2 Аналіз існуючих методів.....18
2.1 Загальний аналіз.....18
2.2 Методи вирішення задач GEC……18
2.3 Множина GEC моделей та їх аналіз.....22
2.3.1 Множина моделей.....22
2.3.2 Опис множини критеріїв та їх характеристика.....26
2.3.3 Опис та аналіз шкал критеріїв.....27
3 Розробка та порівняння розроблених GEC систем……..30
3.1 Опис обраних для розробки технологій……30
3.2 Створення GEC системи з використанням методу правил (ALBERT)……30
3.3 Створення GEC системи з використанням методу «segment-level recurrence» (метод Grammarly)……32
3.4 Порівняння систем……35
Висновок.....37
Перелік джерел посилання.....38
Додаток А.....39
Додаток Б……..40
Додаток В…….48

ВСТУП

У сучасному інформаційному суспільстві важливість якості письмового матеріалу не може бути переоцінена. Повідомлення в чаті, переписка на роботі, написання публікації або написання твору – це те з чим кожен день стикаються більша частина людей. Та навіть найбільш досконалий текст може містити помилки, які впливають на його читабельність та точність, а отже можемо помітити зростання актуальності вирішення питання поліпшеня правопису, а разом с тим помітити високий розвиток сфери обробки природної мови. Саме тому проблема виправлення помилок у текстах є актуальною та потребує уваги.

Ця робота присвячена аналізу та огляду різноманітних методів виправлення помилок у текстах. Метою її є систематичний розгляд доступних підходів та їх порівняльний аналіз з урахуванням їхньої ефективності, швидкості та можливостей застосування в різних сферах.

Дослідження методів виправлення помилок має велике значення для покращення якості текстів у різних контекстах: від наукових публікацій до повсякденного комунікаційного середовища. Ця робота прагне висвітлити різноманітні підходи до виправлення помилок та визначити оптимальні стратегії для досягнення максимальної точності та якості тексту.

Було проведено аналіз різних існуючих систем виправлення помилок у текстах української мови та методів реалізації їх GEC, що надало можливість зробити висновки щодо їхньої ефективності та застосовності у практичних умовах. Окремо було оглянуто доступні GEC датасети та проведено порівняння з іншими існуючими доступними даними, що допоможе визначити шляхи покращення сьогоденних систем виправлення помилок в текстах. Рекомендації, розроблені на основі цього дослідження, можуть стати важливим кроком у поліпшенні якості письмового матеріалу та оптимізації процесу виправлення текстових помилок.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Загальні відомості про Natural Language Processing

Опрацювання природної мови (Воно ж Natural Language Processing) є галуззю штучного інтелекту, яка вивчає комп’ютерні методи та алгоритми для розуміння, аналізу, генерації та взаємодії з людською мовою. Вона включає в себе різні завдання, такі як розпізнавання мови (голосове та текстове), синтез мови (генерація мовлення), машинний переклад, виявлення та корекцію помилок у текстах, аналіз настроїв та емоцій у текстах, взаємодію з користувачами через мовлення та багато іншого.

Головна мета опрацювання природної мови полягає в створенні інтелектуальних систем, які можуть розуміти отримувану текстову або голосову інформацію, виражену людською літературною чи розмовною мовою, та ефективно взаємодіяти з користувачами на цьому рівні. Це розвивається через використання методів машинного навчання, обробки природної мови (NLP), обробки сигналів, статистичних моделей та інших технік, які дозволяють комп’ютерам працювати з мовою як зі складною системою правил та структур.

1.2 Основні етапи Natural Language Processing

Розглянемо фундаментальний процес обробки природної мови (Natural Language Processing, NLP), що є ключовим для розуміння та використання сучасних технологій в цій галузі. Відповідно до актуальних вимог та вибіркової спрямованості наукового аналізу, увага акцентується на важливих етапах NLP, що сприяють зростанню рівня розуміння та ефективності обробки мови комп’ютерними системами.

Розібравши ці процеси конвертації текстових даних у структуровану форму, аналізу синтаксичних та семантичних аспектів мовлення, виявлення іменованих сутностей та їх взаємозв’язків, а також генерації мовлення та машинного перекладу дозволить нам краще зрозуміти технічні аспекти NLP та посприяє подальшій спробі вдосконалення технологій у цій галузі.

Як вже було сказано обробка природної мови (NLP) – це галузь штучного

інтелекту, що вивчає методи та технології для розуміння та обробки людської мови комп’ютерними системами. Основні етапи NLP включають:

– передпроцесинг (попереднє оброблення);
– аналіз;
– витягнення інформації;
– машинний переклад та генерація мови;
– взаємодія з користувачем.

Ці етапи використовуються для розвитку різноманітних застосувань NLP, таких як пошукові системи, чат-боти, системи автоматичної обробки текстів та інші. Проаналізуємо кожен з етапів.

Передпроцесинг представляє з себе опрацювання і підготовку даних у відповідну форму для подальношого аналізу. Зазвичай постпроцесинг відбувається у декілька кроків – сегментація, стемінг, нормалізації та токенізації (див.рис.1.1).

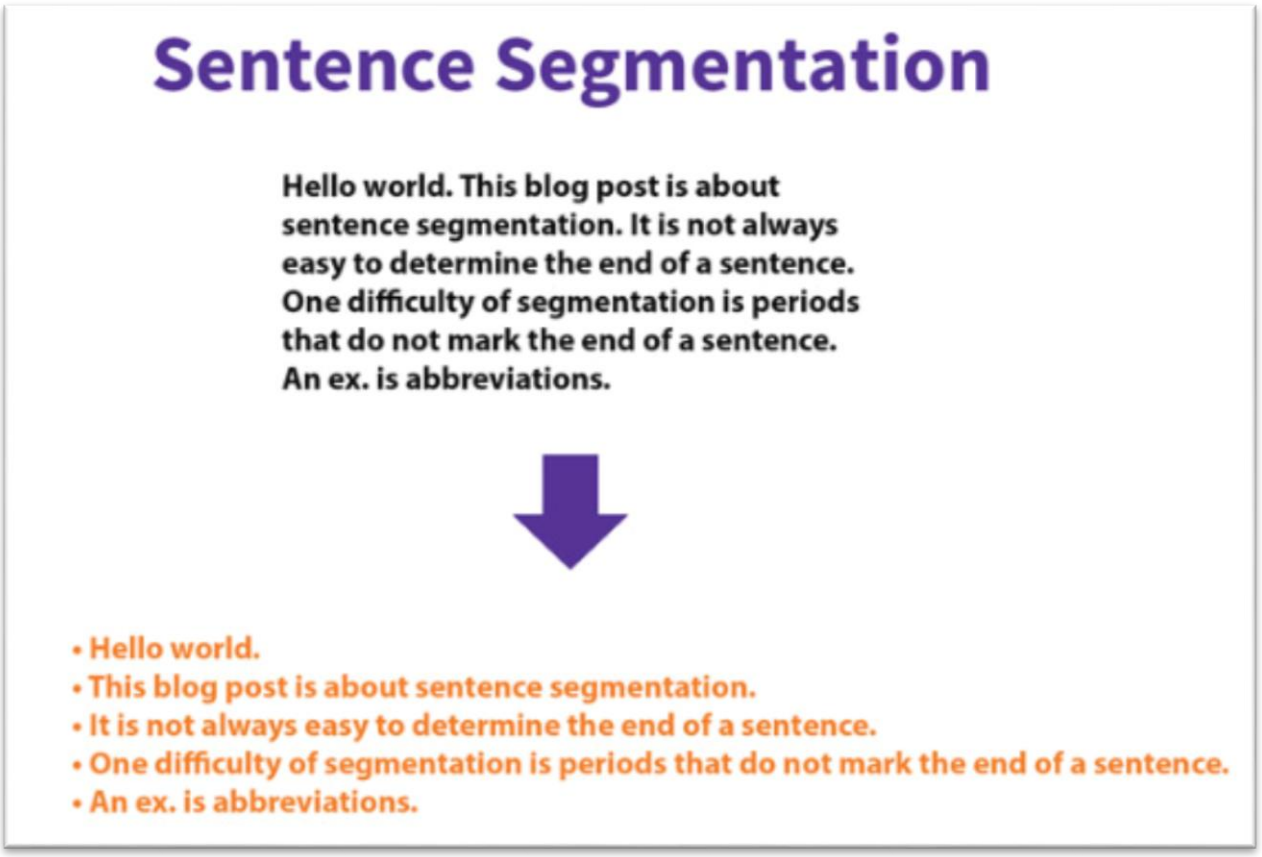

Рисунок 1.1 – Приклад сегментації тексту

При сегментації тексту він розбивається на окремі речення (часто це робиться за допомогою знаходження розділових знаків але є і більш точніші методи, що допомагає працювати навіть з не чітко відформатованомим текстом). [1]

Стемінг є одним із ключових понять у обробці природної мови (Natural Language Processing, NLP). Це процес нормалізації тексту, який спрощує слова до їх базової форми або «стему». Стем – це частина слова, яка залишається після відкидання закінчень та афіксів. Основна мета стемінгу полягає в тому, щоб різні форми одного слова (наприклад, прислівника, дієслова в минулому часі, іменника в різних відмінках) зведені до їхньої базової форми, щоб уникнути дублювання та забезпечити більш точний аналіз тексту.

У стемінгу використовуються правила та алгоритми, щоб визначити стем кожного слова. Один і той же стем може мати кілька варіантів написання, які зводяться до одного стема для подальшої обробки. Наприклад, слова «бігати», «бігав», «бігла» будуть зведені до стема «біг».

Основні алгоритми стемінгу включають в себе алгоритм Портера, Ловінгера, Сноу-Болла та інші. Кожен з цих алгоритмів має свої особливості та правила для визначення стемів слів.

Важливо зазначити, що стемінг не завжди дає точний результат, оскільки може призводити до неправильного зведення деяких слів до одного стема, що може впливати на якість обробки тексту та аналізу даних. Також варто враховувати, що стемінг використовується лише для зведення слів до базової форми і не враховує контексту або значення слів у тексті.

Наступним кроком проводиться токенізація [2]. Токенізація є важливим етапом в обробці природної мови і полягає в розділенні тексту на окремі слова або ж токени. Основна мета токенізації полягає в тому, щоб розбити текст на найменші лексичні одиниці, які можна обробляти окремо. Токенами можуть бути слова, числа, символи пунктуації, фрази тощо.

Після процесу токенізації для кожного токену призначаться маркер який показує призначення токену до частин мови (рис. 1.2-1.3), що допомагає

визначити суть речення та допоможе у подальшому аналізі тексту, такого як синтаксичний аналіз, отримати його семантичне розуміння та провести статистичну обробка текст. Цей процес дозволяє ефективно взаємодіяти з текстовими даними та робити їх доступними для обробки комп'ютерними системами в рамках NLP.

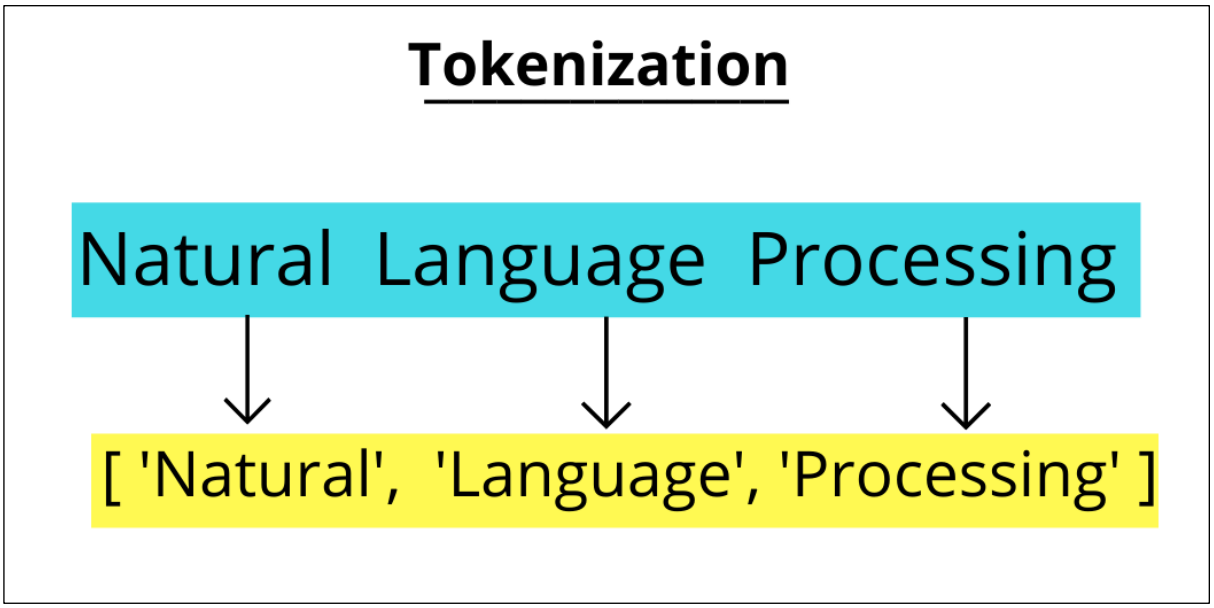

Рисунок 1.2 – Токенізація речення

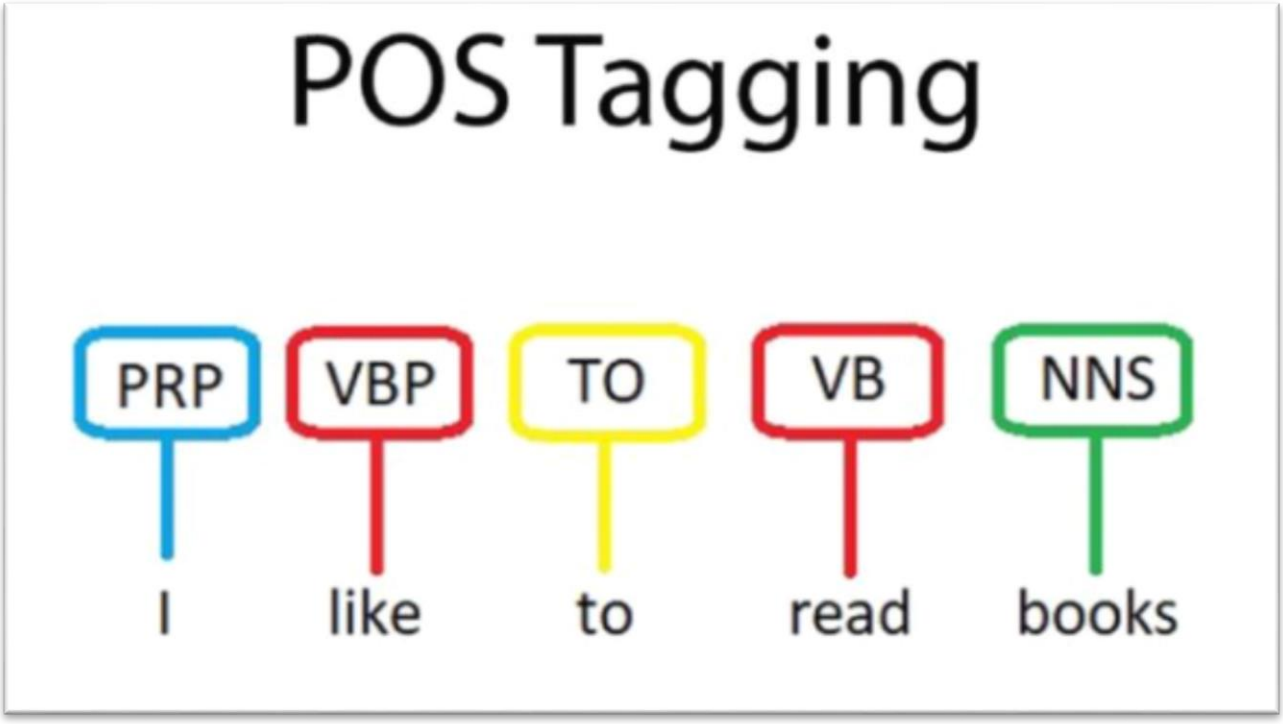

Рисунок 1.3 – Розподілення по частинам мови

Нормалізація [3] в контексті обробки природної мови – це процес перетворення тексту до стандартизованої або однорідної форми, що допомагає полегшити подальший аналіз та обробку даних комп'ютерними системами. Основна мета нормалізації полягає в усуненні різниці у форматі, структурі та представленні текстової інформації для підвищення якості обробки та аналізу мовлення.

До основних видів нормалізації в NLP входять:

- нижній регістр (Lowercasing): Перетворення всіх літер тексту у нижній регістр. Це допомагає уникнути дублювання та розбіжностей через різний регістр літер у словах;
- видалення пунктуації (Punctuation removal): Усунення розділових знаків, символів пунктуації та спеціальних символів із тексту. Це допомагає покращити точність та зрозумілість тексту для подальшого аналізу;
- видалення зайвих пробілів (Whitespace trimming): Позбавлення тексту зайвих пробілів та пробілів у початку та в кінці рядка. Це робить текст більш чистим та зрозумілим для обробки;
- усунення стоп-слів (Stopwords removal): Вилучення загальних та неінформативних слів (стоп-слів), таких як «і», «а», «на», «до» і т. Д., які не несуть значення для аналізу тексту (див.рис.1ю4).

Sample text with Stop Words	Without Stop Words
GeeksforGeeks – A Computer Science Portal for Geeks	GeeksforGeeks , Computer Science, Portal ,Geeks
Can listening be exhausting?	Listening, Exhausting
I like reading, so I read	Like, Reading, read

Рисунок 1.4 – Усунення стоп-слів

Нормалізація тексту в NLP є важливим етапом перед подальшим аналізом, оскільки допомагає зменшити складність та розмір текстових даних, полегшує порівняння та виявлення паттернів у тексті, а також забезпечує більш точні та надійні результати обробки.

1.3 Проблематика яку опрацьовує Natural Language Processing

Natural Language Processing опрацьовує широкий спектр проблематики, пов’язаної з аналізом та розумінням людської мови за допомогою комп’ютерних систем. Приклад проблематик які вирішує о бробка природної мови:

- машинне перекладання: Дозволяє автоматично перекладати тексти з однієї мови на іншу, що сприяє міжнародному спілкуванню, бізнесу та культурній обміні;
- розпізнавання мовлення: За допомогою NLP можна розпізнавати та транскрибувати голосові команди та мовлення людей, що відкриває широкі можливості для інтерфейсів з мовленням, розумних асистентів та систем голосового управління;
- класифікація текстів: Дозволяє класифікувати текстові дані за темами, емоціями, важливістю та іншими параметрами. Це корисно для автоматичної обробки повідомлень, сортування документів, аналізу настроїв та іншого;
- генерація текстів: Може бути використаний для створення нового текстового контенту, такого як новини, описи, рецензії та інше. Це корисно для автоматичного наповнення веб-сайтів, соціальних медіа та інших платформ;
- корекція текстів та редагування: Допомагає виявляти та виправляти граматичні та стилістичні помилки у текстах, що полегшує створення якісного та зрозумілого контенту;
- аналіз відгуків та коментарів: Використовується для аналізу великих обсягів текстових даних, таких як відгуки, коментарі, соціальні мережі тощо, для виявлення тенденцій, відношень та сентименту групи людей;
- автоматизація процесів: Може автоматизувати багато рутинних завдань, пов’язаних з обробкою текстів, що допомагає підвищити ефективність та швидкість роботи.

Natural Language Processing має великий практичний потенціал і

використовується в різних галузях, включаючи інформаційні технології, медіа, маркетинг, науку про дані, медицину, освіту, повсякденне життя та інші сфери.

1.4 Grammatical error correction завдання та підходи його вирішення.

GEC (англ. Grammar Error Correction) – це завдання виявлення та виправлення помилок у вихідному тексті. Його застосовують у різних областях, таких як виправлення пошукових запитів, покращення результатів машинного перекладу (MT), підтримка технологічних процесів (TPP), коректура правопису у браузерах та текстових редакторах тощо.

Методи виправлення помилок у GEC можна розділити на кілька категорій: ті, що базуються на правилах; ті, що використовують синтаксичний аналіз речень; статистичне моделювання; класичні методи машинного навчання (ML); та системи машинного перекладу на основі глибокого навчання (MT) (див.рис.1.5).

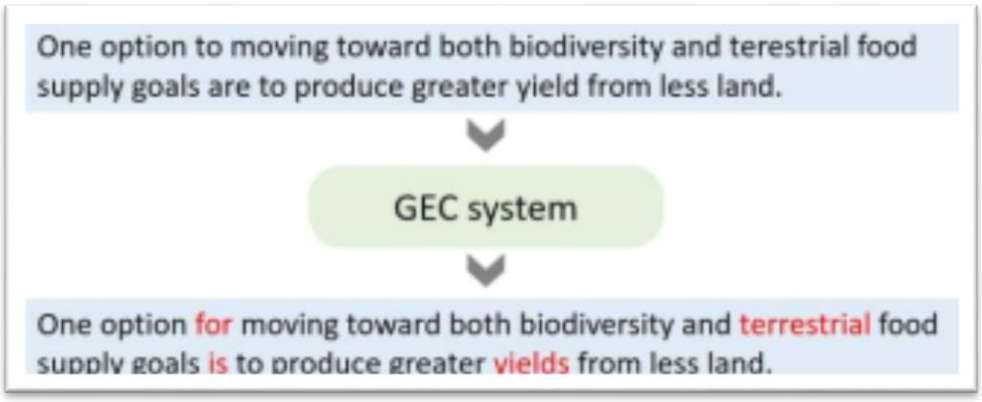

Рисунок. 1.5 – Загальний вигляд роботи GEC

MT метод використовує штучну нейронну мережу, навчену на великій кількості текстів з різних джерел. Ця модель враховує контекст тексту та використовує контекстно-залежні підходи для точної корекції помилок.

GEC використовується в різних сферах, від редагування текстів у веб браузерах до підтримки при написанні наукових статей чи робіт. Вона спрощує процес виправлення помилок і може підвищити якість написаного тексту.

Generative Pre-trained Transformer-based Error Correction представляє собою

нове покоління моделей машинного навчання, які спеціалізуються на виправленні помилок у тексті. Ці моделі засновані на архітектурі трансформерів і володіють вражаючою здатністю розуміти контекст та виправляти помилки граматики, стилю або синтаксису.

Важливим аспектом GEC є їх попереднє навчання на великих обсягах текстових даних. Це дозволяє моделям усвідомлювати та узагальнювати різноманітні структури мови та типові помилки, які зустрічаються у текстах.

Для досягнення високої точності виправлення помилок GEC використовує контекстуальні векторні представлення слів, що дозволяє їм краще розуміти значення слів у конкретному контексті. Це робить їх ефективними при виправленні навіть складних граматичних або структурних помилок.

Застосування GEC може бути корисним у багатьох областях, включаючи автоматичне редагування текстів, підтримку мовного навчання, вирішення завдань у навчанні та бізнесі, а також в підтримці авторів та редакторів у підготовці текстів для публікації.

1.5 Основні характеристики та різновиди GEC

Одним із ключових факторів успішності GEC є їхня здатність до постійного вдосконалення. Ці моделі можуть покращуватися завдяки постійному навчанню на нових данних, що дозволяє їм ставати все ефективнішими та точнішими у виправленні помилок.

Крім того, GEC поєднують у собі різноманітні методи, включаючи правила граматики, контекстуальні векторні представлення, навчання з учителем та без нього, щоб забезпечити широкий спектр можливостей для виправлення помилок у тексті.

Основні характеристики Generative Pre-trained Transformer-based Error Correction Models (GEC) включають:

- трансформерна архітектура: GEC моделі базуються на архітектурі трансформерів, яка дозволяє їм ефективно розуміти контекст тексту та враховувати довгострокові залежності між словами у реченні;

- попереднє навчання на великих даних: Ці моделі навчаються на великому обсязі текстових даних, що дозволяє їм засвоювати мовні особливості та різноманітність граматичних конструкцій;
- контекстуальне розуміння: GEC використовують контекстуальні векторні представлення слів, що дозволяє їм адаптуватися до різних семантичних та граматичних варіацій у тексті;
- методи навчання без учителя та з учителем: Ці моделі використовують різноманітні стратегії навчання, включаючи автокодування (unsupervised learning) та навчання з учителем (supervised learning), для покращення точності виправлення помилок;
- автоматичне виправлення помилок: GEC може автоматично виправляти граматичні, стилістичні та синтаксичні помилки у тексті, полегшуючи процес редагування та покращуючи якість написаного;
- постійне вдосконалення та оновлення: Моделі GEC постійно вдосконалюються за рахунок навчання на нових даних, що дозволяє їм покращувати свої можливості виправлення помилок у тексті.

1.6 Постановка задачі

Цільове дослідження має на меті провести аналіз та порівняння різноманітних методів виправлення помилок у текстах для визначення їхньої ефективності, точності та застосовності з метою поліпшення якості письмового виразу. Основні завдання включають огляд існуючих підходів до виправлення помилок, проведення порівняльного аналізу їхньої продуктивності та точності, а також ідентифікацію переваг та недоліків кожного методу. Дослідження спрямоване на з’ясування оптимального методу або комбінації методів, які можна використовувати для покращення письмового виразу, забезпечення його більшої чіткості та граматичної правильності у різних контекстах та сферах застосування.

Об’єкт дослідження – процеси ідентифікації та корекції граматичних та стилістичних помилок в текстовому контенті. Предмет дослідження – методи та

засоби виправлення граматичних/стилістичних помилок в текстах із застосуванням оптимального конвеєру (pipeline) на основі ТРР, вибору та генерування ознак, алгоритмів машинного навчання, в умовах наявності невеликих за обсягом корпусів анотованих даних. Наукова новизна – застосування нейронних мереж з новою архітектурою, огляд state-of-the-art методів та порівняння різних етапів конвеєру (pipeline) дасть змогу визначити таку їх комбінацію, яка дозволить отримати якісну модель виправлення граматичних помилок в текстах.

Також варто зазначити, що більшість досліджень у напрямі виправлення граматичних та стилістичних помилок зосереджені на корекції помилок в англомовному текстовому контенті, наше ж дослідження пріорітезує саме україномовний контент.

2 АНАЛІЗ ІСНУЮЧИХ МЕТОДІВ

2.1 Загальний аналіз

Наразі ми маємо багато різноманітних підходів, які застосовуються для корекції граматичних, стилістичних та синтаксичних помилок. Наприклад такі як:

- правописні та граматичні правки програмного забезпечення: Багато текстових редакторів та програм для обробки тексту вміють автоматично виправляти помилки, такі як правописні та граматичні, через вбудовані алгоритми. Однак, вони можуть бути обмежені у своїй точності та можуть пропускати складніші помилки;
- моделі машинного навчання для виправлення помилок: Використання нейронних мереж та алгоритмів машинного навчання, які навчаються на великих обсягах даних, щоб автоматично виправляти помилки у тексті. Моделі, такі як GPT (Generative Pre-trained Transformer), можуть бути використані для цієї мети;
- правописні програми та плагіни: Існують спеціалізовані програми та плагіни, що фокусуються саме на виправленні правописних помилок. Вони можуть пропонувати різні варіанти виправлення та використовують правила граматики для корекції;
- користувальницькі розширення браузерів: Деякі розширення для браузерів мають можливість виправляти помилки під час набору тексту в Інтернеті. Вони можуть виявляти і виправляти найпоширеніші граматичні та стилістичні помилки.

Ми ж будемо робити наголос в основному на моделях машинного навчання для виправлення помилок.

2.2 Методи вирішення задач GEC

Для розуміння вирішення проблеми цієї роботи нам необхідно розглянути найвідоміші підходи методів GEC.

З загальнодоступних джерел [4] було знайдено декілька різних підходів, а саме:

- підхід класифікації;
- підхід синтаксичного аналізу;
- підхід на основі правил;
- нейронний машинний переклад.

Підхід на основі класифікацій базується на навчанні моделі класифікації, яка визначає, чи містить конкретне слово, фраза або речення граматичну помилку, і в разі потреби пропонує виправлення. Цей спосіб тісно пов’язаний з методами машинного навчання і інтерпритує GEC як класифікацію даних. Основна ідея полягає в тому, щоб навчити модель класифікації, зазвичай використовуючи модель машинного навчання, таку як логістична регресія або дерево рішень, для класифікації кожного слова або фрази в реченні як помилкового або правильного, щоб розрізняти правильні конструкції від неправильних та пропонувати варіанти виправлень.

На початковому етапі необхідно підготувати навчальний набір даних, який складається з текстів з граматичними помилками та їх вірними версіями. Ці дані допомагають моделі вивчити правильні граматичні структури та правила мови.

Далі проводиться навчання моделі класифікації на основі цих даних. Модель навчається розпізнавати граматичні помилки та визначати, які фрагменти тексту потребують виправлення.

У процесі виявлення граматичних помилок модель перевіряє тексти на наявність неправильних конструкцій. Якщо вона виявляє помилку, вона може запропонувати варіанти виправлень на основі вивчених правильних зразків.

Один з важливих аспектів цього підходу - оцінка якості виправлень. Модель використовує метрики оцінювання, такі як точність, відновлення та F-міра, щоб визначити ефективність своїх виправлень та вдосконалити свою роботу у майбутньому.

Застосування підходу на основі класифікації в GEC дозволяє автоматизувати процес виявлення та виправлення граматичних помилок, що

забезпечує покращення якості текстів та зроблення їх більш зрозумілими та професійними але має недолік у вигляді необхідності великих обсягів даних для навчання. Як приклад маємо такі системи як Grammarly або Gingersoftware. Також в сучасності цей метод поєднували з нейроними мережами (рис. 2.1), що показало непогані результати у знаходженні та виправлені помилок.

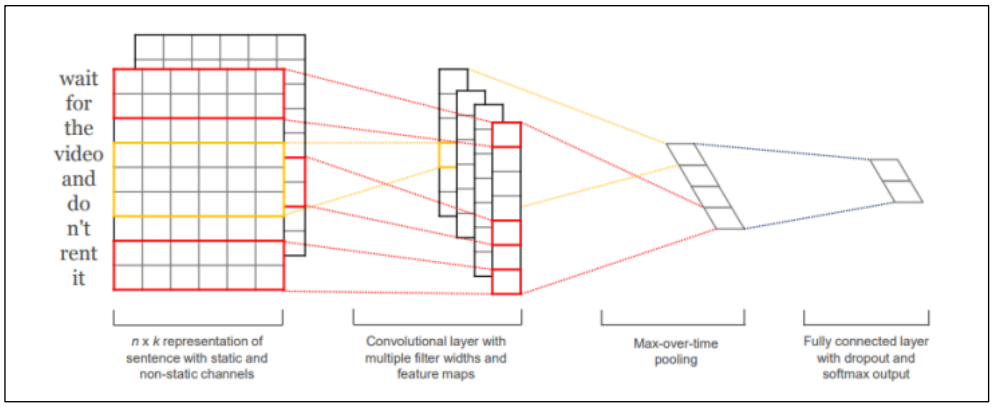

Рисунок 2.1 – Представлення архітектури нейронної мережі на основі класифікації

Синтаксичний аналіз же представляє собою базу даних з лексемами де вхідні дані розбиваються у вигляді дерева, дерево аналізується і повертає як текст або повідомлення про його неправильну структуру або помилки. Це доволі старий метод який, як недолік, має бути відповідно структурований та мати великий набір правил під кожну лексему, що підходить не для кожної мови, а також не завжди може точно визначити правильну структуру речення, особливо для складних речень і є доволі обчислювально витратним. Як приклад маємо LanguageTool і AfterTheDeadline.

Підхід на основі правил. Цей підхід ґрунтується на використанні граматичних правил мови, які визначають правильні структури та конструкції, і використовуються для автоматичного виправлення помилок. Правила зазвичай базуються на знаннях про граматику та морфологію мови.

На першому етапі розробляються або використовуються готові граматичні правила, які визначають типи граматичних помилок та їх можливі варіанти

виправлень. Ці правила можуть охоплювати різні аспекти мови, такі як правопис, граматика, синтаксис, пунктуація та інші.

Далі текст піддається аналізу на предмет виявлення граматичних помилок за допомогою визначених правил. Модель або програма перевіряють текст на відповідність цим правилам і виявляють місця, де можуть бути помилки.

У разі виявлення помилки застосовуються правила виправлення, які вказують на правильні конструкції. Наприклад, якщо у реченні виявлена помилка у формі дієслова, правила виправлення можуть вказувати на коректний час або форму дієслова, яку слід використовувати.

Підхід на основі правил вимагає ретельної розробки та оновлення граматичних правил, оскільки мова може містити велику кількість варіацій та особливостей. Крім того, використання цього підходу може бути менш гнучким у порівнянні з іншими методами, такими як машинне навчання на основі даних.

Однак підхід на основі правил може бути ефективним для виправлення стандартних граматичних помилок та використовується в багатьох системах автоматичної корекції тексту, редакторах, перекладачах та інших програмах, що покликані полегшити роботу з мовою та забезпечити високу якість текстів.

В той же час цей спосіб складний через великі обсяги праці (складним є створити набір правил, який охоплює всі можливі типи помилок), які необхідні при використанні цього способу, та така ж проблема як і у попередньому, а саме не відповідність для кожної з мов. Тут як системи які використовують такий метод можна вказати ALBERT та Aspell.

Нейронний машинний переклад - підхід ґрунтується на використанні нейромережевих моделей, які навчаються перекладати текст з помилками на правильний варіант з урахуванням граматичних правил та контексту.

Для цього методу необхідний навчальний набір даних, який складається з пар текстів з граматичними помилками та їх правильними версіями. Ці дані використовуються для навчання нейронної мережі виправляти помилки в текстах. Модель навчається перекладати текст з граматичними помилками на правильний варіант, враховуючи граматичні правила та контекст тексту.

У процесі виявлення та виправлення помилок модель застосовується до нового тексту для визначення місць, де можуть бути граматичні недоліки. Потім вона перекладає ці фрагменти тексту на правильну форму, враховуючи контекст та граматичні правила.

Перевагами такого методу є його можливість навчатись на великій кількості даних, здатності моделі ураховувати широкий спектр граматичних структур та особливостей мови, що робить його досить ефективним для виправлення різних типів граматичних помилок та здатності ефективно працювати з новими даними на основі попередбного навчання, що робить цей метод доволі ефективним.

2.3 Множина GEC моделей та їх аналіз

2.3.1 Множина моделей

LanguageTool – використовує одну з провідних моделей у сфері обробки природної мови (NLP), що вирізняється своєю здатністю розуміти контекст тексту та враховувати двобічні зв’язки між словами у реченні. Вона базується на трансформерній архітектурі, що дозволяє їй досягати відмінних результатів у завданнях NLP.

Базова модель LanguageTool має 110 мільйонів, розширена версія – 345. Особливістю її архітектури є наявність механізму уваги, завдяки чому дані можуть бути оброблені одночасно (на противагу рекурентним нейронним мережам, де дані сприймаються послідовно). Окрім того, особливістю LanguageTool є попереднє навчання нейронної мережі для вирішення двох завдань: передбачення певного слова у реченні і визначення того, чи є друге речення логічним продовженням першого. Попереднє навчання та механізм уваги дають змогу отримати контекстні векторні вкладення слів.

Однією з ключових переваг LanguageTool є його здатність розуміти семантику слів у контексті. Він використовує техніку маски заміни та наступного передбачення, що дозволяє моделі вирішувати завдання, необхідні для розуміння тексту та правильного розпізнавання контексту.

Його модель використовує самонавчання на великих обсягах текстових

даних, що допомагає йому усвідомлювати широкий спектр синтаксичних та семантичних зв’язків. Це дозволяє моделі бути ефективною в різних мовних завданнях, таких як виправлення помилок, розпізнавання сутностей та інше.

Також слід відзначити, що LanguageTool є платформонезалежною моделлю, тобто його можна успішно використовувати у різних доменних областях та завданнях NLP. Він може бути налаштований та доопрацьований для досягнення високої точності в різних контекстах.

Однак, важливо враховувати, що він вимагає значних обчислювальних ресурсів для навчання та використання, особливо коли мова йде про роботу з великими обсягами даних. Також він може мати обмежену ефективність у роботі з деякими мовними особливостями чи в умовах з обмеженими обсягами даних для навчання.

Модель GPT (Generative Pre-trained Transformer) є однією з ключових трансформерних моделей, яка використовується в обробці природної мови. Вона ґрунтується на трансформерній архітектурі та базується на механізмі уваги, де кожне слово в реченні має змогу взаємодіяти з усіма іншими словами у вхідному тексті.

Формула уваги у трансформерах (включаючи модель GPT) виглядає наступним чином (формула 2.1):

$$Attention(Q,K,V) = softmax(QKT/ dk\text{^}2)V \quad (2.1)$$

де Q, K та V є матрицями запитів, ключів та значень відповідно,

dk є розмірністю ключів та запитів.

Ця формула дозволяє моделі встановлювати зв’язки між словами у тексті та визначати їхню важливість для кожного іншого слова.

Однією з головних переваг моделі GPT є її здатність генерувати текст та розуміти контекст для вирішення завдань NLP. Модель побудована на концепції рекурентних нейронних мереж та автокодування, де вона навчається передбачати наступні слова в тексті на основі контексту попередніх слів.

Також важливо відзначити, що GPT має можливість генерації тексту, адаптованого під різні завдання, що робить її універсальною у вирішенні різноманітних завдань NLP. Однак, вона може стикатися з проблемою відтворення знайомих фраз або занадто шаблонного тексту, особливо в разі обмежених даних для навчання.

Grammarly – має метод з розширенням базової трансформерної архітектури, яке було розроблене з метою покращення можливостей обробки довгих текстів. Ця модель була створена з урахуванням проблеми «забування» в довгих залежностях, що може виникати у звичайних трансформерах під час обробки великих текстових послідовностей.

Grammarly пропонує новий метод під назвою «segment-level recurrence», який дозволяє моделі зберігати попередні представлення та використовувати їх при обробці нових сегментів тексту. Це дозволяє моделі ефективно працювати з довгими текстами, уникнути забування далеких залежностей та підвищити якість обробки інформації у великих послідовностях.

Однією з головних переваг методу Grammarly є його здатність обробляти довгі текстові послідовності, зберігаючи інформацію про контекст у більш ефективний спосіб, що робить його корисним для різноманітних завдань обробки природної мови. Ця модель може бути особливо корисною у великих маштабах завдань, таких як генерація тексту чи мовний аналіз великих корпусів тексту. Але цей метод може вимагати значних обчислювальних ресурсів для ефективної роботи через свою складну архітектуру та додаткові механізми зберігання контексту.

ALBERT (A Lite BERT) – це модель на основі трансформерної архітектури, розроблена з метою ефективного навчання та зменшення обчислювальних витрат по порівнянню зі стандартними BERT-подібними моделями. Основна ідея ALBERT полягає в оптимізації та упакуванні параметрів для підвищення ефективності.

Ця модель пропонує дві важливі стратегії для зменшення числа параметрів: shared-layers та factorized embedding parameterization. Shared-layers дає

можливість спільно використовувати підсистеми моделі для різних вхідних позицій, що дозволяє значно зменшити загальну кількість параметрів. Factorized embedding parameterization зводить до мінімуму розмірність векторів у вбудовуваному просторі слів, зменшуючи кількість параметрів та вимоги до пам’яті.

Однією з ключових переваг ALBERT є його висока ефективність у порівнянні з іншими BERT-подібними моделями при використанні меншої кількості параметрів. Це дозволяє досягати гарних результатів у завданнях обробки природної мови з меншими обчислювальними витратами.

Також важливо відзначити, що ALBERT має великий потенціал у використанні для завдань з обмеженими ресурсами, оскільки вона може досягати близьких результатів до більш складних моделей, використовуючи меншу кількість параметрів. Це робить ALBERT привабливим вибором для широкого спектру завдань у NLP з огляду на ефективність та економію ресурсів.

Gemini – має модель, що базується на контекстуальних векторах слів, розроблена з метою створення представлень слів, які враховують їх контекстуальне значення у реченні. Однією з ключових особливостей Gemini є його здатність генерувати вектори слова, що враховують контекстуальний контекст, використовуючи бідирекційні рекурентні нейронні мережі (bi-directional LSTM).

Gemini використовує два рівні представлення слів: перший рівень створюємо на основі речення, в якому слово контекстуалізується за допомогою бідирекційного LSTM, а другий рівень враховує контекст слова всередині речення. Формально, вектори Gemini вираховуються як зважена сума складових векторів, отриманих на кожному рівні.

Основна перевага методу полягає у його здатності до захоплення семантичних зв’язків та контекстуальних нюансів у тексті, що робить його корисним для завдань, де важлива точність у розумінні контексту слів. Однак, він може бути вимогливим до обчислювальних ресурсів через використання більш складних архітектур.

2.3.2 Опис множини критеріїв та їх характеристика

Опис множини критеріїв та їх характеристика (табл.2.1):

- точність. Метрика: Відношення кількості правильно виправлених помилок до загальної кількості помилок у тексті;
- час виконання. Метрика: Час, необхідний для виправлення помилок у вхідних текстах методом;
- масштабованість. Метрика: Здатність методу ефективно виправляти помилки у великих обсягах тексту без втрати швидкості або точності;
- ресурсомісткість. Метрика: Кількість обчислювальних або пам'ятевих ресурсів, які використовує метод для виправлення помилок;
- покриття помилок. Метрика: Діапазон видів помилок, які здатний виправляти метод (орфографічні, граматичні, синтаксичні тощо);
- адаптивність до мови. Метрика: Ефективність методу для виправлення помилок в конкретній мові, такій як українська.

Таблиця 2.1 – Шаблон для оцінювання моделей

Критерії	Точність	Час виконання	Масштабованість	Ресурсомісткість	Покриття помилок	Адаптивність
LanguageTool	xc_{11}	…	…	…	…	xc_{11}
GPT	…	…	…	…	…	…
Transformer - XL	…	…	xc_{nj}	…	…	…
ALBERT	…	…	…	…	…	…
Gemini	…	…	…	…	…	xc_{55}
Коєфіцієнти	v_1	v_2	…	...	v_n	v_1

2.3.3 Опис та аналіз шкал критеріїв

Точність: Цей критерій оцінює, наскільки ефективно метод виправляє помилки в тексті. Він показує відсоток правильно виправлених помилок у порівнянні з усіма помилками, які потрібно було виправити.

Час виконання: Цей критерій вказує на те, скільки часу потрібно методу для виправлення помилок у тексті. Це важливо, оскільки деякі методи можуть вимагати більше часу для обробки тексту, що може бути неефективним у великих обсягах даних чи у реальному часі.

Масштабованість: Цей критерій оцінює, наскільки добре метод справляється з обробкою великих обсягів тексту якісно та вчасно. Це важливо для того, щоб метод був ефективним у великих проектах або в реальному часі.

Ресурсомісткість: Цей критерій враховує кількість обчислювальних або пам’ятевих ресурсів, які потрібні для роботи. Іноді методи можуть бути дуже вимогливими до ресурсів, що може стати проблемою у великих проектах.

Покриття помилок: Цей критерій показує, наскільки широкий спектр типів помилок здатний виправляти метод. Деякі методи можуть бути спеціалізовані на виправленні певних видів помилок, тоді як інші можуть бути універсальними.

Адаптивність до мови: Цей критерій оцінює, наскільки ефективно метод працює з конкретною мовою. Для деяких методів важливо мати можливість працювати з різними мовами, включаючи українську, адже різні мови можуть мати відмінності у структурі або правилах (див.рис.2.2, табл.2.2-2.4).

$$Z^* = \max_{i=1,m} \sum_{j=1}^{n} \alpha_j \beta_j a_{ij}$$

Рисунок 2.2 – Векторний опис

Таблиця 2.2 – Опис критеріїв

Критерії	Точність	Час виконання	Масштабованість	Ресурсомісткість	Покриття помилок
LanguageTool	8	6	7	5	9
GPT	7	7	8	6	8
Grammarly	7	8	7	8	7
ALBERT	6	7	8	6	8
Gemini	7	7	6	7	7
Коєфіцієнти	0.25	0.20	0.10	0.15	0.20

Таблиця 2.3 – Результат використання на множині альтернатив принципу Парето

Критерії	Точність	Час виконання	Масштабованість	Ресурсомісткість	Покриття помилок
Grammarly	7	8	7	8	7
ALBERT	8	7	8	6	8
Коєфіцієнти	0.25	0.20	0.10	0.15	0.20

Таблиця 2.4 – Результати оцінювання

Критерії	Точність	Час виконання	Масштабованість	Ресурсомісткість	Покриття помилок
Grammarly	1.75	1.6	0.7	1.2	1.4
ALBERT	2	1.4	0.8	0.9	1.6

Отже, за допомогою вирішення багатокритеріальної задачі було визначено цінність моделей GEC і зроблено висновок, що методи NLP основані на

архітектурі трансформерів мають найвищий коєфіцієнт цінності, а отже рекомендовані для користування.

3 РОЗРОБКА ТА ПОРІВНЯННЯ РОЗРОБЛЕНИХ GEC СИСТЕМ

3.1 Опис обраних для розробки технологій

Задля детальнішого практичного порівняння було вирішенно створити програмні застосунки за допомогою мови програмування C# [5], технологій ASP.Net Core та Entity Framework [6]. Причиною вибору саме цих технологій стало:

- C# у поєднанні з ASP.NET Core дозволяє легко працювати з API, що забезпечує великий вибір інструментів роботи з текстом. За допомогою ASP.NET Core можна легко налаштувати проект і працювати з API для наших додатків;
- C# дозволяє використовувати HttpClient [7], який входить до складу .NET, що значно спрощує роботу з HTTP-запитами до зовнішніх API та обробкою її відповіді;
- Entity Framework є ефективним ORM (Object-Relational Mapping) інструментом, що спросить роботу з джерелом даним (у нашому випадку це GEC датасет) і дозволить інтерпретувати дані та працювати з ними як об’єктами C# без потреби створення ручного коду необхідного для створення запитів до джерела даних;
- асинхроність і продуктивність, що важливо у нашому випадку так як оскільки асинхронні запити зменшують час очікування і підвищують час відгуку додатка який працює з великою кількістю даних;
- наявність великої кількості відкритої документації, що дає ресурси для вирішення проблем, які можуть виникнути на ептапі розробки додатку і дасть можливість швикдко вирішити потенційні проблеми.

3.2 Створення GEC системи з використанням методу правил (ALBERT)

Як було вже вказано ALBERT має модель на основі трансформерної архітектури, яка розроблена з метою ефективного навчання та зменшення

обчислювальних витрат. Основна ідея ALBERT полягає в оптимізації та вибору параметрів корегування для підвищення ефективності.

Реалізуємо метод з параметром корегування якого буде виправлення неправильно вжитих артиклів у реченні написаного українською мовою.

```
        public static string CorrectSubjectVerbAgreement(string text)
        {
            Dictionary<string, string> subjectVerbPairs = new
Dictionary<string, string>() {…};
            foreach (var pair in subjectVerbPairs)
            {
                text = Regex.Replace(text, $@"\b{pair.Key}\b",
pair.Value, RegexOptions.IgnoreCase);
            }

            return text;
        }

        public static string CorrectArticleUsage(string text)
        {
            text = Regex.Replace(text, @"\ba ([aeiouAEIOU]\w*)", "an
$1");
            text = Regex.Replace(text, @"\ban ([^aeiouAEIOU\s]\w*)", "a
$1");

            return text;
        }
```

Дана частина реалізує метод трансформерів, дані для навчання якого береться з набору даних (У нашому випадку ним виступає Dictionary subjectVerbPairs) який містить у собі пари-значення параметрів обробки тексту.

Далі тестуємо результати, використавши при цьому бібліотеку Diagnostic [8] задля заміру часу обробки даних та ресурсозатратності процесу.

```
            Console.OutputEncoding = System.Text.Encoding.UTF8;

        string inputText = "я йду в школу, ти йде до магазину, він йде на
роботу, ми йдем в парк, ви йдем в кіно, вони йдут додому.";
        Console.WriteLine("Оригінальний текст: " + inputText);

        Stopwatch stopwatch = new Stopwatch();
        stopwatch.Start();
        TimeSpan startTime =
Process.GetCurrentProcess().TotalProcessorTime;

        string correctedText = CorrectGrammar(inputText);
        Console.WriteLine("Скорегований текст: " + correctedText);
```

```
            stopwatch.Stop();
            TimeSpan endTime =
Process.GetCurrentProcess().TotalProcessorTime;

            Console.WriteLine($"Execution Time:
{stopwatch.ElapsedMilliseconds} ms");
            Console.WriteLine($"Processor Time: {endTime - startTime}");
        }

        public static string CorrectGrammar(string text)
        {
            text = CorrectSubjectVerbAgreement(text);
            text = CorrectArticleUsage(text);

            return text;
        }
```

Структура Stopwatch та відповідні методи дозволять нам заміряти та порівняти витрати ти час обчислювального процесу (див.рис.3.1).

Рисунок. 3.1 – Результати роботи методу правил

Отримуємо час виконання в 103 ms.

3.3 Створення GEC системи з використанням методу «segment-level recurrence» (метод Grammarly)

Даний метод є розширенням методу трансформерів і призначений для роботи з великим обсягами даних і вимагає значних обчислювальних ресурсів для ефективної роботи через свою складну архітектуру та додаткові механізми зберігання контексту.

Реалізуємо додаток використовуючи даний метод.

Використовуючи Postman сформуємо запит який будемо використовувати у подальшій розробці (див.рис.3.2).

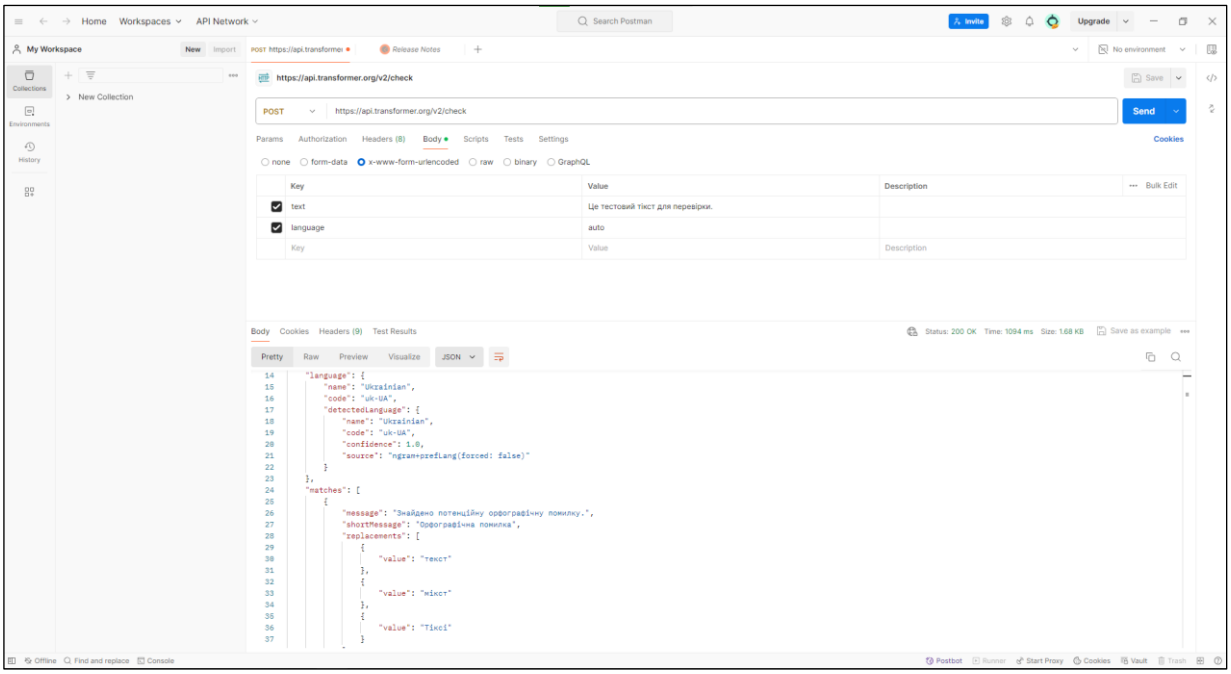

Рисунок 3.2 – Сформований запит у Postman

Далі створимо моделі даних через які будемо взаємодіяти з джерелом даних. Дані моделі представляютимуть собою відповідь від серверу до застосунку.

```
public class Context
{
    public string Text { get; set; }
    public int Offset { get; set; }
    public int Length { get; set; }
}

public class Rule
{
    public string Id { get; set; }
    public string Description { get; set; }
}

public class Match
{
    public string Message { get; set; }
    public string ShortMessage { get; set; }
    public int Offset { get; set; }
    public int Length { get; set; }
    public List<object> Replacements { get; set; }
    public Context Context { get; set; }
    public string Sentence { get; set; }
```

```
        public Rule Rule { get; set; }
    }
```

Клас Context відповідає за текст який передано у запиті та його характеристики, грає роль допоміжного класу. Клас Rule відповідає за дані опису помилки і теж використовується як допоміжний клас у класі Match який вже в свою чергу має всю характеристику обробленого за допомогою GEC методу тексту.

```
Console.OutputEncoding = System.Text.Encoding.UTF8;

string textToCheck = "Ваш тікст для перевірки тут. Тут теж буде
памилка яку ви маєте побачити";

Stopwatch stopwatch = new Stopwatch();
stopwatch.Start();

TimeSpan startTime = Process.GetCurrentProcess().TotalProcessorTime;

using (HttpClient client = new HttpClient())
{
    var values = new Dictionary<string, string>
{
    { "text", textToCheck },
    { "language", "uk" }
};

    var content = new FormUrlEncodedContent(values);

    HttpResponseMessage response = await client.PostAsync(url,
content);
    string responseString = await
response.Content.ReadAsStringAsync();

    var languageToolResponse =
JsonConvert.DeserializeObject<LanguageToolResponse>(responseString);

    if (languageToolResponse != null && languageToolResponse.Matches
!= null)
    {
        foreach (var match in languageToolResponse.Matches)
        {
            Console.WriteLine($"Message: {match.Message}");
            Console.WriteLine($"Context: {match.Context.Text}");
            Console.WriteLine($"Sentence: {match.Sentence}");
            Console.WriteLine($"Rule: {match.Rule.Description}");
            foreach (var i in match.Replacements)
            {
                Console.WriteLine($"Cases of replacement: {i}");
            }
        }
```

```
            }
            else
            {
                Console.WriteLine("No matches found.");
            }
        }

        stopwatch.Stop();
        TimeSpan endTime = Process.GetCurrentProcess().TotalProcessorTime;
        Console.WriteLine($"Execution Time: {stopwatch.ElapsedMilliseconds}
ms");
        Console.WriteLine($"Processor Time: {endTime - startTime}");
```

Цей код реалізовує логіку доставку, прийняття, десеріалізацію та показ даних які були оброблені методом GEC (див.рис.3.3).

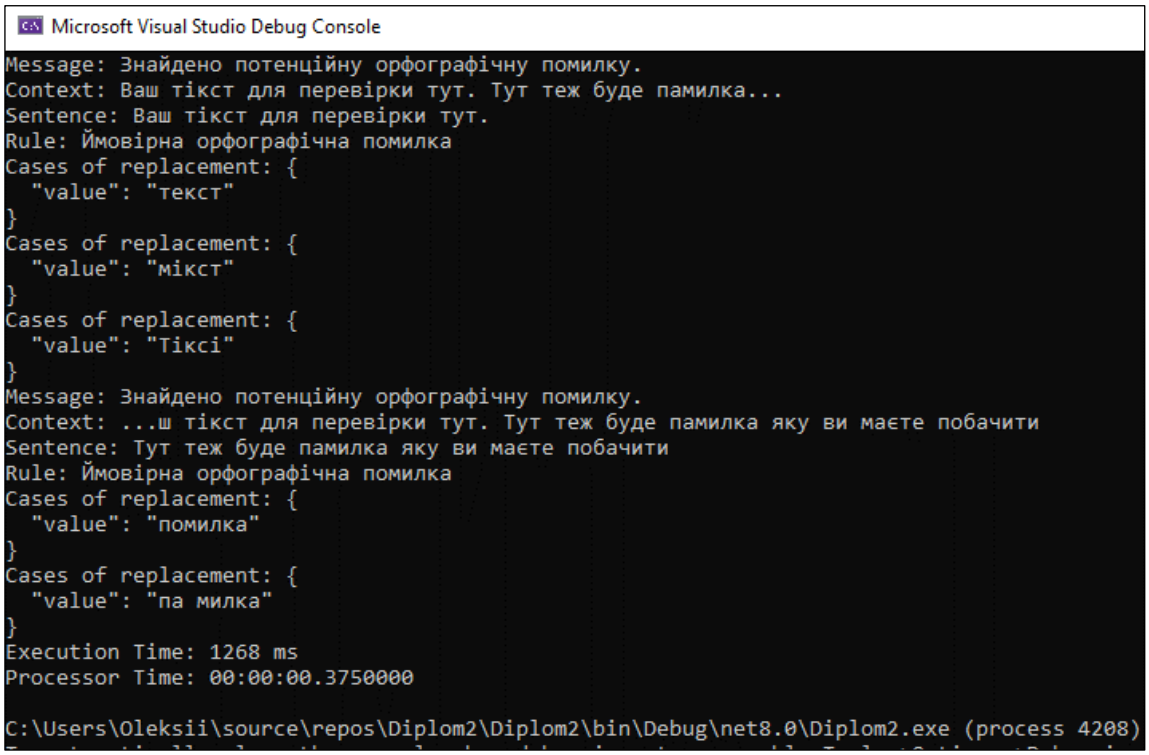

Рисунок 3.3 – Результат обробки тексту

Також так само як і у попередньому випадку за допомогою бібліотеки Diagnostic було здійснено заміри продуктивності та ресурсозатратності обробки тексту

3.4 Порівняння систем

Перший метод є більш ефективним у плані швидкості виконання та використання процесорних ресурсів, оскільки він виконує лише задачі встанолені

йому по окремій категорії корекції помилок і через це має невеликі завтрати по ресурсам системи але в той же час це обмежує систему і змушує масштабувати її конкретно до потреб користувача. Другий метод додає значні затримки через обробку даних по багатьох параметрах одночасно але має більшу ефективність через можливість охоплення великої кількості даних по багатьох параметрах корекції.

ВИСНОВКИ

У даній роботі було проведено аналіз різноманітних підходів до виправлення помилок у тексті, а також порівняння їхньої ефективності та застосовності.

Проаналізували існуючі системи і даній галузі, що відповідають за задачу виправлення помилок в текстах (він же Grammatical Error Correction)

Виявлено, що моделі на основі машинного навчання, виявляють високу точність та здатність до контекстуального розуміння тексту, що робить їх ефективними у виправленні граматичних, стилістичних та синтаксичних помилок але потребує більших ресурсів.

Жоден з методів GEC не є ідеальним, і кожен з них має свої сильні та слабкі сторони і найкращий метод для конкретного завдання буде залежати від конкретних потреб і обмежень задачі.

Було проведено аналіз існуючих систем виправлення помилок в текстах та медоти які вони використовують та обрано найоптимальніші з них.

Для повного завершення роботи було проведено порівняння цих методів та визначення кращих варіантів їх використання для різних цілей.

Було завершено усі поставлені у меті етапи кваліфікаційної роботи.

Отже, можна зробити висновок, що підходи, які базуються на моделях машинного навчання з використанням трансформерів, є перспективними для покращення якості текстового виразу та автоматичного виправлення помилок у текстах. Проте, необхідно продовжувати дослідження та вдосконалення цих моделей для покращення їхньої точності та роботи з різними мовними особливостями та контекстами.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. A New Evaluation Method: Evaluation Data and Metrics for Chinese Grammar Error Correction: https://arxiv.org/pdf/2205.00217
2. The UNLP 2023 Shared Task on Grammatical Error Correction for Ukrainian: https://aclanthology.org/2023.unlp-1.16.pdf
3. Text Normalization for Natural Language Processing (NLP): https://towardsdatascience.com/text-normalization-for-natural-language-processing-nlp-70a314bfa646
4. Detecting Errors to Improve Grammar Error Correction Models: https://www.scribendi.ai/detecting-errors-to-improve-grammar-error-correction-models/
5. C# language documentation: https://learn.microsoft.com/en-us/dotnet/csharp/specification/
6. Entity Framework 6 Introduction: https://www.entityframeworktutorial.net/entityframework6/introduction.aspx
7. HttpClient Class documentation: https://learn.microsoft.com/en-us/dotnet/api/system.net.http.httpclient?view=net-8.0
8. System Diagnostic documentation: https://learn.microsoft.com/en-us/dotnet/api/system.diagnostics?view=net-8.0