Issue 16, 2024

https://doi.org/10.23939/sisn2024.16.011

УДК 004.9

ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ ВИРІШЕННЯ ЗАДАЧІ ВИПРАВЛЕННЯ ПОМИЛОК В УКРАЇНОМОВНИХ ТЕКСТАХ

Ростислав Федчук[1], Вікторія Висоцька[2]

[1, 2] Національний університет "Львівська політехніка",
кафедра інформаційних систем та мереж, Львів, Україна
[1] E-mail: rostyslav.b.fedchuk@lpnu.ua, ORCID: 0009-0002-6669-0369
[2] E-mail: victoria.a.vysotska@lpnu.ua, ORCID: 0000-0001-6417-3689

Ця стаття присвячена дослідженню та аналізу задач виправлення граматичних помилок у текстах українською мовою, що є важливою проблемою у сфері опрацювання природної мови. У статті розглянуто специфічні виклики, які постають перед системами автоматичного виправлення помилок, зумовлені особливостями української мови, як-от морфологічна складність. Наведено приклади типових помилок та проаналізовано, чому наявні методи виправлення граматичних помилок часто виявляються недостатніми для української мови. Огляд літератури охоплює останні дослідження та публікації у сфері виправлення граматичних помилок, зокрема ті, що стосуються інших мов, та висвітлює підходи, які можуть бути адаптовані для української мови. Особливу увагу приділено аналізу наявних корпусів текстів українською мовою, як-от UA_GEC та інші, що використовуються для тренування моделей машинного навчання. Описано їхній обсяг, типи текстів та специфікації, а також їхні переваги та недоліки. Розглянуто інструменти для опрацювання природної мови, що підтримують українську мову, як-от: LanguageTool, NLP-uk, Stanza, NLP-Cube, pymorphy2, Tree_stam. Проаналізовано їхні функціональні можливості, продуктивність та описано перенавчені моделі машинного навчання, зокрема mBART50, mT5, що були адаптовані для української мови, та їхню ефективність у задачах виправлення граматичних помилок. У статті представлено практичні аспекти застосування цих моделей та корпусів для автоматичного виправлення граматичних помилок в текстах українською мовою. Детально описано процес адаптації моделей до специфіки української мови, наведено приклади практичних кейсів та проведено аналіз результатів. Значну частину статті присвячено опису одного з варіантів розроблення системи підтримки прийняття рішень для виправлення помилок у текстах українською мовою. Наведено архітектуру системи, її основні компоненти та процеси, що реалізовані за допомогою UML діаграм. Описано вхідні та вихідні дані, а також надано аналіз отриманих результатів, що демонструють ефективність запропонованих рішень. Результати цього дослідження можуть бути корисними для розробників NLP-систем, дослідників у сфері опрацювання текстів та освітніх установ, які займаються вдосконаленням якості письмових текстів українською мовою.

Ключові слова: ідентифікація та корекція граматичних помилок, виявлення та виправлення помилок, моделі машинного навчання, корпуси текстів, опрацювання природньої мови, лінгвістичні інструменти, українська мова.

Вступ

У сучасному світі, де інформація поширюється миттєво завдяки інтернету та соціальним мережам, здатність писати граматично правильні тексти стає надзвичайно важливою. Граматика не лише впливає на зрозумілість і точність повідомлення, але й на загальне враження про автора

тексту. Помилки у текстах можуть знизити рівень довіри до джерела інформації, викликати непорозуміння або навіть спричинити репутаційні втрати. Виправлення граматичних помилок (GEC) у текстах стає критично важливим завданням, особливо для текстів українською мовою, які мають специфічні особливості та складнощі.

Актуальність теми обумовлена зростаючим використанням української мови в різних сферах, зокрема освіті, медіа, бізнесі та державному управлінні. Попри значний прогрес у галузі опрацювання природної мови (NLP) для англійської та інших популярних мов, розроблення ефективних методів та інструментів для української мови все ще залишається недостатньо вивченою сферою. Українська мова має складну морфологічну структуру: відмінювання, рід, числові форми та багатозначність слів, що створює додаткові виклики для автоматичних систем виправлення помилок.

Цінність цієї теми полягає у підвищенні якості письмових текстів українською мовою, що сприятиме покращенню комунікації, зменшенню кількості помилок у публічних документах та освітніх матеріалах, а також підвищенню рівня грамотності населення. Крім того, розвиток технологій GEC для української мови може стимулювати подальші дослідження в цій галузі та сприяти інтеграції української мови у світову наукову спільноту з NLP.

Особливістю цього дослідження є комплексний підхід, який передбачає аналіз наявних корпусів текстів українською мовою, огляд інструментів та моделей машинного навчання, а також розробку та оцінку системи підтримки прийняття рішень для виправлення граматичних помилок. Це дослідження має на меті не лише виявити основні проблеми та виклики, але й запропонувати рішення, які можуть бути використані розробниками та дослідниками для покращення автоматичних систем опрацювання української мови. Отже, дослідження, представлене у цій статті, спрямоване на заповнення прогалин у знаннях та інструментах для GEC української мови, що є важливим кроком на шляху до створення більш досконалих та точних NLP систем для української мови.

Формулювання проблеми

Опрацювання природної мови є галуззю штучного інтелекту, яка займається взаємодією між комп'ютерами та людською мовою. Вона включає широкий спектр завдань, як-от: розпізнавання мовлення, синтаксичний та семантичний аналіз, машинний переклад, видобування інформації, та GEC. Основна мета NLP полягає в тому, щоб зробити машини здатними аналізувати, розуміти і відтворювати людську мову так само, як це роблять люди.

Grammar Error Correction (GEC) є процесом виявлення та виправлення граматичних, орфографічних, пунктуаційних та стилістичних помилок у текстах. Це завдання є надзвичайно важливим для підвищення якості текстів та поліпшення загальної комунікації. Українська мова має низку специфічних особливостей, що створюють унікальні виклики для завдання виправлення граматичних помилок. Серед них можна виділити:

- Складну морфологічна структуру, притаманну українській мові, яка характеризується багатою системою відмінків (шість відмінків), які змінюють форму слів залежно від їхньої граматичної функції в реченні. Це ускладнює завдання автоматичного аналізу та корекції, оскільки правильний вибір відмінка залежить від контексту.
- Роди та числа, оскільки слова в українській мові мають різні роди (чоловічий, жіночий, середній, спільний) та числа (однина, множина). Правильне узгодження цих граматичних категорій у реченні є складним завданням для автоматичних систем, особливо в складних синтаксичних конструкціях.
- Багатозначність слів, яка властива багатьом словам в українській мові, і правильне значення залежить від контексту. Це ускладнює завдання семантичного аналізу та виправ-

лення помилок, оскільки система повинна правильно інтерпретувати значення слова в конкретному контексті.

- Пунктуація та орфографія, оскільки в українській мові правила можуть бути досить складними та контекстуально залежними, що створює додаткові труднощі для автоматичних систем виправлення помилок.

На додаток до зазначених викликів існують декілька інших факторів, які ускладнюють завдання GEC для української мови:

1. Недостатність ресурсів, оскільки більшість сучасних досліджень, які спрямовані на виконання завдань GEC, написані для виправлення помилок в англомовних текстах [1]. Завдяки цьому існує багато наборів даних, які досягнули суттєвих результатів під час корекції граматики англійської мови. Проте проведених досліджень для морфологічно складних мов, до яких і належить українська, є доволі мало. Існує обмежена кількість великих і якісних корпусів текстів українською мовою, які можна використовувати для тренування моделей машинного навчання. Це обмежує можливості для розроблення високоефективних GEC систем.

2. Недосконаліть правил та стандартів вживання української мови, яка має свої власні правила орфографії, пунктуації та граматики, які не завжди є однозначними або дотримуються авторами текстів. Це може створювати складнощі у визначенні та корекції помилок, особливо тоді, коли автор відхиляється від стандартних правил.

3. Варіативність мови у використанні слів, фраз та граматичних конструкцій залежно від регіону, соціального середовища та контексту. Ця варіативність ускладнює розробку універсальних алгоритмів виправлення помилок.

4. Контекстуальна складність, через багатозначність слів та контекстуальні залежності української мови, які можуть призводити до неправильного розуміння та виправлення помилок. Правильна інтерпретація значення слова або фрази в контексті є важливим етапом у процесі корекції, але це може бути складним завданням для комп'ютерних систем.

5. Обмеженість даних для навчання, пов'язаних з великим обсягом якісних даних, необхідних для тренування моделей машинного навчання. Однак обмежена доступність якісних даних українською мовою ускладнює розвиток ефективних систем виправлення помилок.

Формулювання мети

Метою цієї статті є дослідження і ґрунтовний аналіз сучасних методів та підходів до виправлення граматичних помилок в україномовних текстах із конкретними пропозиціями.

Для досягнення поставленої мети були визначені такі завдання:

1. Провести огляд наявних методів виправлення граматичних помилок, включно з правилами, синтаксичними методами, статистичними підходами та методами машинного навчання. Проаналізувати останні дослідження та публікації у сфері GEC для української мови.

2. Визначити та описати наявні корпуси текстів українською мовою, які використовуються для навчання та тестування моделей GEC.

3. Оцінити наявні інструменти опрацювання природної мови для текстів українською мовою та їхню придатність для завдань GEC.

4. Проаналізувати навчені моделі машинного навчання, які можуть бути застосовані для виправлення граматичних помилок у текстах українською мовою.

5. Розробити та описати можливий варіант системи підтримки прийняття рішень для виправлення помилок у текстах українською мовою.

Об'єктом дослідження є процес виявлення та виправлення граматичних помилок у текстах українською мовою за допомогою сучасних методів опрацювання природної мови та машинного навчання.

Предметом дослідження є методи та інструменти виправлення граматичних помилок, зокрема правила, синтаксичні підходи, статистичні методи, а також моделі машинного навчання, які використовуються для автоматичного корегування текстів українською мовою.

Дослідження у сфері виправлення граматичних помилок є надзвичайно важливим для покращення якості автоматично згенерованих текстів, що використовується у різноманітних застосунках, як-от системи перевірки правопису, автоматичний переклад, системи запитань-відповідей, а також в освітніх і професійних інструментах. Удосконалення цих методів для української мови сприятиме підвищенню якості текстової продукції та комунікації, забезпечуючи більш точне і зрозуміле спілкування.

Новизна цього дослідження полягає в комплексному підході до аналізу та оцінки сучасних методів виправлення граматичних помилок у текстах українською мовою, зокрема із застосуванням передових методів машинного навчання. Результати дослідження можуть бути використані для розроблення ефективних інструментів та систем, що підвищують якість текстів українською мовою.

Аналіз останніх досліджень та публікацій

Від початку формулювання завдання GEC минуло вже > 40 років. Дослідники використовували різноманітні підходи та методи для виконання цього складного завдання. Серед них виділяються методи на основі правил, на основі синтаксису, статистичні методи та методи машинного навчання.

Методи на основі правил використовують набір правил граматики для виправлення помилок у тексті [2]. Йдеться про залучення правил орфографії, синтаксичних правил та правил коректного вживання слів. Одним із підходів є ручне складання набору правил, але цей підхід може бути часто ресурсномістким. Головна перевага методів на основі правил є їхня прозорість із можливістю точного контролю над процесом виправлення помилок. Однак ці методи не здатні виявляти складні або контекстуальні помилки, які не відповідають встановленим правилам. Для максимальної ефективності роботи з різноманітними типами текстів цей метод потребує формування надзвичайно великої кількості правил, а отже – знань із лінгвістики. Тому основним очевидним мінусом цього підходу є труднощі з покриттям усіх можливих помилок правилами. Хоча цей метод є одним і з перших і доволі старих, проте він застосовується і досі через свою легкість в реалізації. Перспективним розвитком досліджень є побудова алгоритмів, які автоматизовано генерують правила з використанням машинного навчання.

Методи на основі синтаксису використовують аналіз синтаксичної структури речень для виявлення та виправлення граматичних помилок. Вони базуються на моделях синтаксичного розбору, як-от дерева залежностей або дерева складних структур на основі відомих досліджень у напрямку до формальних мов та граматик Наума Хомські, що викладають усім студентам із технічних наук на перших курсах дискретної математики [3]. Ці методи можуть бути ефективними для виявлення контекстуально залежних помилок, таких як невідповідність числа та роду між словами у реченні. Головною проблемою синтаксичного підходу є потреба повної граматики, яка охоплює всі типи текстів. Досі не існує надійного та доступного синтаксичного аналізатора з широким охопленням. Окрім того, синтаксичні аналізатори страждають від природної неоднозначності мови, тому зазвичай навіть для правильних речень повертається більше ніж один результат.

Статистичні методи виправлення помилок ґрунтуються на аналізі великого обсягу текстових даних для виявлення типових граматичних помилок та їхнього усунення. Ці методи можуть використовувати статистичні моделі, як-от моделі ймовірностей. Для кожного слова у тексті обчислюються ймовірності його виправлення, заміни, видалення або вставки на основі ймовірностей, які визначені в моделі. Наприклад, слово з низькою ймовірністю може бути видалено або замінено на більш ймовірне слово. Після обчислення ймовірностей для різних варіантів виправлення помилки

обирається той, який має найвищу ймовірність. Статистичні методи дають змогу ефективно виправляти помилки в тексті, особливо якщо доступно велике різноманіття правильних текстів для навчання моделі. Однак ці методи також можуть мати обмеження у виправленні складних або контекстуальних помилок, які не відображаються в статистичних властивостях тексту.

Методи машинного навчання для виправлення помилок у тексті використовують різні статистичні й математичні концепції та алгоритми, які дають їм змогу передбачувати ймовірне виправлення тексту [4]. Попри те, що передбачення алгоритмів є дуже важливими, не менш важливим є збір великого обсягу даних, які будуть використовуватися для навчання та тестування моделі. Дані можуть збиратися з різних джерел, зокрема з інтернету, книг, новин тощо. Потім дані піддаються попередньому опрацюванню. Попереднє опрацювання тексту є важливим етапом у опрацюванні природної мови (NLP), оскільки вона дає змогу підготувати текстові дані для подальшого аналізу та опрацювання. Нижче наведено основні методи й техніки, які використовуються для попереднього опрацювання тексту в NLP:

1. Токенізація полягає у розбитті тексту на окремі слова або фрази (токени), оскільки це базовий етап опрацювання тексту, який дає змогу виділити окремі елементи тексту для подальшого аналізу. Існує два типи токенізації – розбиття тексту на окремі слова та розбиття тексту на окремі речення.

2. Лематизація і стемінг використовуються для приведення слів до їхньої базової або початкової форми. Різниця цих двох понять лише в тому, що лематизація приводить слово до його базової (кореневої форми) з урахування контексту, тоді як стемінг шляхом відсікання суфіксів та префіксів без урахування контексту.

3. Видалення стоп-слів, тобто загальних слів (наприклад, “і”, “але”, “або”), які не несуть значущого змісту для аналізу. Видалення стоп-слів допомагає зменшити обсяг тексту і зосередитися на більш значущих словах.

4. Нормалізація тексту передбачає приведення тексту до стандартної форми, зокрема перетворення до нижнього регістру, видалення спеціальних символів і пунктуації.

5. Розпізнавання сутностей полягає у виявленні та класифікації сутностей у тексті, як-от: імена людей, назви організацій, дати, місця тощо.

6. Частиномовний аналіз передбачає призначення кожному слову в тексті його частини мови (іменник, дієслово, прикметник тощо), що допомагає у розумінні синтаксичної структури речень.

7. Векторизація тексту – перетворення тексту у числові вектори для машинного навчання.

8. Синтаксичний аналіз полягає у визначенні граматичної структури речень, побудові дерев синтаксичного розбору та виявленні залежностей між словами.

9. Опрацювання неструктурованих даних – витягнення корисної інформації з неструктурованих даних (наприклад, HTML-теги, JSON-формат) та зведення тексту до структурованого формату.

Перевагами використання методів машинного навчання є здатність адаптуватися до різних типів помилок та контекстів тексту, а також автоматично навчатися на нових даних задля постійного покращення ефективності. Ці методи потребують великих обсягів даних та обчислювальних ресурсів.

Одне з рішень полягає в тому, щоб розглядати GEC як завдання класифікації. Це означає, що система, яка працює з GEC, може розглядати кожну частину тексту як окремий приклад і класифікувати його на основі того, чи містить він граматичні помилки, чи ні. Основна ідея полягає в тому, що система отримує на вхід текст та повертає класифікацію кожного фрагмента цього тексту з подальшими інструкціями. До того ж завдання GEC можна розглядати як завдання машинного перекладу. Для вирішення завдання GEC за допомогою машинного перекладу, ми можемо використовувати аналогічні підходи та моделі, які використовуються для перекладу тексту з однієї мови на іншу. Модель перекладу навчається на парах речень, де одне речення є “помилковим”, а інше – “правильним”, і вивчає відповідності між ними. Інтерпретація GEC як

машинного перекладу відкриває можливість використання широкого спектру методів та технік, розроблених для машинного перекладу.

Нейронний машинний переклад (NMT) використовує глибокі нейронні мережі для автоматичного перекладу тексту, тобто для моделювання відображення між вхідним та вихідним текстами. Основними компонентами NMT є кодер, декодер, механізм уваги і функція втрат. Основна перевага NMT полягає в тому, що вона може здійснювати корекцію помилок у контексті, тобто враховувати семантику та граматику тексту при генерації виправлення. Це допомагає забезпечити більш якісні та природні переклади, особливо в складних або низько ресурсних мовах, як-от українська. NMT виявилася дуже ефективною для багатьох завдань машинного перекладу і зараз широко застосовується в багатьох системах перекладу та інших галузях опрацювання природної мови.

Підхід на основі статистичного машинного перекладу (SMT) – це підхід до машинного перекладу, який використовує статистичні моделі для перекладу тексту з однієї мови на іншу [5]. Основна ідея полягає в тому, щоб побудувати модель, яка може виробляти найбільш імовірний переклад, виходячи з аналізу статистики великої кількості паралельних текстів (текстів, що містять однаковий зміст, але написані на різних мовах). Проведені дослідження CoNLL-2013 [6] та CoNLL-2014 [7–8] показали ефективність застосування навченої SMT моделі для виправлення різних типів помилок. Проте такі моделі стають не дуже ефективними під час довготривалого врахування контекстів, що продемонстровано в роботах [6–8]. Статистичний машинний переклад був одним із популярних підходів до машинного перекладу перед приходом нейронних мереж. Хоча він усе ще використовується в деяких ситуаціях, багато новіших моделей машинного перекладу виявилися ефективнішими.

Рекурентні нейронні мережі (RNN) є класом нейронних мереж, які призначені для роботи з послідовними даними, такими як текст або часові ряди [9]. Однією з ключових особливостей RNN є здатність запам’ятовувати і використовувати інформацію з попередніх кроків для опрацювання поточного вхідного значення. Однією з основних проблем RNN є виникнення проблеми зниклих та вибуваючих градієнтів під час тренування на довгих послідовностях. Це може призводити до того, що модель не може ефективно враховувати віддалені залежності в даних. Для вирішення проблеми вибуваючих градієнтів RNN у [10] було запропоновано новий клас моделей – Transformer.

Transformers – це архітектура глибинного навчання, яка отримала широку популярність в останні роки, особливо в опрацюванні природної мови. Transformers базується на механізмі уваги та є ефективним засобом для опрацювання паралельних даних. Це дає змогу Transformers краще моделювати довгострокові залежності в тексті, а також отримувати кращі результати на багатьох завданнях опрацювання природної мови. Основні переваги Transformers:

- Універсальність – вражаюча ефективність в різних завданнях опрацювання природної мови без значних змін в архітектурі. Це робить їх універсальними і зручними для застосування в різних сферах, зокрема в разі виправлення помилок у тексті.
- Увага на контексті є однією з ключових особливостей, який дає змогу моделі враховувати контекст кожного слова в реченні під час опрацювання. Модель краще розумітиме зв’язки між словами і враховуватиме це при виправленні помилок.
- Паралельність обчислень, що прискорює роботу, оскільки опрацювання великих обсягів даних здійснюється паралельно. Це робить їх ефективними для великих корпусів тексту та великих обсягів даних.
- Можливість вивчення довгих залежностей для деяких завдань, зокрема виправлення помилок в тексті; важливо мати здатність розрізняти та виправляти помилки, які віддалені від самих слів. Transformers здатні вивчати довгі залежності в тексті завдяки механізму уваги [11], що дає їм можливість враховувати весь контекст прийняття рішення.

- Можливість перенавчання на великих корпусах даних на великих наборах даних, як-от Wikipedia або новинні статті, що дає змогу вивчати широкий спектр знань про мову та може підвищити їхню ефективність під час виправлення помилок у тексті.

Підготовлені вхідні дані використовуються для навчання моделі. Навчена потокова модель потім використовується для прогнозування вихідних даних для нових вхідних послідовностей. Дуже поширеним є алгоритм, коли спочатку попередньо тренують ваги моделі на більшому синтетичному наборі даних, а потім тренують на меншому наборі текстів, специфічних для GEC паралельними текстами. На етапі виведення створюється правильна кінцева вихідна послідовність.

Механізм уваги в Transformer дає змогу моделі концентрувати увагу на різних частинах вхідних даних під час опрацювання. Він використовується і в кодері, і в декодері. Ось короткий опис алгоритму його роботи:

1. Запит (Query) – це вектор, який представляє поточний токен (слово або символ). Запит використовується для обчислення схожості між цим токеном та всіма іншими токенами в послідовності.
2. Для кожного токен у послідовності має ключ та значення, який використовується для обчислення схожості між поточним запитом та іншими токенами, а значення є інформацією, яка пов’язана з цим токеном.
3. Для кожного токену у послідовності обчислюється “складна” увага, яка враховує схожість між поточним запитом та ключем. Це обчислення зазвичай виконується за допомогою функції подібності, як-от скалярний добуток або косинусна подібність. Потім ця увага нормалізується, щоб отримати розподіл уваги, який додається до одиниці.
4. Відповідь моделі формується як зважена сума значень, де ваги визначаються обчисленою увагою. Саме завдяки цьому модель здатна приділяти більшу частину уваги важливим частинам послідовності та менше уваги неважливим.

Декодер в Transformer своєю чергою використовується для генерації послідовностей на основі внутрішнього представлення вхідних даних, яке було згенеровано кодером. Це внутрішнє представлення містить інформацію про контекст вихідної послідовності, яка використовувалася під час опрацювання вхідних даних. Як і кодер, декодер також використовує позиційні ембедінги для передачі інформації про позицію кожного токену в послідовності. Саме через це декодер може враховувати порядок слів під час генерації вихідної послідовності.

Аналоги та інструменти опрацювання природньої української мови. Майже всі наявні програмні інструменти для вирішення завдань GEC українською мовою використовують Rule-based підхід. Однією із таких програм є LanguageTool [12], яка може перевіряти граматичні, стилістичні та орфографічні помилки в текстах, написаних багатьма мовами, зокрема українською. Проте існує конкретний проєкт демонстрації LanguageTool API для української мови – NLP-uk [13]. Цей проєкт містить інструменти для нормалізації, очищення, токенізації, лематизації, POS-тегування текстів, а також для роботи із двозначностями в українській мові. З метою вирішення NLP-завдань для української мови можна використовувати Stanza [14]. Stanza є пакетом Python для токенізації, лематизації, POS тегування, розбору залежностей, вирішення задач NER. Існує також інший пакет Python – NLP-Cube [15], призначення якого виконувати завдання токенізації, розбиття речень на частини, багатослівної токенізації, лематизації, тегування частин мови та розбору залежностей. У [16] подано морфологічний аналізатор для української мови – pymorphy2. Його функціонал передбачає POS-тегування, механізм відмінювання слів та лематизацію. Для української мови також існує стемер для опрацювання природньої української мови – Tree_stam [17]. Він створений на основі машинного навчання та використовується для зведення слова до його базової форми. Наприклад, для слів “Пробіжка”, “бігун”, “бігати” і “підбігти” стемер може повернути однаковий стем “біг”. Це допомагає зменшити кількість унікальних слів у тексті та спростити подальший аналіз. Tree_stam перевершує всі доступні на сьогодні день стемери, а також деякі лематизатори.

Він також має найнижчий відсоток помилок, пов'язаних із недооцінкою, порівняно з наявними алгоритмами розпізнавання.

Переднавчені моделі машинного навчання для української мови. Використання попередньо навчених моделей має кілька переваг порівняно зі створенням моделей "з нуля", а саме:

- Менша потреба в даних для навчання для досягнення добрих результатів.
- Зменшення часу тренування, оскільки лишається лише потреба налаштувати модель під конкретне завдання (процес fine-tunning).
- Ефективне використання ресурсів, оскільки такі моделі вже мають велику кількість параметрів, які були оптимізовані під час тренування на великих обсягах даних.
- Загальні знання про дані дають змогу таким моделям краще узагальнювати та робити більш точні прогнози.
- Попередньо навчені моделі можуть уже мати побудовані додаткові функціональності або властивості, що може бути важко або часозатратно реалізувати з нуля.

Під час вибору моделі для налаштування потрібно враховувати завдання, що виконується. Модель повинна бути така, яка була попередньо навчена на схожому або пов'язаному завданні та зі схожою архітектурою. Тобто для моделей GEC для текстів українською мовою найперше потрібно вибирати моделі, які навчені саме на них. Алгоритми моделей GEC іншими мовами не завжди ефективні, оскільки українська мова відноситься до морфологічно складних, а тому часто потребує окремих підходів. Оскільки завдання GEC відноситься до опрацювання тексту, то архітектурою варто вибирати моделі sequence2tagging, NMT або з архітектурою Transformers. *Нижче наведено список перенавчених моделей, які можна використовувати для виконання завдань GEC українською мовою:*

- mT5 [18] – модель багатомовного перекладу.
- pythia-uk – доопрацьована mT5 на wiki та oasst1 для чатів українською мовою.
- M2M-100 [19] – машинний переклад української з/на 100 мов.
- mBART50 [20] – розширена модель mBART і призначена для багатомовного перекладу.
- youscan/ukr-roberta-base [21] – модель української мови, яка вирішує завдання знаходження замаскованого слова – fill_mask.
- uk-punctcase [22] – модель відновлення пунктуації та регістру на основі XLM-RoBERTa-Uk.
- punctuation_uk_bert [23] – відновлення пунктуації та регістру на основі bert-base-multilingual-cased.
- xlm-roberta-base-uk [24] – урізана версія XLM-RoBERTa для української мови.
- aya-101 – модель, яка підтримує 101 мову. Має 13 мільярдів параметрів.
- UAlpaca – доопрацьована Llama fine-tuned для виконання інструкцій з машинного перекладу набору даних Alpaca.
- XGLM –регресійна модель, яка має 4.5 мільярдів параметрів.
- Tereveni-AI/GPT-2
- uk4b and haloop inference toolkit – моделі GPT-2 малого, середнього та великого розмірів, навчені на вікіпедії, новинах та книгах UberText 2.0.
- Electra – одна з новітніх моделей у сфері представлення тексту, яка відзначається високою ефективністю та точністю. Основна ідея за моделлю ELECTRA полягає в тому, що вона використовує дві моделі, які взаємодіють між собою, – генератор і дискримінатор. Генератор створює шумні приклади, замінюючи деякі токени вхідного тексту на штучно створені токени. Дискримінатор же намагається розрізнити оригінальні приклади від згенерованих генератором. Electra використовує більш ефективний метод попереднього навчання, порівняно з BERT. Замість навчати модель на завдання маскування токенів, Electra навчає генератор штучно створювати приклади. Electra досягає високої точності

навіть із використанням меншої кількості параметрів порівняно з іншими моделями як BERT/GPT.

- Helsinki-NLP – машинний переклад української з/на 25 мов.
- MITIE NER Model, ukr-models/uk-ner, lang-uk/flair-uk-ner, dchaplinsky/uk_ner_web_trf_large – моделі для вирішення задач NER.
- lang-uk/flair-uk-pos – модель для виконання завдань POS-тегування.
- ukrainian-word-stress – додає наголос у словах.

Наявні корпуси української мови. Алгоритми машинного навчання стали важливою складовою для вирішення різноманітних завдань, їхні можливості у прогнозуванні, класифікації та опрацюванні даних роблять їх потужним інструментом для вирішення складних проблем. Проте важливо пам'ятати, що якість та ефективність алгоритмів машинного навчання значною мірою залежать від якості та репрезентативності даних, на яких вони навчалися. І саме тут роль відіграють корпуси і набори даних.

Корпуси – це колекції текстових або інших даних, які використовуються для навчання, тестування та валідації моделей машинного навчання. Вони можуть бути зібрані з різних джерел, зокрема інтернет, книги, наукові статті, соціальні медіа тощо. Корпуси надають можливість алгоритмам машинного навчання вивчати закономірності в даних і вдосконалювати свої прогностичні та аналітичні здібності. Вони також можуть використовуватися для генерації нових даних, наприклад, для розвитку та покращення синтетичних мовленнєвих моделей. Однак корпуси – це лише частина екосистеми даних. Перед опрацюванням корпуси часто потребують очищення, анотації та інших методів підготовки, щоб забезпечити високу якість та репрезентативність. На сьогодні вже існує чимала кількість корпусів текстів українською мовою, призначених спеціально для машинного навчання, стали вагомим інструментом для розвитку та вдосконалення моделей машинного навчання. Однак порівняно з корпусами текстів англійською мовою ресурсів для дослідження текстів українською мовою ще не так багато, і вони не покривають так багато сфер NLP, як хотілося б. Через відсутність великого безкоштовного корпусу, у [25] автори створили UberText 2.0**.** Наразі UberText 2.0 є одним із найбільших корпусів текстів українською мовою, маючи 3,274 мільярди токенів, складається з 8,59 мільйонів текстів і займає 32 гігабайти. Окрім тексту UberText 2.0, містить нормалізацію текстів, виявлення мови, сегментацію на речення, токенізацію (зі збереженням пунктуації), лематизацію та POS-тегування. Нижче наведено деякі з найбільш відомих корпусів текстів українською мовою для машинного навчання:

- UberText 2.0 – новини, Вікіпедія, соціальна, художня та юридична література.
- UA-GEC – анотований GEC корпус.
- OSCAR [26] – дані, витягнуті з Common Crawl, 28 Гб.
- CC-100 – документи, витягнуті з Common Crawl.
- mC4 – дані з Common Crawl, 196 Гб.
- KUM [27] – текстовий корпус текстів українською мовою.
- GRAC [28] – довідковий корпус текстів українською мовою, укладений вручну.
- ukTenTen – український корпус з Мережі, 7,5 млрд токенів.
- Ukrainian Twitter corpus – Український Твіттер-корпус для виявлення токсичного тексту.
- BRUK [29]– корпус текстів сучасною українською мовою, 1 млн слів.
- Zvidusil [30] – веб-корпус із синтаксичною анотацією.
- Malyuk – сукупність корпусів OSCAR, UberText 2.0 та Українські Новини, 113 Гб тексту
- Ukrainian forums – 250 тисяч речень, зібраних із форумів.
- ZNO – ~4000 запитань та відповідей.
- Yakaboo Book Reviews – анотований корпус текстів, що складається з оглядів книг, рейтингів та описів.
- Polish-Ukrainian Parallel Corpus.

- ua-news – анотований копус текстів новин.
- ВЕСУМ – словник POS-тегів.
- Heteronyms – набір омонімів.
- OPUS.
- Tatoeba MT Challenge data sets.
- Back-translated monolingual Wiki data.
- Wiki Edits – 5 млн речень з історії редагувань української Вікіпедії.
- NER-uk – анотований BRUK для завдань виявлення сутностей (NER).
- UA-SQuAD – анотований корпус даних про відповіді на запитання.
- WSC Dataset – анотований корпус текстів ручного перекладу.
- Ukrainain news headlines – заголовки новин, 5,2 млн.
- obscene-ukr – словник ненормативної лексики.
- Word stress dictionary – наголос для 2.7M словоформ.

Проте зараз існує лише один корпус, який призначений спеціально для виправлення помилок у текстах українською мовою – UA-GEC (рис. 1) [31].

Рис. 1. Набори даних GEC для різних мов на основі даних з [31]

UA-GEC був створений 2021 року та містив 1011 текстів з помилками, що налічують 20715 речень [32]. Ці тексти зібрані з різних джерел, як-от: чати, статті, пости із соціальних мереж, есе або навіть листи у формальному стилі. Професійні редактори дозволяють аналізувати тексти на наявність помилок, що стосувалися вільного мовлення, граматики, пунктуації та орфографії. Вже 2022 року UA-GEC було оновлено до другої версії, яка вже містить удвічі більше корпусів текстів, а саме – 34 000 речень [33]. Категорії граматики і стилістики розділено, що дає змогу окремо виправляти і граматичні помилки, і використання помилкового стилю, що є доволі важким завданням. До категорії помилок залучено пунктуацію, орфографію, граматику та стиль (Fluency). Загалом категорію “Граматика” було збільшено на 13 підкатегорій, а категорію “Стиль” – на 5. Зафіксовано, що на орфографію припадає 19 % виправлень, натомість пунктуаційні виправлення трапляються частіше через суворі правила пунктуації в українській мові. Корпус GEC+Fluency містить дані від 828 авторів, 1872 текстів, 33 735 речень і 500 618 токенів із рівнем помилок 8,2 % для всього корпусу (рис. 2) [31–32]. Корпус підходить для розроблення та оцінювання систем GEC текстів українською мовою, а також для дослідження багатомовного та низько ресурсного NLP, морфологічно багатих мов та GEC на рівні документів, зокрема корекцію вільного володіння мовою. Нижче можна побачити статистичні дані українського корпусу GEC порівняно з корпусами інших мов (рис. 3).

Split	Writers	Texts	Sentences	Tokens	Annotations	Error rate
Train	752	1,706	31,038	457,017	38,383	8.1%
Test	76	166	2,697	43,601	7,865	9.0%
TOTAL	828	1,872	33,735	500,618	46,248	8.2%

Рис. 2. Статистика корпусу GEC+Fluency

Language	Corpus	Sent.	Er.
English	Lang-8	1,147,451	14.1
	NUCLE	57,151	6.6
	FCE	33,236	11.5
	W&I+L	43,169	11.8
	JFLEG	1,511	
	CWEB	13,574	1.74
Czech	AKCES-GEC	47,371	21.4
German	Falko-MERLIN	24,077	16.8
Romanian	RONACC	10,119	
Russian	RULEC-GEC	12,480	6.4
Spanish	COWS-L2H [3]	12,336	
Ukrainian	**UA-GEC**	**33,735**	**8.2**

Рис. 3. Порівняльна статистика корпусу UA_GEC з іншими мовами

*Застосування моделей та корпусів машинного навчання для української мови***.** Більшість систем GEC для англійської мови, які демонструють найкращі результати в тестах, базуються на архітектурі NMT. У [34] команда “Правописця” проводить дослідження використання аугментації (augmentation) даних, тобто застосування синтетично створених даних для навчання моделі GEC. За основу обрано mBart-50 як модель NMT, XLM-ROBERTa як seq2tag модель, а також корпус UA_GEC, на якому здійснено навчання. Для seq2tag токени позначалися KEEP, DELETE, APPEND або REPLACE тегами залежно від того, що з ними потрібно зробити. Модель MBart-50-large налаштована із завданням перекладу з української на українську, доопрацьована на UA-GEC корпусі з аугментацію синтетично згенерованим набором даних. До цієї моделі додано 5 тисяч речень, створених за допомогою кругової транслітерації (ukr-rus-ukr), 10 тисяч речень за допомогою генерації пунктуаційних помилок, 2 тисяч розмивання тексту, 10 тисяч росіянізмів. У результаті моделі на основі NMT показали кращі результати, ніж моделі на основі правил та моделі seq2tag на оцінювальному тесті (рис. 4). Після дослідження більшості найсучасніших підходів GEC у текстах англійською мовою та спробі їхньої адаптації для текстів українською мовою було виявлено, що найефективнішу систему GEC можна розробити за допомогою підходу NMT, проте seq2tag має багато можливостей для дослідження. Найкраща модель отримала оцінку 0.632 для F0.5 на наборі даних UA-GEC. До того ж результати показують, що додавання до навчальних даних UA-GEC синтетичних дає найкращі результати. Стає очевидним, що якісний корпус даних набагато більш важливий за розмір цього набору. Тому автори [34] зробили висновок, що найбільш перспективним напрямком для майбутніх досліджень є використання даних GEC, анотованих людиною.

GrammarUA(smartik/mbart-large-50-finetuned-gec) використовує доопрацьовану mBART50, яка показала гарні результати для мов з обмеженими ресурсами на спільному завданні [35]. Модель schhwmn/mt5-base-finetuned-ukr-gec створена та навчена mT5 модель на копусі UA-GEC. Особливістю цієї моделі є те, що можна використати не весь корпус UA-GEC, а лише ту частину, де є помилки.

Type	TP	FP	FN	Total P	Total R	Total F0.5
NMT (baseline)	685	302	1068	0.694	0.391	0.601
NMT (best)	691	241	1047	0.741	0.398	0.632
seq2tag (baseline)	399	753	953	0.346	0.295	0.335
seq2tag (most data)	461	1324	901	0.258	0.339	0.271
rule-based	104	1064	1194	0.089	0.080	0.087

Рис. 4. Аналіз моделей команди Правописця

WebSpellChecker [36] використовували власну архітектуру Transformer під назвою RedPenNet, що застосовує попередньо навчений кодер MLM разом із неглибоким декодером для генерації токенів заміни та проміжків для правлення випадків GEC. Під час генерації маркерів правлення ваги уваги кодера і декодера визначають діапазони правлення (початок і кінець), які вказують на позицію правлення у вихідному реченні. Токени редагування прогнозуються регресійним способом. Основною перевагою RedPenNet є можливість реалізувати будь-яке перетворення джерело-ціль за допомогою мінімальної кількості кроків авторегресії, що дає змогу ефективно вирішувати випадки GEC, зокрема взаємопов'язані правки. Однак через специфічну архітектуру, RedPenNet не підходить для навчання або детального налаштування. Отже, зручні інструменти, як-от інфраструктура HuggingFace, не можуть бути використані для швидкого налаштування та розгортання моделі. Найкращою моделлю виконання завдань GEC для текстів українською мовою є модель команди QC-NLP [37], що була представлена на спільному завданні [35]. Команда налаштувала велику багатомовну модель (mT5) у два етапи – спочатку на синтетичних даних, а потім на UA_GEC даних. Вони також використали меншу модель seq2seq Transformer, попередньо навчену на синтетичних даних і допрацьовану на UA_GEC корпусі. Ключова інновація, яку започаткувала команда QC-NLP як поетапне налаштування, спочатку на синтетичних, а потім на "золотих" даних. Показники моделі зображені на рисунку нижче (рис. 5).

		GEC+Fluency			GEC only		
Model	**Number of params.**	**P**	**R**	**$F_{0.5}$**	**P**	**R**	**$F_{0.5}$**
mT5 large	1.3B	73.21	53.22	68.09	76.81	61.39	73.14
seq2seq	275M	69.91	53.78	65.96	72.32	63.13	70.27

Рис. 5. Показники моделі GEC команди QC-NLP

Результати

Протягом генерації вихідної послідовності декодер використовує маски самоподібності для того, щоб кожен токен у вихідній послідовності не мав можливості "підглядати" на майбутні токени. У такий спосіб забезпечується правильність генерації послідовності. Декодер також використовує механізм уваги для того, щоб зосередитися на важливих частинах вхідного представлення під час створення вихідної послідовності. Тому декодер враховує контекст вхідних даних під час генерації кожного токену вихідної послідовності. Після того, як декодер згенерував вихідну послідовність, функція втрат порівнює її з фактичною вихідною послідовністю. На основі цього порівняння оновлюються параметри моделі з метою покращення її якості. У фіналі декодер генерує вихідну послідовність токенів, один за одним, використовуючи механізм уваги та інші механізми для врахування контексту вхідних даних. Кожен наступний токен генерується на основі попереднього та контексту вхідних даних. Коли кінцеву довжину вихідної послідовності досягнуто або ж появився спеціальний символ завершення, то процес зупиняється. На рис. 6 продемонстровано архітектуру Transformer. Розглянемо приклад роботи кодера для тексту "Десь тутт захована помілка.". Для спрощення припустимо, що використовується простий Transformer з одним шаром стеку кодування.

1. Токенізація передбачає, що спершу текст буде розділений на окремі токени: [“Десь”, “тутт”, “захована”, “ помілка”]. Також вхідні послідовності попередньо обробляються.

2. Ембедінги передбачає, що кожен токен буде перетворений у векторне представлення (ембедінг). Наприклад, “захована” може бути вектором [0.1, -0.8, 0.5, ...], “помілка” – [0.1, 0.3, -0.3, ...] і так далі.

3. Подання в кодер забезпечує, що кожен токен з його ембедінгом подається на вхід кодеру. Кодер виконує послідовність операцій, зокрема механізми уваги та нейронні шари, для опрацювання кожного токену та його контексту у вхідній послідовності.

4. Вихід кодера дає змогу отримати послідовність внутрішніх представлень кожного токена після опрацювання кодером. Ці представлення містять контекстуальну інформацію про кожен токен у вхідному тексті. Наприклад, вектор для “захована” може мати вигляд як [0.3, 0.2, -0.9, ...], для “помілка” – [0.7, -0.4, 0.1, ...] і так далі.

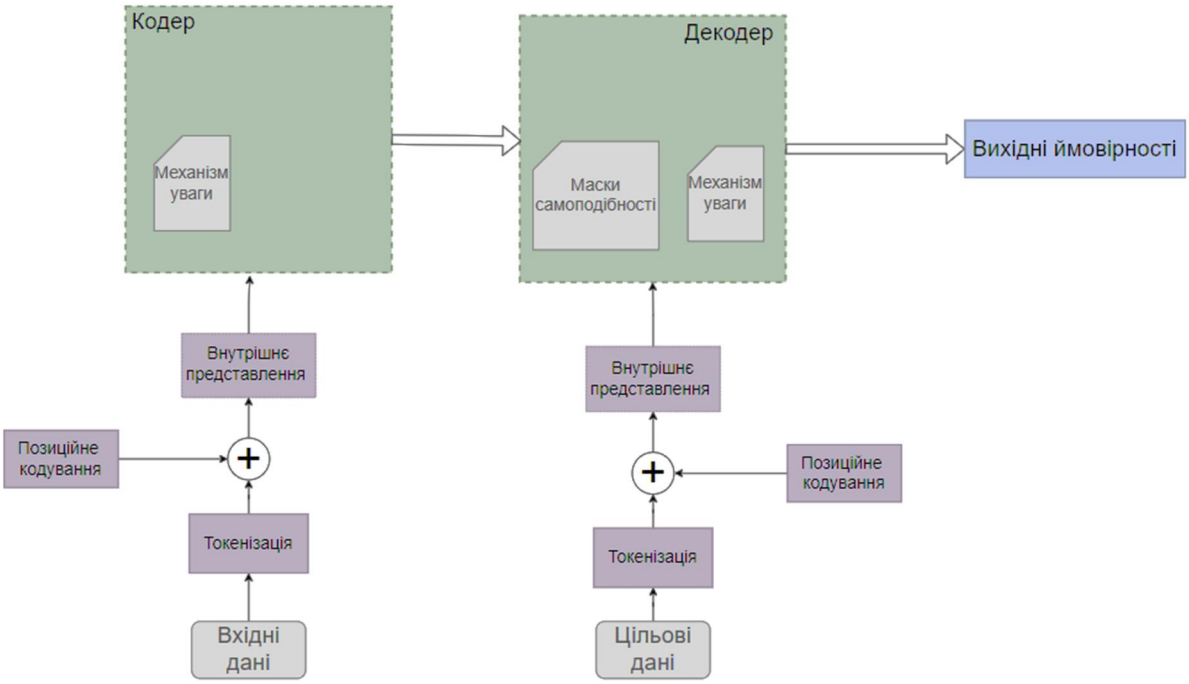
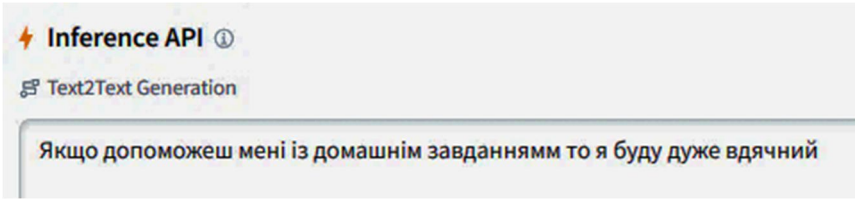


Рис. 6. Архітектура Transformer

На рис. 7–11 можна спостерігати приклади роботи моделі GEC команди Правописця.



</> JSON Output

```
[
  {
    "generated_text": "Якщо допоможеш мені із домашнім завданням, то я буду дуже вдячний."
  }
]
```

Рис. 7. Приклад виправлення пунктуаційних помилок на основі моделі GEC команди Правописця

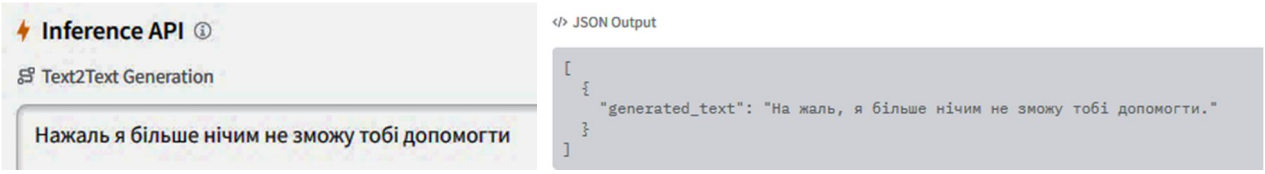


Рис. 8. Приклад успішного виправлення орфографічних та пунктуаційних помилок на основі моделі Правописця

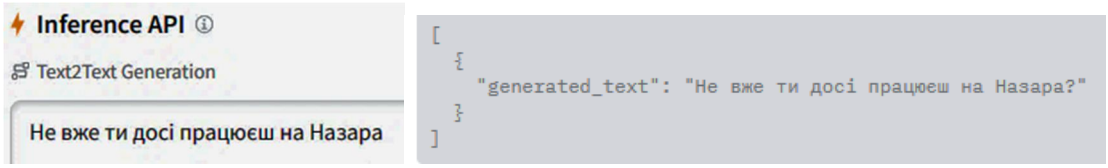


Рис. 9. Приклад виправлення не всіх помилок у тексті на основі моделі GEC команди Правописця

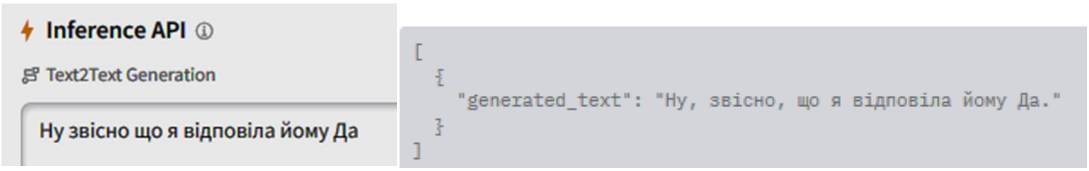


Рис. 10. Приклад неможливості виправити росіянізм у тексті на основі моделі GEC команди Правописця

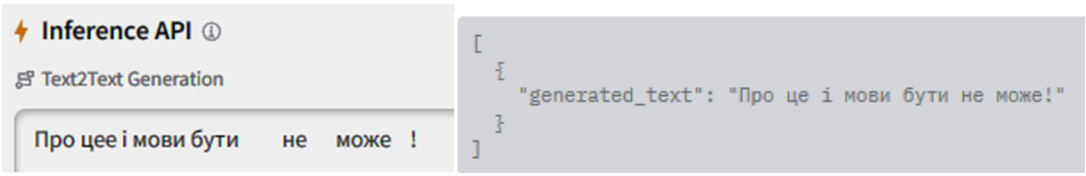


Рис. 11. Приклад успішного форматування тексту на основі моделі GEC команди Правописця

Приклад роботи моделі **GrammarUA** можна спостерігати на рис. 12–14.

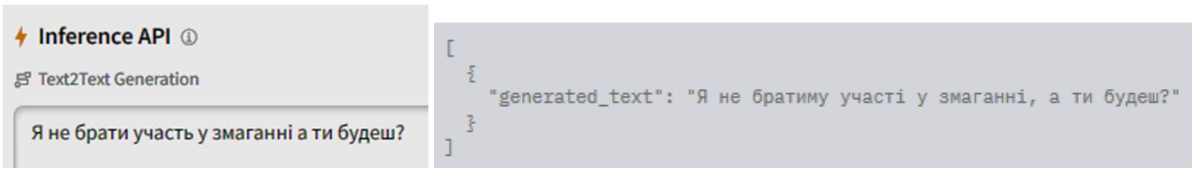


Рис. 12. Приклад виконання моделі GrammarUA

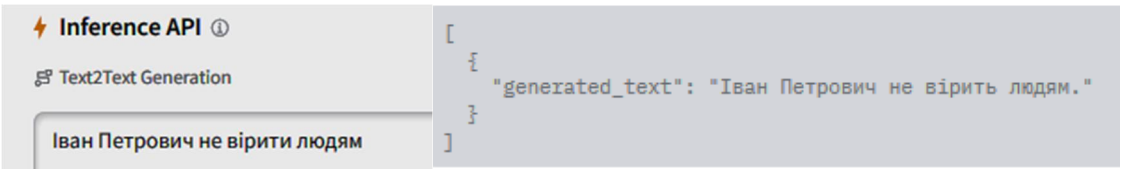


Рис. 13. Приклад успішного виконання моделі.

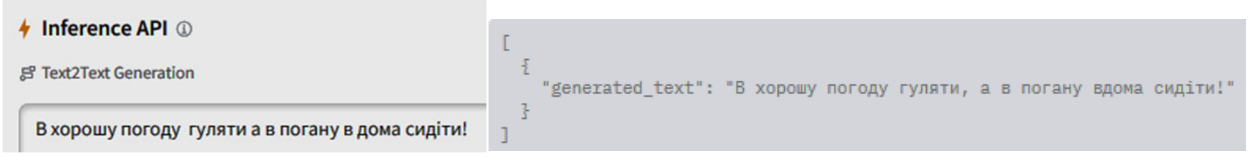


Рис. 14. Приклад некоректного виправлення тексту

Отже, існує декілька можливих шляхів вирішення проблем GEC для текстів українською мовою:

- Розроблення нових та вдосконалення старих моделей машинного/глибинного навчання.
- Інтеграція методів машинного навчання та правил для створення гібридних систем виправлення помилок.
- Використання перенавчених моделей, що забезпечує адаптацію популярних моделей для виконання завдань GEC англійською до української мови та їхнє використання може дати значний приріст у точності та ефективності.
- Розширення корпусів текстів шляхом збільшення кількості та різноманітності корпусів текстів українською мовою, що сприятиме кращому тренуванню моделей машинного навчання. Йдеться про поповнення вже наявних корпусів і створення нових.
- Інтеграція з іншими NLP-інструментами може покращити якість виправлень та забезпечити більш комплексний аналіз тексту.
- Проведення масштабних експериментів для оцінки ефективності запропонованих методів на різних корпусах текстів.

Через морфологічну складність української мови виникає необхідність використання різних методів GEC. Уже визначено, що навчання моделі “з нуля” вимагає величезної кількості часу, великих паралельних корпусів [37–41], а тому варто використовувати вже навчену модель [42–45]. Найкращі результати показали моделі mT5 і mBART-50, тому запроповано взяти їх за основу. Очікується, що основним типом інформації є власне сам текст, який потребує перевірки. Однак це не всі вхідні дані. Нижче наведено список основних типів вхідних даних для системи:

- Словники та граматичні правила (словники, граматичні правила та БД, які містять правильні слова, фрази та граматичні конструкції для порівняння з уведеним текстом і виявлення помилок).
- Дані для машинного навчання (навчальні набори текстів, анотовані дані з виправленнями помилок, функції або функції для аналізу тексту тощо).
- Метадані та контекстна інформація.
- Текстові дані (текстові рядки, які можуть бути представлені у вигляді введеного користувачем тексту, документів або посилань на сайти).
- Дані про користувача (мінімальний набір відомостей про автора тексту).

Вихідні дані системи виправлення помилок українського тексту призначені для надання користувачу або системі інформації про виявлені помилки та їхнє виправлення. До вуихідних даних входять:

- виправлені тексти, які можуть бути представлені у форматі оригінального тексту з виправленнями або в окремому файлі, де помилки позначені та супроводжуються пропозиціями щодо виправлення;
- примітки, призначені для надання користувачеві інформації про наявність помилок у реальному часі;
- статистика як загальна кількість помилок, їхні типи (орфографічні, граматичні тощо), розподіл помилок за категоріями, частота помилок у різних текстах тощо;
- звіти про помилки як тип помилки, місце в тексті, причина помилки та пропозиції щодо виправлення;
- підказки для користувача як пропозиції щодо покращення стилю чи форми вираження;
- дані для аналізу та вдосконалення системи.

Структура даних для системи підтримки прийняття рішень (СППР) для виправлення помилок у тексті українською мовою організована у вигляді бази даних, яка містить параметри / критерії та правила для виявлення та виправлення помилок. Схема бази даних подано на рис. 15.

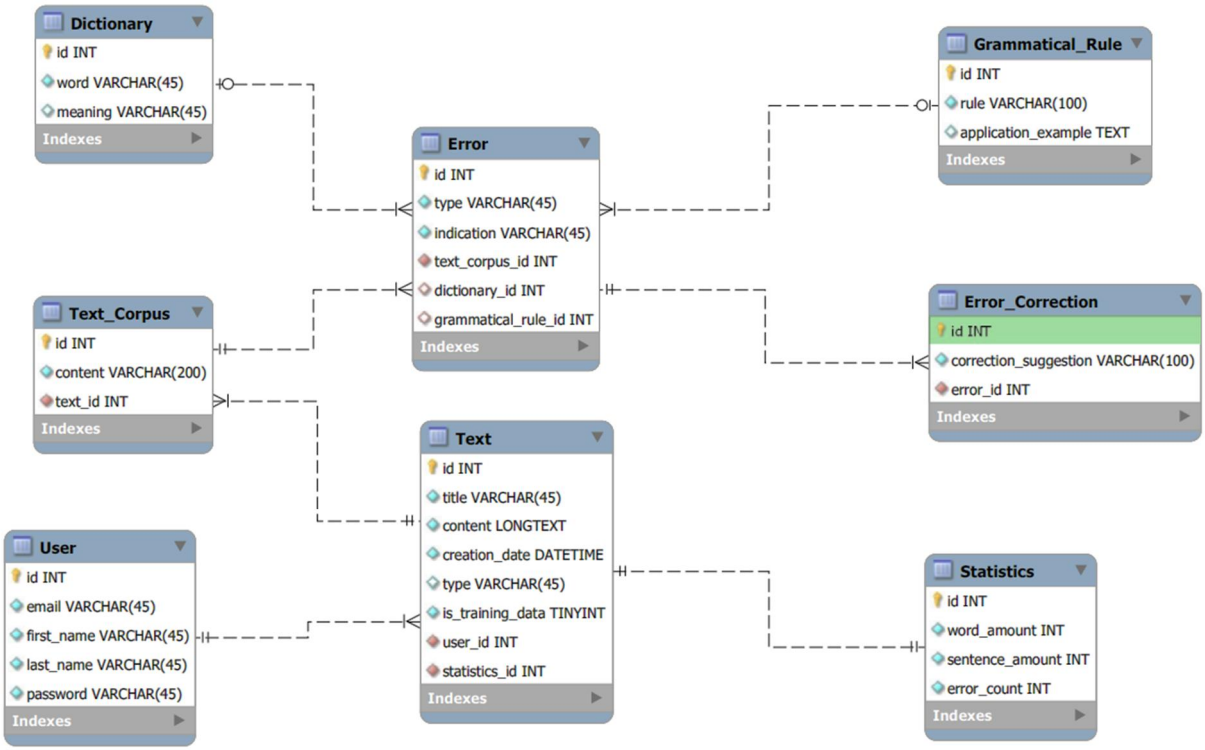


Рис. 15. Реляційна модель БД для виправлення помилок у текстах українською мовою

Діаграма варіантів використання (рис. 16) дає змогу описати взаємодію користувачів із системою та їхні функціональні вимоги. Завдяки цьому можна моделювати функціональну поведінку системи на високому рівні абстракції та розуміти, як система повинна взаємодіяти з користувачем для досягнення бізнес-цілей. Діаграма варіантів використання дає змогу отримати загальне уявлення про те, що відбуватиметься в системі. Користувач повинен увійти в систему, щоб отримати доступ до її функцій. Авторизований користувач може завантажити текст у систему різними способами: ввести текст вручну, з файлу або за покликанням на сайт із текстом. Система дає можливість перевірити правильність тексту трьома способами: перевірка тексту за словником, правилами граматики та моделлю машинного навчання. Результат роботи програми відображається користувачеві з текстовою статистикою, знайденими в тексті помилками та різними пропозиціями. Варто зазначити, що питання “Як буде працювати система?” або “Як буде впроваджуватися система?” діаграма варіантів використання не дає відповідей. Для більш детального ознайомлення із системою було побудовано діаграму кооперації (рис. 17). На цій діаграмі кооперації зображені основні компоненти та взаємодії системи підтримки прийняття рішень для ідентифікації та корекції помилок в україномовному тексті. Діаграма ілюструє, як різні елементи системи взаємодіють між собою для виконання завдань, пов’язаних з опрацюванням тексту. Діаграма кооперації є важливим інструментом для розробників та аналітиків, оскільки дає змогу візуалізувати, як компоненти системи взаємодіють між собою, що є корисним для розуміння роботи системи та її складових частин.

Взаємодія між компонентами системи така:

1. Авторизація користувача, що входить у систему, яка підтверджується сервером.
2. Завантаження тексту, яке може відбуватися вручну, з файлу або шляхом введення URL. Текст передається на сервер для опрацювання.
3. Перевірка правильності тексту, яку здійснює сервер, перевіряючи текст за допомогою граматичних правил, словника та моделі машинного навчання.

4. Виведення результатів перевірки шляхом надсилання користувачу через ПЗ, де можуть бути показані статистичні дані, наявні помилки та припущення для корекції тексту.
5. Збереження результатів у БД для подальшого використання.

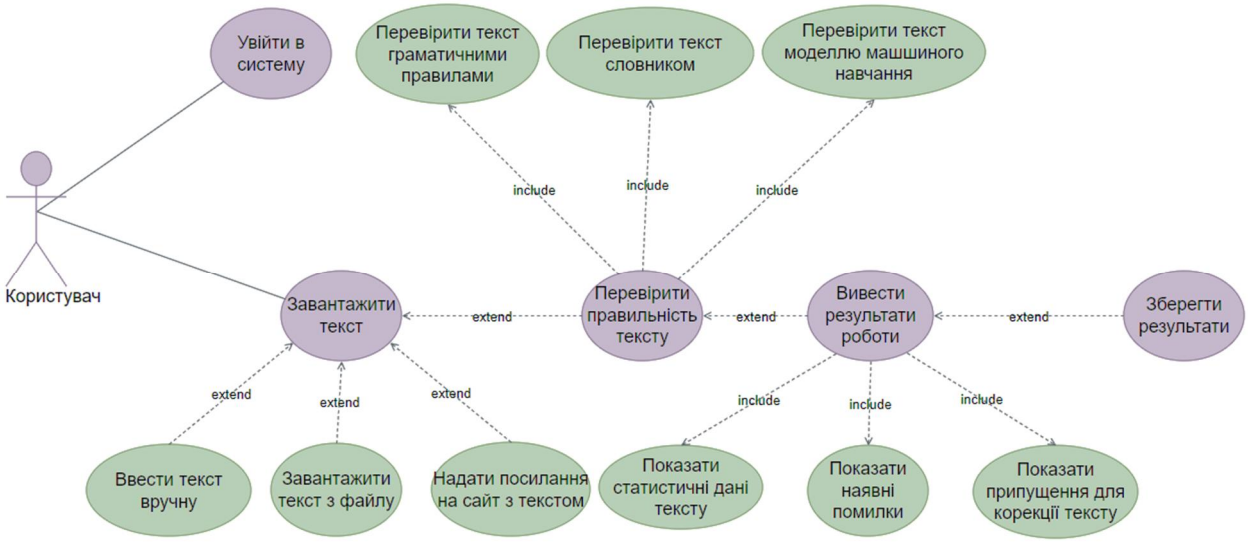


Рис. 16. Діаграма варіантів використання СППР для у виправлення помилок у тексті українською мовою

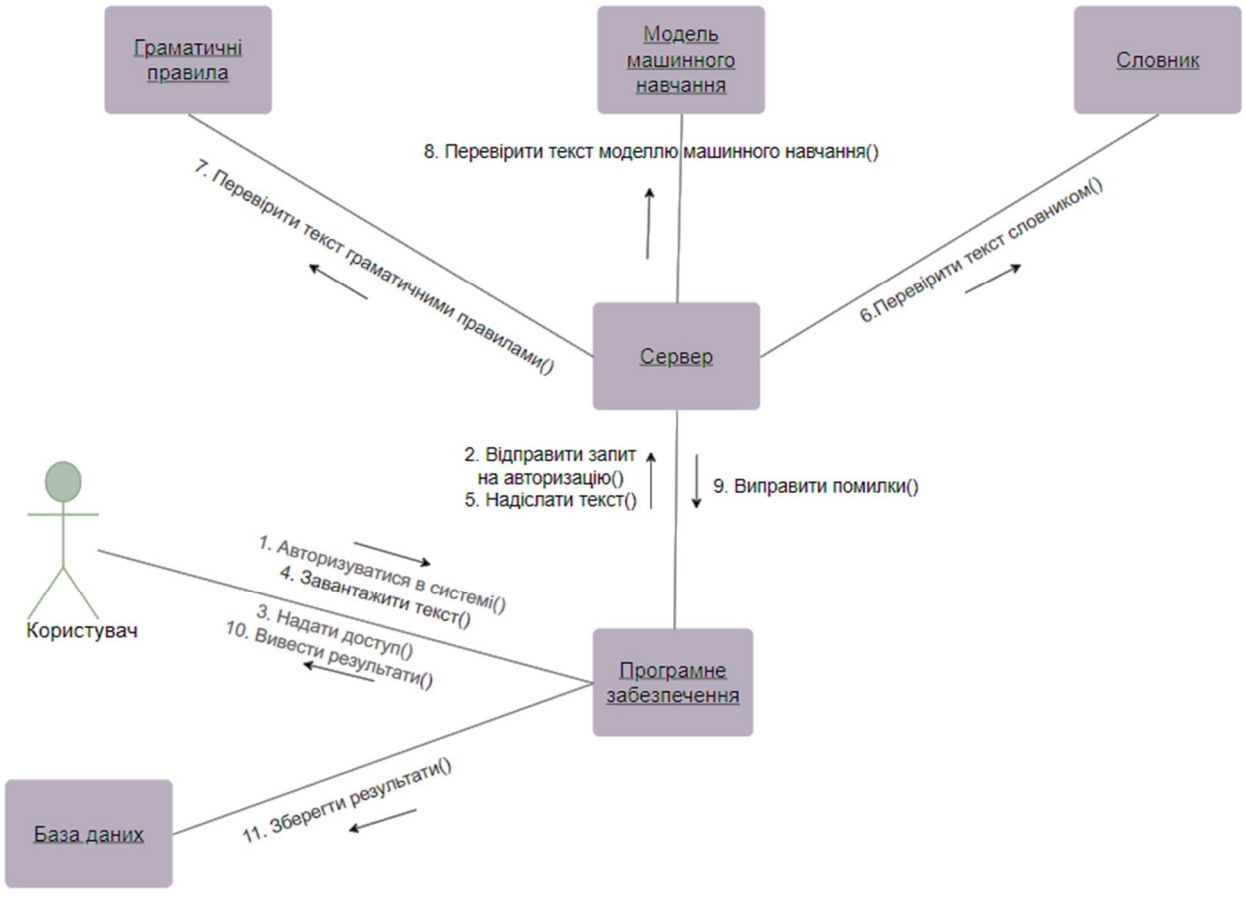


Рис. 17. Діаграма кооперації СППР для виправлення помилок у тексті українською мовою

Щоб детальніше побачити описану взаємодію та послідовність окремих компонентів системи, була створена діаграма послідовності. На рис. 18 видно, що є етапи авторизації, створення та збереження текстів. База даних використовується для зберігання параметрів правильності / зразків для подальшої персоналізації правильності, запропонованої системою. Спочатку користувач вибирає варіант завантаження даних, вставляючи текст, відкриваючи файл або надаючи посилання на сайт. Персональне програмне забезпечення, доступне в інтернеті, надсилає дані на сервер для аналізу. На схемі показано лише один можливий підхід до побудови системи граматичної правильності. Він поєднує в собі перевірку орфографії, правила, аналізатори та використання алгоритмів машинного навчання.

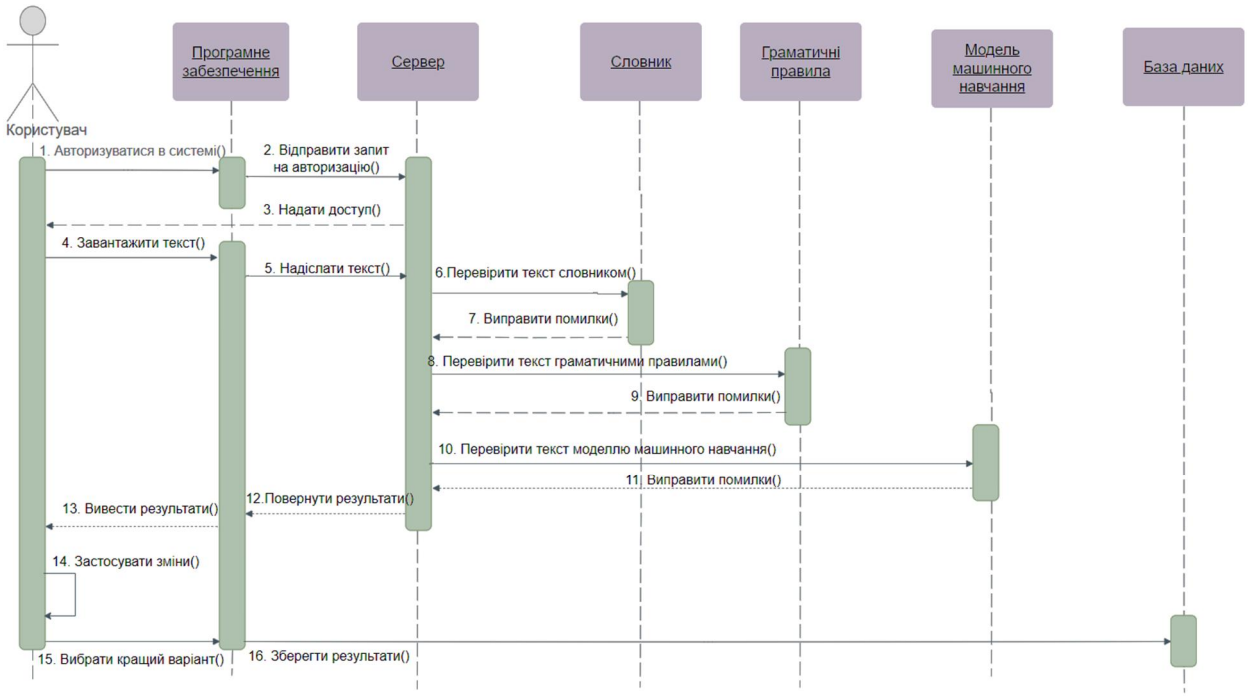


Рис. 18. Діаграма послідовності СППР для виправлення помилок в текстах українською мовою

Завершальною дією є побудова діаграми станів (рис. 19). З рис. 19 можна бачити, які стани існують у системі та в яких станах ця сама система перебуває залежно від певних подій. Взаємодія користувача із системою починається з авторизації системи, яка є її першим станом. Користувач вводить свої дані в систему і чекає відповідь. Далі йде умова: “Чи має вказаний користувач доступ до системи та її функціоналу?”. Якщо цей користувач зареєстрований у системі, то доступ надається. якщо ж ні, то система завершується для цього користувача. Після надання доступу до системи користувач може завантажувати текст у систему, що є другим станом. Варто відзначити, що цей стан є загальним поняттям для завантаження тексту з файлу вручну і з будь-якого сайту. На цьому етапі система виконує багато загальних перевірок тексту, а саме: перевіряє, чи наданий текст є українським, чи текст не пустий, чи підлягає перевірці та аналізу. Коли текст пройшов базову перевірку та був успішно завантажений у систему, програма починає процес перевірки тексту на наявність граматичних помилок. Спочатку програма спробує перевірити текст за словником. Це перший етап перевірки на наявність граматичних помилок. Цей етап є початковим, оскільки вже, починаючи з цього моменту, можна виявити певний вид помилки та заощадити час. Наступним етапом перевірки тексту та стану системи є “Перевірка тексту за правилами граматики”. Тут необхідні правила зчитуються з бази даних і система намагається застосувати їх до тексту. Останнім етапом перевірки тексту є “Перевірка тексту за моделлю машинного навчання”. Варто за-

значити, що для виправлення помилок не обов'язково будуть застосовані всі три методи (етапи) перевірки. Деякі можуть бути пропущені через відсутні слова чи правила. Тепер залишилося тільки застосувати знайдені рішення до тексту. Цей стан називається "Застосування корекції". Результати виправлення тексту відображаються користувачеві системи у вигляді статистики, знайдених помилок, запропонованих виправлень і рекомендацій щодо покращення тексту. Результати роботи програмного забезпечення, необхідні для подальшого вдосконалення роботи системи, зберігаються в базі даних. Це останній і остаточний стан системи виправлення помилок для україномовного текстового контенту. Якщо користувач хоче проаналізувати інший текст, система знову проходить усі вищезазначені стани.

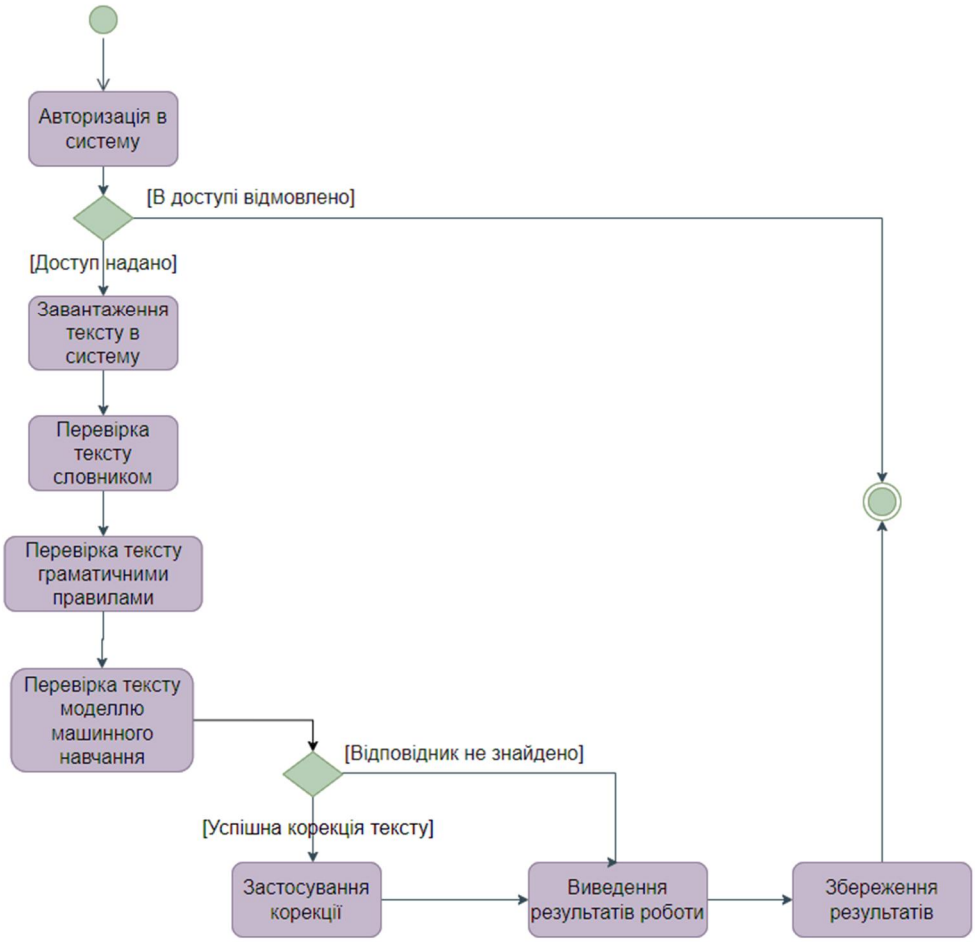


Рис. 19. Діаграма станів СППР для виправлення помилок у тексті українською мовою

Для досліджень обрано набір даних UA-GEC [31], який містить 850 і 160 текстів у навчальній і тестовій вибірці відповідно. Набір даних містить усього 20 715 речень (як з помилками, так і без). На момент написання роботи вбудовані методи ітерації по корпусу UA-GEC дають змогу лише отримувати повні документи, а не окремі речення. Текст анотований за форматом I {like=>likes:::error_type=Grammar} turtles. Для опрацювання та розділення текстів застосована бібліотека Regex. Правильні пари не видаляються, однак для подальшого опрацювання обрані речення, де кількість токенів є більшою за 4 (два з них – спеціальні токени початку і кінця речення).

Усього в результаті розподілу текстів за вищезазначеними виразами отримуємо 15 599 і 2205 речень у навчальній та тестовій вибірках відповідно. Нейронна мережа навчалася протягом 15 епох з такими гіперпараметрами: n_epochs = 15, batch_size = 8 та lr = 0.00001. Після 15-ї епохи значення

функції втрат на тестовій вибірці збільшується (рис. 22), тому 15 епох є найбільш оптимальним значенням у цьому разі. Нейронну мережу досліджували протягом 10-ти епох, однак мінімальне значення функції втрат у тестовій вибірці досягнуто під час другої епохи.

```
sorted(tokenizer.vocab.items(), key=lambda x: x[1])
```

```
[('<s>', 0),
 ('<pad>', 1),
 ('</s>', 2),
 ('<unk>', 3),
 ('<mask>', 4),
 ('!', 5),
 ('"', 6),
 ('#', 7),
 ('$', 8),
 ('%', 9),
 ('&', 10),
 ("'", 11),
 ('(', 12),
 (')', 13),
 ('*', 14),
 ('+', 15),
```

Рис. 20. Словник токенів “Ukrainian Roberta”

```
[15738 rows x 5 columns]
```

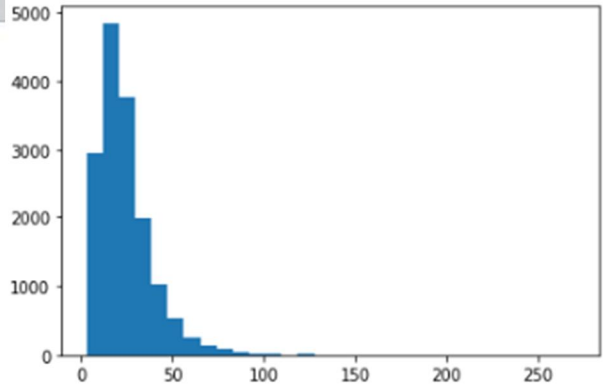


Рис. 21. Розподіл кількості токенів у навчальній вибірці

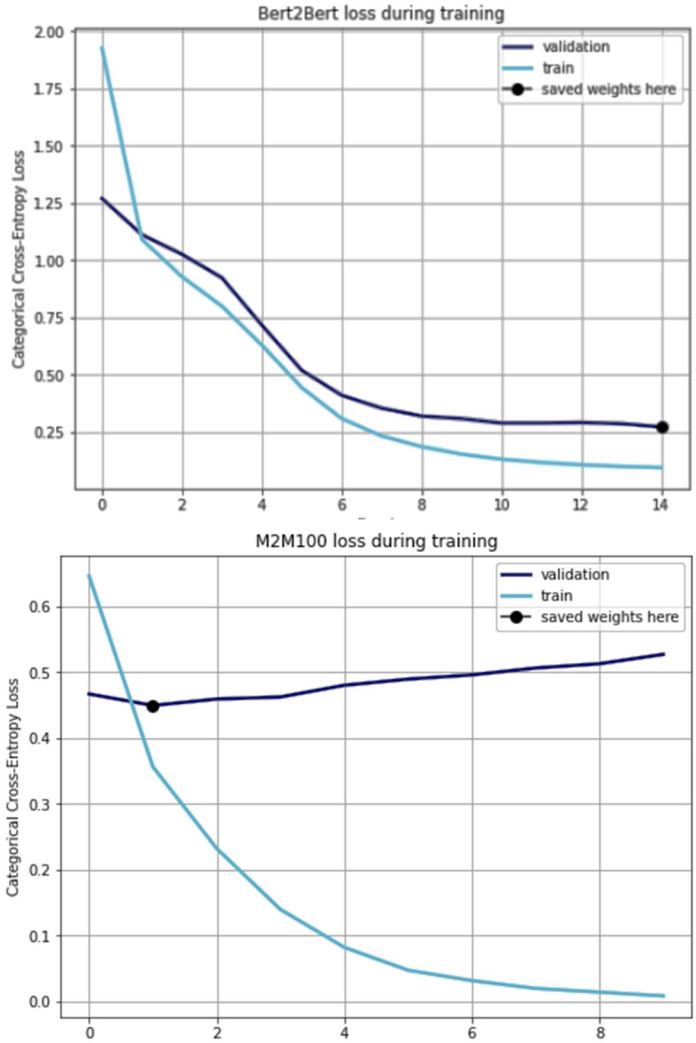


Рис. 22. Крива функції втрат під час навчання та тестування нейронної мережі

Враховуючи обмеження доступних обчислювальних ресурсів, найдовше речення має становити 20 токенів, розмір пакету – 2 ноти. Оскільки нейронна мережа попередньо навчена на різномовних текстах (як можливо присутній суржик, так і слова / словосполучення та навіть речення іншою мовою), відповідний словник лексем містить не цілі слова, а частини українських слів. Тому одне слово кодується більшою кількістю лексем, що через обмежену довжину речення призводить до некоректного представлення навчальних прикладів та їхнього змісту. Максимальна кількість токенів обмежена обчислювальними ресурсами. Також можлива некоректне подання навчальних прикладів.

Висновки

У статті розглянуто проблему ідентифікації та корекції помилок (GEC) у текстах українською мовою, її актуальність та складності, а також сучасні підходи до вирішення цього завдання. Проведено аналіз наявних корпусів текстів українською мовою, інструментів опрацювання природної мови (NLP) та попередньо навчених моделей машинного навчання для текстів українською мовою. Виявлено основні труднощі, пов'язані з граматичною та синтаксичною різноманітністю української мови, відсутністю великих анотованих корпусів текстів та нестачею ресурсів для навчання моделей. Описано різні підходи до GEC, зокрема методи на основі правил, синтаксичні та статистичні методи, а також методи машинного навчання. Було виявлено, що найефективнішими є методи на основі машинного навчання, зокрема ті, що використовують моделі Transformer. Проведено огляд доступних корпусів та інструментів для опрацювання текстів українською мовою. Визначено, що їхня кількість та якість є обмеженими порівняно з ресурсами англійською мовою. Обговорено наявні попередньо навчені моделі для текстів українською мовою та їхні модифікації, які можуть бути адаптовані для завдань GEC. Наприкінці роботи запропоновано архітектуру та описано основні функції системи підтримки прийняття рішень для автоматичної корекції помилок в україномовних текстах. Наведено UML-діаграми, що ілюструють ключові аспекти системи, як-от: варіанти використання, послідовність дій, кооперація та стани.

Подальші дослідження у сфері GEC для текстів українською мовою є надзвичайно важливими, оскільки автоматична корекція помилок сприятиме підвищенню якості текстів, що пишуть українською мовою, та забезпечить їхню відповідність граматичним і стилістичним нормам. Дослідження в цій сфері сприятимуть розвитку технологій опрацювання природної мови для текстів українською мовою, що своєю чергою позитивно вплине на розвиток інших технологій та застосунків. Подальший розвиток архітектури нейронних мереж може покращити якість корекції. Використання більш потужних та глибинних моделей може допомогти покращити ситуацію. Моделі повинні краще враховувати синтаксичні та семантичні зв'язки між словами. Наразі усі моделі машинного навчання використовують перенавчені моделі, які не були створені з урахуванням морфологічної складності української мови. Саме тому покращення систем GEC-напряму залежить від збільшення кількості досліджень, спрямованих на вивчення особливостей української мови. Врахування відмінювання, словотвору та інших особливостей мови є важливим.

Підсумовуючи, необхідно сконцентрувати увагу та зусилля на створенні спеціалізованої моделі, яка враховуватиме всі нюанси української мови. Крім того, необхідно забезпечити набагато більший ручний корпус "золотого" стандарту виключно для задач GEC. Також необхідно продовжити дослідження з навчанням моделей на синтетично створених наборах даних.

Подяка

Наведена стаття підготовлена завдяки грантовій підтримці Національного Фонду Досліджень України, реєстраційний номер проєкту 187/0012 від 1/08/2024 (2023.04/0012) "Розроблення інформаційної системи автоматичного виявлення джерел дезінформації та неавтентичної поведінки користувачів чатів" за конкурсом "Наука для зміцнення обороноздатності України".

REFERENCES

1. Bryant, C., Yuan, Z., Qorib, M. R., Cao, H., Ng, H. T., Briscoe, T. (2023). Grammatical Error Correction: A Survey of the State of the Art. *Computational Linguistics, 49*(3), 643–701. DOI: 10.48550/arXiv.2211.05166.

2. Smith, O. B., Ilori, J. O., Onesirosan, P. (1984). The proximate composition and nutritive value of the winged bean Psophocarpus tetragonolobus (L.) DC for broilers. Anim. Feed Sci. *Technol., 11*, 231–237

3. Chomsky, N. (1961). *On the notion "rule of grammar"*, 155–210, USA: American Mathematical Society.

4. Naghshnejad, M., Joshi, T., Nair, V. N. (2020) *Recent Trends in the Use of Deep Learning Models for Grammar Error Handling*, arXiv:2009.02358.

5. Brockett, C., Dolan, W. B., Gamon, M. (2006). Correcting ESL Errors Using Phrasal SMT Techniques. *Association for Computational Linguistics,* Proceedings of the 21st Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics, 249–256. DOI: 10.3115/1220175.1220207

6. Yoshimoto, I., Kose, T., Mitsuzawa, K., Sakaguchi, K., Mizumoto, T., Hayashibe, Y., Komachi, M., Matsumoto, Y. (2013). NAIST at 2013 CoNLL Grammatical Error Correction Shared Task. *Association for Computational Linguistics,* 26–33. https://aclanthology.org/W13-3604

7. Felice, M., Yuan, Z., Andersen, E., Yannakoudakis, H., Kochmar, E. (2014). Grammatical error correction using hybrid systems and type filtering. *Association for Computational Linguistics,* 15–24. DOI:10.3115/v1/W14-1702

8. Junczys-Dowmunt, M., Grundkiewicz, R. (2014). The AMU System in the CoNLL-2014 Shared Task: Grammatical Error Correction by Data-Intensive and Feature-Rich Statistical Machine Translation. Association for Computational Linguistics, Proceedings of the Eighteenth Conference on Computational Natural Language Learning: *Shared Task,* 25–33. https://doi.org/10.3115/v1/W14-1703

9. Cho, K., Merriënboer, B. V., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., Bengio, Y. (2014). Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. *Association for Computational Linguistics,* 1724–1734. https://doi.org/10.3115/v1/D14-1179.

10. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., Polosukhin, I. (2017). *Attention Is All You Need.* https://doi.org/10.48550/arXiv.1706.03762

11. Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M. et al. (2020). Transformers: State-of-the-Art Natural Language Processing. *Association for Computational Linguistics,* 38–45. https://doi.org/10.18653/v1/2020.emnlp-demos.6

12. We believe that anyone can write beautifully and professionally. LanguageTool. https://languagetool.org/about

13. LanguageTool API NLP UK. Github. https://github.com/brown-uk/nlp_uk

14. Stanza – A Python NLP Package for Many Human Languages. Stan for DNLP. https://stanfordnlp.github.io/stanza

15. NLP-Cube. Github. https://github.com/adobe/NLP-Cube.

16. Pymorphy. Github. https://github.com/pymorphy2/pymorphy2

17. Tree_stem. Github. https://github.com/amakukha/stemmers_ukrainian

18. MT5: Multilingual T5. Github. https://github.com/google-research/multilingual-t5

19. Multilingual Machine Translation. https://github.com/facebookresearch/fairseq/tree/main/examples/m2m_100

20. MBART50. https://github.com/facebookresearch/fairseq/tree/main/examples/multilingual#mbart50-models

21. Ukrainian Roberta base model. Hugging Face. https://huggingface.co/youscan/ukr-roberta-base

22. Uk-punctcase model. Hugging Face. https://huggingface.co/ukr-models/uk-punctcase

23. Ukrainian model to restore punctuation and capitalization. https://huggingface.co/dchaplinsky/punctuation_uk_bert

24. XML Roberta Base Uk model. Hugging Face. https://huggingface.co/ukr-models/xlm-roberta-base-uk

25. Chaplynskyi, D. (2023). Introducing UberText 2.0: A Corpus of Modern Ukrainian at Scale. Association for Computational Linguistics, Proceedings of the Second Ukrainian Natural Language Processing Workshop (UNLP), 1–10. https://doi.org/10.18653/v1/2023.unlp-1.1

26. Abadji, J., Suarez, P. O., Romary, L., Sagot, B. (2022). Towards a Cleaner Document-Oriented Multilingual Crawled Corpus. European Language Resources Association, *Proceedings of the Thirteenth Language Resources and Evaluation Conference,* 4344–4355. https://aclanthology.org/2022.lrec-1.463

27. Darchuk, N. (2017). *Possibilities of semantic marking of the corpus of the Ukrainian language* (KUM). Digital Repository Dragomanov Ukrainian State University. https://enpuir.npu.edu.ua/handle/123456789/17838

28. Shvedova, M., et al. (2017–2022). *General Regionally Annotated Corpus of Ukrainian Language* (GRAC). Network for ukrainian studies jena. https://doi.org/10.48550/arXiv.1911.02116

29. *BRUK: Braunskyi korpus ukrainskoi movy*. Github. https://github.com/brown-uk/corpus

30. Kotsyba N., et al. (2018). *Laboratorija ukrajins'koji.* https://mova.institute/

31. UA-GEC. https://github.com/grammarly/ua-gec

32. Syvokon, O., Nahorna, O. (2021). *UA-GEC: Grammatical Error Correction and Fluency Corpus for the Ukrainian Language.* https://doi.org/10.48550/arXiv.2103.16997

33. Syvokon O., Nahorna O., Kuchmiichuk P. Osidach N. (2023). *UA-GEC: Grammatical Error Correction and Fluency Corpus for the Ukrainian Language.* Association for Computational Linguistics, Proceedings of the Second Ukrainian Natural Language Processing Workshop (UNLP), 96–102. https://doi.org/10.18653/v1/2023.unlp-1.12

34. Bondarenko, M., et. al. (2023). *Omparative Study of Models Trained on Synthetic Data for Ukrainian Grammatical Error Correction.* Association for Computational Linguistics, Proceedings of the Second Ukrainian Natural Language Processing Workshop (UNLP), 103–113. https://doi.org/10.18653/v1/2023.unlp-1.13

35. Romanyshyn M. (2023) *Proceedings of the Second Ukrainian Natural Language Processing Workshop* (UNLP). Association for Computational Linguistics. https://aclanthology.org/2023.unlp-1.pdf

36. Didenko, B., Sameliuk, A. (2023). *RedPenNet for Grammatical Error Correction: Outputs to Tokens, Attentions to Spans.* Association for Computational Linguistics, Proceedings of the Second Ukrainian Natural Language Processing Workshop (UNLP), 121–131. https://doi.org/10.18653/v1/2023.unlp-1.15

37. Gomez, F. P., Rozovskaya, A., Roth, D. (2023). *A Low-Resource Approach to the Grammatical Error Correction of Ukrainian.* Association for Computational Linguistics, Proceedings of the Second Ukrainian Natural Language Processing Workshop (UNLP), 114–120. https://doi.org/10.18653/v1/2023.unlp-1.14.

38. Vysotska, V. (2024). Linguistic intellectual analysis methods for Ukrainian textual content processing. CEUR Workshop Proceedings. https://ceur-ws.org/Vol-3722/paper25.pdf.

39. Vysotska, V. (2024). *Linguistic intellectual analysis methods for Ukrainian textual content processing.* CEUR Workshop Proceedings. https://ceur-ws.org/Vol-3722/paper18.pdf.

40. Vysotska, V., Holoshchuk, S., Holoshchuk, R. (2021). *A Comparative Analysis for English and Ukrainian Texts Processing Based on Semantics and Syntax Approach.* https://ceur-ws.org/Vol-2870/paper26.pdf.

41. Vysotska, V. (2024). Computer Linguistic Systems Design and Development Features for Ukrainian Language Content Processing. In *COLINS* (3) (pp. 229–271). https://ceur-ws.org/Vol-3688/paper18.pdf.

42. Kholodna, N., et.al. (2022, November). Machine Learning Model for Paraphrases Detection Based on Text Content Pair Binary Classification. In MoMLeT+ DS (pp. 283–306). https://ceur-ws.org/Vol-3312/paper23.pdf

43. Lytvyn, V., et. al. (2023). Identification and Correction of Grammatical Errors in Ukrainian Texts Based on Machine Learning Technology. *Mathematics, 11*(4), 904. DOI: 10.3390/math11040904

44. Kholodna, N., et. al. (2021). *A Machine Learning Model for Automatic Emotion Detection from Speech.* In MoMLeT+ DS (pp. 699–713). https://ceur-ws.org/Vol-2917/paper42.pdf.

45. Kholodna, N., et. al. (2023). Technology for grammatical errors correction in Ukrainian text content based on machine learning methods. *Radio Electronics, Computer Science, Control, 1,* 114. DOI: 10.15588/1607-3274-2023-1-12

INFORMATION TECHNOLOGIES FOR SOLVING THE PROBLEM OF CORRECTING ERRORS IN UKRAINIAN-LANGUAGE TEXTS

Rostyslav Fedchuk[1], **Victoria Vysotska**[2]

[1, 2] Lviv Polytechnic National University,
Information Systems and Networks Department, Lviv, Ukraine
[1] E-mail: rostyslav.b.fedchuk@lpnu.ua, ORCID: 0009-0002-6669-0369
[2] E-mail: victoria.a.vysotska@lpnu.ua, ORCID: 0000-0001-6417-3689

This article is dedicated to the study and analysis of grammatical error correction (GEC) tasks in Ukrainian language texts, which is a significant issue in the field of natural language processing (NLP). The paper addresses the specific challenges faced by automatic error correction systems due to the peculiarities of the Ukrainian language, such as its morphological complexity and contextuality. Examples of typical errors are provided, and the reasons why existing GEC methods often prove insufficient for Ukrainian are analysed. The literature review covers recent research and publications in the GEC field, particularly those related to other languages, and highlights approaches that can be adapted for Ukrainian. Special attention is given to the analysis of existing Ukrainian text corpora, such as the UA_GEC and others used for training machine learning models. Their volume, text types, specifications, advantages, and disadvantages are described. Tools for natural language processing that support Ukrainian, such as LanguageTool, NLP-uk, Stanza, NLP-Cube, pymorphy2, Tree_stam, are examined. Their functionalities, performance, and accuracy are analysed. Pre-trained machine learning models, including mBART50 and mT5 were adapted for Ukrainian with description of their effectiveness in GEC tasks. The article presents practical aspects of applying these models and corpora for automatic grammatical error correction in Ukrainian texts. The process of adapting models to the specifics of the Ukrainian language is detailed, practical case examples are provided, and results are analysed. A significant part of the paper is devoted to the description of the developed decision support system for correcting errors in Ukrainian language texts. The system's architecture, its main components, and processes are presented through UML diagrams. The input and output data are described, along with an analysis of the obtained results, demonstrating the effectiveness of the proposed solutions. The results of this study can be useful for NLP system developers, researchers in text processing, and educational institutions focused on improving the quality of written texts in Ukrainian.

Keywords: identification and correction of grammatical errors, error detection and correction, machine learning models, text corpora, natural language processing (NLP), linguistic tools, Ukrainian language.