УДК 004.9

ТЕХНОЛОГІЯ ВИПРАВЛЕННЯ ГРАМАТИЧНИХ ПОМИЛОК В УКРАЇНОМОВНОМУ ТЕКСТОВОМУ КОНТЕНТІ НА ОСНОВІ МЕТОДІВ МАШИННОГО НАВЧАННЯ

Холодна Н. М. – магістр кафедри «Інформаційні системи та мережі», Національний університет «Львівська політехніка», Львів, Україна.

Висоцька В. А. – канд. техн. наук, доцент, доцент кафедри «Інформаційні системи та мережі», Національний університет «Львівська політехніка», Львів, Україна.

АНОТАЦІЯ

Актуальність. Більшість досліджень у напрямі виправлення граматичних та стилістичних помилок зосереджені на корекції помилок в англомовному текстовому контенті. Завдяки наявності великих наборів даних досягнуто суттєвого підвищення точності корекції граматики англійської мови. На жаль, досліджень інших мов мало. Системи в для англійської мови постійно розвиваються і наразі активно використовують методи машинного навчання: класифікацію (sequence tagging) та машинний переклад. Для створення якісної моделі машинного навчання для корекції граматичних/стилістичних помилок у текстах тих мов, які є складними морфологічно, необхідна велика кількість паралельних або вручну розмічених даних. Ручна анотація даних вимагає багато зусиль професійних лінгвістів, що робить створення корпусів текстів, особливо морфологічно багатих мов, зокрема, української, часо- та ресурсозатратним процесом.

Мета – є розроблення технології виправлення помилок в українськомовних текстах на основі методів машинного навчання з використанням невеликого набору анотованих паралельних даних.

Метод. Для даного дослідження при розробці системи корекції помилок в україномовних текстах із застосуванням оптимального конвеєру (pipeline), що включає в себе попереднє опрацювання текстового контенту, вибір та генерування ознак, обрані алгоритми машинного навчання, в умовах наявності невеликих за обсягом корпусів анотованих даних. Застосування нейронних мереж з новою архітектурою, огляд state-of-the-art методів та порівняння різних етапів конвеєру дасть змогу визначити таку їх комбінацію, яка дозволить отримати якісну модель корекції помилок в україномовних текстах.

Результати. Розроблено модель машинного навчання для корекції помилок в україномовних текстах. Запропоновано універсальну схему розробки системи корекції помилок для різних мов. Відповідно до отриманих результатів, нейронна мережа має здатність виправляти прості речення, написані українською, однак розроблення повноцінної системи вимагатиме застосування перевірки орфографії за допомогою словників і перевірки правил, як простих, так і заснованих на результаті парсингу залежностей або інших ознак. З-поміж трьох моделей, найкращі показники має попередньо навчена модель нейронного перекладу mT5. З метою економії обчислювальних ресурсів можливим також є застосування попередньо навченої нейронної мережі типу BERT, використовуючи її як у якості енкодера, так і декодера. Така нейронна мережа має вдвічі менше параметрів, ніж інші попередньо навчені моделі машинного перекладу, і показує задовільні результати при виправленні граматичних та стилістичних помилок.

Висновки. Створена модель показує відмінні результати класифікації на тестових даних. Розраховані метрики якості машинного перекладу дають змогу лише частково порівняти моделі, оскільки більшість слів і словосполучень у початковому та виправленому реченні співпадають. Найкраще значення як BLEU (0.908) , так і METEOR (0.956) отримано для mT5, що співпадає із аналізом прикладів, у якому найбільш точні виправлення помилок без зміни початкового значення речення отримані для такої нейронної мережі. M2M100 має більшу оцінку BLEU (0.847), ніж “Ukrainian Roberta” Encoder-Decoder (0.697), однак, суб’єктивно оцінюючи результати виправлення прикладів, M2M100 значно гірше справляється із подібним завданням, ніж дві інші моделі. Для METEOR також M2M100 (0.925) має більшу оцінку, ніж “Ukrainian Roberta” Encoder-Decoder (0.876).

КЛЮЧОВІ СЛОВА: NLP, text pre-processing, корекція помилок, виправлення граматичних помилок, машинне навчання, глибинне навчання, аналіз тексту, класифікація тексту, нейронна мережа.

АБРЕВІАТУРА

БД – база даних;
ІС – інтелектуальна система;
ІТ – інформаційна технологія;
ПЗ – програмне забезпечення;
ПО – предметна область;
BERT – bidirectional encoder representations from transformers;
GEC – grammatical error correction;
GECS – grammatical error correction system;
LSTM – long short-term memory;
MT – machine translation;
ML – machine learning;
NLP – natural language processing;
NN – neural network;
TPP – text pre-processing.

НОМЕНКЛАТУРА

S – система граматичної корекції;
I – множина вхідних даних;
O – множина вихідних даних;
R – основні правила опрацювання потоку вхідних даних в ІС граматичної корекції;
U – параметри опрацювання вхідних даних;
N – нейронна мережа;
α – оператор скачування вхідних даних;
β – оператор опрацювання вхідних даних;
γ – оператор збереження вхідних даних;

© Холодна Н. М., Висоцька В. А., 2023
DOI 10.15588/1607-3274-2023-1-12

μ – TPP-оператор;

χ – оператор пошуку помилок;

ω – оператор машинного навчання ІС на достовірних текстових даних;

λ – оператор граматичної корекції тексту;

i_1 – множина даних ідентифікації;

i_2 – множина вхідного текстового контенту;

i_3 – множина шаблонів/правил помилок;

i_4 – підтвердження правки від автора/користувача;

o_1 – маркований/тегований текст з помилками;

o_2 – колекція пропозицій корекції тексту;

o_3 – множина підтверджених автором правок;

r_1 – правила алгоритму взаємодії;

r_2 – NLP-правила;

r_3 – правила алгоритму нейронної мережі;

r_4 – правила алгоритму корекції помилок;

u_1 – множина рівнів доступу;

u_2 – множина вимог доступу;

u_3 – множина NLP-вимог;

u_4 – множина метрик машинного навчання;

u_5 – множина вимог корекції помилок.

ВСТУП

GEC – задача ідентифікації та усунення помилок у вхідному тексті. GEC застосовують в різних сферах, включаючи виправлення пошукових запитів і MT-результатів, TPP, перевірку правопису в браузерах і текстових процесорах тощо. GEC-методи поділяють на категорії: методи, засновані на правилах; методи, засновані на синтаксичному аналізі речень; статистичне моделювання; класичні ML-методи; MT на основі глибинного навчання.

Перевірка на основі правил використовує набір попередньо визначених шаблонів помилок для відповідності тексту. Всі правила розробляють зазвичай вручну. Текст є помилковим, якщо відповідає одному з правил [1]. Переваги підходу: швидкодія, інтерпретація результатів, можливості ітеративного розвитку ІС. Однак метод має недоліки: складність ІС збільшується в міру появи різних типів помилок, створення правил є ресурсозатратним і вимагає експертних знань з ПО, зокрема лінгвістики. Одночасно для покриття усіх можливих випадків необхідна величезна кількість правил.

При перевірці на основі синтаксису повністю аналізується морфологія та синтаксис тексту. Для цього потрібна лексична БД, морфологічний і синтаксичний аналізатори (парсери). Залежно від граматики мови, синтаксичний парсер визначає синтаксичну структуру кожного речення у вигляді дерева. Якщо повний аналіз не був успішним, тоді текст є помилковим [1]. Недоліком синтаксичного підходу є необхідність розробки додаткових правил для уточнення необхідних виправлень. Ці правила мають покривати усі можливі варіанти помилок.

Для автоматичного отримання правил із великої кількості тексту використовують статистичні моделі, які навчаються на великій кількості речень і можуть призначати ймовірність новій послідовності слів на основі кількості спостережуваних сполучень слів у навчальному корпусі. Поширені та більш вірогідні послідовності, які часто зустрічаються в корпусі, вважаюьб правильними, тоді як рідкі послідовності можуть містити помилки [2]. Переваги: автоматичне створення правил та відсутність необхідності застосування експертних знань для опрацювання даних або створення ознак. Недоліки: складна інтерпретація результатів, залежність точності моделі від якості навчального набору даних, необхідність застосування великого за обсягом набору даних.

При класифікації модель навчається передбачувати виправлення для кожного слова/тегу, що позначають дію над певним токеном, яку потрібно виконати для виправлення вхідного речення (sequence labelling). Для класифікації використовують складні системи із рекурентними NN або трансформерами, а також класичні ML-методи: наївний Байєсів класифікатор, метод опорних векторів, випадковий ліс тощо. Останні вимагають ручного створення ознак, як-от POS-теги слів у реченні, парсинг залежностей, відмінки слів, головних і другорядних членів речення тощо. Окрім необхідності застосування експертних знань при побудові моделі, головним недоліком є припущення про незалежність помилок у реченні або контекст слова не містить помилок. Цей метод не дає змоги одночасно виправити кілька співзалежних помилок.

Глибинне навчання архітектури рекурентних NN використовують для багатьох NLP-завдань. На відміну від методу класифікації, моделі глибокого навчання не вимагають розробки ознак, оскільки NN можуть досліджувати їх автоматично. Це є великою перевагою, оскільки генерація ознак вимагає експертних знань з лінгвістики. Особливо популярною варіацією рекурентних NN є LSTM. Рекурентні NN використовують для передбачення тегу, що позначає необхідну дію над токеном для виправлення речення. Дана NN-архітектура використовується у GEC-задачі для MT тексту, що містить помилки, у його правильний варіант. Для створення кращої MT-моделі використовують велику кількість пар речень (паралельних корпусів), точність системи у цьому випадку буде залежати від якості набору даних. Окрім того, рекурентні NN сприймають токени послідовно, що сповільнює час навчання та передбачення, унеможливлює паралельне опрацювання даних. У випадку комерційного застосування ІС час очікування є критичним, тому все більше досліджень наразі спрямовані на застосування іншої NN-архітектури, що називається трансформер.

Трансформер – модель глибинного навчання, яка замінює механізм рекурентності на механізм уваги, що забезпечує контекст для будь-якого положення токену у вхідній послідовності. Ця властивість надає змогу розпаралелювати набагато більше процесів у порівнянні з рекурентними нейронними мережами, і відтак знижує тривалість навчання [3]. Трансформери

© Холодна Н. М., Висоцька В. А., 2023
DOI 10.15588/1607-3274-2023-1-12
OPEN ACCESS
CC BY SA

швидко стали домінуючою архітектурою для NLP [4], випереджаючи такі альтернативи, як згорткові та рекуррентні NN, особливо для завдань розуміння мови (класифікація, переказ і узагальнення тексту, МТ) та її генерації. Архітектура масштабується відповідно до навчальних даних і фіксує особливості тексту на великій відстані. Попереднє навчання моделі дозволяє навчати трансформери на великих відкритих неанотованих корпусах і згодом легко налаштовувати їх до конкретних завдань, отримуючи в результаті високу якість системи.

Підвид трансформерів BERT призначений для попереднього навчання глибоких двонаправлених представлень з неанотованого набору даних. В результаті попередньо навчена модель BERT може бути налаштована лише одним додатковим вихідним шаром для різних NLP-задач без істотних модифікацій внутрішньої архітектури [5].

Метою дослідження є проектування та створення GEC-системи в текстах українською мовою за допомогою ML-методів з використанням невеликого набору анотованих паралельних даних. До задач, які необхідно вирішити для досягнення поставленої мети, належать:

– опис функціональності та вимог проектованої ІС;

– порівняння state-of-the-art методів для GEC;

– проектування і застосування нейронних мереж з різною архітектурою;

– вибір найбільш оптимальної моделі у ТРР-контексті, векторного вкладення або векторизації, вибору та генерування ознак, ML-алгоритму та його параметрів.

Об’єкт дослідження – процеси ідентифікації та корекції граматичних та стилістичних помилок в україномовному текстовому контенті. Предмет дослідження – методи та засоби виправлення граматичних/стилістичних помилок в україномовних 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 україномовних текстах.

1 ПОСТАНОВКА ПРОБЛЕМИ

GEC-систему S подано кортежем:

$$S = < I, O, R, U, N, \alpha, \beta, \gamma >,$$

де $I = \{i_1, i_2, i_3, i_4\}$, $O = \{o_1, o_2, o_3\}$, $R = \{r_1, r_2, r_3, r_4\}$, $U = \{u_1, u_2, u_3, u_4, u_5\}$.

Основними процесами ІС граматичної корекції є «ТРР», «Пошук помилок», «Машине навчання» та «Виправлення граматичних помилок». ТРР-процес ІС граматичної корекції опишемо суперпозицією:

$$C_{AU} = \mu \circ \beta \circ \alpha,\ C_{AU} = \mu(\beta(\alpha(i_1, i_2, i_4), r_1, u_1), u_2).$$

Процес «Пошук помилок» ІС граматичної корекції опишемо суперпозицією: $C_{CU} = \chi \circ \beta \circ \alpha$, тобто

$$C_{CU} = \chi(\beta(\alpha(C_{AU}, i_2, i_3, i_4), r_1, u_3), r_2).$$

Процес машинного навчання на достовірних даних ІС граматичної корекції опишемо суперпозицією:

$$C_{UL} = \omega \circ \gamma \circ \beta \circ \alpha,\ C_{UL} = \omega(\gamma(\beta(\alpha(C_{CU}, i_2), i_3), u_4), r_3).$$

Процес «Виправлення граматичних помилок» ІС граматичної корекції на основі GEC-методів опишемо суперпозицією:

$$C_{US} = \lambda \circ \gamma \circ \beta \circ \alpha,\ C_{US} = \lambda(\gamma(\beta(\alpha(C_{US}, i_2), i_4), u_5), r_4).$$

GEC-методи, засновані на правилах, синтаксичному аналізі або статистичному моделюванні, описані у дослідженнях, що спрямовані на побудову системи перевірки та виправлення текстів, написаних данською [7], грецькою [8], латвійською [9], слов’янської [10], пенджабі [11], філіппінською [12] та арабською [13] мовами. Системи GEC для англійської постійно розвиваються і наразі активно використовують ML-методи: класифікацію (sequence tagging) та МТ. Для створення якісної ML-моделі для GEC у текстах тих мов, які є складними морфологічно, необхідна велика кількість паралельних або вручну розмічених даних. Проблему отримання анотованих даних без ручного маркування частково вирішують за допомогою алгоритмів, що змінюють початковий текст, додаючи у нього «шум» (noise injection). Окрім того, використовують зворотний переклад безпомилкових текстів для того, щоб отримати їх неграматичні відповідники. Таким чином автоматично згенеровують певну кількість розмічених корпусів паралельних текстів [14].

2 АНАЛІЗ ЛІТЕРАТУРНИХ ДЖЕРЕЛ

Основними аналогами розроблюваної ІС є два додатки: Grammarly та LanguageTool.

Grammarly – комерційна онлайн-платформа компанії Grammarly Inc., яка перевіряє й виправляє не тільки граматичні помилки, а й пропонує рекомендації щодо чіткості (стислість та зрозумілість), захопливості (словниковий запас та розмаїття) та тону повідомлення (формальність, ввічливість і впевненість) [16–17]. Для покращення перевірки і точності пропозицій платформа використовує ML-алгоритми і методи глибинного навчання [18]. 30 мільйонів людей і 30 тисяч команд використовують Grammarly щодня [18]. Платформа потрапила до рейтингу Time «Time100 Most Influential Companies of 2022» та FastCompany «The 10 most innovative companies in artificial intelligence of 2022», також до списку Forbes «Cloud 100» та The Software

© Холодна Н. М., Висоцька В. А., 2023
DOI 10.15588/1607-3274-2023-1-12

Report «Top 100 Software Companies» [18]. Наразі Grammarly підтримує лише англійську мови та її діалекти: американський, британський, канадський, австралійський [19].

LanguageTool – GEC-програма із відкритим кодом, що у своїй основі використовує перевірку на основі правил. Усього система більше 30 мов, зокрема і українську [20]. LanguageTool розвивається з 2003 року і наразі налічує 5 415 правил для англійської мови, 1 022 – для української [21]. Щодня користувачі перевіряють більше 20 мільйонів текстів за допомогою цієї платформи [22]. Для перевірки правильності написання україномовних текстів доступні правила таких категорій: варваризми (наприклад, приймати участь – брати), великі літери, граматика, логічні помилки (наприклад, неправильна дата), орфографія, оформлення, пунктуація, стиль, типографія [21]. Недоліком LanguageTool для перевірки україномовних текстів є невелика навіть у порівнянні з англійською мовою кількість правил, які не можуть покрити усі можливі граматичні помилки.

Sai Muralidhar Jayanthi, Danish Pruthi, Graham Neubig розробили систему NeuSpell [23] для виправлення орфографічних помилок англомовних текстів, що заснована на методах глибинного навчання. Система складається з 10 різних нейронних мереж, які дають змогу фіксувати контекст навколо орфографічних помилок. Для навчання нейронних мереж використані синтетичні навчальні дані, створені за допомогою кількох стратегій додання «шуму». NeuSpell є програмою із відкритим кодом і наразі підтримує лише англійську мову.

Hunspell [24] – система для виправлення орфографічних помилок, створена для мов з багатою морфологією, складеними словами і різним кодуваннями символів. Використовує спеціальний формат словника, який визначає, які стеми та афікси є дійсними в певній мові. Система забезпечує токенізацію, стемінг і перевірку орфографії для майже будь-якої мови чи алфавіту.

В [6] запропоновано підхід перевірки граматичної правильності текстів на основі «мінімального навчання з вчителем» (minimal supervision). Даний метод полягає у застосуванні невеликого анотованого набору даних і додання штучних помилок до корпусу новин, художньої літератури та інших жанрів (усього 18 мільйонів слів). Дослідники розробили класифікатори для кількох поширених типів граматичних помилок: прийменники, відмінок іменника, форма дієслова та узгодження з дієсловом (розподіл на число та рід). Окрім того, вони застосували метод нейронного МТ тексту, що містить помилки, у його правильний варіант. Alla Rozovskaya та Dan Roth зауважили, що МТ-точність є занадто низькою у зв'язку з недостатньою кількістю навчальних даних. Запропонований метод «мінімального навчання з вчителем» збільшує як точність класифікаторів, так і МТ-якість. За допомогою експериментів показали, що МТ-метод показує незадовільні метрики якості (F-score = 10.6) з використанням набору даних відповідних текстів, що містить приблизно 200 тис. вручну анотованих і виправлених слів. Цю МТ-систему значно перевершує запропонований у [6] ML-підхід (різновидність навчання з учителем, яке використовує немарковані дані для тренування – зазвичай невелику кількість маркованих даних та велику кількість немаркованих).

Дослідники Oleksiy Syvokon, Olena Nahorna презентували набір даних – корпус текстів [15], що професійно анотований для GEC та вільного редагування українською мовою. Дослідники зібрали тексти з помилками (20 715 речень – 328 779 токенів) від різноманітних авторів, у тому числі носіїв мови. Дані охоплюють найрізноманітніші сфери письма, від текстових чатів і есе до офіційного письма. Професійні коректори виправляли та анотували корпус на помилки, пов'язані з вільним мовленням, граматикою, пунктуацією та орфографією. За [1, 15], цей корпус можна використовувати для розробки та оцінки систем GEC українською мовою і дослідження морфологічно багатих мов. За [1], найбільш очевидною проблемою, пов'язаною з GEC-системами, є потреба у високоякісних навчальних корпусах, що містять велику кількість навчальних прикладів.

В [25] розроблено морфологічний парсер pymorphy2 для української мови у форматі бібліотеки для мови програмування Python. pymorphy2 аналізує частину мови, число, відмінок, час, лему та стем заданого слова. Морфологічний парсер заснований на словниках OpenCorpora, що конвертовані до формату XML. Користувачі також мають змогу додавати власні слова та правила, це дає можливість проводити морфологічний аналіз текстів певної ПО без зміни вихідного коду pymorphy2 та адаптувати pymorphy2 для роботи з іншими мовами.

В [26] розроблено систему ТРР, морфологічного та синтаксичного аналізу україномовних текстів. Для токенізації, розподілу на речення та пошуку email-адрес дослідники використали бібліотеку NLTK мови програмування Python та регулярні вирази, для видалення стоп-слів та пошуку іменованих сутностей – відповідні словники. Для морфологічного аналізу слів, їх лематизації або стемінгу була використана бібліотека pymorphy2, що підтримує українську мову. Окрім того, дослідники реалізували графічний інтерфейс додатку за допомогою бібліотеки PyQt.

В [27] проаналізовано існуючі методи пошуку іменних груп в англомовних і україномовних текстах та розробили метод детектування іменних груп на основі дерева залежностей речення і моделі розпізнавання іменованих сутностей. Довели, що методи аналізу англомовного тексту не можуть бути використані для україномовних документів, адже вони створені з урахуванням особливостей структури побудови речень тільки в англомовних текстах.

В [28] адаптовано алгоритм стемінгу Портера для ТРР україномовних текстів. Окрім того, дослідники реалізували алгоритм на мові програмування PHP і

© Холодна Н. М., Висоцька В. А., 2023
DOI 10.15588/1607-3274-2023-1-12
OPEN ACCESS
CC BY SA

виклали його у вільний доступ. Довели, що кращого результату можна досягти через лематизацію, однак для цього потрібно провести додаткові дослідження.

В [29] розроблено GEC-систему в англомовних текстах за допомогою спеціальної моделі глибинного навчання – Transformer [3]. Виправлення помилок відбувається завдяки використанню даної моделі для задачі перекладу неправильного тексту у його відкоригований відповідник. Особливістю даної системи є вибір TPP-методів: автоматичне виправлення орфографічних помилок, токенізація [30], метод кодування ознак Byte Pair Encoding. Довели, що дана ІС має достатню точність (F-score = 60.93 на завданні CoNLL-2014 [31]) і швидкість (10 слів за секунду) для подальшого її впровадження і застосування на онлайн-платформах.

В [32] застосовано нейронну мережу архітектури encoder-decoder, що складається із шарів згортки та механізму уваги, для GEC у англомовних текстах за допомогою МТ. Для процесу декодування використані ансамблі однакових моделей, що ініціалізовані випадковим чином. Для фінального вибору речень з-поміж можливих кандидатів оцінки їх ймовірностей змінені за допомогою додатково створених ознак. ІС значно перевершила якість попередньо існуючих МТ-систем, окрім того і на основі рекурентних нейронних мереж.

В [33] розроблено GEC-систему у англомовних текстах, що заснована на підході анотації послідовностей (sequence tagging). Окрім стандартних тегів (keep, delete, append, replace), що позначають дію, яку необхідно виконати над токеном для виправлення речення, також запропоновані так звані g-transformations: зміна регістру першої літери, об'єднання або розділення токену, зміна числа іменника або форми дієслова. Для виправлень токенів застосований ітеративний підхід, у якості ML-методу – різні моделі сімейства Transformer. Такий підхід дозволив отримати найкращі на момент публікації статті оцінки якості системи та збільшення швидкості опрацювання даних у 10 разів у порівнянні із іншими ІС, заснованих на архітектурі Transformer.

В [34] описано GEC-систему, засновану на підході анотації послідовностей за допомогою класичного ML-алгоритму – наївного класифікатора Байєса. Усього використано п'ять моделей – відповідно до наявних у корпусі типів граматичних помилок. Попри те, що система була однією з найкращих на завданні CoNLL [35], аналіз результатів показує, що досягнення дуже високої точності при граматичній корекції вимагає більш складних NLP-методів.

В [9] розроблено засновану на правилах програму перевірки граматики для латвійської мови. Усього реалізовано дві групи правил: правила, що описують правильні речення, і правила, що описують граматичні помилки. На корпусі текстів, написаних людьми, що не є носіями мови, система досягла F1-score 62,4% і 40,2% на корпусі студентських робіт. Недолік: не перевіряє правильність слова в залежності від його контексту. Відповідно, правильно написані та узгоджені слова, що не належать реченню, будуть маркуватися як безпомилкові.

В [8] описано ІС для перевірки правильності текстів, написаних на грецькій мові. Система заснована на правилах та парсингу синтаксису речення. ІС аналізує текст користувача та надає виправлення, опис помилок, а також правила щодо стилю та семантичної інформації в тексті. Системне оцінювання проводилося як паралельне виправлення одних і тих самих текстів програмою та людиною. Система перевірки грецької граматики наблизилася до 90% виправлення людського. Однак ця ІС не може опрацювати всі можливі граматичні помилки.

В [36] розроблено універсальну ML-модель для GEC у текстах, написаних на різних мовах. Дослідники стверджують, що для отримання якісної багатомовної GEC-системи мають бути виконані такі два кроки: автоматична генерація достатньої кількості навчальних даних і використання ML-моделей сімейства Transformer із величезною кількістю параметрів (до 11 мільярдів). Таким чином, автори досягнули високої точності виправлення помилок для чотирьох мов: англійської, чеської, німецької.

В [37] досліджено здатність GEC-моделей до узагальнення граматичних правил для корекції нових помилок у тексті. ML-модель основі Transformer протестована з використанням невідомих їй прикладів. Отримано незадовільні результати навіть у випадку простих правил і зменшеного словнику, що може свідчити про те, що алгоритму бракує можливості узагальнення, необхідної для виправлення нових помилок у наданих тестових прикладах. Для більш якісного GEC за меншого обсягу навчального корпусу існуючі ІС на основі ML варто поєднати з окремою перевіркою певних правил.

В [38] запропоновано мовно-незалежну стратегію для розробки багатомовної GEC-системи. Для її імплементації необхідні лише попередньо навчена МТ-модель та корпус паралельних навчальних даних для перекладу з англійської на обрану мову. Перш за все, МТ-модель генерує нові синтетичні дані на обраній мові, що застосовують у якості помилкових вхідних текстів. Отриманий корпус використовують для попереднього навчання моделі, після чого її потрібно додатково налаштувати, використовуючи анотований набір даних для GEC у текстах обраної мови. Досить гарні показники точності виправлення досягнуті для німецької, китайської мов.

В [39] адаптовано фреймворк Break-It-Fix-It (BIFI), що оригінально використовувався для виправлення коду програм за умови відсутності ідеальних зразків, до завдання перевірки граматичної правильності речень. Дослідники використали попередньо навчену мовну модель для розробки бінарного класифікатора LM-Critic, який виконує завдання попереднього сортування речень і вказує, чи містить певний текст граматичні помилки. LM-Critic визначає речення граматично правильним, якщо йому відповідає

© Холодна Н. М., Висоцька В. А., 2023
DOI 10.15588/1607-3274-2023-1-12
OPEN ACCESS
CC BY SA

більший числовий показник ймовірності, ніж його неправильним «сусідам». Утворені пари правильних і неправильних речень використовують для навчання «коректора», що опрацьовує лише речення, позначені LM-Critic як ті, що містять граматичні помилки.

В [40] для вирішення проблеми необхідності застосування великих за обсягом корпусів навчальних даних згенерували помилкові версії великих неанотованих наборів текстів за допомогою запропонованої функції шуму. Отримані паралельні корпуси згодом використовують для попереднього навчання моделей на основі архітектури Transformer. Потім необхідно додатково налаштувати ML-модель відповідно до ПО та стилю датасету. Дана GEC-система ускладнена використанням нейронної перевірки орфографії, що сортує запропоновані варіанти виправлень в залежності від контексту.

В [41] змінили архітектуру ML-моделі Transformer, впровадивши новий механізм уваги, що заснований на дереві залежностей слів у реченні. Так як неправильно побудоване речення може спричинити помилку у парсингу дерева залежностей, дослідники також навчили NN виправляти графи, що містять помилки. Додатково застосувавши запропонований метод аугментації даних, отримали гарні показники GEC-точності навіть без попереднього навчання NN.

В [42] застосували нейронну модель sequence-to-sequence на рівні символів для того, щоб уникнути проблем зі словами OOV (out of vocabulary words, відсутні у словнику). У якості енкодера застосовують двонапрямлену рекурентну NN, декодер – також рекурентна нейронна мережа, поєднана із механізмом уваги. Декодер генерує вихідне речення посимвольно. Навіть попри те, що такий підхід допомагає нейронній мережі опрацьовувати невідомі їй слова, вона не може ефективно опрацьовувати інформацію на рівні слова: ця модель отримала оцінку F-score = 40,56 на тестовому наборі CoNLL [31].

В [43] застосували новий підхід до автоматичної генерації навчальних анотованих прикладів та GEC за допомогою генеративних змагальних мереж (GANs). Генеративні змагальні мережі складаються з генератора, що постійно генерує навчальні приклади, і дискримінатора, що класифікує отримані дані. У [43] генератором є NN Grammatical Error Labeler, що навчається додавати помилки, аналогічні помилкам у навчальній вибірці текстів, дискримінатор – NN Grammatical Error Detector, що навчається розпізнавати правильну мітку, що позначає дію, яку необхідно виконати над певним токеном.

В [44] розробили дизайн платформи для GEC і створили відповідну ML-модель, поєднану з перевіркою правил. Система складається з трьох модулів: виправлення помилок, адміністрування системи, фільтрування відгуків. GEC-Модуль включає TPP. На TPP-етапі також визначається валідність введених користувачем даних. Модуль фільтрування відгуків дозволяє користувачам скорегувати неправильні виправлення системи. Такий підхід дає змогу виправити можливі помилки, наявні у наборі даних, оскільки отримані відгуки користувачів можуть використовуватись для додаткового навчання нейронної мережі.

В [45] презентували корпус анотованих даних для створення GEC-системи та редагування текстів, написаних чеською мовою. Усього набір даних містить 42 210 речень. Окрім того, дослідники використали MT-підхід для GEC. Нейронна мережа архітектури Transformer попередньо навчена на додатково синтезованих з головного набору даних реченнях. Окрім чеської мови, автори статті застосували даний підхід для виправлення помилок в німецькомовних текстах.

В [46] запропонували та реалізували новий метод автоматичного створення навчальних даних із застосуванням двох різних за якістю моделей перекладу. «Погана» система перекладу є статистичною моделлю на рівні речень, якісна – навченою нейронною мережею. Даний метод дав змогу отримати 10 мільйонів паралельних речень для навчання нейронної мережі архітектури Transformer. Даний метод дасть змогу отримати великі датасети навчальних даних для інших мов, для яких великі корпуси анотованих даних не є доступними.

В [47] використали генеративні змагальні мережі для створення системи автоматичного виправлення помилок у англомовних текстах. На відміну від алгоритму в [43], в даному дослідженні генератор виправляє помилки, «перекладаючи» помилкові речення у правильні, а дискримінатор навчається оцінювати якість перекладу та наявність помилок у опрацьованому генератором реченні. Дискримінатор, що приймає на вхід два речення, заснований на сіамських рекурентних або згорткових NN.

3 МАТЕРІАЛИ ТА МЕТОДИ

Система перевірки граматичної правильності речень може використовуватись для перегляду запропонованих змін та їх пояснень, а також для автоматичного виправлення правильності речень.

Така система може використовуватись як і індивідуальним користувачем для перевірки власних текстів, так і сторонньою NLP-програмою для попереднього або фінального опрацювання речень.

Зовнішніми сутностями є: користувач, NLP-програма, адміністратор системи.

Зацікавлені особи прецеденту та їх вимоги:

– користувач створює обліковий запис, завантажує або відкриває документи, перевіряє граматичну правильність речень, застосовує запропоновані зміни;

– стороння NLP-програма відправляє текст на перевірку за допомогою API системи, отримує змінений текст або перелік можливих виправлень;

– адміністратор системи виконує її налаштування, навчання нейронної мережі, перевірку запропонованих користувачем виправлень.

© Холодна Н. М., Висоцька В. А., 2023
DOI 10.15588/1607-3274-2023-1-12

OPEN ACCESS
CC BY SA

Користувач ПС: 1) фізична особа, що використовує систему для перевірки власних текстів; 2) NLP-програма.

Передумови прецеденту:

– комп'ютер, за допомогою якого здійснюватиметься аналіз, підключений до Інтернету та має встановлене необхідне ПЗ;

– користувач має бути успішно авторизованим у системі;

– словник (якщо використовується перевірка орфографії) містить достатню кількість слів і (або) правил їх утворення, є доступним;

– реалізовані усі ключові модулі, що забезпечують основний функціонал системи;

– штучна нейронна мережа (якщо застосовуються ML-методи) є попередньо натренованою на виконання необхідного завдання і має достатньо високі показники якості.

Основний успішний сценарій:

– користувач завантажує програму або розширення, або відкриває онлайн-додаток у браузері;

– користувач завантажує або створює документ і заповнює його;

– система виконує перевірку тексту;

– система відображує необхідні виправлення;

– користувач підтверджує або скасовує застосування запропонованих виправлень;

– користувач зберігає документ. Зберігається також і історія змін.

Альтернативні потоки:

– Власний файл з даними не відповідає типу файлу, який може відкрити програма.

1. Програма повідомляє користувача про помилку і скасовує відкриття файлу.

2. Користувач обирає файл з правильним розширенням.

3. Програма відкриває файл з правильним розширенням, зчитує дані, проводить їх попередню обробку (точка повернення в основний сценарій).

– Помилка у роботі програми:

1. Користувач звертається до служби технічної підтримки (розробника) і повідомляє про помилку у програмі.

2. Розробник усуває помилку і надає нову версію або надає інформацію про способи її самостійного усунення (точка повернення).

Пост-умови:

– Користувач запропонував власний GEC-приклад у реченні у випадку отриманих незадовільних результатів;

– Запропоновані користувачем виправлення переглянуті адміністратором для подальшого налаштування системи або збережені для персоніфікації подальших виправлень;

– Дані, налаштування і файли користувача занесені до бази даних.

Спеціальні СВ:

– ІС повинна відповідати бажаному рівню точності, яка перевіряється спочатку на навчальних даних, потім – на тестових даних, у режимі перевірки власного файлу точність не перевіряється;

– система має підсвічувати неправильні слова та словосполучення, надавати обґрунтування запропонованих змін;

– для доступу до функціоналу системи сторонній NLP-додаток використовує її API-сервіс.

Список необхідних ІТ та додаткових пристроїв:

– користувач повинен мати комп'ютер із встановленим ПЗ та підключенням до мережі Інтернет для отримання даних;

– система використовує розподілену нереляційну базу даних для зберігання даних користувачів;

– для додаткового опрацювання текстів використовуються попередньо навчені ML-моделі.

Для побудови діаграми варіантів використання для системи автоматичного виправлення помилок україномовних текстів (рис. 1) спочатку необхідно визначити основних акторів: Користувач (User) і NLP-Програма (NLP Application), а також основні варіанти використання як Apply correction (застосувати корекцію), Check text (перевірити текст), Suggest correction (запропонуйте виправлення), Upload text (завантажити текст), Check text for all types of errors (перевірити текст на всі види помилок), View suggestions (переглянути пропозиції).

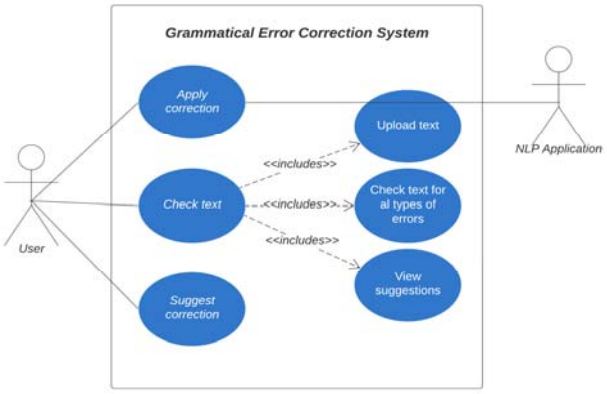


Рисунок 1 – Діаграма варіантів використання GECS

GECS – це десктопний / мобільний додаток, онлайн-платформу або модуль-доповнення до певного ПЗ, наприклад, браузеру або текстового редактору. ПЗ для аналізу природньої мови у свою чергу може використовувати API сервісу автоматичного виправлення речень, обмінюючись повідомленнями за допомогою архітектури REST взаємодії між додатками. GEC-система включає в себе також сервер опрацювання даних, оскільки за умови застосування ML-методів, у тому числі і мовних моделей на основі архітектури Transformer з мільйонами параметрів, додаток потребуватиме багато обчислювальних ресурсів для обробки текстів. Збільшення кількості користувачів, і, відповідно, збільшення обсягу даних також потребуватимуть додаткових комп'ютерних потужностей. До складу вищезазначеної ІС також входить сервер БД користувачів, що зберігає документи у обліковому

© Холодна Н. М., Висоцька В. А., 2023
DOI 10.15588/1607-3274-2023-1-12

записі. Взаємодія із серверами опрацювання та зберігання даних відбувається через графічний інтерфейс ІС, онлайн-платформи або доповнення. Альтернативним варіантом є встановлення запропонованого ПЗ на потужному комп'ютері, що має достатньо ресурсів для опрацювання даних локально. У іншому випадку, локальна версія ПЗ може мати спрощений функціонал. На рис. 2 подана розширена діаграма варіантів використання із уточненням функціоналу ІС з відповідними варіантами використання як Apply correction (застосувати корекцію), Check text (перевірити текст), Suggest correction (запропонуйте виправлення), Create a document (створити документ), Login/Signup (вхід/вихід), Upload text (завантажити текст), Check text for all types of errors (перевірити текст на всі види помилок), View suggestions (переглянути пропозиції), Upload a document (завантажити документ), Send request (відправляти запит), Get list of suggested corrected (отримати список запропонованих виправлень).

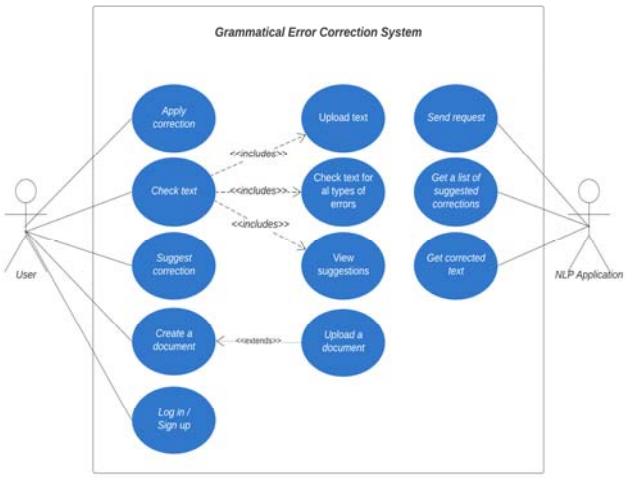


Рисунок 2 – Доповнена діаграма GECS

В залежності від виду взаємодії користувача і GECS не всі типи перевірок можуть бути доступними. Для зберігання та редагування особистих файлів користувач має бути зареєстрованим та авторизованим (варіант використання Log in / Sign up) у системі. Документи, логін, пароль та інші дані користувачів зберігаються на окремому сервері.

Варіант використання «Перевірити текст» (Check text) включає в себе три інших: Завантажити текст, Перевірити текст на всі типи помилок, Переглянути пропозиції. Завантаження тексту може відбуватись у кілька варіантів: вставлення свого тексту у textbox, відкриття файлу або створення нового документу. Створення (завантаження) документу виділено у окремий варіант використання, оскільки даний функціонал може бути реалізований лише для онлайн-платформи або локально встановленого додатку. До базових належать помилки, пов'язані з вільним мовленням, граматикою, пунктуацією та орфографією. В залежності від реалізованого функціоналу система може пропонувати заміни слів на синоніми для збільшення лексичної різноманітності тексту або заміни відповідно до певного стилю письма. Варіант використання Переглянути пропозиції позначає почерговий перегляд запропонованих виправлень разом із поясненням порушеного правила або причини заміни. Окрім того, частини тексту, що містять помилки, мають бути підсвічені відповідно до типу порушених правил. Користувач повинен мати змогу застосувати лише обрані виправлення, ті, які є необхідними у даному випадку. Якщо користувач вважає якість запропонованих виправлень незадовільною, він повинен мати змогу самостійно виправити текст і зберегти його як зразок для подальшого налаштування системи або навчання нейронної мережі. Таким чином, наступні виправлення системи матимуть кращу якість і будуть більш персоніфікованими для кожного окремого користувача. Варіант використання Застосувати виправлення дає змогу користувачеві автоматично виправити усі знайдені системою граматичні помилки без перегляду детальних пояснень порушених правил.

NLP-Додатки також можуть використовувати GEC-систему для попередньої або пост-обробки текстів. В цьому випадку додаток має відправити запит, використовуючи Application Programming Interface (API) сервісу, система – відправити результат аналізу або одразу виправлений текст. Отже, GEC може бути реалізоване як для індивідуального використання у вигляді застосунку, веб-сайту або розширення для додатків, так і як модуль у системі аналізу або генерації природньої мови.

Для побудови діаграми класів (рис. 3) у контексті інформаційної GEC-системи визначено вісім основних класів: GEC Software (Програма, що реалізує користувацький інтерфейс системи), User (Користувач), User Database (БД користувачів), Document (Документ), Data Processing Server (Сервер обробки даних), Check Result (Результат перевірки), Dictionary (Словник), ML model (ML-Модель), NLP application (NLP-Додаток).

Клас User (користувач) описує індивідуального користувача системи, який завантажує додаток, встановлює розширення або перевіряє тексти за допомогою онлайн-платформи. Цей клас містить такі атрибути як ім'я, електронна пошта і дані для авторизації у програмі – логін і пароль. Відповідно, він може авторизуватися у системі, завантажити і перевірити текст, створити і зберегти документ, застосувати виправлення, надане системою, або запропонувати власне. Також, у контексті інформаційної системи, користувач створює документ, що зберігає безпосередньо сам текст, а також мета-дані про історію внесених змін і виправлень, автора документу і дату створення. Клас User Database (БД користувача) може зберігати, записувати, оновлювати, повертати інформацію і документи, додати або верифікувати користувача.

© Холодна Н. М., Висоцька В. А., 2023
DOI 10.15588/1607-3274-2023-1-12

OPEN ACCESS
CC BY SA

Клас GEC Software (ПЗ GEC) постає посередником між користувачем, сервером обробки даних та базою даних. ПЗ може авторизувати користувача, запросити дані із сховища, відправити текст на аналіз, отримати результати та візуалізувати їх, застосувати запропоновані виправлення і зберегти результат, зберегти введені користувачем виправлення. Як вже зазначалось, альтернативним варіантом є використання локальних обчислювальних ресурсів без під'єднання до зовнішнього серверу. GEC Software також надає API сервісу для запитів на аналіз текстів та автоматичне виправлення помилок: до атрибутів цього класу належать тип запиту, адреса для запиту і тип даних, що відправляє сервер як відповідь. Даний ресурс може обробляти запити, автоматично застосовувати необхідні виправлення та відправляти правильний текст на сервер, з якого прийшов запит. Подібні сервіси визначають тип користувача за токеном.

Клас NLP Application (NLP-застосунок) позначає будь-яку програму для аналізу або генерації природньої мови, що використовує GEC-систему для попередньої або пост-обробки даних. Така програма має свою структуру і призначення, метод doItsMagic() відповідає за пряме її застосування. Ця програма може відправляти запити певного типу і обробляти пакет даних, отриманий від додатку GEC Software.

Клас Data Processing Server (сервер опрацювання даних) описує поведінку серверу обробки даних. Такі сервери мають свої технічні характеристики (кількість ядер процесора, об'єм операційної та постійної пам'яті, наявність графічного процесора тощо), які напряму пливають на швидкість обробки великого масиву даних.

Для коректного аналізу граматичної коректності речень сервер повинен спочатку виконати ТРР, після цього – представити записи як вектори. Потім – парсинг дерева залежностей слів у реченні, частин мови та іменованих сутностей. Отримане дерево залежностей аналізується за допомогою правил. Для автоматичного виправлення використовується підхід передбачення міток-тегів, що вказують на дію, яку необхідно виконати над кожним токеном. Для речень із багатьма помилками, парсинг дерев залежнотей яких дає некоректні результати, застосовується МТ-метод тексту із помилками у правильний його варіант.

Перевірка орфографії є першим етапом перевірки граматичної коректності речень. Словник Dictionary позначає окремий клас тому, що для перевірки орфографії використовують словники певної мови, що також можуть містити правила утворення слів, а результати ранжуються за зростанням Edit Distance або модифікаціями цієї метрики.

Клас ML model (ML-модель) описує поведінку ML-моделі, до методів якого належать тренування моделі на навчальних даних та передбачення результатів. Кратність зв'язків «0…*» та відношення агрегації вказують на те, що система не має обов'язково використовувати МТ-алгоритми для перевірки граматичної правильності речень та їх автоматичного виправлення.

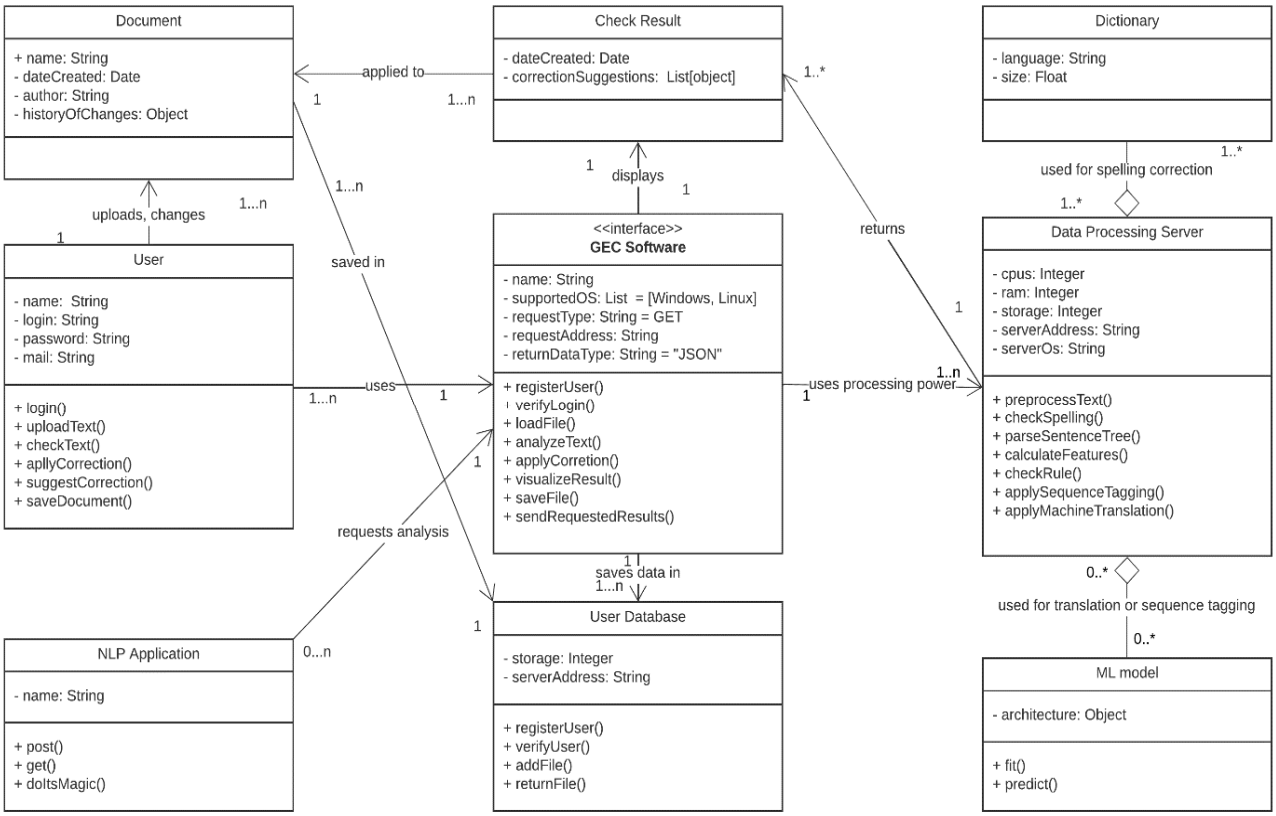


Рисунок 3 – Діаграма класів для GEC-системи

© Холодна Н. М., Висоцька В. А., 2023
DOI 10.15588/1607-3274-2023-1-12

Клас Check Result (результат перевірки) означає результат перевірки одного тексту. Його основним атрибутом є список запропонованих виправлень. Виправлення містять індекси початкового і кінцевого символів, виправлену стрічку, пояснення порушеного правила або причини необхідної заміни. Кратності усіх зв'язків та їх назви вказані на рис. 3.

Послідовність кроків взаємодії із системою перевірки граматичної правильності текстів може бути представлена за допомогою двох діаграм послідовності відповідно для індивідуального користувача додатку та для NLP-застосунку, який використовує виправлення текстів у граматично правильний варіант як одну із складових власного конвеєру (pipeline).

На рис. 4 подано діаграму послідовності для режиму перевірки власного тексту користувача відповідними компонентами як Software (програмне забезпечення), ML-модель Processing server (сервер опрацювання), ML model (), User Database (база даних користувачів). На схемі пропущені кроки реєстрації та авторизації, а також створення і збереження документу. Основний алгоритм такий:

1. Завантажити текст (Load text).
2. Надіслати текст (Send text).
3. Перевірити орфографію (Check spelling).
4. Виправити орфографічні помилки (Correct spelling errors).
5. Проаналізувати дерево залежностей або непередбачених ситуацій (Parse dependency or contingency tree).
6. Перевірити розширені правила на основі дерева (Check advanced rules based on tree).
7. Перевірити за допомогою моделі машинного навчання (Check with machine learning model).
8. Застосувати корекцію машинного навчання (Apply machine learning correction).
9. Повернути результати (Return results).
10. Повернути пропозиції у форматі: [i_початок, i_кінець, пропозиція, пояснення] (Return suggestions in format: [i_start, i_end, suggestion, explanation]).
11. Показати пропозиції (Display suggestions).
12. Застосувати виправлення (Apply fixes).
13. Запропонувати виправлення (Suggest correction).
14. Запам'ятати вподобання (Remember preferences).

БД користувачів використовується для збереження вподобань / зразків виправлень користувачів для подальшої персоніфікації запропонованих системою виправлень. Спочатку користувач має обрати варіант завантаження даних –вставкою тексту, відкриття файлу або створення і заповнення нового документу. ПЗ, встановлене на персональному комп'ютері, або доступне у форматі онлайн-сервісу чи розширення, відправляє дані для аналізу на сервер. На цій діаграмі послідовностей наведений лише один із можливих підходів до побудови системи граматичної

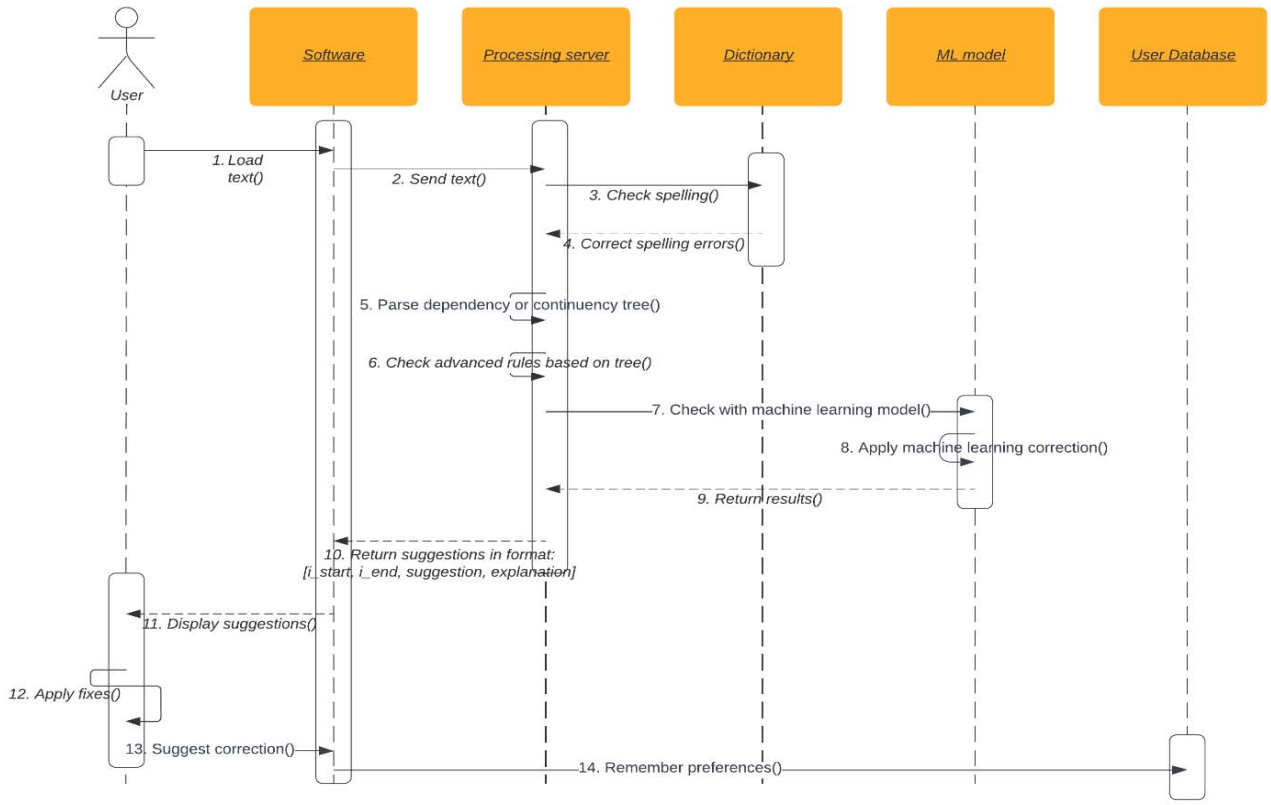


Рисунок 4 – Діаграма послідовності для першого режиму роботи системи

© Холодна Н. М., Висоцька В. А., 2023
DOI 10.15588/1607-3274-2023-1-12

корекції, що поєднує у собі перевірку орфографії, правил, парсинг синтаксичних дерев, перевірку правил на основі дерев і ML-алгоритми. Після перевірки орфографії, що відбувається з використанням словника певної мови, система має перевірити виконання базових правил, що не потребують парсингу синтаксичних дерев, як-от чергування «у-в», правопис «пів-напів», вживання апострофа тощо. Застосування перевірки правил та парсингу синтаксичних дерев обумовлене двома факторами:

– відсутністю достатньо великих анотованих / паралельних корпусів української мови для навчання повністю автоматичних моделей, що засновані на алгоритмах глибинного навчання;

– недостатньою спроможністю мовних моделей штучного інтелекту узагальнювати правила.

У випадку, коли у системі не реалізовані усі можливі правила правопису, в тому числі і ті, що засновані на парсингу синтаксичних дерев, одним із варіантів є додаткове використання ML-методів для передбачення тегів, що позначають дію, яку необхідно виконати над токенами, або для «перекладу» тексту із помилками у правильний варіант. Надалі ПЗ візуалізує і застосовує запропоновані виправлення, зберігає документ у базі даних за необхідності.

Другий режим застосування системи – аналіз граматичної коректності отриманих за допомогою API запитів (рис. 5). На цій діаграмі послідовності зображені етапи відправки, отримання і опрацювання запитів від стороннього NLP додатку (NLP Application) до системи (GEC System), зокрема:

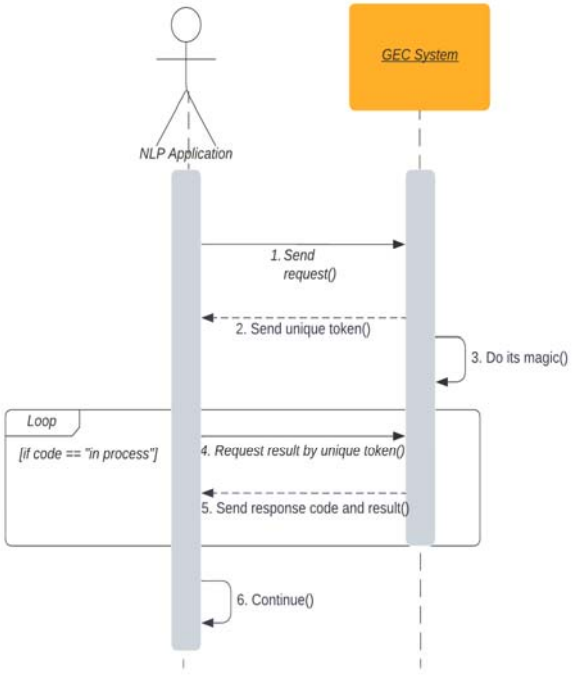


Рисунок 5 – Діаграма послідовності для другого режиму роботи

1. Надіслати запит (Send request).

2. Надіслати унікальний маркер (Send unique token).

3. Опрацювання запитів (Do its magic) з циклом (Loop) типу [якщо код == “в процесі”] ([if code == “in process”]).

4. Результат запиту за унікальним маркером (Request result by unique token)

5. Надіслати код відповіді та результат (Send response code and result).

6. Продовжити (Continue).

Об’єкт GEC System позначає сукупну систему граматичної корекції разом з усіма складовими, включаючи графічний інтерфейс, сервер для обробки даних, попередньо навчені ML-моделі. Повідомлення 3. Do its magic(), у свою чергу, позначає повний аналіз тексту та застосування необхідних виправлень.

GEC System надає кожному запиту унікальний токен, за допомогою якого можна отримати результати після завершення системою опрацювання запиту. Оскільки, в залежності від навантаження на систему, обробка даних може зайняти певний час, то NLP додаток має з певним інтервалом відправляти новий запит на отримання результатів. NLP додаток продовжує свою діяльність після отримання виправлених текстів. На діаграмі діяльності (рис. 6) зображений один із-поміж множини можливих підходів до розробки системи автоматичного виправлення граматичних помилок. Алгоритм:

1. Якщо доступний мовний словник [Language dictionary is available], то застосувати виправлення орфографії, що не запам’ятовується (Apply low-recall spelling correction).

2. Якщо мовний словник недоступний [Language dictionary is not available], то застосувати перевірку основних правил (Apply check of basic rules).

3. Якщо доступний синтаксичний аналізатор залежності/виборчої групи [Dependency/Constituency parser is available], то застосувати перевірку більш конкретних правил (Apply check of more specific rules).

4. Якщо правила можуть охоплювати всі випадки Rules can cover all cases], то використати підхід, заснований на правилах (Use a rule-based approach).

5. Якщо правила не можуть охоплювати всі випадки [rules can’t cover all cases] і/або якщо парсер недоступний [parser is not available], то перевірити або визначити розмір корпусу.

6. Якщо достатньо великі корпуси недоступні [large enough corpora is not available], то застосувати методи збільшення/генерування даних (Apply data augmentation / generation techniques).

7. доступні великі корпуси [large corpora is available], то визначити характеристики корпусу.

8. Якщо доступні великі анотовані корпуси [large annotated corpora is available], то використати підхід позначення послідовності (Use sequence tagging approach).

9. Якщо доступні великі паралельні корпуси [large parallel corpora is available], то використати підхід

© Холодна Н. М., Висоцька В. А., 2023
DOI 10.15588/1607-3274-2023-1-12

OPEN ACCESS
CC BY SA

нейронного машинного перекладу (Use neural machine translation approach).

10. Якщо доступні великі неанотовані корпуси [large unannotated corpora is available], то використати підхід на основі статистики (Use statistical-based approach).

Перш за все, якщо словник слів (або правил утворення слів) є доступним для певної мови, пропонується застосовувати його для виправлення орфографічних помилок. Варто зазначити, що модуль для автоматичного виправлення орфографічних помилок має мати низькі показники повноти (recall).

Значення повноти також можна інтерпретувати як відношення кількості правильно визначених позитивних випадків до усіх істинно позитивних випадків, тобто як частку загального числа позитивних зразків, що було знайдено.

Значення влучності (precision), у свою чергу, можна оцінити як частку правильно визначених істинно позитивних випадків.

В залежності від якості класифікатора та значення порогу прийняття рішення значення влучності буде залежним від значення повноти (рис. 7).

Низькі показники повноти (і високі – влучності) позначають те, що класифікатор (у даному випадку функція, що визначає орфографічні помилки) має бути достатньо «впевненим» у своєму «рішенні» про виправлення слова саме для того, щоб заміна не могла причинити зміну семантики речення.

Після перевірки і виправлення орфографії важливим є застосування перевірки базових правил граматики. Приклади для української мови: чергування «у-в», правопис «пів-», правопис апострофа та м'якого знака тощо.

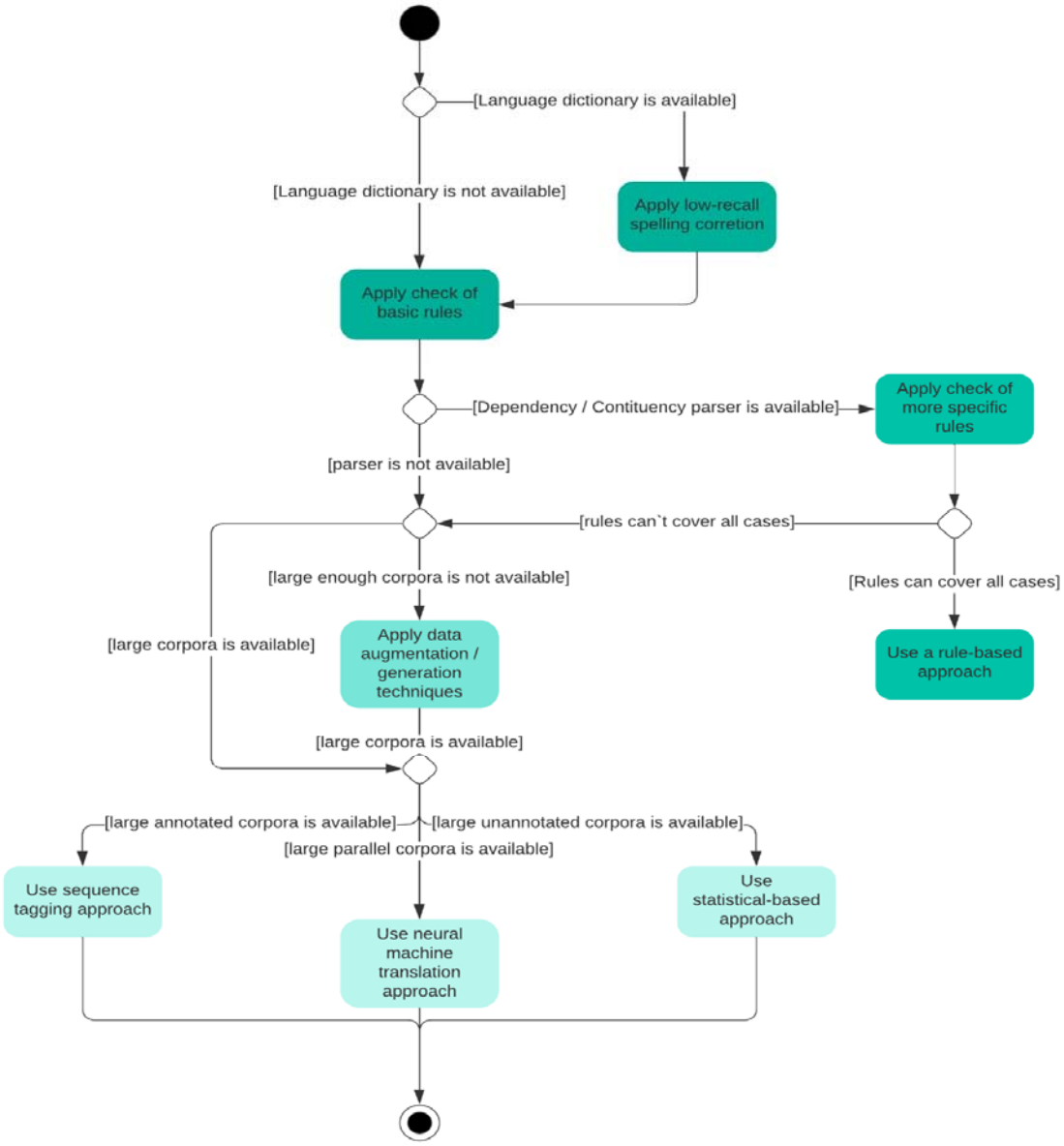


Рисунок 6 – Діаграма діяльності процесу розробки GEC-системи

© Холодна Н. М., Висоцька В. А., 2023
DOI 10.15588/1607-3274-2023-1-12

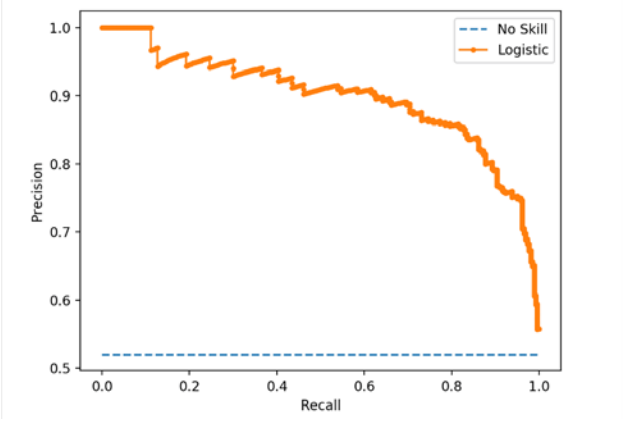


Рисунок 7 – Приклад кривої Precision-Recall

В залежності від того, чи є доступними парсери залежностей, груп слів, іменованих сутностей, частин мови тощо, можна застосовувати перевірку додаткових правил на основі отриманих результатів. Якщо теоретично можливий ітеративний розвиток системи і впроваджені правила можуть пояснити усі можливі помилки певної мови, використання методів і алгоритмів не є необхідним.

У випадку, якщо перевірки додаткових правил не є достатньо (особливо у випадку морфологічно багатих мов), для виправлення залишкових помилок можливим є застосування ML-методів. Якщо навчальних даних не є достатньо або вони взагалі не є доступними, для отримання паралельного / анотованого корпусу застосовують методи аугментації та генерації даних. Також, в залежності від того, чи є дані анотованими або корпус містить набір паралельних текстів, використовують відповідні підходи до розробки і навчання моделі. У випадку достатньо великих і якісних неанотованих корпусів також можна побудувати мовну модель на основі статистичних методів.

На діаграмі розгортання GEC-системи (рис. 8) зображено 3 основні процесори на основі трьох пристроїв/програм, зокрема : Пристрій користувача (:User Device), Сервер БД (:DB Server) та Сервер опрацювання даних (:Data Processing Server). :User Device застосовує програму/браузер (Program/ Browser). :DB Server використовує База даних NoSQL (NoSQL Database). :Data Processing Server побудований на основі реалізації ML-моделі (ML model), словника (Dictionary), модуля перевірки правил (Rule-check module) та модуля огляду користувачів (User review module).

Користувач має доступ до функціоналу системи за допомогою графічного інтерфейсу завантаженої програми, розширення або онлайн-платформи. У БД зберігається інформація про користувачів, приклади запропонованих виправлень, документи. Така БД за своїм типом не є реляційною.

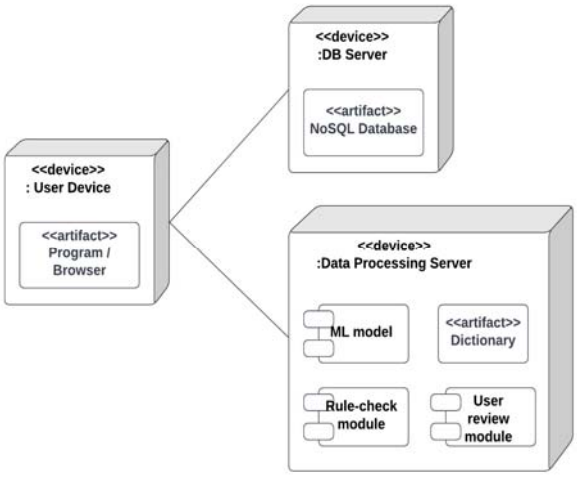


Рисунок 8 – Діаграма розгортання GEC-системи

Сервер обробки даних містить модулі ML-моделі, перевірки правил і перевірки запропонованих користувачами виправлень. Остання може застосовуватись для додаткового налаштування загальної системи перевірки та виправлення.

Альтернативним варіантом схеми розгортання є використання власних обчислювальних ресурсів персонального комп'ютера для аналізу даних, однак, з врахуванням того, що для отримання достовірного результату необхідно застосувати кілька етапів перевірки, особливо з використанням ML-методів, така архітектура не є доцільною.

Вибір набору даних. Набір україномовних UA-GEC анотованих текстів презентований дослідниками з Grammarly. Цей набір даних був професійно анотований для GEC та редагування україномовних текстів. Наскільки відомо на час публікації, це перший корпус GEC з української мови. Дослідники зібрали тексти з помилками (20 715 речень – 328 779 токенів) від різноманітних авторів, у тому числі носіїв мови. Дані охоплюють найрізноманітніші сфери письма, від текстових чатів і есе до офіційного письма. Професійні коректори виправляли та анотували корпус на помилки, пов'язані з вільним мовленням, граматикою, пунктуацією та орфографією. Даний корпус можна використовувати для розробки та оцінки систем GEC українською мовою і дослідження морфологічно багатих мов.

Методи. Основні підходи виявлення помилок, на яких зосереджені академічні дослідження, можна розділити на такі категорії: методи, засновані на правилах; методи, засновані на синтаксичному аналізі речень; статистичне моделювання; класичні ML-методи; MT на основі глибинного навчання. Для покращення якості системи використовують різні методи генерації синтетичних навчальних даних.

MT. Поява у 2018 році нової архітектури глибинного навчання Transformer, що містить у собі механізм уваги, дозволила значно спростити розробку мовних моделей, що тепер може відбуватися без

© Холодна Н. М., Висоцька В. А., 2023
DOI 10.15588/1607-3274-2023-1-12

застосування експертних знань домену, у даному випадку – лінгвістики. Більша частина досліджень застосовує модель Transformer для впровадження нейронного МТ тексту із помилками у його правильний варіант. Значним недоліком такого підходу для розробки моделей для різних мов є необхідність у великих корпусах анотованих або паралельних даних.

Transformer – модель глибинного навчання, особливістю якої є застосування механізму уваги, що роздільно зважують важливість кожної частини даних входу.

Будівельними блоками трансформера є вузли масштабованої скалярнодобуткової уваги (scaled dot-product attention units). Коли моделі Transformer передають речення, ваги уваги між усіма лексемами обчислюються одночасно. Вузол уваги виробляє вкладення для кожної лексеми в контексті, які містять інформацію про саму лексему, разом зі зваженим поєднанням інших релевантних лексем, кожну з яких зважено за вагою уваги до неї.

Альтернативою архітектурі Transformer є згорткові або рекурентні нейронні мережі архітектури encoder-decoder.

– Згорткові нейронні мережі (CNN, Convolutional Neural Network)

CNN складається з шарів входу та виходу, а також із декількох прихованих шарів. Приховані шари CNN зазвичай складаються зі згорткових шарів, агрегувальних шарів, повноз'єднаних шарів та шарів нормалізації. Згорткові шари застосовують до вхідних даних операцію згортки, передаючи результат до наступного шару.

Згорткові мережі можуть включати шари локального або глобального агрегування, які об'єднують виходи кластерів нейронів одного шару до наступного шару. Наприклад, максимізаційне агрегування використовує максимальне значення з кожного з кластерів нейронів попереднього шару. Іншим прикладом є усереднене агрегування, що використовує усереднене значення з кожного кластеру нейронів попереднього шару. Повноз'єднані шари з'єднують кожен нейрон одного шару з кожним нейроном наступного шару .

– Рекурентні нейронні мережі (Recurrent Neural Networks, RNN)

Рекурентні нейронні мережі це мережі, що містять зворотні зв'язки і дозволяють зберігати інформацію. Наявність зворотного зв'язку дозволяє передавати інформацію від одного кроку навчання мережі до іншого.

Одним з різновидів RNN є LSTM-мережі. LSTM (Long Short Term Memory) – це RNN, здатні до навчання довготривалих залежностей. LSTM-мережі складаються з повторюваних елементів. Кожен такий елемент містить чотири шари і відрізняється тим, що має комірку довгої короткочасної пам'яті.

Можливість LSTM-мереж успішно вивчати дані з довготривалими залежностями робить їх оптимальним вибором для розв'язання задач, у котрих як вхідна, так і вихідна інформація представляються у вигляді послідовностей деяких елементів (наприклад, літер, слів, речень).

Sequence tagging. Окрім методу нейронного перекладу, ML-методи у задачі виявлення та виправлення орфографічних помилок також застосовують для передбачення міток, що позначають дію, яку необхідно виконати над кожним токеном. В залежності від типу помилок, кількості наявних класів міток та загальної складності завдання використовують як глибинні ML-методи, так і класичного.

До класичних ML-методів із вчителем відносяться такі алгоритми: логістична регресія, дерево рішень, наївний байєсів класифікатор, метод опорних векторів, k-найближчих сусідів, випадковий ліс тощо.

– Логістична регресія

Логістична регресія використовується для завдань бінарної класифікації, тобто коли на виході потрібно отримати відповідь, до якого з двох класів належить об'єкт. Логістична функція перетворює будь-яке значення в число в межах від 0 до 1. Так і передбачається ймовірність належності до одного чи другого класу. Тобто, на відміну від логістичної регресії, цей метод не передбачає значення числової змінної, виходячи з вибірки вихідних значень, а замість цього значенням функції є ймовірність того, що це початкове значення належить до певного класу.

– Дерево рішень

Дерево рішень будує моделі класифікації або регресії у вигляді деревоподібної структури. Метод розбиває набір даних на менші й менші підмножини. Кінцевий результат – дерево з вузлами рішення і кінцевими вузлами. Що глибше дерево, то складніші правила ухвалення рішень і точніша модель.

Однією з основних переваг дерев рішень є їхня інтуїтивність: класифікаційна модель, що представлена у вигляді дерева рішень, легко інтерпретується користувачем та спрощує розуміння завдання, яке потрібно вирішити, тому що дозволяє зрозуміти, чому конкретний об'єкт належить до відповідного класу. Ще однією важливою перевагою дерев рішень є їхня універсальність для вирішення задач класифікації та регресії. Але є в алгоритму і недоліки. По-перше, це нестабільність процесу: невеликі зміни в наборі даних можуть призводити до побудови іншого дерева. Це пов'язано з ієрархічністю дерева: зміни в вузлах на верхньому рівні можуть призвести до змін у всьому дереві нижче. По-друге, це складність контролю розміру дерева, що є критичним фактором, який зумовлює якість вирішення завдання.

– Наївний Байєсів класифікатор

Наївний метод Байєса – це набір методів класифікації, заснованих на теоремі Байєса. Модель складається з двох типів ймовірностей, які розраховуються за допомогою тренувальних даних: імовірність кожного класу і умовна ймовірність для

© Холодна Н. М., Висоцька В. А., 2023
DOI 10.15588/1607-3274-2023-1-12
OPEN ACCESS
CC BY SA

кожного класу при кожному значенні x. Наївний байєсів класифікатор називається наївним, тому що алгоритм передбачає, що кожна вхідна змінна є незалежною. Це припущення не завжди відповідає реальним даним. Проте даний алгоритм вельми ефективний для цілого ряду складних завдань на зразок класифікації спаму або розпізнавання рукописних цифр.

– Метод опорних векторів

Метод опорних векторів використовується для задач класифікації тексту, як-от призначення категорії і виявлення спаму. Метод опорних векторів – один із найбільш популярних для класичної класифікації, тому що він простий та швидкий. Виходячи з того, що об'єкт, який перебуває в N-вимірному просторі, належить до одного з двох класів, метод опорних векторів будує гіперплощину з розмірністю (N – 1), щоб всі об'єкти виявилися в одній з двох груп. Що далі від гіперплощини лежать точки даних, то більше впевненість в тому, що вони були правильно класифіковані. Що менше схожих ознак між даними, то менша ймовірність належності до одного класу.

– Метод k-найближчих сусідів

Для алгоритму k-найближчих сусідів передбачення для нової точки робиться шляхом пошуку k найближчих сусідів в наборі даних і підсумовування вихідної змінної для цих k примірників. Ідея найближчих сусідів може погано працювати з багатовимірними даними, що негативно позначиться на ефективності алгоритму при вирішенні задачі.

– Випадковий ліс

Випадковий ліс належить до сімейства ансамблевих алгоритмів. Як зрозуміло з його назви, він містить велику кількість окремих дерев рішень, які діють як ансамбль. Кожне окреме дерево у випадковому лісі обчислює прогноз класифікації, і клас із найбільшою кількістю голосів стає прогнозом моделі. Результатом обирається той варіант, який випадав найчастіше. Велика кількість некорельованих дерев рішень, тобто тих, що знаходять рішення незалежно одне від одного й діють спільно, перевершить будь-яке рішення, отримане одним деревом рішення.

Саме некорельовані моделі можуть створювати ансамблеві прогнози, які є точнішими, ніж будь-який з окремо взятих прогнозів. Причиною такого ефекту є те, що кожне дерево рішень «захищає» одне одного від індивідуальних помилок: хоча деякі дерева рішень можуть бути неправильними, більшість із них будуть правильними, тому як група дерев буде рухатися в правильному напрямку.

До переваг методу можна віднести можливість ефективно обробляти дані з великою кількістю ознак і класів та високу масштабованість. Серед недоліків можна виділити великий розмір моделей, що будуються. Що більша модель, то вища її обчислювальна складність та швидкість знаходження рішень.

– AdaBoost

Бустінг – це сімейство ансамблевих алгоритмів, суть яких полягає в створенні сильного класифікатора на основі декількох слабких. Для цього спочатку створюється одна модель, потім інша модель, яка намагається виправити помилки в першій. Моделі додаються до тих пір, поки тренувальні дані не будуть ідеально класифіковані або поки не буде перевищено максимальну кількість моделей.

AdaBoost використовують разом з короткими деревами рішень. Після створення першого дерева перевіряється його ефективність на кожному тренувальному об'єкті, щоб зрозуміти, скільки уваги має приділити наступне дерево всіх об'єктах. Тим даним, які складно передбачити, дається більшу вагу, а тим, які легко передбачити, – меншу. Моделі створюються послідовно одна за одною, і кожна з них оновлює ваги для наступного дерева. Після побудови всіх дерев робляться прогнози для нових даних, і ефективність кожного дерева залежить від того, наскільки точним воно було на тренувальних даних.

Для обрання найкращого методу класифікації для його подальшого застосування необхідно порівняти показники точності, повноти, влучності, F-міри для всіх вищезазначених алгоритмів.

Морфологічні парсери. Перевірка правил відмінювання та узгодження слів у реченні може бути заснована на парсингу дерев залежності (Dependency Parsing) або приналежності (Constituency Parsing).

Термін парсинг залежностей відноситься до процесу дослідження залежностей між фразами речення з метою визначення його граматичної структури. Речення ділиться на багато розділів здебільшого на основі цього. Процес заснований на припущенні, що між кожною мовною одиницею в реченні існує прямий зв'язок. Ці гіперпосилання називаються залежностями. На відміну від аналізу залежностей, парсинг приналежностей розподіляє речення на фрази певного типу (іменникова, дієслівна групи тощо). У цьому випадку термінальний вузол – це мовна одиниця або фраза, яка має батьківський вузол та тег частини мови.

Парсинг залежностей та приналежностей слів у реченнях певних мов можна виконати за допомогою бібліотеки для NLP-NLTK, однак вона не підтримує аналіз україномовних текстів.

Підтримка української мови. Бібліотека мови Python pymorphy2 підтримує обробку української мови. pymorphy2 аналізує частину мови, число, відмінок, час, лему та стем заданого слова. Морфологічний парсер заснований на словниках OpenCorpora, що конвертовані до формату XML. Користувачі також мають змогу додавати власні слова та правила, це дає можливість проводити морфологічний аналіз текстів певної предметної області без зміни вихідного коду pymorphy2 та адаптувати pymorphy2 для роботи з іншими мовами.

Мова і бібліотеки. Для ML найпопулярнішими мовами є Python, R, Java, C++. Перевагою Python з-

© Холодна Н. М., Висоцька В. А., 2023
DOI 10.15588/1607-3274-2023-1-12

OPEN ACCESS

поміж інших мов саме для створення системи визначення емоційного забарвлення текстового контенту є його підтримка великої кількості бібліотек:

– для роботи з класичними ML-методами: Scikit-Learn;

– для створення штучних нейронних мереж, глибинного навчання: TensorFlow, Keras, PyTorch;

– для NLP: NLTK, spaCy, WordNet;

– для роботи із масивами, матрицями: NumPy;

– роботи з таблицями: pandas;

– візуалізації даних (в тому числі, й інтерактивної): Matplotlib, seaborn, Plotly.

Бібліотека Scikit-Learn підтримує TPP, зменшення розмірності даних, вибір ML-моделей для регресії, класифікації або кластерного аналізу. Однак Scikit-Learn не має комплексної підтримки для створення моделей глибинного навчання.

Для створення штучних нейронних мереж була обрана бібліотека Keras, що слугує високорівневим API для TensorFlow 2. Keras дозволяє будувати послідовні моделі у вигляді графу, вершинами якого є шари (layers) певного типу із заданою кількістю вузлів. Також, за допомогою Keras можна поєднувати результати роботи кількох окремих частин нейронної мережі для їх подальшої обробки, така структура не є лінійною.

TensorFlow дає можливість імпортувати навчену ML-модель для її подальшого використання у інших програмах. TensorFlow також підтримує виконання низькорівневих операцій із тензорами за допомогою центральних процесорів, графічних процесорів та тензорних блоків обробки.

Бібліотека NLTK у цьому дослідженні застосовується для попередньої обробки тексту: токенізації, видалення стоп-слів, стемінгу, лематизації. Також за допомогою функцій з цієї бібліотеки можна виявляти найпопулярніші n-грами і частини мови окремих токенів, розпізнавати іменовані сутності тощо.

До додаткових бібліотек, що спрощують роботу із природньою мовою, належать Regex та emoji – для використання регулярних виразів і заміни емоджі словами відповідно.

Середовище розробки. Для роботи над задачами з дослідження даних і застосування ML зручним інструментом є Jupyter Notebook, який дозволяє запускати написаний код невеликими фрагментами – комірками. Одним із онлайн-сервісів, що надає змогу використовувати Jupyter Notebook без локального встановлення, є Google Colab. Цей сервіс дає можливість використовувати графічні процесори GPU і TPU, що значно пришвидшують навчання штучних нейронних мереж.

4 ЕКСПЕРИМЕНТИ

Опрацювання набору даних. Обраний набір даних UA-GEC [15] містить 850 і 160 текстів у навчальній і тестовій вибірці відповідно. За даними авторів, набір даних містить усього 20 715 речень (як з помилками, так і без). На момент написання роботи вбудовані методи ітерації по корпусу UA-GEC дозволяють лише отримувати повні документи, а не окремі речення.

Текст анотований за наступним форматом:

```
I {like=>likes:::error_type=Grammar} turtles.
```

Для опрацювання та розділення текстів була застосована бібліотека ReGex. Регулярні вирази для розподілу текстів на речення та виявлення помилок мають наступний формат:

```
split_pattern = r'\n+'
additional_split_pattern = r'(?<=[^А-Я].[.?!(\.\.\.)…])
+(?=[А-Я“"])'
error_pattern =
r'\{(([^\{]*\(*\)*)=>([^\{]*\(*\)*):::error_type=([^\{]*))\}'
```

Правильні пари не видаляються, однак для подальшого опрацювання були 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.

Попередньо навчені нейронні мережі.

Морфологічна складність української мови зумовлює потребу у поєднанні різних GEC-методів.

Один із модулів запропонованої системи – опрацювання речення за допомогою штучних нейронних мереж. Навчання нейронної мережі «з нуля» з випадково ініціалізованими вагами потребує паралельних корпусів великих обсягів. Єдиний на момент написання роботи україномовний набір даних містить усього тисячу текстів, що є критично мало для навчання мовної моделі для обробки морфологічно багатих мов.

Тому, навчання нейронної мережі GEC-завданню вимагає її попереднього налаштування за допомогою великих корпусів. У цьому випадку існує кілька попередньо налаштованих моделей, що підтримують українську і параметри яких є доступними у відкритому доступі: RoBERTa від YouScan [48], MT-моделі (від Google – MT5 [49], від Facebook – M2M100 [50] та mBART-50 [51]).

Модель архітектури енкодер-декодер на основі RoBERTa. У 2020 році команда дослідників із YouScan презентувала нейронну мережу «Ukrainian Roberta» [48], що була попередньо натренована на україномовних текстах загальним обсягом 2,5 мільярди слів. «Ukrainian Roberta» має 125 мільйонів параметрів, що налаштовуються, її навчання тривало 85 годин.

Складністю використання цієї нейронної мережі є те, що вона за своєю архітектурою є тільки енкодером і використовується для отримання контекстних векторних вкладень слів. «Ukrainian Roberta» була навчена вирішувати завдання передбачення замаскованого токену у реченні (рис. 9). Попереднє навчання нейронної мережі дозволяє їй отримати попереднє «знання» граматики мови.

© Холодна Н. М., Висоцька В. А., 2023
DOI 10.15588/1607-3274-2023-1-12

OPEN ACCESS
CC BY SA

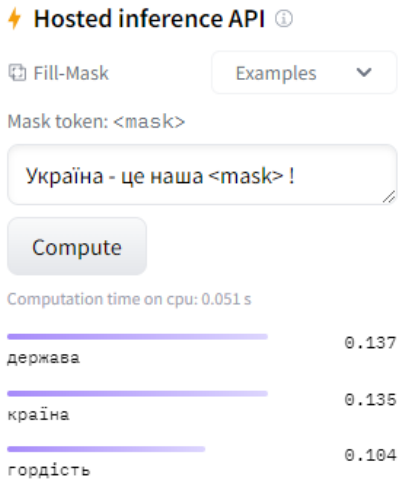


Рисунок 9 – Приклад передбачення замаскованого токену

BERT – модель глибинного навчання для NLP архітектури Transformer. Базова модель BERT має 110 мільйонів, розширена версія – 345. Особливістю архітектури Transformer є наявність механізму уваги, завдяки чому дані можуть бути оброблені одночасно (на противагу рекурентним нейронним мережам, де дані сприймаються послідовно). Окрім того, особливістю BERT 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.

Попередньо навчену нейронну мережу RoBERTa можна вважати покращеним аналогом BERT: її архітектура є аналогічною до архітектури BERT, основна відмінність полягає у підборі гіперпараметрів під час попереднього навчання, збільшенні об'єму навчального корпусу, застосуванні динамічного маскування токенів для задачі передбачення слова у реченні: слово, яке необхідно передбачити у певному реченні, змінюється з кожною епохою. Окрім того, у цьому випадку відсутнє передбачення того, чи є друге речення логічним продовженням першого. Наприклад, для класифікації речень додається шар повнозв'язних нейронів, де кількість нейронів відповідає наявним класам [52].

Нейронні мережі типу денкодер утворюють констекстні векторні представлення слів у реченні фіксованої довжини незалежно від довжини вхідного повідомлення. У свою чергу, BERT є складовою нейронних мереж типу енкодер-декодер, що використовуються для завдань sequence-to-sequence. Декодери, у свою чергу, генерують токени послідовно, кожний наступний токен залежить від попередніх. Застосування попередньо навченої «Ukrainian Roberta» є можливим за аналогією до підходу, запропонованого у [53]. У випадку ініціалізації BERT у якості декодера, для передбачення наступного слова на основі контекстного векторного вкладення вхідних токенів, необхідно додати шар перехресної уваги (cross-attention layer) із випадковим чином згенерованими вагами. Окрім того, для передбачення розподілу ймовірностей наступного слова використовується LM Head (шар або шари повнозв'язних нейронів, де кількість вихідних нейронів відповідає кількості токенів у словнику). Вагові коефіцієнти LM Head ініціалізуються ваговими коефіцієнтами шару векторного вкладення BERT W_emb. Також, двонапрямлений механізм уваги BERT необхідно замінити на однонапрямлений для генерації токену в залежності лише від попередніх [52, 57]. Словник токенів, що відповідає попередньо навченій моделі «Ukrainian Roberta», містить 52 тис. елементів, де перші записи є спеціальними токенами, розділовими знаками та найбільш поширеними словами (рис. 10). Оскільки словники токенів є унікальними для кожної попередньо навченої моделі, TPP теж має свої особливості. Після токенізації текстів і речень у навчальній вибірці розподіл кількості токенів має наступний вигляд, поданий на рис. 11. Максимальна довжина речення – 100 токенів, довші речення обрізаються.

Нейронна мережа навчалась протягом 15 епох з наступними гіперпараметрами:

- n_epochs = 15
- batch_size = 8
- lr = 0.00001

Після 15-ї епохи значення функції втрат на тестовій вибірці збільшується (рис. 12), тому 15-ть епох є найбільш оптимальним значенням у даному випадку.

Налаштування попередньо навченої МТ-моделі. Facebook та Google надали відкритий доступ до ваг попередньо навчених нейронних мереж для МТ.

```
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),
```

Рисунок 10 – Словник токенів «Ukrainian Roberta»

© Холодна Н. М., Висоцька В. А., 2023
DOI 10.15588/1607-3274-2023-1-12

OPEN ACCESS
CC BY SA

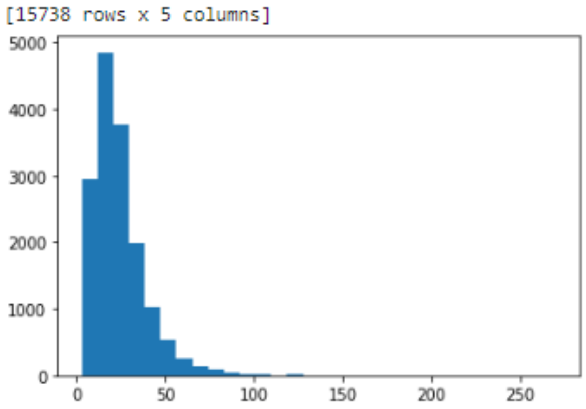


Рисунок 11 – Розподіл кількості токенів у навчальній вибірці

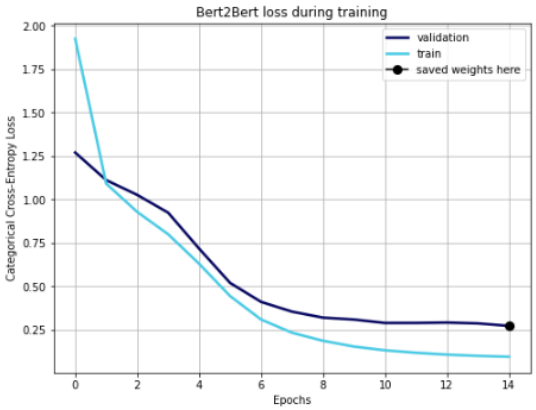


Рисунок 12 – Крива функції втрат при навчанні NN

mT5 [49] заснована на архітектурі Transformer і була попередньо навчена MT-завданню із застосуванням корпусу, заснованого на даних Common Crawl. Усього нейронна мережа була попередньо натренована на текстах, написаних 101 мовою, серед яких є і українська. Базова версія, що застосовується у цьому дослідженні, має 580 мільйонів параметрів, що налаштовується, версія mT5-XXL – 13 мільярдів параметрів.

У зв'язку з обмеженням наявних обчислювальних ресурсів, найбільша довжина речення складає 20 токенів, batch size – 2 записи. Варто зауважити, що, оскільки нейронна мережа була попередньо натренована за допомогою текстів, написаних різними мовами, то відповідний словник токенів радше містить частини україномовних слів, аніж повні слова. Тому одне слово кодується більшою кількістю токенів, а у контексті обмеженої довжини речення це може призвести до некоректного представлення навчальних прикладів і їх значення, що не є повністю завершеним. Нейронна мережа навчалась протягом 10 епох, найнижче значення функції втрат на тестовій вибірці було досягнуто на 4 епосі, тоді ж і були збережені ваги нейронної мережі.

Відповідно до [50], M2M100 має низку переваг у порівнянні із попередніми MT-моделями. За ствердженнями дослідників, попередні MT-моделі навчаються на «англо-центричних» даних, тобто для перекладу між двома мовами такі моделі навчаються перекладу з першої мови на англійську, потім – з англійської на другу мову. Такий підхід може призвести до втрати початкового значення речення, і M2M100 – перша нейронна мережа для MT, що була навчена прямому перекладу з першої мови на другу без проміжного перекладу на англійську. Такий підхід дозволив отримати найкращі результати метрики BLEU у порівнянні з іншими дослідженнями, збільшивши показник на 10 пунктів. Усього нейронна підтримує 2 200 напрямків перекладу – у 10 разів більше, ніж попередня англо-центрична мовна модель. NN попередньо натренована на тестах, написаних 100 мовами, у тому числі й українською. Базова версія, що застосовується у дослідженні, містить 418 мільйонів параметрів, найбільша – 15 мільярдів. Після розрахунку кількості токенів для текстів навчальної вибірки отримуємо наступний розподіл, поданий на рис. 13.

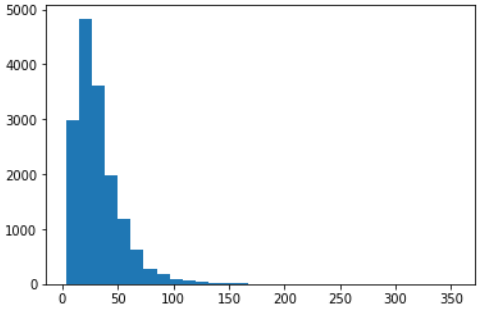


Рисунок 13 – Розподіл кількості токенів у навчальній вибірці

Аналогічно до попереднього випадку, максимальна кількість токенів є обмеженою у зв'язку з обмеженими обчислювальними ресурсами. В цьому випадку, також можливе некоректне представлення навчальних прикладів. Нейронна мережа навчалась протягом 10 епох, однак найменше значення функції втрат на тестовій вибірці було досягнуто на другій епосі (рис. 14). Для порівняння також було проведене навчання нейронної мережі протягом 10 епох, ваги були збережені після останньої.

Варто зауважити, що на якість генерації тексту впливає низка параметрів, а саме:

– Значення гіперпараметрів.
– Обсяг навчальної вибірки.
– Кількість навчальних епох.
– Архітектура і кількість параметрів нейронної мережі.
– Кількість токенів вхідного тексту.
– Навчальна вибірка (чи застосовувались валідаційні дані або був використаний весь корпус).

Приклад навчання NN MT5. Імпорт бібліотек. Основними бібліотеками у даному проекті є PyTorch, HuggingFace Transformers, Regex, pandas (рис. 15).

© Холодна Н. М., Висоцька В. А., 2023
DOI 10.15588/1607-3274-2023-1-12

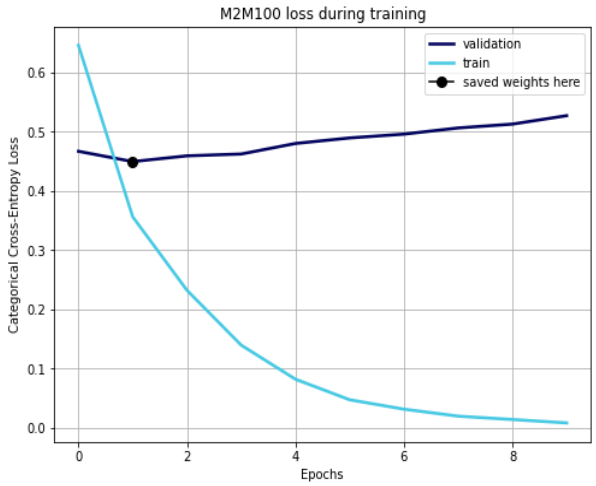


Рисунок 14 – Крива функції втрат при навчанні NN

```
################################  IMPORTS  ################################

from collections import defaultdict
from ua_gec import Corpus
from ua_gec.corpus import Document

import pandas as pd
import regex as re
import copy
import numpy as np
import matplotlib.pyplot as plt

from transformers import AdamW, AutoModelForSeq2SeqLM, AutoTokenizer
from transformers import get_linear_schedule_with_warmup
from transformers import Seq2SeqTrainer, Seq2SeqTrainingArguments

from IPython.core.display import display, HTML
from tqdm import tqdm_notebook

import torch
from torch.utils.data import Dataset, DataLoader
```

Рисунок 15 – Імпорт бібліотек

Константи на налаштування. Встановлюємо такі налаштування: регулярні виразу для розділення текстів на речення, регулярний вираз для пошуку анотації помилок, гіперпараметри (рис. 16). Завантаження попередньо навчено моделі. Завантаження попередньо навченої NN для її подальшого налаштування відбувається за допомогою кількох команд (рис. 17). Створення PyTorch Dataset. Перед створенням класу Датасет та його об'єктів необхідно провести попереднє дослідження даних, після цього агрегуючи ці етапи (рис. 18). Додаткові налаштування подані на рис. 19. При використанні PyTorch необхідно кожного разу реалізувати власну функцію навчання NN (рис. 20).

```
################################  SETTINGS  ################################

split_pattern = r'\n+'
error_pattern = r'\{{([^\{]*\(*\)*)=>([^\{]*\(*\)*):::error_type=([^\{]*)}\}'
additional_split_pattern = r'(?<=[^A-Я].[.?!(\.\.\.)…]) +(?=[A-Я""])'

regex_special_symbols = {
                '(': '\(',
                ')': '\)',
                '?': '\?',
                '[': '\[',
                ']': '\]',
                '*': '\*',
                '$': '\$'
            }

model_name = 'google/mt5-base'

n_epochs = 25
batch_size = 4
lr = 0.00001
max_token_length = 50

device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
```

Рисунок 16 – Змінні із налаштуваннями

```
#########################  LOAD PRETRAINED MODEL  #########################

tokenizer = AutoTokenizer.from_pretrained(model_name)
model = AutoModelForSeq2SeqLM.from_pretrained(model_name)
model = model.to(device)
```

Рисунок 17 – Завантаження попередньо навченої NN

```
train_dataset = Parallel_Dataset('train', tokenizer, max_token_length)
test_dataset = Parallel_Dataset('test', tokenizer, max_token_length)
```

Рисунок 18 – Завантаження датасету

```
############################  ADDITIONAL  ############################

test_loader = DataLoader(test_dataset, batch_size=batch_size)
train_loader = DataLoader(train_dataset, batch_size=batch_size)

n_batches = int(np.ceil(len(train_dataset)) / batch_size)
n_batches_test = int(np.ceil(len(test_dataset)) / batch_size)

total_steps = n_epochs * n_batches

n_warmup_steps = int(total_steps * 0.01)

optimizer = torch.optim.AdamW(model.parameters(), lr=lr)
scheduler = get_linear_schedule_with_warmup(optimizer,
                                            n_warmup_steps,
                                            total_steps)
```

Рисунок 19 – Додаткові налаштування

```
model, history = train(model, optimizer, scheduler,
                       n_epochs, len(train_dataset), len(test_dataset),
                       train_loader, test_loader)
```

Рисунок 20 – Виклик функції навчання NN

Збереження ваг NN: torch.save(model, 'mt5.pt'). Отримання передбачення подано на рис. 21.

© Холодна Н. М., Висоцька В. А., 2023
DOI 10.15588/1607-3274-2023-1-12

```
def predict(sentence):
    inputs = tokenizer(sentence, padding="max_length", truncation=True, max_length=200, return_tensors="pt")
    input_ids = inputs.input_ids.to("cuda")
    attention_mask = inputs.attention_mask.to("cuda")

    outputs = model.generate(input_ids, attention_mask=attention_mask)
    print(outputs)

    output_str = tokenizer.batch_decode(outputs, skip_special_tokens=True)

    display(HTML(f'<h3>Input sentence</h3>'))
    display(HTML(f'<p>{sentence}</p>'))
    display(HTML(f'<h3>Result</h3>'))
    display(HTML(f'<p>{output_str[0]}</p>'))

predict('учора був чудове день')
```

Рисунок 21 – Функція отримання передбачення

5 РЕЗУЛЬТАТИ

Для демонстрації роботи нейронних мереж використані власні приклади та завдання Зовнішнього Незалежного Оцінювання. Отримані результати: «Ukrainian Roberta» encoder-decoder. Нейронна мережа непогано розставляє розділові знаки у простому реченні (рис. 22).

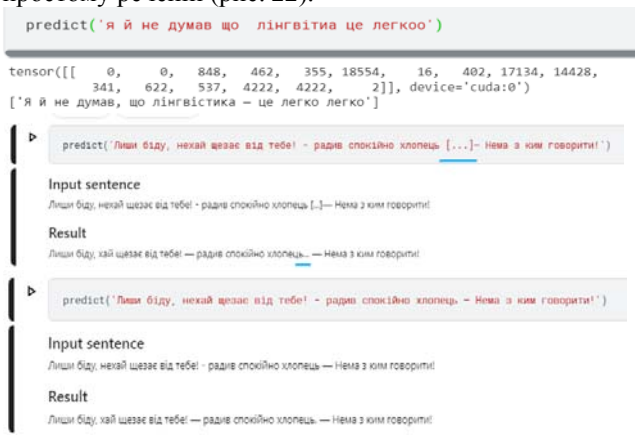


Рисунок 22 – Розставлення розділових знаків

Наступний приклад: NN правильно виправила розділові знаки, однак самостійно змінила речення, використовуючи відомі їй словосполучення (рис. 23). Інколи NN «не розуміє» суті поставленого завдання (рис. 24 – залишити правильне словосполучення у початковому вигляді). Інколи NN взагалі «не розуміє» суті поставленого завдання (рис. 25). Однак у цьому випадку речення має логічну структуру. На рис. 26 NN правильно змінила відмінок іменника з називного на кличний, однак гуаш – фарба, іменник жіночого роду, тут він не є правильно узгодженим, ймовірно тому, що аналогічних прикладів немає у навчальному наборі даних.

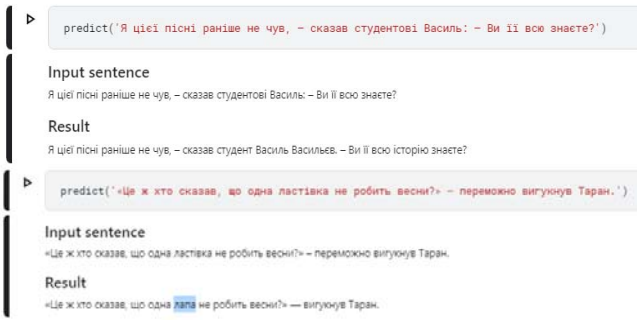


Рисунок 23 – Зміна речення ейронною мережею

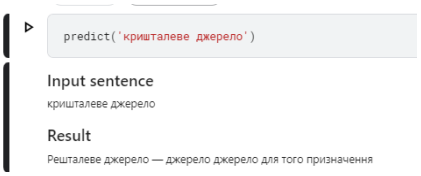


Рисунок 24 – Нерозуміння задачі NN

Рисунок 25 – Некоректна зміна речення

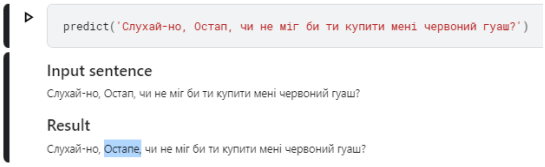


Рисунок 25 – Зміна відмінку іменника

На рис. 26 NN правильно виправила числівник, але завершила речення своєю «творчістю». У прикладах, що містять лише слова або словосполучення, нейронна мережа «намагається» завершити думку, самостійно добудовуючи речення. Такий результат є прямим наслідком того, що приклади у навчальному наборі містять лише речення, а не словосполучення.

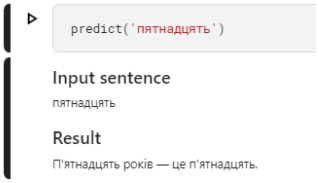


Рисунок 26 – «Творчість» NN

Однак, для перевірки слів і словосполучень в загальному застосовуються інші методи: перевірка за словником та перевірка простих правил (апостроф, чергування у-в, правопис пів-, напів- тощо).

Попередні результати тестування NN показують, що NN неправильно визначає суть поставленого завдання (виправлення помилок при збереженні початкового змісту речення / словосполучення), що може бути результатом відсутності окремих слів та словосполучень у навчальному наборі даних. На противагу цьому, NN виправляє речення паттернами з набору навчальних даних, що навпаки може свідчити про її перенавчання. Окрім того, причиною таких помилок може бути архітектура моделі – RoBERTa оригінально застосовується як енкодер для отримання контекстних представлень токенів. На рис. 27 NN виправила «чим…тим» на «що…то». Деякі

© Холодна Н. М., Висоцька В. А., 2023
DOI 10.15588/1607-3274-2023-1-12

OPEN ACCESS
CC BY SA

мовознавці вважають, що цей сполучник не є власне українським, а суржиком. Прості речення є зазвичай простим завданням для NN (рис. 28), але є складності (рис. 29).

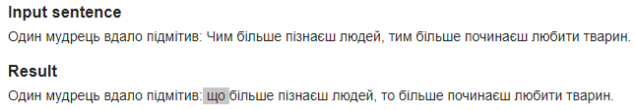


Рисунок 27 – Виправлення NN суржика

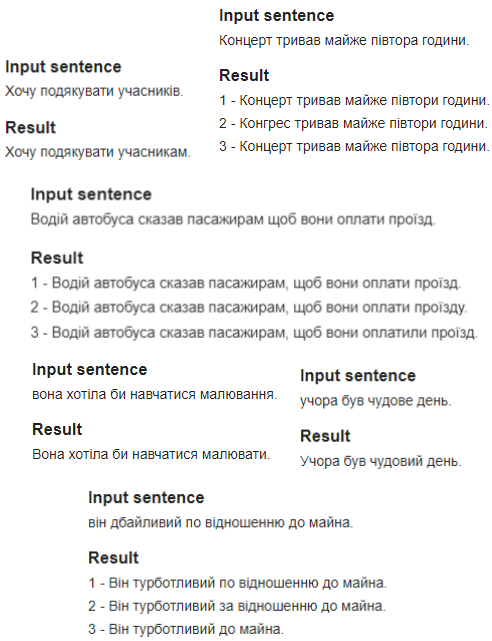


Рисунок 28 – Виправлення простих речень

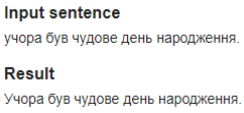


Рисунок 29 – Некоректне виправлення

Правильне речення є серед пропозицій моделі, однак проблемним є той факт, що більша ймовірність була призначена неправильному реченню (рис. 30). На рис. 31а NN правильно замінила словосполучення «за професії», однак слово «архітектор» замінене на найближче за значенням слово. Аналогічно, більшу ймовірність має неправильне речення (рис. 31б), але правильне виправлення є серед запропонованих.

У деяких реченнях mT5 пропонує більш правильні виправлення (рис. 32). На даних прикладах найкращу точність виправлення має МТ-модель mT5, однак для остаточної оцінки і порівняння NN необхідний достатньо великий тестовий корпус даних.

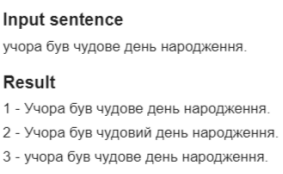
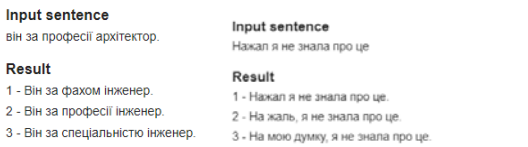


Рисунок 30 – Варіанти виправлення



а б

Рисунок 31 – Коректне виправлення

Рисунок 32 – Варіанти корекції різним NN-моделями

Із наведених прикладів видно, що mT5 краще вдається виправляти граматичні без зміни початкового змісту речення. Однак дана модель має вдвічі більше параметрів, ніж енкодер-декодер, заснований на Roberta (580 мільйонів проти 250), що вимагає значних обчислювальних ресурсів для навчання NN та отримання передбачення. Нейронна мережа M2M100 значно гірше виправляє граматичні помилки, ніж дві попередні моделі (рис. 33)

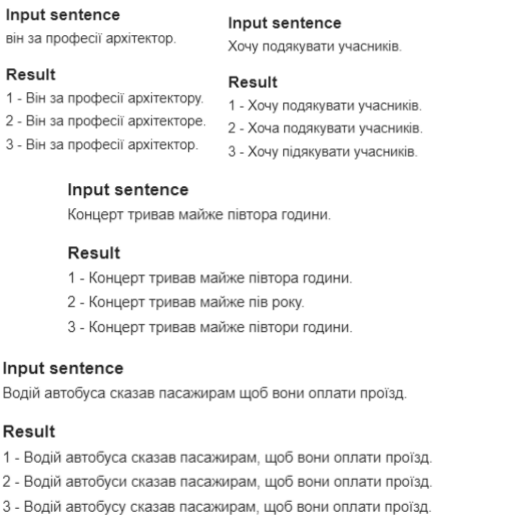


Рисунок 33 – Робота нейронної мережі M2M100

© Холодна Н. М., Висоцька В. А., 2023
DOI 10.15588/1607-3274-2023-1-12

6 ОБГОВОРЕННЯ

Порівняння нейронних мереж за допомогою спеціальних метрик. Відповідно до [54], для оцінки GEC-якості не існує базових, уставлених метрик. Із метою оцінки таких систем в загальному використовують метрики МТ-якості, однак і вони мають певні недоліки.

BLEU (bilingual evaluation understudy) [55] – це алгоритм для оцінки МТ-якості. Оцінки розраховуються для окремих перекладених сегментів (зазвичай речень) шляхом порівняння їх з набором правильних перекладів хорошої якості. Ці бали потім усереднюються по всьому корпусу, щоб отримати оцінку загальної якості перекладу. Оцінка за BLEU завжди є числом від 0 до 1. Це значення вказує, наскільки текст-кандидат подібний до еталонних текстів, а значення, ближчі до 1, означають більш подібні тексти. Значення BLEU засноване на розрахунку кількості спільних N-грам у початковому реченні та його перекладі. Недоліком метрики BLEU є те, що вона заснована на порівняннях стрічок, і, відтак, ігнорується той факт, що деякі граматичні помилки можна виправити кількома способами. Окрім того, результати отримані результати GEC-якості не можна порівняти із результатами певних систем перекладу, оскільки у першому випадку більша кількість н-грамів у реченнях буде співпадати (або співпадати повністю, якщо речення не містить граматичних помилок і не було змінене).

METEOR (Metric for Evaluation of Translation with Explicit ORdering) [56] – ще одна метрика для оцінювання МТ-якості. Вона заснована на використанні n-gram та орієнтована на використання статистичної та точної оцінки вихідного тексту. На відміну від метрики BLUE, дана метрика використовує функції співставлення синонімів разом із точною відповідністю слів. Метрика була розроблена, щоб вирішити проблеми, які мала BLUE, а також відтворити хорошу кореляцію з оцінкою експертів на рівні словосполучень або речень.

В результаті запуску метрики на рівні словосполучень кореляція з людським рішенням становила 0.964, тоді кореляція з BLUE становила 0,817 на тому ж наборі вхідних даних. На рівні речень максимальна кореляція з оцінкою експертів була 0,403 [56].

Як і в метриці BLUE, основна одиниця для оцінки – речення, алгоритм спочатку проводить вирівнювання тексту між двома реченням, рядком еталонного перекладу та рядком вхідного тексту для оцінювання. Дана метрика використовує декілька етапів встановлення відповідності між словами МТ й еталонного перекладу для зіставлення двох рядків:

1. Точне встановлення відповідності – визначаються рядки, що є ідентичними в еталонному і машинному перекладі.

2. Встановлення відповідності основ – проводиться стемінг (виділення основи слова) і визначаються слова з однаковим коренем в еталонному і машинному перекладі.

3. Встановлення відповідності синонімів – визначаються слова, що є синонімами відповідно до WordNet.

Розраховані метрики МТ-якості дають змогу лише частково порівняти моделі, оскільки більшість слів і словосполучень у початковому та виправленому реченні співпадають. Найкраще значення як BLEU (0,908) , так і METEOR (0,956) отримано для mT5, що співпадає із аналізом прикладів, у якому найбільш точні виправлення помилок без зміни початкового значення речення отримані саме для цієї нейронної мережі. M2M100 має більшу оцінку BLEU (0,847), ніж «Ukrainian Roberta» Encoder-Decoder (0,697), однак, суб'єктивно оцінюючи результати виправлення прикладів, M2M100 значно гірше справляється із цим завданням, ніж дві інші моделі. Для METEOR також M2M100 (0.925) має більшу оцінку, ніж «Ukrainian Roberta» Encoder-Decoder (0,876). Щодо розробки проекту можна зробити такі висновки:

– Для створення якісної ML-моделі для GEC у текстах тих мов, які є складними морфологічно, необхідна велика кількість паралельних або вручну розмічених даних. Ручна анотація даних вимагає багато зусиль професійних лінгвістів, що робить створення корпусів текстів, особливо морфологічно багатих мов, часо- та ресурсозатратним процесом.

– Вирішення завдання автоматичного виявлення та виправлення помилок в україномовних текстах вимагає подальших досліджень у зв'язку з невеликою кількістю робіт, що фокусуються на обробці саме української мови. Окрім того, відповідно до результатів дослідження Погорілого С. Д. і Крамова А. А. [27], методи, які застосовуються для обробки англійської мови, не можуть бути використані для української, оскільки остання є значно складнішою і морфологічно багатшою мовою.

– Поява у 2017 році нової архітектури глибинного навчання Transformer [3], що містить механізм уваги, дозволила значно спростити розробку мовних моделей, що тепер може відбуватися без застосування експертних знань домену, у даному випадку – лінгвістики. Більша частина досліджень застосовує модель Transformer для впровадження нейронного МТ тексту із помилками у його правильний варіант. Значним недоліком такого підходу для розробки моделей для різних мов є необхідність у великих корпусах анотованих або паралельних даних. Єдиний (на момент написання роботи) україномовний набір даних [15] містить усього 20 715 речень, такої кількості навчальних зразків не є достатньо для створення автоматичної інтелектуальної МТ-системи.

– Застосування перевірки правил та парсингу синтаксичних дерев при розробці GEC-системи в україномовних текстах може бути обумовлене такими двома факторами:

1) відсутністю достатньо великих анотованих / паралельних корпусів української мови для навчання

© Холодна Н. М., Висоцька В. А., 2023
DOI 10.15588/1607-3274-2023-1-12

OPEN ACCESS
CC BY SA

повністю автоматичних моделей, що засновані на алгоритмах глибинного навчання;

2) недостатньою спроможністю лінгвістичних моделей штучного інтелекту узагальнювати граматичні правила природньої мови [37].

– розробка якісної системи перевірки граматичної правильності речень в україномовних текстах вимагатиме поєднання ML-алгоритмів із кількома різними типами методів, зокрема і застосування експертних знань з комп'ютерної лінгвістики.

– Відповідно до результатів досліджень, для отримання найкращої GEC-точності за допомогою нейронної мережі, потрібно використовувати попередньо навчені MT-моделі, що підтримують українську мову.

– Найкраще значення як BLEU, так і METEOR було отримано для MT-моделі mT5, результати співпадають із аналізом власних прикладів, у якому найбільш точні виправлення помилок без зміни початкового значення речення були отримані саме для цієї нейронної мережі. M2M100 має більшу оцінку BLEU, ніж «Ukrainian Roberta» Encoder-Decoder, однак, суб'єктивно оцінюючи результати виправлення власних прикладів, M2M100 значно гірше справляється із цим завданням, ніж дві інші моделі. З метою економії обчислювальних ресурсів можливим також є застосування попередньо навченої нейронної мережі типу BERT, використовуючи її як у якості енкодера, так і декодера. Така нейронна мережа має вдвічі менше параметрів, ніж інші попередньо навчені MT-моделі, і показує задовільні GEC-результати.

ВИСНОВКИ

Результатом роботи є розроблена ML-модель для GEC в україномовних текстах. Ми також пропонуємо універсальну схему розробки GEC-системи для різних мов. Відповідно до отриманих результатів, нейронна мережа має здатність виправляти прості речення, написані українською, однак розробка повноцінної системи вимагатиме застосування перевірки орфографії за допомогою словників і перевірки правил, як простих, так і заснованих на результаті парсингу залежностей або інших ознак.

З-поміж трьох моделей, найкращі показники має попередньо навчена модель нейронного перекладу mT5. Найкраще значення як BLEU (0,908) , так і METEOR (0,956) отримано для mT5, що співпадає із аналізом прикладів, у якому найбільш точні виправлення помилок без зміни початкового значення речення отримані саме для цієї нейронної мережі. M2M100 має більшу оцінку BLEU (0,847), ніж "Ukrainian Roberta" Encoder-Decoder (0,697), однак, суб'єктивно оцінюючи результати виправлення прикладів, M2M100 значно гірше справляється із цим завданням, ніж дві інші моделі. Для METEOR також M2M100 (0,925) має більшу оцінку, ніж «Ukrainian Roberta» Encoder-Decoder (0,876).

© Холодна Н. М., Висоцька В. А., 2023
DOI 10.15588/1607-3274-2023-1-12

З метою економії обчислювальних ресурсів можливим також є застосування попередньо навченої нейронної мережі типу BERT, використовуючи її як у якості енкодера, так і декодера. Така нейронна мережа має вдвічі менше параметрів, ніж інші попередньо навчені MT-моделі, і показує задовільні GEC-результати.

ПОДЯКИ

Роботу виконано в рамках держбюджетної теми «Методи та засоби функціонування систем підтримки прийняття рішень на основі онтологій» (ID:839 2017-05-15 09:20:01 (2459-315)). Дослідження провадилось в межах спільних наукових досліджень кафедри інформаційних систем та мереж НУ «Львівська політехніка» на тему «Дослідження, розроблення і впровадження інтелектуальних розподілених інформаційних технологій та систем на основі ресурсів баз даних, сховищ даних, просторів даних та знань з метою прискорення процесів формування сучасного інформаційного суспільства». Наукові дослідження провадилися також в рамках ініціативної тематики досліджень кафедри ICM НУ «Львівська політехніка» на тему «Розроблення інтелектуальних розподілених систем на основі онтологічного підходу з метою інтеграції інформаційних ресурсів».

ЛІТЕРАТУРА

1. Naghshnejad M. Recent Trends in the Use of Deep Learning Models for Grammar Error Handling / M. Naghshnejad, T. Joshi, V. N. Nair // ArXiv. – 2020. DOI: 10.48550/arXiv.2009.02358
2. Automated Grammatical Error Detection for Language Learners, Second Edition / [C. Leacock, M. Chodorow, M. Gamon, J. Tetreault] // Synthesis Lectures on Human Language Technologies. – 2014, Berlin : Springer. – 154 p. DOI: 10.1007/978-3-031-02153-4
3. Attention Is All You Need / [A. Vaswani, N. Shazeer, N. Parmar et al.] // ArXiv. – 2017. DOI: 10.48550/arXiv.1706.03762
4. Transformers: State-of-the-Art Natural Language Processing / [T. Wolf, L. Debut, V. Sanh et al.] // EMNLP 2020: Conference on Empirical Methods in Natural Language Processing: System Demonstrations, Online, ALC Anthology, Oct. 2020 : proceedings. – P. 38–45. DOI: 10.18653/v1/2020.emnlp-demos.6
5. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding / [J. Devlin, M.-W. Chang, K. Lee, K. Toutanova] // Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies Minneapolis, Minnesota, ALC Anthology, June 2019 : proceedings. – P. 4171–4186. DOI: 10.18653/v1/N19-1423
6. Rozovskaya A. Grammar Error Correction in Morphologically Rich Languages: The Case of Russian / A. Rozovskaya, D. Roth // Transactions of the Association for Computational Linguistics. – 2019. – Vol. 7. – P. 1–17. DOI: 10.1162/tacl_a_00251
7. Bick E. DanProof: Pedagogical Spell and Grammar Checking for Danish / E. Bick // International Conference Recent Advances in Natural Language Processing, Hissar, Bulgaria, Sep. 2015 : proceedings. – P. 55–62.
8. Design and construction of the Greek grammar checker / [P. Gakis, C. T. Panagiotakopoulos, K. N. Sgarbas et al.] //

OPEN ACCESS

Digital Scholarship in the Humanities. – 2017. – Vol. 32. – P. 554–576. DOI: 10.1093/llc/fqw025
9. Deksne D. A New Phase in the Development of a Grammar Checker for Latvian / D. Deksne // Human Language Technologies – The Baltic Perspective. – IOS Press, 2016. – P. 147–152. DOI: 10.3233/978-1-61499-701-6-147
10. Sorokin A. Spelling Correction for Morphologically Rich Language: a Case Study of Russian / A. Sorokin // 6th Workshop on Balto-Slavic Natural Language Processing, Valencia, Spain, Apr. 2017 : proceedings. – P. 45–53. DOI: 10.18653/v1/W17-1408.
11. Gill M. S. A Grammar Checking System for Punjabi / M. S. Gill, G. S. Lehal // Coling 2008: Companion volume: Demonstrations, Manchester, UK, Aug. 2008 : proceedings. – P. 149–152.
12. Go M. P. Developing an Unsupervised Grammar Checker for Filipino Using Hybrid N-grams as Grammar Rules / M. P. Go, A. Borra // PACLIC: 30th Pacific Asia Conference on Language, Information and Computation: Oral Papers, Seoul, South Korea, Oct. 2016 : proceedings. – P. 105–113.
13. Shaalan K. F. Arabic GramCheck: a grammar checker for Arabic / K. F. Shaalan // Software: Practice and Experience. – 2005. – Vol. 35(7). – P. 643–665. DOI: 10.1002/spe.653
14. A Comprehensive Survey of Grammar Error Correction / [Y. Wang, Y. Wang, J. Liu, Z. Liu] // ArXiv. – 2020. DOI: 10.48550/arXiv.2005.06600
15. Syvokon O. UA-GEC: Grammatical Error Correction and Fluency Corpus for the Ukrainian Language / O. Syvokon, O. Nahorna // ArXiv. – 2021. DOI: 10.48550/arXiv.2103.16997
16. Lardinois F. Grammarly goes beyond grammar / F. Lardinois // TechCrunch. – 2019. – Access mode: https://techcrunch.com/2019/07/16/grammarly-goes-beyond-grammar/.
17. Lardinois F. Grammarly gets a tone detector to keep you out of email trouble / F. Lardinois // TechCrunch. – 2019. – Access mode: https://techcrunch.com/2019/09/24/grammarly-gets-a-tone-detector-to-keep-you-out-of-email-trouble/.
18. Grammarly Inc. About Us / Grammarly Inc. – Access mode: https://www.grammarly.com/about.
19. Grammarly Inc. Does Grammarly support languages other than English? / Grammarly Inc. – Access mode: https://support.grammarly.com/hc/en-us/articles/115000090971-Does-Grammarly-support-languages-other-than-English-.
20. LanguageTool. Languages / LanguageTool. – Access mode: https://dev.languagetool.org/languages.
21. LanguageTool. Error Rules for LanguageTool / LanguageTool Community. – Access mode: https://community.languagetool.org/rule/list?offset=0&max=10&lang=uk&filter=&categoryFilter=&_action_list=%D0%A4%D1%96%D0%BB%D1%8C%D1%82%D1%80.
22. LanguageTool. About / LanguageTool. – Access mode: https://languagetool.org/about.
23. Jayanthi S. NeuSpell: A Neural Spelling Correction Toolkit / S. Jayanthi, D. Pruthi, G. Neubig // ArXiv. – 2020. DOI: 10.48550/arXiv.2010.11085
24. Hunspell, Github. – 2021. – Access mode: https://github.com/hunspell/hunspell.
25. Korobov M. Morphological Analyzer and Generator for Russian and Ukrainian Languages / M. Korobov // ArXiv. – 2015. DOI: 10.48550/arXiv.1503.07283
26. Tmienova N. System of Intellectual Ukrainian Language Processing / N. Tmienova, B. Sus // ITS: the XIX International Conference on Information Technologies and Security, Kyiv, Ukraine, Nov. 28, 2019 : proceedings. – P. 199–209.
27. Pogorilyy S. Method of noun phrase detection in Ukrainian texts / S. Pogorilyy, A. A. Kramov // ArXiv. – 2020. DOI: 10.48550/arXiv.2010.11548
28. Глибовець А. Алгоритм токенізації та стемінгу для текстів українською мовою / А. Глибовець, В. Точицький // Наукові записки НаУКМА. Комп'ютерні науки. – 2017. – Т. 198. – С. 4–8.
29. Hao S. A Research on Online Grammar Checker System Based on Neural Network Model / S. Hao, G. Hao // Journal of Physics. – 2020. – Vol. 1651. – P. 1–8. DOI: 10.1088/1742-6596/1651/1/012135
30. Батюк Т. М. Технологія соціалізації особистостей за спільними інтересами на основі методів машинного навчання та seo-технологій / Т. М. Батюк, В. А. Висоцька // Радіоелектроніка, інформатика, управління. – 2022. – № 2 (61). – С. 53–68. DOI: 10.15588/1607-3274-2022-2-6
31. The CoNLL-2014 Shared Task on Grammatical Error Correction / [H. T. Ng, S. M. Wu, T. Briscoe, et al.] // Conference on Computational Natural Language Learning: Shared Task, Baltimore, Maryland, Jun. 2014 : proceedings. – P. 1–14. DOI: 10.3115/v1/W14-1701
32. Chollampatt S. A Multilayer Convolutional Encoder-Decoder Neural Network for Grammatical Error Correction / S. Chollampatt, H. T. Ng // ArXiv. – 2018. DOI: 10.48550/arXiv.1801.08831
33. GECToR – Grammatical Error Correction: Tag, Not Rewrite / [K. Omelianchuk, V. Atrasevych, A. N. Chernodub, O. Skurzhanskyi] // ArXiv. – 2020. DOI: 10.48550/arXiv.2005.12592
34. The University of Illinois System in the CoNLL-2013 Shared Task / [A. Rozovskaya, K.-W. Chang, M. Sammons, D. Roth] // Conference on Computational Natural Language Learning: Shared Task, Sofia, Bulgaria, Aug. 2013 : proceedings. – P. 13–19.
35. The CoNLL-2013 Shared Task on Grammatical Error Correction /[H. T. Ng, S. M. Wu, Y. Wu et al.] // Conference on Computational Natural Language Learning: Shared Task, Sofia, Bulgaria, Aug. 2013 : proceedings. – P. 1–12.
36. A Simple Recipe for Multilingual Grammatical Error Correction / [S. Rothe, J. Mallinson, E. Malmi et al.] // ArXiv. – 2021. DOI: 10.48550/arXiv.2106.03830
37. Mita M. Do Grammatical Error Correction Models Realize Grammatical Generalization? / M. Mita, H. Yanaka // ArXiv. – 2021. DOI: 10.48550/arXiv.2106.03031
38. A Unified Strategy for Multilingual Grammatical Error Correction with Pre-trained Cross-Lingual Language Model / [X. Sun, T. Ge, S. Ma et al.] // ArXiv. – 2022. DOI: 10.48550/arXiv.2201.10707
39. Yasunaga M. LM-Critic: Language Models for Unsupervised Grammatical Error Correction / M. Yasunaga, J. Leskovec, and P. Liang // ArXiv. – 2021. DOI: 10.48550/arXiv.2109.06822
40. A Neural Grammatical Error Correction System Built on Better Pre-training and Sequential Transfer Learning / [Y. J. Choe, J. Ham, K. Park, Y. Yoon] // Workshop on Innovative Use of NLP for Building Educational Applications, Florence, Italy, Aug. 2019 : proceedings. – P. 213–227. DOI: 10.18653/v1/W19-4423

© Холодна Н. М., Висоцька В. А., 2023
DOI 10.15588/1607-3274-2023-1-12

41. Wan Z. A Syntax-Guided Grammatical Error Correction Model with Dependency Tree Correction / Z. Wan, X. Wan // ArXiv. – 2021. DOI: 10.48550/arXiv.2111.03294
42. Neural Language Correction with Character-Based Attention / [Z. Xie, A. Avati, N. Arivazhagan et al.] // ArXiv. – 2016. DOI: 10.48550/arXiv.1603.09727
43. Parnow K. Grammatical Error Correction as GAN-like Sequence Labeling / K. Parnow, Z. Li, H. Zhao // ArXiv. – 2021. DOI: 10.48550/arXiv.2105.14209
44. Wang X. Research and Implementation of English Grammar Check and Error Correction Based on Deep Learning / X. Wang, W. Zhong // Scientific Programming. – 2022. – Vol. 2022. – Article ID 4082082. DOI: 10.1155/2022/4082082
45. Grammatical Error Correction in Low-Resource Scenarios / J. Náplava, M. Straka // W-NUT: 5th Workshop on Noisy User-generated Text, Hong Kong, China, Nov. 2019 : proceedings. – P. 346–356. DOI: 10.18653/v1/D19-5545.
46. Improving Grammatical Error Correction with Machine Translation Pairs / [W. Zhou, T. Ge, C. Mu et al.] // ArXiv. – 2020. DOI: 10.48550/arXiv.1911.02825
47. Raheja V. Adversarial Grammatical Error Correction / V. Raheja, D. Alikaniotis // EMNLP: Findings of the Association for Computational Linguistics, Online, Nov. 2020 : proceedings. – P. 3075–3087. DOI: 10.18653/v1/2020.findings-emnlp.275
48. Radchenko V. Ukrainian Roberta / V. Radchenko. Access mode: https://github.com/youscan/language-models.
49. mT5: A Massively Multilingual Pre-trained Text-to-Text Transformer / [L. Xue, N. Constant, A. Roberts et al.] // Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Online, Jun. 2021 : proceedings. – P. 483–498. DOI: 10.18653/v1/2021.naacl-main.41.
50. Beyond English-Centric Multilingual Machine Translation / [A. Fan, S. Bhosale, H. Schwenk et al.]// ArXiv. – 2020. DOI: 10.48550/arXiv.2010.11125
51. Multilingual Translation with Extensible Multilingual Pretraining and Finetuning / [Y. Tang, C. Tran, Xian Li et al.] // ArXiv. – 2020. DOI: 10.48550/arXiv.2008.00401
52. Platen von Patrick Leveraging Pre-trained Language Model Checkpoints for Encoder-Decoder Models / Platen von Patrick. – Access: https://huggingface.co/blog/warm-starting-encoder-decoder.
53. Rothe S. Leveraging Pre-trained Checkpoints for Sequence Generation Tasks / S. Rothe, S. Narayan, A. Severyn // ArXiv. – 2019. DOI: 10.48550/arXiv.1907.12461
54. Ground truth for grammatical error correction metrics / [C. Napoles, K. Sakaguchi, M. Post, J. Tetreault] // 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing, Beijing, China, July 2015 : proceedings. – P. 588–593. DOI: 10.3115/v1/P15-2097
55. Bleu: a Method for Automatic Evaluation of Machine Translation / [K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu] // 40th Annual Meeting of the Association for Computational Linguistics, Philadelphia, Pennsylvania, USA, July 2002 : proceedings. – P. 311–318. DOI: 10.3115/1073083.1073135.
56. Banerjee S. METEOR: An Automatic Metric for MT Evaluation with Improved Correlation with Human Judgments / S. Banerjee, A. Lavie // ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization, Michigan, Jun. 2005 : proceedings. – P. 65–72.
57. Platen von Patrick Encoder-Decoder models don't need costly pre-training to yield state-of-the-art results on seq2seq tasks / Platen von Patrick. – Access mode: https://twitter.com/patrickplaten/status/1325844244095971328.

Accepted 19.09.2022.
Received 03.12.2022.

UDC 004.9

TECHNOLOGY FOR GRAMMATICAL ERRORS CORRECTION IN UKRAINIAN TEXT CONTENT BASED ON MACHINE LEARNING METHODS

Kholodna N. – PhD student of Information Systems and Networks Department, Lviv Polytechnic National University, Lviv, Ukraine.

Vysotska V. – PhD, Associate Professor of Information Systems and Networks Department, Lviv Polytechnic National University, Lviv, Ukraine.

ABSTRACT

Context. Most research in grammatical and stylistic error correction focuses on error correction in English-language textual content. Thanks to the availability of large data sets, a significant increase in the accuracy of English grammar correction has been achieved. Unfortunately, there are few studies on other languages. Systems for the English language are constantly developing and currently actively use machine learning methods: classification (sequence tagging) and machine translation. A large amount of parallel or manually labelled data is required to build a high-quality machine learning model for correcting grammatical/stylistic errors in the texts of those morphologically complex languages. Manual data annotation requires a lot of effort by professional linguists, which makes the creation of text corpora, especially in morphologically rich languages, mainly Ukrainian, a time- and resource-consuming process.

Objective of the study is to develop a technology for correcting errors in Ukrainian-language texts based on machine learning methods using a small set of annotated parallel data.

Method. For this study, machine learning algorithms were selected when developing a system for correcting errors in Ukrainian-language texts using an optimal pipeline, including pre-processing and selecting text content and generating features in small annotated data corpora. The neural network's use with a new architecture, a review of state-of-the-art methods, and a comparison of different pipeline stages will make it possible to determine such a combination of them, allowing a high-quality error correction model in Ukrainian-language texts.

Results. A machine learning model for error correction in Ukrainian-language texts has been developed. A universal scheme for creating an error correction system for different languages is proposed. According to the results, the neural network can correct simple sentences written in Ukrainian. However, creating a full-fledged system will require spell-checking using dictionaries and checking rules, both simple and based on the result of parsing dependencies or other features. The pre-trained neural translation

© Холодна Н. М., Висоцька В. А., 2023
DOI 10.15588/1607-3274-2023-1-12

model mT5 has the best performance among the three models. To save computing resources, it is also possible to use a pre-trained BERT-type neural network as an encoder and a decoder. Such a neural network has half the number of parameters as other pre-trained machine translation models and shows satisfactory results in correcting grammatical and stylistic errors.

Conclusions. The created model shows excellent classification results on test data. The calculated machine translation quality metrics allow only a partial comparison of the models since most of the words and phrases in the original and corrected sentences are the same. The best value for both BLEU (0.908) and METEOR (0.956) is obtained for mT5, which is consistent with the case study in which the most accurate error corrections without changing the initial value of the sentence are obtained for such a neural network. The M2M100 has a higher BLEU score (0.847) than the "Ukrainian Roberta" Encoder-Decoder (0.697). However, subjectively evaluating the results of the correction of examples, the M2M100 does a much worse job than the other two models. For METEOR, M2M100 (0.925) also has a higher score than the "Ukrainian Roberta" Encoder-Decoder (0.876).

KEYWORDS: NLP, text pre-processing, error correction, grammatical error correction, machine learning, deep learning, text analysis, text classification, neural network.

REFERENCES

1. Naghshnejad M., Joshi T., Nair V. N. Recent Trends in the Use of Deep Learning Models for Grammar Error Handling, *ArXiv,* 2020. DOI: 10.48550/arXiv.2009.02358
2. Leacock C., Chodorow M., Gamon M., Tetreault J. Automated Grammatical Error Detection for Language Learners, Second Edition, *Synthesis Lectures on Human Language Technologies,* 2014. Berlin, Springer, 154 p. DOI: 10.1007/978-3-031-02153-4
3. Vaswani A., Shazeer N., Parmar N., Uszkoreit J., Jones L., Gomez A. N., Kaiser L., Polosukhin I. Attention Is All You Need, *ArXiv,* 2017. DOI: 10.48550/arXiv.1706.03762
4. Wolf T., Debut L., Sanh V., Chaumond J., Delangue C., Moi A., Cistac P., Rault T., Louf R., Funtowicz M., Davison J., Shleifer S., Platen P. v., Ma C., Jernite Y., Plu J., Xu C., Scao T. L., Gugger S., Drame M., Lhoest Q., Rush A. Transformers: State-of-the-Art Natural Language Processing, *EMNLP 2020: Conference on Empirical Methods in Natural Language Processing: System Demonstrations, Online, ALC Anthology, Oct. 2020 : proceedings*, pp. 38–45. DOI: 10.18653/v1/2020.emnlp-demos.6
5. Devlin J., Chang M.-W., Lee K., Toutanova K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies Minneapolis.* Minnesota, ALC Anthology, June 2019 : proceedings, pp. 4171–4186. DOI: 10.18653/v1/N19-1423
6. Rozovskaya A., Roth D. Grammar Error Correction in Morphologically Rich Languages: The Case of Russian, *Transactions of the Association for Computational Linguistics*, 2019, Vol. 7, pp. 1–17. DOI: 10.1162/tacl_a_00251
7. Bick E. DanProof: Pedagogical Spell and Grammar Checking for Danish, *International Conference Recent Advances in Natural Language Processing*. Hissar, Bulgaria, Sep. 2015, proceedings, pp. 55–62.
8. Gakis P., Panagiotakopoulos C. T., Sgarbas K. N., Tsalidis C., Verykios V. S., Design and construction of the Greek grammar checker, *Digital Scholarship in the Humanities,* 2017, Vol. 32, pp. 554–576. DOI: 10.1093/llc/fqw025
9. Deksne D. A New Phase in the Development of a Grammar Checker for Latvian. Human Language Technologies, The Baltic Perspective, IOS Press, 2016, pp. 147–152. DOI: 10.3233/978-1-61499-701-6-147
10. Sorokin A. Spelling Correction for Morphologically Rich Language: a Case Study of Russian, *6th Workshop on Balto-Slavic Natural Language Processing, Valencia, Spain, Apr. 2017, proceedings*, pp. 45–53. DOI: 10.18653/v1/W17-1408.
11. Gill M. S., Lehal G. S. A Grammar Checking System for Punjabi, *Coling 2008: Companion volume: Demonstrations, Manchester, UK, Aug. 2008 : proceedings,* pp. 149–152.
12. Go M. P., Borra A. Developing an Unsupervised Grammar Checker for Filipino Using Hybrid N-grams as Grammar Rules, *PACLIC: 30th Pacific Asia Conference on Language, Information and Computation, Oral Papers*. Seoul, South Korea, Oct. 2016 : proceedings, pp. 105–113.
13. Shaalan K. F. Arabic GramCheck: a grammar checker for Arabic, *Software: Practice and Experience,* 2005, Vol. 35(7), pp. 643–665. DOI: 10.1002/spe.653
14. Wang Y., Wang Y., Liu J., Liu Z. A Comprehensive Survey of Grammar Error Correction, *ArXiv*, 2020. DOI: 10.48550/arXiv.2005.06600
15. Syvokon O., Nahorna O. UA-GEC: Grammatical Error Correction and Fluency Corpus for the Ukrainian Language *ArXiv,* 2021. DOI: 10.48550/arXiv.2103.16997
16. Lardinois F. Grammarly goes beyond grammar, *TechCrunch,* 2019. Access mode: https://techcrunch.com/2019/07/16/grammarly-goes-beyond-grammar/.
17. Lardinois F. Grammarly gets a tone detector to keep you out of email trouble, *TechCrunch,* 2019. Access mode: https://techcrunch.com/2019/09/24/grammarly-gets-a-tone-detector-to-keep-you-out-of-email-trouble/.
18. Grammarly Inc. About Us / Grammarly Inc. Access mode: https://www.grammarly.com/about.
19. Grammarly Inc. Does Grammarly support languages other than English? / Grammarly Inc. Access mode: https://support.grammarly.com/hc/en-us/articles/115000090971-Does-Grammarly-support-languages-other-than-English-.
20. LanguageTool. Languages, LanguageTool. Access mode: https://dev.languagetool.org/languages.
21. LanguageTool. Error Rules for LanguageTool / LanguageTool Community. Access mode: https://community.languagetool.org/rule/list?offset=0&max=10&lang=uk&filter=&categoryFilter=&_action_list=%D0%A4%D1%96%D0%BB%D1%8C%D1%82%D1%80.
22. LanguageTool. About / LanguageTool. Access mode: https://languagetool.org/about.
23. Jayanthi S., Pruthi D., Neubig G. NeuSpell: A Neural Spelling Correction Toolkit, *ArXiv,* 2020. DOI: 10.48550/arXiv.2010.11085
24. Hunspell, Github, 2021, Access mode: https://github.com/hunspell/hunspell.
25. Korobov M. Morphological Analyzer and Generator for Russian and Ukrainian Languages, *ArXiv,* 2015. DOI: 10.48550/arXiv.1503.07283
26. Tmienova N., Sus B. System of Intellectual Ukrainian Language Processing, *ITS: the XIX International Conference on Information Technologies and Security*, Kyiv, Ukraine, Nov. 28, 2019 : proceedings, pp. 199–209.

© Холодна Н. М., Висоцька В. А., 2023
DOI 10.15588/1607-3274-2023-1-12

27. Pogorilyy S., Kramov A. A. Method of noun phrase detection in Ukrainian texts, *ArXiv,* 2020. DOI: 10.48550/arXiv.2010.11548
28. Hlybovets A., Tochytskyi V. Algorithm of tokenization and stemming for texts in the Ukrainian language, *Scientific notes of NaUKMA. Computer Science,* 2017, Vol. 198, pp. 4–8.
29. Hao S., Hao G. A Research on Online Grammar Checker System Based on Neural Network Model, *Journal of Physics*, 2020, Vol. 1651, pp. 1–8. DOI: 10.1088/1742-6596/1651/1/012135
30. Batiuk T. M., Vysotska V. Technology for Personalities Socialization by Common Interests Based on Machine Learning Methods And SEO-Technologies, *Radio Electronics, Computer Science, Control,* 2022, Vol. 2 (61), pp. 53–68. DOI: 10.15588/1607-3274-2022-2-6
31. Ng H. T., Wu S. M., Briscoe T., Hadiwinoto C., Susanto R. H., Bryant C. The CoNLL-2014 Shared Task on Grammatical Error Correction, *Conference on Computational Natural Language Learning: Shared Task.* Baltimore, Maryland, Jun. 2014, proceedings, pp. 1–14. DOI: 10.3115/v1/W14-1701
32. Chollampatt S., Ng H. T. A Multilayer Convolutional Encoder-Decoder Neural Network for Grammatical Error Correction, *ArXiv,* 2018. DOI: 10.48550/arXiv.1801.08831
33. Omelianchuk K., Atrasevych V., Chernodub A. N., Skurzhanskyi O. GECToR – Grammatical Error Correction: Tag, Not Rewrite, *ArXiv,* 2020. DOI: 10.48550/arXiv.2005.12592
34. Rozovskaya A., Chang K.-W., Sammons M., Roth D. The University of Illinois System in the CoNLL-2013 Shared Task, *Conference on Computational Natural Language Learning: Shared Task*. Sofia, Bulgaria, Aug. 2013, proceedings, pp. 13–19.
35. Ng H. T., Wu S. M., Wu Y., Hadiwinoto C., Tetreault J. The CoNLL-2013 Shared Task on Grammatical Error Correction, *Conference on Computational Natural Language Learning: Shared Task*. Sofia, Bulgaria, Aug. 2013, proceedings, pp. 1–12.
36. Rothe S., Mallinson J., Malmi E., Krause S., Severyn A. A Simple Recipe for Multilingual Grammatical Error Correction, *ArXiv,* 2021. DOI: 10.48550/arXiv.2106.03830
37. Mita M., Yanaka H. Do Grammatical Error Correction Models Realize Grammatical Generalization?, *ArXiv,* 2021. DOI: 10.48550/arXiv.2106.03031
38. Sun X., Ge T., Ma S., Li J., Wei F., and Wang H. A Unified Strategy for Multilingual Grammatical Error Correction with Pre-trained Cross-Lingual Language Model, *ArXiv,* 2022. DOI: 10.48550/arXiv.2201.10707
39. Yasunaga M., Leskovec J., and Liang P. LM-Critic: Language Models for Unsupervised Grammatical Error Correction, *ArXiv,* 2021. DOI: 10.48550/arXiv.2109.06822
40. Choe Y. J., Ham J., Park K., Yoon Y. A Neural Grammatical Error Correction System Built on Better Pre-training and Sequential Transfer Learning, *Workshop on Innovative Use of NLP for Building Educational Applications*, Florence. Italy, Aug. 2019 : proceedings, pp. 213–227. DOI: 10.18653/v1/W19-4423
41. Wan Z., Wan X. A Syntax-Guided Grammatical Error Correction Model with Dependency Tree Correction, *ArXiv*, 2021. DOI: 10.48550/arXiv.2111.03294
42. Xie Z., Avati A., Arivazhagan N., Jurafsky D., Ng A. Neural Language Correction with Character-Based Attention, ArXiv, 2016. DOI: 10.48550/arXiv.1603.09727
43. Parnow K., Li Z., Zhao H. Grammatical Error Correction as GAN-like Sequence Labeling, *ArXiv,* 2021. DOI: 10.48550/arXiv.2105.14209
44. Wang X., Zhong W. Research and Implementation of English Grammar Check and Error Correction Based on Deep Learning, *Scientific Programming,* 2022, Vol. 2022, Article ID 4082082. DOI: 10.1155/2022/4082082
45. Náplava J., Straka M. Grammatical Error Correction in Low-Resource Scenarios, *W-NUT: 5th Workshop on Noisy User-generated Text.* Hong Kong, China, Nov. 2019 : proceedings, pp. 346–356. DOI: 10.18653/v1/D19-5545.
46. Zhou W., Ge T., Mu C., Xu K., Wei F., Zhou M. Improving Grammatical Error Correction with Machine Translation Pairs, *ArXiv,* 2020. DOI: 10.48550/arXiv.1911.02825
47. Raheja V., Alikaniotis D. Adversarial Grammatical Error Correction, *EMNLP: Findings of the Association for Computational Linguistics, Online, Nov. 2020 : proceedings*, pp. 3075–3087. DOI: 10.18653/v1/2020.findings-emnlp.275
48. Radchenko V. Ukrainian Roberta. Access mode: https://github.com/youscan/language-models.
49. Xue L., Constant N., Roberts A., Kale M., Al-Rfou R., Siddhant A., Barua A., Raffel C. mT5: A Massively Multilingual Pre-trained Text-to-Text Transformer, Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Online, Jun. 2021 : proceedings, pp. 483–498. DOI: 10.18653/v1/2021.naacl-main.41.
50. Fan A., Bhosale S., Schwenk H., Ma Z., El-Kishky A., Goyal S., Baines M., Celebi O., Wenzek G., Chaudhary V., Goyal N., Birch T., Liptchinsky V., Edunov S., Grave E., Auli M., Joulin A. Beyond English-Centric Multilingual Machine Translation, *ArXiv,* 2020. DOI: 10.48550/arXiv.2010.11125
51. Tang Y., Tran C., Li Xian, Chen P.-J., Goyal N., Chaudhary V., Gu J., Fan A. Multilingual Translation with Extensible Multilingual Pretraining and Finetuning, *ArXiv,* 2020. DOI: 10.48550/arXiv.2008.00401
52. Platen von Patrick Leveraging Pre-trained Language Model Checkpoints for Encoder-Decoder Models. Access: https://huggingface.co/blog/warm-starting-encoder-decoder.
53. Rothe S., Narayan S., Severyn A. Leveraging Pre-trained Checkpoints for Sequence Generation Tasks, *ArXiv,* 2019. DOI: 10.48550/arXiv.1907.12461
54. Napoles C., Sakaguchi K., Post M., Tetreault J. Ground truth for grammatical error correction metrics, *53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing.* Beijing, China, July 2015 : proceedings, pp. 588–593. DOI: 10.3115/v1/P15-2097
55. Papineni K., Roukos S., Ward T., and Zhu W.-J. Bleu: a Method for Automatic Evaluation of Machine Translation, *40th Annual Meeting of the Association for Computational Linguistics, Philadelphia, Pennsylvania, USA, July 2002 : proceedings*, pp. 311–318. DOI: 10.3115/1073083.1073135.
56. Banerjee S., Lavie A. METEOR: An Automatic Metric for MT Evaluation with Improved Correlation with Human Judgments, *ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization.* Michigan, Jun. 2005 : proceedings, pp. 65–72.
57. Platen von Patrick Encoder-Decoder models don't need costly pre-training to yield state-of-the-art results on seq2seq tasks. Access mode: https://twitter.com/patrickplaten/status/1325844244095971328.

© Холодна Н. М., Висоцька В. А., 2023
DOI 10.15588/1607-3274-2023-1-12