КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА
Факультет комп'ютерних наук та кібернетики
Кафедра інтелектуальних програмних систем

Кваліфікаційна робота

на здобуття освітнього рівня бакалавра

за спеціальністю 121 Інженерія програмного забезпечення

на тему:

ОБРОБКА ПРИРОДНОЇ МОВИ З ВИКОРИСТАННЯМ ТЕХНОЛОГІЙ ГЛИБИННОГО НАВЧАННЯ

Виконала студентка 4-го курсу
Анна ХОДИРЕВА _____
(підпис)

Науковий керівник:
доцент, кандидат фіз.-мат. наук
Ярослав ЛІНДЕР _____
(підпис)

Засвідчую, що в цій роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Роботу розглянуто й допущено до захисту на засіданні кафедри інтелектуальних програмних систем
28 травня 2021 р.,
протокол № 14
Завідувач кафедри
Олександр ПРОВОТАР _____
(підпис)

Київ - 2021

РЕФЕРАТ

Обсяг роботи 43 сторінок, 2 ілюстрацій, 2 таблиці, 13 джерел посилань.

Ключові слова: PYTHON, ОБРОБКА ПРИРОДНОЇ МОВИ, ЗМІНА СТИЛЮ ТЕКСТУ, НЕЙРОННІ МЕРЕЖІ, НАВЧАННЯ З ПІДКРІПЛЕННЯМ, ТРАНСФОРМЕРИ, ФОРМАЛІЗАЦІЯ ТЕКСТУ, ГЕНЕРАТИВНО-ЗМАГАЛЬНІ МЕРЕЖІ.

Мета роботи: проектування та розробка системи обробки природної мови з використанням технологій глибинного навчання, яка дозволяє виконувати перетворення стилю тексту зі збереженням його семантичного значення на прикладі задач формалізації та деформалізації тексту, теоретичний опис та систематизація поточних методів та алгоритмів обробки природної мови, застосування їх на практиці, оцінка їх роботи, порівняння ефективності розроблених методів для протилежних задач.

Методи розробки: мова програмування – Python. Використано бібліотеку PyTorch для глибинного навчання, бібліотеку Transformers з попередньо навченими моделями GPT і BERT.

Результати роботи: в ході проведеної роботи вдалося навчити модель, здатну виконувати перетворення стилю тексту зі збереженням його семантичного значення без використання паралельного корпусу даних, використовуючи техніки змагального тренування і тренування з підкріпленням. Оцінка моделі показала добрі результати. В процесі роботи був проаналізований розвиток обробки природної мови за останні роки, були отримані знання стосовно найсучасніших архітектур моделей для обробки природної мови (GPT – для генерації природної мови, BERT – для аналізу природної мови), механізму уваги, застосування навчання з підкріпленням та генеративно-змагальних мереж в обробці природної мови, перетворення текстового стилю та оцінки його якості.

Областю застосування мовних моделей є інтерфейси типу машина-людина, задачі аналізу великих об’ємів текстових або голосових даних. Областю застосування систем перетворення стилю тексту є редагування тексту для покращення у відповідності з деяким критерієм, фільтрація небажаного контенту. Система формалізації тексту може бути застосована як щоденний інструмент для редагування написаних повідомлень, офіційних документів, тощо.

Дана робота може бути використана як основа для подальших досліджень в області обробки природної мови, перетворення стилю тексту, формалізації тексту.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАЧЕННЯ

NLP – Natural Language Processing, обробка природної мови;

LSTM – Long Short-Term Memory, довга короткочасна пам’ять;

CNN – Convolutional Neural Network, згорткова нейронна мережа;

RNN – Recurrent Neural Network, рекурентна нейронна мережа;

Seq2Seq (модель) – Sequence-to-sequence, тип моделей, в яких послідовність на вході перетворюється в іншу послідовність на виході;

GPT – Generative Pre-trained language model, модель OpenAI на базі архітектури Трансформер для генерації текста, близького до написаного людиною;

BERT - Bidirectional Encoder Representations from Transformers, мовна модель від Google на базі архітектури Трансформер, що показала передові результати в задачах розуміння природної мови;

TPU – Tensor Processing Unit, тензорний процесор;

GAN – Generative-Adversarial Networks, генеративно-змагальні нейронні мережі;

WMD – Word Mover’s Distance, алгоритм для обчислення відстані між документами;

IDE – Integrated Development Editor, інтегрована середа розробки.

ВСТУП

Оцінка сучасного стану об’єкта розробки. Останнім часом область обробки природної мови стрімко розвивається завдяки росту доступності даних і обчислюваних потужностей, а також нещодавно запропонованим архітектурам і методам. Зростаючий вплив NLP у світі аналізу великих даних обумовлений тим, що нові просування в області обробки природної мови вирішують багато проблем попередників і доводять, що машини здатні навіть перевершити людські показники в деяких задачах. Застосування NLP включає аналіз настроїв – де модель NLP може передбачити тип настрою, який виражає фрагмент тексту, віртуальні чат-боти – це роботи, які взаємодіють з людьми за допомогою тексту, маючи здатність розуміти та забезпечувати логічні відповіді на текстові повідомлення, надіслані людьми, розпізнавання мови – технологія, яка зазвичай використовується у перетворенні мови на текст, сьогодні є стандартною функцією мобільних телефонів.

Актуальність роботи та підстави для її виконання: Враховуючи велику кількість хаотичних мовних даних, яку виробляють люди кожного дня, можливість їх аналізувати в неупередженій та впорядкованій манері є вкрай актуальною. В області обробки природної мови існує мало прикладів застосування навчання з підкріпленням до моделей Трансформерів, попередні подібні роботи використовують RNN. Ця робота пропонує архітектуру моделі для навчання попередньо натренованих моделей без паралельного корпусу. Гарні результати в області перетворення стилю тексту досягаються за допомогою паралельного корпусу для двох стилів, але задача перетворення стилю значно ускладнюється, коли для неї немає паралельного навчального корпусу даних. В багатьох реальних сценаріях отримання високоякісних анотованих даних є

дорогим і трудомістким; навпаки, «сирі» приклади, що характеризують цільове завдання, загалом можна легко зібрати.

Мета й завдання роботи: Метою даної роботи є розробка системи обробки природної мови, що призначена для перетворення стилю тексту з ціллю його формалізації або деформалізації зі збереженням семантичного значення тексту. В роботі поставлені такі задачі як аналіз та опис поточних здобутків в області нейронних мереж, обробки природної мови, мовних моделей, архітектури Трансформер, а також представлення фреймворку для поєднання двох мовних моделей за допомогою методів навчання з підкріпленням в застосуванні до задачі перетворення стилю тексту з метою формалізації або деформалізації, аналіз результатів навчання моделі та висновки, які допоможуть вести подальші дослідження в цій області.

Об’єкт, методи й засоби дослідження або розроблення: Об’єктом дослідження вистувають мовні моделі, нейронні мережі. Об’єктом розроблення є система для навчання нейронної мережі, що комбінує в собі декілька мереж архітектури Трансформер та використовує змішаний багатоетапний стиль навчання.

Можливі сфери застосування: Сферами застосування мовних моделей є будь-які мовні задачі. Найактуальнішими є задачі аналізу великих об’ємів текстових або голосових даних, задачі побудування інтерфейсів для людського використання.

1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Основні поняття

Обробка природної мови (NLP, Natural Language Processing) – підрозділ лінгвістики, комп’ютерних наук і штучного інтелекту, що фокусується на взаємодії машин з людськими мовами. Якщо точніше, метою цього підрозділу є навчити комп’ютери обробляти і розуміти природну мову у спосіб, який не відрізняється від людського.

NLP поєднує в собі обчислювальну лінгвістику - моделювання людської мови, засноване на правилах, - зі статистичними моделями, моделями машинного навчання та глибокого навчання. Разом ці технології дозволяють комп’ютерам обробляти людську мову у вигляді текстових чи голосових даних та «розуміти» її повний зміст разом із задумами та настроями мовця чи автора. Основна здатність NLP – це структуризація величезної кількості хаотичних даних. NLP надає можливість розв’язати багатозначність природної мови і це робить його вкрай необхідним в задачах, де потрібен безкомпромісно правильний аналіз.

Обробка природної мови має дві складові: аналіз (розуміння або моделювання природної мови) та синтез (генерація текстів природною мовою). Розв'язання цих задач наближує до створення найбільш природної форми спілкування комп'ютера й людини, допоможе швидко обробляти великі об'єми «сирих» мовних даних без участі людини і робити це точніше, дешевше та надійніше.

Людська мова сповнена багатозначності, що надзвичайно ускладнює написання програм, які точно визначають контекст мовлення. Омоніми, омофони, сарказм, ідіоми, метафори, винятки з граматики та вживання, варіації в

структурі речень - це лише деякі з засобів людської мови, для вивчення яких навіть людям потрібні роки.

Формалізація тексту – це різновид перетворення текстового стилю. Перетворення текстового стилю – це перефразування тексту із вхідного стилю до цільового стилю зі збереженням семантичного значення. Перетворення стилю намагається контролювати деякі характеристики тексту, такі як емоціональність, формальність, ввічливість, а іноді навіть копіювати стиль якихось відомих авторів. Гарні результати в цій області досягаються за допомогою анотованого паралельного корпусу для двох стилів. Але задача перетворення текстового стилю значно ускладнюється, коли для неї немає паралельного навчального корпусу даних.

1.2 Нейронні мережі

Нейронна мережа – це обчислювальна система, яка складається з деякої кількості сильно зв’язаних вузлів, які здатні обробляти інформацію даючи відповідь на зовнішні вхідні дані, динамічно змінювати свій стан [4].

Нейронні мережі, як правило, мають шарову організацію. Шари складаються з ряду взаємозв’язаних вузлів, які містять «функцію активації». Дані подаються в мережу через вхідний шар, який передає дані одному або декільком прихованим шарам, де здійснюється фактична обробка системою зважених з'єднань.

Щільний шар – найпростіший і найпоширеніший у використанні шар нейронної мережі. Він є повнозв’язним, тобто кожен вузол одного шара пов’язаний з кожним вузлом іншого. Щільний шар виконує операцію на вході та повертає значення на вузли вихідного шару.

Більшість нейронних мереж мають певний оптимізаційний процес навчання, який шукає найкращі параметри нейронної мережі відповідно до функції втрат. Навчання нейронної мережі передбачає визначення специфічних обмежень, що виділяють навчання нейронних мереж серед інших задач оптимізації: астрономічне число параметрів, необхідність високого паралелізму при навчанні, багатокритеріальність вирішених завдань, необхідність знайти достатньо широку область, в якій всі значення мінімізованих функцій близькі до мінімального.

Функція втрат – це будь-яка функція, що обчислює помилку нейронної мережі.

Для задач класифікації найчастіше використовується перехресна ентропія, яка в загальному випадку обчислює різницю між двома розподілами ймовірностей.

Для задач регресії найпоширенішою функцією втрат являється середня квадратична похибка (Mean squared error). Ця функція обчислюється за середнім значенням квадратичних різниць між фактичними (цільовими) та прогнозованими значеннями.

З точки зору оптимізаційного процесу нейронна мережа – це параметрична функція виду:

$$y = F_\theta(x),$$

де θ – це параметри функції.

Як правило, модель нейронної мережі навчається за допомогою алгоритму градієнтного спуску, а ваги оновлюються за допомогою алгоритму зворотного розповсюдження помилки. Це додає нейронній мережі як параметричній функції окрім умови обчислюваності умову на диференційованість.

Алгоритм: Backpropagation

1) Ініціалізувати ваги w_{ij}

2) Подати дані x_i *на вхід та отримати вихід* o_i

3) Порахуати значення помилки

$\delta_k = o_k(1 - o_k)(t_k - o_k)$

4)Визначити загальну помилку

$\delta_j = o_j(1 - o_j) \sum \delta_k w_{j,k}$

5) Оновити значення вагів

$\Delta w_{i,j} = \alpha \Delta w_{i,j} + (1 - \alpha)\eta \delta_j o_i$

$w_{i,j} = w_{i,j} + \Delta w_{i,j}$

6) Повторити кроки 2-5 до отримання бажаного результату

Рисунок 1 – Псевдокод алгоритму зворотного розповсюдження помилок

Метод зворотного розповсюдження помилки працює за допомогою алгоритму градієнтного спуску, який є методом оптимізації моделі за допомогою руху в напрямку градієнта цільової функції. Класичний метод градієнтного спуску має стабільну швидкість збіжності (лінійну), але повільну швидкість ітерації, особливо при великому наборі навчальних даних. Це обумовлено тим, що лише для одного кроку потрібно пропустити через нейронну мережу увесь набір даних, обчислити помилку і корекцію вагів для кожного екземпляра, лише після цього обчислити сумарну корекцію вагів для усього набору і виконати її. Це з урахуванням великих розмірів сучасних нейронних мереж та корпусів даних є неефективним за часом, витраченим на одну ітерацію.

Крім того, в реальних умовах присутні такі проблеми, що погіршують роботу звичайного градієнтного спуску:

- Коли моделі потрапляє в локальний мінімум або сідлову точку, вона затримується в ній, і можливо ніколи не зможе з неї вийти, виходить, що навчання зупиняється.

- Складний ландшафт цільової функції: плато поряд з регіонами сильної нелінійності. Похідна на плато практично дорівнює нулю, а раптовий обрив, навпаки, може перенести занадто далеко.
- Занадто мала швидкість навчання змушує алгоритм збігатися дуже довго і затримуватись в локальних мінімумах, занадто велика – пропускати вузькі мінімуми або взагалі розходитися.
- Деякі параметри оновлюються значно рідше інших, особливо коли в даних зустрічаються інформативні, але рідкісні ознаки, що погано позначається на нюансах узагальнюючого правила мережі. З іншого боку, надання занадто великої значимості взагалі всім рідко зустрічається ознаками може привести до перенавчання.

Щоб подолати ці проблеми існує багато модифікацій алгоритму градієнтного спуску, кожен з яких використовує різні прийоми щоб поліпшити показники навчання.

Широко використовується стохастичний градієнтний спуск, який дозволяє на кожному кроці градієнтного спуску використовувати для обчислення корекції вагів випадково обрану підмножину вхідного набору даних. Це дає значно швидші ітерації в обмін на зменшену швидкість збіжності алгоритму. Також обчислені градієнти не співпадають з реальним напрямком зростання функції. Можна підібрати такий розмір підмножини вхідного набору, щоб обчислення одного кроку не було занадто довгим, а точність градієнту була достатня.

Adam (Adaptive Momentum) – це алгоритм оптимізації, який можна використовувати замість стохастичного градієнтного спуску для оновлення вагових коефіцієнтів мережі ітеративно на основі навчальних даних. [1] Це метод з адаптивною швидкістю навчання, тобто він обчислює індивідуальну швидкість навчання для різних параметрів. Adam використовує оцінки першого та другого моментів градієнта, щоб адаптувати швидкість навчання для кожної ваги нейронної мережі.

Ослаблення ваг або L_2 регуляризація – це метод регуляризації, що застосовується до ваг нейронної мережі [3]. Він намагається звести до мінімуму функцію втрат, яка порушує як основну функцію втрат, так і штраф за L_2 нормою ваг:

$$L_{new}(w) = L_{original}(w) + \lambda w^T w$$

де λ – це значення, що визначає силу штрафу.

AdamW – це виправлена модифікація алгоритма Adam, в якій було змінено типову імплементацію ослаблення ваг, відокремивши накладення штрафу ослаблення ваг від оновлення градієнту. В Adam ослаблення ваг додається до градієнту, а в AdamW це значення напряму віднімається від ваг. L_2 регулярізація в Adam зазвичай виконується за такою формулою, де w_t – це швидкість ослаблення ваг в час t:

$$g_t = \nabla f(\theta_t) + w_t \theta_t$$

в той час як AdamW коригує ослаблення ваг, щоб воно відображалося в оновленні градієнта, :

$$\theta_{t+1,i} = \theta_{t,i} - \eta \left(\frac{1}{\sqrt{\hat{v}_t + \epsilon}} * \hat{m}_t + w_{t,i} \theta_{t,i} \right), \forall t$$

1.3 Мовні моделі

Мовна модель – це модель, яка здатна апроксимувати ймовірнісний розподіл послідовності лінгвістичних елементів (токенів, в ролі яких можуть виступати слова, морфеми, речення, тощо). Мовні моделі застосовують цей розподіл в мовних задачах для точного прогнозування або створення нових послідовностей.

Є два типи мовних моделей:

- Статистичні мовні моделі: Ці моделі використовують традиційні статистичні методи, такі як N-грами, приховані моделі Маркова (Hidden Markov Models) та певні лінгвістичні правила для вивчення розподілу ймовірностей слів.
- Нейронні мовні моделі: Це нові моделі в області НЛП, які за своєю ефективністю перевершили статистичні мовні моделі. Вони використовують різні види нейронних мереж для моделювання мови. Таку мовну модель можна навчити за допомогою стандартних нейронних алгоритмів, таких як стохастичний градієнтний спуск із зворотним поширенням. Далі в роботі будемо розглядати тільки нейронні моделі, адже вони майже витіснили в застосуванні інші типи окрім найпростіших задач.

1.4 Seq2Seq моделі

Для більшості задач NLP популярністю користуються Seq2Seq моделі. Seq2Seq - це архітектура нейронної мережі, яка перетворює задану послідовність елементів, наприклад послідовність слів у реченні, в іншу послідовність. Моделі Seq2Seq особливо підходять для задач перекладу, коли послідовність слів з однієї мови перетворюється на послідовність різних слів на іншій мові. Насправді будь-яку задачу, де послідовність перетворюється на послідовність, можна звести до задачі перекладу, якщо використати розширене поняття мови. Це значить - можна припустити, що вхідна та вихідна послідовності задані якимись абстрактними невідомими мовами, і дозволити моделі Seq2Seq самостійно виявити закономірності перекладу між ними. Цей підхід довів свою ефективність на практиці.

Моделі Seq2Seq складаються з Енкодера (Encoder) та Декодера (Decoder). Суть моделі Seq2Seq в тому, що для перекладу мови A в мову B використовується деяка третя абстрактна мова C (мова C зазвичай має вищий вимірний простір, тобто текст такою мовою – це n-вимірний вектор), яка є спільною для Енкодера і Декодера. Енкодер приймає вхідну послідовність мовою A, відображає її у мову C. Цей абстрактний вектор далі передається Декодеру. Оскільки Декодер здатний читати мову C, він перетворює його у вихідну послідовність мовою B. Вихідна послідовність може бути іншою мовою, символами, копією вводу в залежності від задачі. Таким чином виходить, що модель здатна перекладати мову A в мову B.

1.5 Рекурентні нейронні мережі

Модель, яка була типовою для задач NLP впродовж багатьох років, називається рекуррентною нейронною мережею (RNN), зокрема один із її варіантів, мережа довготривалої пам'яті (LSTM), яка створена для того, щоб вирішувати проблему зникаючого градієнта [12] .

RNN читає текст в одному напрямку слово за словом, це значить, що вона розуміє слова, з якими стикається в тексті, враховуючи значення слів, які зустрічалися раніше. Тож RNN, як правило, має набагато більше інформації, щоб зрозуміти значення слова наприкінці текстової послідовності ніж на початку, оскільки вона накопичує контекст із кожним наступним прочитаним вхідним словом. Частина контексту, необхідна для розуміння слова, може бути далі за реченням, яке ще не було до кінця прочитане, і це погіршує ефективність роботи моделі. Цей недолік можна частково виправити, обробляючи речення у двох напрямках за допомогою двоспрямованих LSTMS. RNN моделі також мають проблеми з пам’яттю. Вони погано запам’ятовують інформацію про дальні залежності (слова, які зустрічались давно, але якимось чином пов’язані із

наступним словом) тому, що їм доводиться читати кожне слово поетапно. Доведено, що чим більше кроків потребує визначення контексту, тобто чим далі знаходяться ключові слова одне від одного, тим більше часу потрібно RNN для його вивчення [7].

Послідовний характер RNN також ускладнює ефективне використання сучасних швидких обчислювальних пристроїв, таких як TPU та GPU, яким для повного розкриття можливістей необхідне розпаралелювання виконуваного програмного забезпечення.

1.6 Механізм уваги

Згодом з’явився новий підхід, в якому відсутні усі недоліки RNN – механізм уваги. Розглянемо детальніше принцип його роботи.

Механізм уваги приймає на вхід два речення, перетворює їх на матрицю, де слова одного речення утворюють стовпці, а слова іншого речення утворюють рядки, а потім він шукає відповідності і проставляє їм значення в матриці, визначаючи контекст між реченнями [5]. Це дозволяє набагато більше, ніж просто зіставляти два речення написані різними мовами і співвідносити їх значення. Можна поставити речення проти самого себе вздовж стовпців та рядків, щоб зрозуміти, як деякі частини цього речення співвідносяться з іншими.

Нейронна мережа, озброєна механізмом уваги, насправді може зрозуміти наприклад, де в тексті знаходиться ключове слово, на яке вказує займенник. Тобто вона знає, як ігнорувати шум і зосередитись на тому, що актуально, як з'єднати контекстом слова, які самі по собі не вказують одне на одного. Отже, механізм уваги, на відміну від RNN, дозволяє поглянути на велику картину сукупності речень, визначити контекст між ними, не орієнтуючись на близькість між словами та послідовність їх читання. Це найбільш сильно з існуючих моделей

нагадує те, як люди в реальності сприймають текст: не з позиції послідовності слів, а з позиції великої картини, охоплюючи декілька слів одразу.

1.7 Трансформер

Хоч спочатку механізм уваги використовувався в комбінації з іншими алгоритмами (LSTM), пізніше виявилось, що він навіть краще працює сам по собі.

В статті «Увага - це все, що вам потрібно» (2017) [5] представлено нову модель під назвою Трансформер. Як вказує заголовок, ця модель використовує механізм уваги, який було розглянуто вище. Трансформер, як і класичні Seq2Seq моделі, використовує Кодер і Декодер для перетворення однієї послідовності в іншу. Проте архітектура Трансформера відрізняється від інших Seq2Seq моделей тим, що відмовилась від використання рекурентних мереж (GRU, LSTM тощо). Все, з чого складається Трансформер – декілька шарів уваги та шари прямого поширення між ними, що надають деякий простір для проміжної підготовки даних.

Трансформер виконує невелику, постійну кількість кроків (обраних емпірично). На кожному кроці він застосовує механізм самоуваги, який безпосередньо моделює взаємозв'язки між усіма словами в реченні, незалежно від їх положення. Таким чином, Трансформер представляє кожне слово вектором, в якому закодований весь його контекст. Механізм уваги потрібен для збору відповідного контексту.

Будь-яке слово може мати кілька значень і по-різному взаємодіяти з іншими словами. Багатоголова увага (Multi-Head Attention) – це модуль механізму уваги, який здатен виконати звичайний процес уваги паралельно кілька разів і потім об'єднати незалежні результати [5]. Це може допомогти

наприклад інтерпретації анафор. Наприклад, "Кішка не переходила вулицю, бо вона була занадто ... (втомлена/широка)". Тут "вона" може означати кішку або вулицю в залежності від останнього слова.

Моделі типу Трансфомер є підмножиною нейронних мереж на графах. Вони представляють вхідне речення як повнозв’язний граф із словами вершинами і використовують увагу, щоб для кожної вершини зібрати інформацію про її сусідей.

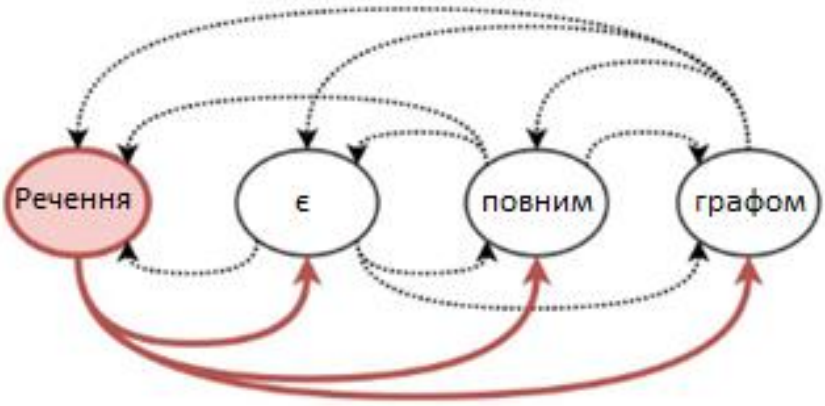

Рисунок 2 – Представлення речення у вигляді повного графа

Класична архітектура Трансформер, що складається з Енкодера і Декодера найкраще підходить для машинного перекладу. Подальші експерименти показали, що можна відкинути Енкодер або Декодер, тобто використовувати лише один стек блоків Трансформера, і таким чином отримати переваги для інших задач. Так з’явились архітектури Трансформер-Енкодер і Трансформер-Декодер.

Однією з найбільших проблем NLP є дефіцит даних для тренування. Більшість наборів даних для конкретних задач містить лише кілька тисяч прикладів позначених людьми. Щоб подолати цей розрив, в NLP почати використовувати трансферне навчання. Трансферне навчання – це така стратегія

навчання, за якої модель, навчена під одну (зазвичай генералізовану) задачу, потім повторно використовується як відправна точка для моделі другої (більш конкретизованої) задачі.

1.8 Моделі GPT і BERT

Однією з перших і найпопулярніших мовних моделей, що використовує трансферне навчання, стала GPT (Generative Pre-trained Transformer), розроблена OpenAI на базі архітектури Трансформер. GPT навчена за допомогою попереднього тренування з використанням величезної кількості нефільтрованого контенту, в тому числі із Інтернету. Завдяки цьому все, що потрібно користувачу моделі, щоб отримувати досить непогані результати – це тонке налаштування невеликим об’ємом анотованих даних під конкретну прикладну задачу. Ця модель показала потужність попереднього тренування, його здатність покращувати генералізацію моделі та економічність до ресурсів, оскільки тонке налаштування під конкретну задачу потребує значно менших затрат часу та даних, ніж навчання з нуля.

Значну роль відіграла і кількість параметрів. Кількість параметрів GPT-2 перевищує 1.5 мільярдів. Остання розробка OpenAI GPT-3 має кількість параметрів 175 мільярдів.

GPT – це авторегресійна модель. Це означає, що вона генерує один токен (слово) за раз, додає його до вхідного тексту, і на наступному кроці ця нова послідовність подається за вхід GPT. Авторегресійні нейронні мережі можуть бути використані для моделювання розподілів будь-яких складних елементів, що розкладаються на впорядковану послідовність більш простих елементів, в якій кожен наступний елемент залежить залежить від попередніх.

BERT (Bidirectional Encoder Representations from Transformers) – модель попереднього навчання на базі Трансформер для обробки природної мови, яка розроблена в Google [6]. Вона має тип двоспрямований енкодер, а також стала першою повністю двоспрямованою мовною моделлю. Це означає, що вона обумовлена як лівим, так і правим контекстом. Для того, щоб слова не бачили самих себе в контексті при попередньому навчанні, BERT використано таку техніку: близько 15% слів у вхідному реченні скриваються маскою, а моделі ставиться за ціль правильно вгадати ці слова.

Для того, щоб BERT міг краще справлятися з визначенням зв'язків в декількох реченнях, попереднє навчання включає додаткове завдання: дано два речення A і B; яка ймовірність того, що B слідує після A?

GPT працює традиційним методом для моделювання мови, тобто ця модель генерує наступне слово в реченні. Цей метод односпрямований, тобто обумовлений лише лівим контекстом, тому GPT значно відстає від BERT в аналізі мови і зв’язків слів в тексті, вона може часто помилятися. Але головна перевага такого способу в тому, що GPT здатна генерувати з нуля цілі довгі параграфи тексту. BERT, яка більш орієнтована саме на коректний аналіз текстів, не досягла такого успіху в генерації тексту, але вона і не була створена для цього.

Оскільки метою BERT є лише моделювання природної мови, для неї необхідний лише енкодер, тому BERT використовує архітектуру Трансформер-Енкодер. В той час як GPT для генерації наступного слова використовує архітектуру Трансформер-Декодер. Одна з головних відмінностей між доступними моделями GPT і BERT різного розміру полягає в висоті стека Енкодерів/Декодерів та в кількості параметрів, які також значно впливають на результат.

1.9 Огляд підходів до підвищення різноманітності генерації тексту

Моделі мовлення, які працюють за допомогою причинно-наслідкових зв’язків, такі як GPT, навчені прогнозувати ймовірність наступного слова з урахуванням певного контексту. Якщо кожного разу для генерації обирати слово з найбільшою ймовірністю, то модель буде потрапляти у цикл і видавати однакові конструкції кожного разу. Так трапляється, тому що найбільша частина уваги моделі приділена одним і тим самим словам.

Для уникнення цього часто використовують методи, основані на семплінгу з випадкового розподілу [2]. Але цей підхід також не ідеальний. Його проблема в тому, що хоча нижні елементи розподілу кожен окремо мають низьку ймовірність, їх дуже багато, і вони в сумі можуть займати високу частину ймовірності. Якщо наприклад ця сумарна ймовірність дорівнює 25%, це означає, що кожен раз при генерації наступного слова шанс відійти від лінії розповіді дорівнює ¼.

Щоб побороти цю низку помилок найпопулярнішими є методи температури та найбільших k.

Метод температури заснований на положенні з термодинаміки, що висока температура робить виникнення станів з низькою енергією менш ймовірним [2]. В імовірнісних моделях логіти відіграють роль енергії, і ми можемо поділити логіти на температуру перед тим як подавати їх на вхід softmax щоб отримати ймовірності вибірки. Зниження температури робить модель більш впевненою у виборі, тоді як високі температури знижують впевненість. Нескінченно висока температура відповідає рівномірному розподіленню, тоді як нульова температура відповідає функції argmax.

Метод найбільших k означає сортування за ймовірністю та відкидання всіх елементів, що знаходяться нижче k-ї позиції. Здається, цей метод дозволяє значно зменшити шанс відійти від теми, тому що хвіст видаляється. Але він не враховує, що в деяких випадках є широкий вибір слів, які підходять для застосування в даній ситуації, а в деяких випадках немає.

Для подолання цієї проблеми був створений метод найбільших p [2]. В цьому методі відсортовані за ймовірністю елементи відбираються по черзі, для них обчислюється кумулятивний розподіл p, і як тільки p досягне деякого значення, всі інші елементи відкидаються.

1.10 Генеративно-змагальні мережі

GAN (Генеративно-змагальні мережі) – це метод тренування нейронних мереж, основною ідеєю якого є створення Генератора та Дискримінатора, які змагаються впродовж навчання. Поняття застосовується до будь-якого типу задач, для яких можуть бути побудовані Генератор і Дискримінатор. Генератор, який здатний створювати зразки даних схожі на реальні, намагається ввести в оману Дискримінатора, який навчений щоб відрізняти правильні зразки даних від неправильних. Більш формальний опис роботи GAN можна виразити так: моделі-дискримінатори вивчають межі між класами, моделі-генератори моделюють розподіл окремих класів.Таким чином ціллю мережі Генератора є підвищити процент помилок Дискримінатора, а ціллю Дискримінатора є навпаки покращення точности розпізнавання. Перевага GAN в тому, що вона не потребує великого вхідного набору даних. В багатьох реальних сценаріях отримання високоякісних анотованих даних є дорогим і трудомістким; навпаки, «сирі» приклади, що характеризують цільове завдання, загалом можна легко зібрати.

GAN не може бути застосована безпосередньо до задач області природної мови, оскільки простір, у якому задаються послідовності мовних елементів, не є неперевним і, отже, не диференційованим.

1.11 Навчання з підкріпленням

Навчання з підкріпленням – це метод тренування нейронних мереж, суть якого в тому, що цільова модель в процесі тренування взаємодіє з деякою середою і з'ясовує, як коригувати свої ваги так, щоб максимізувати винагороду, отриману від цієї середи. Навчання з підкріпленням нещодавно почало набирати популярність в задачах NLP.

На відміну від навчання з учителем, навчання з підкріпленням використовує скалярне оціночне значення винагороди замість явних міток тренувальних зразків. Цей підхід дозволяє використовувати непаралельний корпус даних для навчання моделі, а також підтримує не диференційовані методики тренування.

Методи градієнту стратегій (policy gradients methods) – це техніки навчання з підкріпленням, що базуються на оптимізації параметризованих стратегій щодо очікуваної винагороди від середи за допомогою градієнтного спуску.

Вони не страждають від багатьох проблем, які властиві традиційним підходам до навчання з підкріпленням, таким як відсутність гарантій наявності ціннісної функції, проблема взаємодії, яка виникає внаслідок невизначеної інформації про стан, та проблема складності, що виникає внаслідок постійних станів та дій.

Задача агента навчання з підкріпленням - максимізувати очікувану винагороду при дотриманні параметризованої лінії поведінки π. Припустимо θ –

це набір параметрів (коефіцієнти чи ваги нейронної мережі), а винагорода деякого сценарію τ – це $r(\tau)$. Тоді справедливе таке означення:

$$J(\theta) = \pi[r(\tau)]$$

Як і будь-якій іншій проблемі машинного навчання, якщо знайти параметри θ, які максимізують J, задачу вирішено. Стандартним підходом до вирішення цієї проблеми максимізації є використання градієнтного підйому (або спуску). У градієнтному підйомі необхідно обходити параметри, використовуючи наступне правило оновлення:

$$\theta_{t+1} = \theta_t + \alpha \nabla J(\theta_t)$$

Теорема градієнту стратегій: похідною очікуваної винагороди є очікування добутку винагороди та градієнта логарифма стратегії π.

Переваги методів градієнту стратегій:

- Їх можна використовувати в задачах з дуже великою або неперервною множиною дій,
- Уникає конфлікту між експлуатацією та дослідженням, тому що оптимізує стохастичну стратегію π.

Недоліком є досить низька швидкість роботи, що є слідством приблизного обчислення градієнту.

1.12 Оцінка збереження змісту тексту Word Mover's Distance

WMD (Word Mover's Distance) – це метод оцінювання семантичної відстані двох документів. Він дозволяє виміряти змістовну різницю між документами навіть якщо в них немає однакових слів. Семантична оцінка здійснюється за допомогою переводу слів в їх векторні представлення використовуючи алгоритм word2vec.

Для того щоб можна було отримати векторні преставлення слів, повинно бути проведено попереднє тренування моделі. Тоді розбіжність між документами можна визначити як мінімальну кумулятивну відстань, яку потрібно подолати словам одного документа щоб досягти семантичного значення слів іншого документа.

Семантична відстань між двома документами задається за допомогою векторного представлення документів, метрики схожості та матриці потоку. В якості метрики схожості використовується евклідова відстань між двома векторами слів.

Важливо зазначити, що WMD не підтримує слова, що не входять до попередньо заданого словника, він напряму їх ігнорує.

WMD погано застосовується до документів з великою кількістю унікальних слів тому, що його складність алгоритму $O(p^3\ log\ p)$, де p – це кількість унікальних слів в обох документах.

2 ПОСТАНОВКА ЗАДАЧІ

В цьому розділі будуть описані задачі формалізації та деформалізації тексту, представлені вимоги до розроблюваної системи, обґрунтований вибір засобів для її реалізації.

2.1 Опис початкових даних і вимог до системи

Сформулюємо задачу перетворення стилю тексту. Нехай є клас A текстових послідовностей, що мають стиль A. Аналогічно, до класу B належать всі послідовності, що мають стиль B. Задача полягає в тому, щоб поставити у відповідність для вхідної послідовность з класу A іншу послідовність з класу B так, щоб семантична відстань між ними була достатньо малою. Також вихідний текст повинен бути достатньо виразним і грамотним з точки зору мовної моделі і людської оцінки.

Перша частина задачі – це створення і навчання системи перетворення текстових послідовностей за правилами вище. Друга частина – це оцінювання її роботи.

Інформацію про стиль тексту неможливо витягнути з тексту без людської оцінки або методів глибокого навчання, тому що інформація про стильові характеристики дуже хаотично структурована і не має очевидної математичної моделі. Так як вимоги до системи полягають в тому, щоб обійтися без паралельного корпусу даних та без людських анотацій, додаткова задача полягає в тому, щоб створити систему глибокого навчання для оцінки перетворення стилю тексту.

Стиль тексту погано корелює зі збереженням змісту тексту, тому гарним рішенням буде розподілити задачу оцінки якості перетворення тексту на два модулі з відповідним призначенням.

2.2 Вимоги до апаратних, програмних і комунікаційних інтерфейсів

Розроблювана програма є кросплатформеною, тому доступна для використання на операційних системах Linux та Windows. Для запуску програми потрібна середа Python мінімальної версії 3.7.4, встановлені бібліотеки PyTorch [13], Transformers [8].

2.1 Мова розробки

Для розробки систем штучного інтелекту слід використовувати мову програмування, яка є стабільною, гнучкою та має доступні інструменти. Тому для розробки обрана мова Python, яка має все вищезазначене, а також відома своєю простотою, незалежністю від платформи та великою кількістю математичних бібліотек і бібліотек для машинного навчання.

Python дозволяє писати лаконічний, зрозумілий людям код. Тоді як задачі машинного навчання вимагають використання складних алгоритмів, мова Python дозволяє зконцентруватись на вирішенні конкретної задачі замість технічних деталей.

2.4 Бібліотека PyTorch

Для глибинного навчання обрано бібліотеку PyTorch, яка підтримує тензорні обчислення з розвинутою підтримкою прискорення на GPU та має доступні можливості для контролю графу обчислень [13]. PyTorch відноситься до бібліотек з динамічним графом обчислень. Це значить, що граф обчислень будується в момент його виконання. Це дає можливість в будь який момент його перебудувати, навіть коли модель вже збережена. Подібний підхід дає максимальну гнучкість і розширюваність, дозволяє використовувати в обчисленнях всі можливості мови програмування

2.5 Інструменти для розробки

Перейдемо до вибору та обґрунтування технологій розробки. Для перших спроб тренування моделей був обраний Google Colab, тому що він дозволяє використати облачні TPU для підвищення швидкості тренування. Як IDE для розробки програми був обраний текстовий редактор Atom в зв’язці з плагіном Hydrogen, що надає Atom деякий функціонал IDE не перегружаючи його інтерфейс, і не жертвуючи при цьому простотою і швидкістю роботи.

3 МОДЕЛЬ

В даному розділі буде представлено обрану архітектуру моделі та обґрунтований її вибір. Використаємо часткову GAN-архітектуру Генератор-Дискримінатор за принципами навчання з підкріпленням. Головна ідея в тому, що Генератор намагається максимізувати оцінку, видану йому Дискримінатором. Також потрібно правильно вибрати метод обрахування оцінки так, щоб вона була збалансованою відносно вимог до вихідного тексту. Модуль Дискримінатора поділимо на декілька модулів для того, щоб відокремити задачу перетворення стилю від задачі збереження семантичного значення.

Модель складається з таких підсистем: Генератор, Стилістичний дискримінатор, Семантичний оцінювач.

Для Генератора використаємо архітектуру GPT-2 Small (12 шарів, 768 скритих шарів, 12 голов уваги, 117M параметрів), для Дискримінатора стилю викомисаємо архітектуру BERT Small (12 шарів, 768 скритих шарів, 12 голов уваги, 109M параметрів). Семантичний оцінювач працює на базі алгоритму Word Mover’s Distance.

В якості алгоритму оптимізації використовуємо AdamW, який на сьогодні є стандартом за швидкістю навчання.

Перейдемо до опису структури моделі і призначення кожного її компоненту.

3.1 Підсистема «Генератор»

Генератор побудований на базі авторегресивної моделі OpenAI GPT-2. Він приймає на вхід текст в деякому стилі і намагається перефразувати його так, щоб в результаті вийшов текст у деякому іншому стилі.

Формат моделі являє собою послідовність елементів, який починається з токена <|startoftext|>, і закінчується токеном <|endoftext|>.

Для того щоб формат завдання відповідав можливостям GPT-2, в якості вихідних даних використовуєься конкатенація двох текстів. Ці два тексти відокремлюються один від одного за допомогою спеціального токену ~@, який обрано за принципом найменш часто вживаних символів. Векторне представлення спеціальних токенів разом із векторними представленнями словника тексту передається моделі для того, щоб вона їх відрізняла від звичайних слів.

3.2 Евристичні фільтри

Очевидним доповненням до оцінки Дискримінаторів для підвищення якості згенерованого тексту є такі прості евристичні фільтри:

1. відхиляти створення нових, неіснуючих слів,
2. відхиляти фрази зі словами, що повторюються кілька разів,
3. відхиляти фрази з двома незв'язаними дієсловами поспіль,
4. відхиляти фрази без розділових знаків або із занадто великою кількістю розділових знаків.

3.3 Підсистема «Стилістичний дискримінатор»

Задача Дискримінатора – оцінити наскільки добре Генератору вдалося виконати перетворення стиля тексту, щоб ця оцінка була використана при подальшому навчанні Генератора. Дискримінатор є тернарним класифікатором, побудованим на основі моделі BERT.

Попереднє навчання не гарантує, що Дискримінатор правильно вивчить патерни стилю, для цього використане змагальне тренування, що було запозичене у GAN (Generative Adversarial Networks). Дискримінатор навчається таким чином, щоб відрізняти написані людиною тексти від таких, що створив Генератор. Також BERT здатен оцінити перетворене речення з точки зору мовної моделі завдяки попередньому тренуванню.

3.4 Підсистема «Семантичний оцінювач»

Для оцінки збереження змісту тексту використаємо відомий та перевірений підхід, що називається Word Mover’s Distance. Він дозволяє виміряти семантичну різницю між документами. Для отримання семантичної оцінки сформованого речення Беремо негативне значення Word Mover’s Distance для двох речень і ділимо його на довжину довшої послідовності.

3.5 Попереднє тренування

Генератор і Стилістичний дискримінатор попередньо натреновані перед стадією навчання з підкріпленням.

Попереднє тренування дало такі переваги:

- Генератор навчився розуміти семантику та стиль з цільового корпусу даних;

- Генератор отримав набір початкових параметрів, що призвело до швидшого навчання моделі. Це є значним результатом, враховуючи, що навчання з підкріпленням потребує більше часу, ніж навчання під наглядом.
- Для Стилістичного дискримінатора попереднє тренування означало те, що він достатньо навчився розрізняти формальний і неформальний класи стилю тексту, щоб вдалося запустити процес змагального навчання з підкріпленням.

.

3.6 Навчання з підкріпленням

Виведений Генератором зразок тексту відправляється на оцінку Стилістичному дискримінатору, Семантичному дискримінатору і Мовному модулю. Ці модулі дають оцінку роботі Генератора з ціллю його тренування, що надає можливість покращити майбутню генерацію зразків. Для обчислення градієнту моделі був вибраний метод градієнту стратегій.

Опишемо задачу в термінах навчання з підкріпленням. Визначимо Визначимо дію n_t у час t як слово в послідовності під номером t, перші t – 1 слів вважаємо вже згенерованими.

Припустимо n_t – це дія, яка підлягає оцінці. Нехай $J_{style}(n_t)$ – це оцінка від дискримінатора стилю тексту, $J_{semantics}(n_t)$ – це оцінка збереження змісту тексту, $J_{language}(n_t)$ – це оцінка грамотності тексту. Оцінка $J(n_t)$ стану N_t визначається як зважена сума оцінок від оцінюючих модулів, де α, β, γ – це позитивні гіперпараметри:

$$J(n_t) = \alpha\ J_{style}(n_t) + \beta\, J_{semantics}(n_t) + \gamma\, J_{language}(n_t)$$

В поточній задачі визначаємо значення гіперпараметрів методом підбору:

$$\alpha = 0.7\ ,\ \beta = 0.4\ ,\ \gamma = 0.4.$$

Тоді кумулятивна оцінка $Q(N_t)$ від послідовності дій N_t визначається як сума оцінок окремих слів:

$$Q(N_t) = \sum_{i=1}^{t} J(n_t) \quad (1)$$

Оцінювальні модулі розроблені таким чином, що приймають на вхід повні речення, тому не можна порахувати на кожному кроці $J(n_t)$ напряму. Для цього використовується техніка семплінгу. Після генерації наступного слова речення доповнюється до кінця N разів, кожен з N зразків подається на вхід оцінювальним модулям, і обчислюється зважена оцінка $J(n_t)$. Значення N емпірично обране і дорівнює 100.

Функція втрат L_θ Генератора визначається як негативне значення кумулятивної оцінки $Q(N_t)$:

$$L_\theta = -\mathrm{Q}(N_t)$$

З рівняння (1) можемо знайти градієнт Генератора як:

$$\nabla_\theta L_\theta = \sum_{i=1}^{t} \nabla_\theta J(n_t)$$

Змінюючи параметри формули оцінки, можна напряму впливати на навчання Генератора. Псевдокод алгоритму навчання з підкріпленням Генератора та Дискримінатора виглядає так:

Цикл:

Підрахувати та зберегти траєкторії, перевірити кінцеві умови

Цикл:

Семплінг міні-батчу дій або переходів

Обчислити градієнт стратегій Генератора та Дискримінатора

Оновити параметри нейронних мереж

3.7 Змагальне навчання

Стилістичний дискримінатор попередньо натренований на великому корпусі тексту так, щоб відрізняти тексти в одному стилі від текстів в іншому стилі. Але без паралельного корпусу цього виявилось недостатньо для дискримінатора, щоб вивчити деякі елементи стилю. Його точність була близько 0.7.

Запозичивши ідею у генеративно-змагальних мереж, Дискримінатор паралельно з Генератором тренується використовуючи згенеровані їм тексти. Для цього був створений випадково впорядкований датасет текстів, що на 60% складається із зразків людського походження і на 40% – із згенерованих моделлю. В цьому датасеті старі згенеровані зразки поступово замінювались новими, які очікувано покращувались впродовж тренування Генератора. Згенеровані зразки потрібно включити для того, щоб Дискримінатор навчився відрізняти їх від людських зразків.

Нехай Дискримінатор D має параметри μ. Вихідне значення $D_\mu(n_t)$ Дискримінатора D – це ймовірність того, що речення написане в цільовому стилі.

Поставимо за мету змагального навчання Дискримінатора мінімізацію такої функції втрат, де n_i^{human} – речення людського походження, а n_i^G – речення створені Генератором:

$$L_\mu = \frac{1}{K}\left(-\sum_{i=1}^{K}\log(1 - D_\mu(n_i^{human}))\right) - \sum_{i=1}^{K}\log(D_\mu(n_i^G))$$

4 ЕКСПЕРИМЕНТ

В цій роботі проведено експеримент, використовуючи для цього дві протилежні задачі: перетворення стилю тексту з неформального у формальний, і навпаки з формального у неформальний, проаналізовано результати експерименту, зроблено висновки. Для цього були натреновані дві моделі, що здатні виконувати такі перетворення.

4.1 Набір даних

Дані для навчання взяті з таких наборів даних:

- Grammarly’s Yahoo Answers Formal Corpus (GYAFC) – це доступний за запитом набір текстових даних, розміщених на форумі з питаннями-відповідями (Yahoo Відповіді) і написаних в неформальному стилі. Ці речення були переписані вручну в офіційному стилі. В навчанні використовувались дані з розділу «сім'я та стосунки». Незважаючи на те, що корпус є паралельним, паралельна інформація не була використана в роботі щоб відповідати вимогам проекту [10].
- Enron Email Dataset – це публічний набір даних, що містить близько 0.5M електронних листів, які написані 150 співробітниками корпорації Enron. Це єдиний датасет з електронними листами, який міститься в публічному доступі [9].

словник	тип	train	test	dev
31129	Формальний стиль	130911	3019	1397
	Неформальний стиль	242167	3019	1397

Таблиця 1 – Розмір вихідного набору даних та його словника, використаного для навчання моделей, отриманого методом злиття та обробки вибраних та завантажених наборів даних

4.2 Хід експерименту

Першим кроком експерименту була попередня підготовка даних:

1. Токенізація.
2. Вилучення зайвих символів.
3. Вилучення зайвих слів.
4. Створення словника.
5. Нормалізація даних.

Обрані набори даних були злиті в один набір та розділені на частини для використання при тренуванні, тестуванні та валідації.

Далі було проведене форматування і реалізована токенізація.

Після підбирання та встановлення гіперпараметрів мереж були почергово натреновані за вищеописаним алгоритмом дві моделі для протилежних задач формалізації тексту, єдина фактична відмінність яких заміна оцінки Дискримінатора стилю тексту на протилежну.

5 ОЦІНКА МОДЕЛІ

Використане як людське, так і автоматичне оцінювання моделі, щоб перевірити систему з точки зору збереження змісту, сили перетворення стилю, та загальної мовної моделі.

Сила перетворення стилю означає ступінь з якою вихідний текст довелося змінити, щоб отримати текст в вихідному стилі.

Оцінка точності визначається як відсоток згенерованих речень, які були класифіковані критиком стилю як ті, що мають цільовий стиль. Точність використана для оцінки якості перетворення стилю, і чим вища точність, тим краще згенеровані речення відповідають цільовому стилю.

Загалом система значно краще показала себе в задачі перетворення з формального тексту на неформальний, ніж навпаки. Це підтверджує те, що задача формалізації тексту трохи складніша для нейронних мереж, ніж задача деформалізації. Ймовірне пояснення цьому в тому, що неформальний стиль є більш вільним, і тексти в такому стилі є більш різноманітними, тому їх простіше згенерувати. Ще однією проблемою є незначні орфографічні помилки в неформальних даних, зібраних з Інтернету. Це пояснюється тим, що слова – це найменші семантичні елементи моделі, тому вона не здатна побачити близькість слів з різницею в один-два символи.

Існує компроміс між збереженням значення і силою перетворення стилю. Це справедливо тому, що незмінені вхідні дані показують найкраще збереження вмісту при нульовій силі перетворення стилю. Так само екземпляри з найбільшим показником сили перетворення стилю майже в ніякій мірі не зберігають значення тексту.

Тип оцінки	Неформальний –> Формальний		Формальний –> Неформальний	
	Семантика	Стиль	Семантика	Стиль
Людська	0.433	0.515	0.591	0.649
Автоматична	0.698	0.715	0.787	0.862

Таблиця 2. Результати автоматичного та людського оцінювання натренованих моделей після їх тренування.

Для об’єднання двох оцінок сили перетворення стилю $S_{semantics}$ та збереження значення S_{style} використаємо таку формулу загальної оцінки S:

$$S = \frac{S_{semantics} * S_{style}}{S_{semantics} + S_{style}}$$

Для перевірки коректності автоматичної оцінки, було проведено людське оцінювання моделі. Воно було виконане у такий спосіб. Було взято по 50 зразків тексту Генератора і зібрано людські оцінки цих зразків по трьом параметрам від 1 до 5: сила перетворення стилю, збереження змісту тексту та грамотність. Для підрахунку сумарної оцінки було взято середнє значення і переведене в шкалу від 0 до 1.

ВИСНОВКИ

- В цій роботі був втілений експеримент з застосування навчання з підкріпленням та змагального навчання для задачі перетворення стилю тексту, була розроблена і продемонстрована багатомодульна архітектура відповідної моделі.
- Була розроблена система обробки природної мови для зміни стилю тексту за характеристиками формальність/неформальність.
- Були навчені дві моделі, що змінюють формальність/неформальність тексту в один і в інший бік.
- Була перевірена ефективність моделі за допомогою людської та автоматичної оцінки на двох задачах: формалізації та деформалізації тексту.
- Були визначені проблеми моделі, в тому числі невміння опрацьовувати незначні орфографічні помилки, які часто роблять люди, компроміс між силою зміни стилю та збереженням значення, проблеми із опрацюванням структурних змін в тексті.
- Були теоретично описані поточні методи та алгоритми обробки природної мови, окремо були розглянуті такі теми як Трансформери, генеративно-змагальні мережі і навчання з підкріпленням.
- Виконане порівняння ефективності глибокого навчання для задач формалізації та деформалізації тексту.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Diederik P. Kingma, Jimmy Ba, Adam: "A Method for Stochastic Optimization" ArXiv:1412.6980 – (2014) – (доступ до ресурсу: https://arxiv.org/abs/1412.6980).
2. Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, Yejin Choi: "The Curious Case of Neural Text Degeneration" arXiv:1904.09751 – (2019) – (доступ до ресурсу: https://arxiv.org/abs/1904.09751).
3. Ilya Loshchilov, Frank Hutter: "Decoupled Weight Decay Regularization" arXiv:1711.05101 – (2017) – (доступ до ресурсу: https://arxiv.org/abs/1711.05101).
4. "Neural Network Primer: Part I" by Maureen Caudill, AI Expert, Feb. 1989.
5. Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin: "Attention Is All You Need" arXiv:1706.03762 – (2017) – (доступ до ресурсу: https://arxiv.org/abs/1706.03762)
6. Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova: "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding" arXiv:1810.04805 – (2018) – (доступ до ресурсу: https://arxiv.org/abs/1810.04805).
7. Hochreiter Sepp, Jürgen Schmidhuber: "Long short-term memory." Neural computation 9, no. 8 (1997).
8. Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, Alexander M. Rush: "HuggingFace's Transformers:

State-of-the-art natural language processing." arXiv:1910.03771 – (2019) – (доступ до ресурсу: https://arxiv.org/abs/1910.03771).

9. William W. Cohen: "Enron Email Dataset" – (2015) – (доступ до ресурсу: https://www.cs.cmu.edu/~./enron/)
10. Sudha Rao, Joel Tetreault: "Dear Sir or Madam, May I introduce the GYAFC Dataset: Corpus, Benchmarks and Metrics for Formality Style Transfer" arXiv:1803.06535 – (2018) – (доступ до ресурсу: https://arxiv.org/abs/1803.06535)
11. Artetxe, M., Labaka, G., Agirre, E., and Cho, K. 2017: "Unsupervised neural machine translation" arXiv:1710.11041 – (2017) – (доступ до ресурсу: https://arxiv.org/abs/1710.11041).
12. Ralf C. Staudemeyer, Eric Rothstein Morris: "Understanding LSTM – a tutorial into Long Short-Term Memory Recurrent Neural Networks" arXiv:1909.09586 – (2019) – (доступ до ресурсу: https://arxiv.org/abs/1909.09586)
13. Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, Soumith Chintala: "PyTorch: An Imperative Style, High-Performance Deep Learning Library" arXiv:1912.01703 – (2019) – (доступ до ресурсу: https://arxiv.org/abs/1912.01703).

ДОДАТКИ

Приклади роботи моделі Informal-To-Formal:

Неформальний стиль	Формальний стиль
I was a bit tired when she came.	I was somewhat tired when she dropped in.
Flora got together with Tim in experimental study.	Flora collaborated with Tim in experimental study.
Gotta see both sides of the story.	You have to consider both sides of the story.
You realize that's my private stuff.	I hope you understand that it is my private property.

Приклади роботи моделі Formal-To-Informal:

Формальний стиль	Неформальний стиль
He has major psychotic episodes and the episodes usually happen at times when he is highly stressed.	He has big psychotic episodes and these seem to happen at times when he has mega stress.
Can I ask you not to tweet about how gr8 you are, please don’t forget your roots, remain calm.	U don’t tweet how gr8 u r, don’t forget ur roots, stay humble.
My apologies. I’m excited too much!	I’m so sorry but I’m way too excited!!
I do believe it to be punk.	I’d say it is punk though.